UNIVERSITY OF THE AEGEAN
SCHOOL OF BUSINESS SCIENCES
DEPARTMENT OF SHIPPING, TRADE AND TRANSPORT

«IMPLEMENTATION OF THE POLARIS EXPERT SYSTEM»

Thesis for the Post Graduate Programme
«Shipping, Trade and Transport – S.T.T.»

Theodossios Scholiadis

11 October 2007

CHIOS

Theodossios Scholiadis

Implementation of the POLARIS Expert System

11 October 2007

Thesis for the Post Graduate Programme
«Shipping, Trade and Transport – S.T.T.»

Department of Shipping, Trade & Transport

Author: .....Mr. T. Scholiadis.....

Supervisor: .....Mr. N. Nikitakos.....

CHIOS

Synopsis

Η σωστή διαχείρηση μιας επιχείρησης είναι ένα από τα πιο δύσκολα πράγματα που μπορεί να κάνει κάποιος. Για να το πετύχει αυτό ένας manager πρέπει να έχει τις σωστές αλλά και τις σχετκές, με την απόφαση, πληροφορίες που θέλει να πάρει. Η συλλογή αυτών των πληροφοριών μπορεί να είναι εύκολη. Η διαχείρηση αυτών όμως, αποτελει δύσκολη υπόθεση. Η σωστή ή στη χειρότερη περίτωση, η λανθασμένη ερμηνεία των διαθέσιμων πληροφοριών μπορεί να καθορίσουν την επιτυχία ή την αποτυχία της επιχείρησης.

Εδώ εμφανίζεται η χρησιμότητα των Expert Systems. Με απλά λόγια, τα Expert Systems είναι συστήματα που βοηθάνε στη διαδικασία λήψης αποφάσεων, είτε αυτές είναι απλές (π.χ. ποιά είναι η πιο διασκεδαστική δραστηριότητα που μπορεί να κάνει κάποιος), είτε περίπλοκες (π.χ. που θα πρέπει ένα κράτος να ξοδέψει τα περιορισμένα έσοδά του, ώστε να επιτύχει καλύτερη απόδοση). Το POLARIS (POLicy Leading ARtificial Inteligence System) αποτελεί ένα τέτοιο Expert System. Χρησημοποιεί παλιές πληροφορίες οργανισμών και κρατών για να βρει ποιοί από αυτούς τους οργανισμούς έχουν πετύχει τους στόχους που θέλει να πετύχει ο χρήστης. Έπειτα, από τους οργανισμούς που πέτυχαν τους στόχους του χρήστη, βρίσκει τους οργανισμούς που δραστηριοποιούνται σε κράτη παρόμοια με αυτό που δραστηριοποιείτε ο οργανισμός του χρήστη.

Το POLARIS είναι ένα Web Application (εφαρμογή διαδικτύου), το οποίο δημιουργήθηκε από τον συγγραφέα της πτυχιακής αυτής, με τη βοήθεία του κ. Γ. Φύκαρη, χρησημοποιώντας τη φιλοσοφία WAMP (Windows, Apach, MySQL & PHP). Η πτυχιακή είναι χωρισμένη σε τρία μέρη και περιγράφει το POLARIS με τον εξής τρόπο:

- **Μέρος Ι: Θεωρία**
  Αυτό το μέρος αναλύει τις θεωρίες πίσω από το POLARIS. Αυτές είναι η θεωρία της Artificial Intelligence και η θεωρία των Expert Systems. Επιπλέον, για την Artificial Intelligence γίνετε ανάλυση των βασικών τμημάτων της, μια από τις οποίες είναι το Case-Based Reasoning, δηλαδή συλλογισμοί βασισμένοι σε παλιές περιπτώσεις παρόμοιων προβλημάτων. Για τα Expert Systems γίνεται μία ανάλυση ενός τομέα που λέγετε Decision Support Systems.

Αυτοί οι δύο τομείς, δηλαδή το Case-Based Reasoning και το Decision Support Systems, είναι οι δύο βασικές θεωρίες στις οποίες βασίζεται το POLARIS. Επίσης, σε αυτό το μέρος γίνετε και μία αναλυτική περιγραφή του POLARIS, δηλαδή το «τι κάνει», και όχι το «πώς το κάνει».

- **Μέρος II: Σχεδιασμός**
  Αυτό το μέρος της πτυχιακής περιέχει το τεχνικό κομμάτι. Πρώτα δίνει τους λόγους για τους οποίους χρησημοποιήθηκε η φιλοσοφία WAMP ενάντι κάποιων άλλων. Έπειτα συνεχίζει με μία αναλυτική περιγραφή της βάσης δεδομένων και των λόγων για την επιλογή του συγγεκριμένου σχεδιασμού. Τελικά, δίνει μια ανάλυση του γενικού σχεδιασμού του POLARIS με βάση το «πώς λειτουργεί» και με ποιά σειρά πρέπει να εκτελεστούν κάποιες δραστηριότητες.

- **Μέρος III: Τελικές Παρατηρήσεις**
  Αυτό είναι το μέρος στο οποίο δίνονται τα τελικά συμπεράσματα της πτυχιακής. Πρώτα γίνετε μια αναλυτική περιγραφή της περαιτέρω έρευνας που μπορεί να γίνει επάνω στο POLARIS, και πώς μπορεί να αναπτυχθεί η εφαρμογή στο μέλλον. Τελος, ο συγγραφέας εξηγεί πώς καταλήγει στο εξής συμπέρασμα:

  «Το POLARIS είναι ένα πολύ χρήσιμο, ευέλικτο και προσαρμόσιμο εργαλέιο για κάθε βιομηχανία. Με την προϋπόθεση ότι χρησημοποιείτε σωστά, μπορεί να γίνει ένας αξιόπιστος "βοηθός" ενός ειδικού της ναυτιλιακής βιομηχανίας, και στο μέλλον, μπορεί και σε άλλες βιομηχανίες. Αυτό δεν σημαίνει ότι θα είναι η καλύτερη εφαρμογή του είδους, αλλά σίγουρα θα έχει τη δυνατότητα να γίνει μία από τις πιο ανταγωνιστικές.»

(Keywords: POLARIS, Expert, Decision)

Table of Contents

Table of Figures

1. INTRODUCTION

Managing and running a successful business is one of the most difficult things one can do. In order to do this, the manager needs relevant information to the decisions he/she wishes to make. Collecting this information is the easy part. Knowing what to do with it is a whole different matter, and by far the most difficult. The correct or, in the worst case, erroneous interpretation of the available information can determine the success, or failure, of the business.

This is where Expert Systems come in handy. In a few words, Expert Systems are systems that aid in a decision making process, be it a simple one (e.g. what would be a person's most enjoyable activity) or of a more complicated nature (e.g. where should a country spend its limited tax income in order to have the most desirable outcome). POLARIS (POLicy Leading ARtificial Intelligence System) is one such Expert System. It uses past information on organisations and Countries in order to find which of these organisations meet the user's search criteria. It then finds which of these organisations are similar to the one the user is in and which are in a country that is similar to the one that the user's organisation is situated in.

POLARIS is a Web App (Web Application) that was developed/programmed by the author of this thesis (with the help of Mr. G. Fykaris) using the WAMP (Windows, Apache, MySQL & PHP) philosophy. The thesis briefly describes the theory of Expert Systems that was used for POLARIS. It also provides an

analysis of the design for the implementation of POLARIS. In order to achieve this, the thesis is divided into three parts:

- **Part I: Theory**
  Part I is the theoretical aspect of the thesis. It is separated into two chapters:

  Chapter 3 gives a brief description of the theory behind Artificial Intelligence as well as explaining some of its key aspects. It then continues to give a brief description behind the theory of Expert Systems, and more specifically analyses Decision Support Systems.

  Chapter 4 gives a description of what POLARIS is, what it does, and what its uses are in the business world.

- **Part II: Design**
  Part II is the technical aspect of the thesis. It is separated into three chapters:

  Chapter 5 gives an explanation for the reasons that the WAMP philosophy was used as opposed to any other.

  Chapter 6 gives a detailed analysis of the database design and, more importantly, the reasons behind its design.

  Chapter 7 explains the “flow” of POLARIS (flow: how the user “moves through” the application), and the reason behind the chosen design.

- **Part III: Final Remarks**
  Part III is the section of the thesis that provides the relevant conclusions. It is separated into two chapters:

  Chapter 8 gives a detailed description of various recommendations for future development on the POLARIS Web App.

  Finally, Chapter 9 provides the final conclusions for the thesis.

The Appendix will contain Listings of the source code for POLARIS. Please note that some key functions will be left out for Intellectual Property reasons.

2. PROBLEM DEFINITION AND RESEARCH QUESTION

There is one key question that needs to be answered for this thesis to have any meaning:

"Why create POLARIS?"

POLARIS is an Expert System designed to aid business managers in making difficult decisions and selecting appropriate strategies for their businesses, when wanting to achieve certain goals.

So, what makes POLARIS different from other Expert Systems? To begin with, even though it has the capability of being applied to any industry, it was specifically designed for the Shipping Industry. After extensive research by the author and Mr. Fykaris, it was found that there is no Expert System in the market that caters for the shipping industry. Mr. Fykaris, through his research, has developed rules that apply to key issues in the shipping industry (explained in Chapter 4), which will be implemented in POLARIS.

Thus, POLARIS was developed to take into account these rules, process the users' requests and provide appropriate solutions.

PART I: THEORY

3. ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Before going into detail about what POLARIS is or what it does, and analytically describing its functions and future prospects, some basic concepts need to be covered. It has already been mentioned that POLARIS is a Case-Based Reasoning Expert System, but there is no point in making reference to that grandiose title if no one knows what it means.

Before understanding *what* POLARIS does and *how* it does it, one needs to understand the basic concepts of the theory that it is based on. This is what this chapter aims to do. It aims to explain what a Case-Based Reasoning System is (as part of the field of Artificial Intelligence) as well as what Expert Systems are. Even though one can understand how to use POLARIS without this knowledge, his/her knowledge of POLARIS would be lacking, and could, therefore, unknowingly make fundamental mistakes in the use of POLARIS.

3.1 Brief Description of Artificial Intelligence

A basic definition that can be given for Artificial Intelligence is:

> *"A branch of computer science that studies how to endow computers with capabilities of human intelligence"*[1]

Concepts on Artificial Intelligence have been around since the time of Aristotle (*"Philosophy of Nature"*) and have been worked on and improved throughout history by people like Descartes (*"Meditations"*), Leibiz, and Euler (*"Graph Theory"*). This work has led to the theory of *"State Space Search"*, which involves the solving of a problem through the analysis of various situations in which the problem exists. This analysis links the situations through Inference Rules, thus creating a "problem diagram" leading to a feasible and viable solution to the problem.[2]

Researchers like Charles Babbage (*"Difference Engine"*), Ada Lovelace, Gottlob Frege (*"Foundation on Arithmetic"*) and George Boole (*"Boolean Algebra"*) added their invaluable input to this growing field of research. This allowed Russell & Whitehead to broaden the field by creating a system that solved a problem "step by step" and was based solely on mathematics. With this method, a problem's details and processes were able to be separated from their real world equivalents. Later, Alfred Tarski developed the *"Reference Theory"*, which linked Russell & Whitehead's mathematical formulas to the real world.[3]

During the 1950's a large number of developments were made in the field of Artificial Intelligence, the most important of which were:

- A.M. Turing's *"Computing Machinery and Intelligence"* (also known as the *"Turin Test"*),
- The GPS (*"General Problem Solver"*[4]), considered to be the first "thinking machine" to be created,
- The development of the LISP (LISt Processor) programming language[5], used in a variety of Artificial Intelligence applications,
- The paper titled *"Programs with common sense"*,
- The paper titled *"Pandemoneum"*[6],

[1] www.netdictionary.com
[2] G. Luger & W. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, Sec1.3 & Ch.3, Benjamin/Cummings Publishing Company, 1989
[3] Tarski, 1944, 1956
[4] Newell, Shaw & Simon (1957)
[5] John McCarthy (MIT), 1958

- The paper titled *"Some Methods of Heuristic Programming and Artificial Intelligence"*[7], and
- Margaret Masterman's *"Semantic nets"* theory.

From the year 1961 to the year 1989 various programs and systems were developed by prominent figures of the "Artificial Intelligence" field that aimed in mimicking real world situations. The programs ranged from systems that could hold a simple conversation in English[8], to systems that could devise a complete "architectural solution" to a problem[9], to autonomous land vehicles based on neural networks[10].

The 1990's were a milestone decade for the field of Artificial Intelligence, as a series of new methodologies and innovations were added to the already great wealth of knowledge in the field. These aspects included the following:

- Machine Learning,
- Intelligent Tutoring,
- Case-Based Reasoning,
- Multi-agent Planning,
- Planning & Scheduling,
- Uncertain Reasoning,
- Data Mining, and
- Natural Language Understanding and Translation.

For completion purposes, a more detailed description of each aspect of Artificial Intelligence mentioned above follows:

[6] Oliver Selfridge, 1958
[7] By Marvin Minsky
[8] ELIZA, by J. Weizenbaum
[9] SOAR (J. Laird, P. Rosenbloom & A. Newell, 1983)
[10] ALVINN, by D. Pomerleau, 1989

3.1.1 Machine Learning

Machine Learning is what a program uses when changes that occur in the program through the constant flow of information lead to the improvement and more accurate performance of the program. Machine learning programs are equipped with specialised algorithms that detect errors/bugs in the program and "teach" the program not to repeat them in the future. This results in the more accurate performance of the program.[11]

3.1.2 Intelligent Tutoring

Intelligent Tutoring programs are training/teaching programs that are equipped with components based on Artificial Intelligence, and are based on research in the fields of Artificial Intelligence and Clinical Psychology. Although it might seem obvious why Artificial Intelligence is useful in these programs (see section 3.1.1 above), it might not be as obvious why Clinical Psychology is required. Clinical Psychology looks for the mechanisms behind the human thought process, the use of language, mathematical reasoning and the use of memory.

While a student uses an Intelligent Tutoring program, the program analyses the student's strengths and weaknesses, and provides appropriate guidance to the student, with the aim of improving the student's abilities.[12]

3.1.3 Case-Based Reasoning

Case-Based Reasoning is a method of reasoning used by programs which solve problems. The solving of these problems is done by comparing the present problem to passed cases of problems very similar to the present one, using a solution that worked in the past. The three steps used for solving a problem using this method, are the following:

1. Find the past case of a problem that is MOST similar to the present problem,
2. Apply the solution used in the past case, and
3. Store the results of the use of the solution to the present problem so that the present problem can be used as a case for future problems.[13]

It is important to note that Case-Based Reasoning is the method that POLARIS is based on.

[11] N. Nilsson, *What is Machine Learning*, Ch. 1, 1996
[12] J. Potts, *Carnegie Melon Today*, 2005
[13] A. Aamodt & E. Plaza, 1994

3.1.4 Multi-agent Planning

Multi-Agent Systems (MAS) are systems in which many programs interact with each other through cooperation or through competition. These programs could be working towards a common goal, or towards the solution of independent problems.

These systems are usually used in (and originated from) the fields of Biology, Sociology, Economics, Philosophy, and other fields with similar characteristics.[14]

3.1.5 Planning & Scheduling

Planning and Scheduling are the parts of two part systems know as "Planning & Scheduling Systems". These are systems that provide reasons for the order and time that certain tasks are completed in, for the timely completion of a project. Scheduling refers to the "when" part of the problem, whereas Planning refers to the "order" the tasks are completed in. Scheduling on its own, or Planning on its own, are fairly trivial problems to solve, however, combining the two concepts gives rise to a problem that is no longer trivial, and requires the use of fairly sophisticated algorithms in order to find a solution.[15]

3.1.6 Uncertain Reasoning

Uncertain Reasoning systems use the power of computational systems to infer useful solutions to problems, using the probability of the occurrence of certain events.[16] This is possible through the "Bayes Theory" and the "Markof Hidden Models" theory. The central idea behind both theories is the fact that the probability of something happening in the future can be foreseen if it is based on the frequency of its occurrence in the past.[17]

3.1.7 Data Mining

Data Mining is a tool that is used by Artificial Intelligence systems, and is used for the extraction of data and metadata from large databases. It is based on the development of specialised algorithms which are able to manage and process the data, in order to retrieve the required information.[18]

---

[14] K. Sykara, 1998

[15] T. Dean & S. Kambhampati, *Planning and Scheduling*, CRC Handbook of Computer Science and Engineering, 1996

[16] D. Leake, *Artificial Intelligence*, University of Indiana, 2002

[17] *Van Nostrand Scientific Encyclopedia*, Ninth Edition, Wiley, New York, 2002

[18] D. Michie, *A prototype knowledge refinery*

3.1.8 Natural Language Understanding and Translation

Natural Language Understanding and Transalation uses the ability of computers to process and understand natural language, as well as to translate words and phrases from one language to another. "Linguistics" is the creation of programs and/or models that can be used by a computer in order to understand and translate the natural languages.[19]

3.1.9 Artificial Intelligence related to other fileds of Computer Science

The Figure below shows how Artificial Intelligence fits in relation to the other fields of Comuter Science:

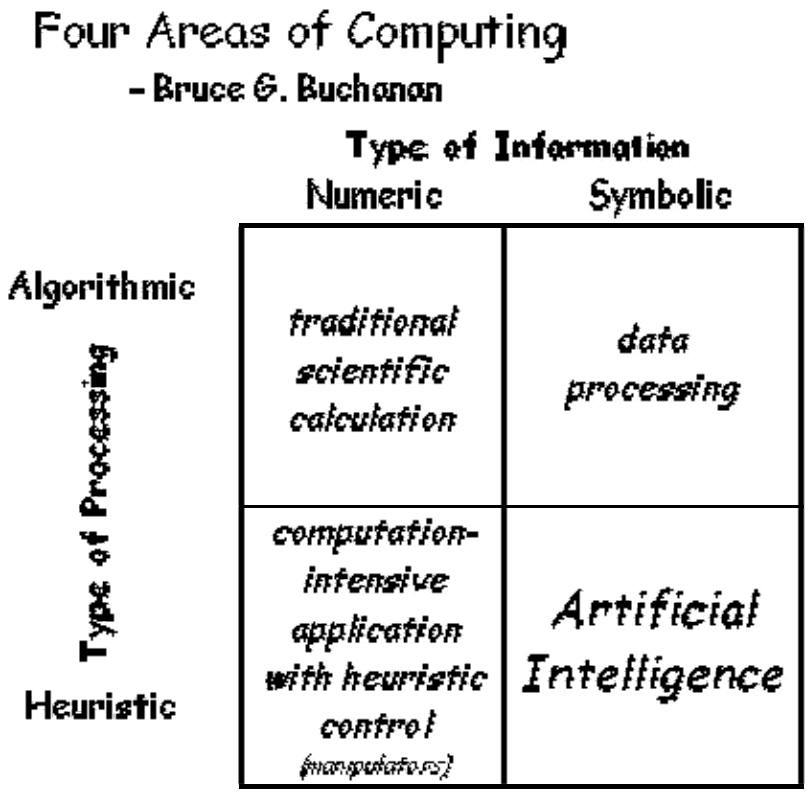


Figure 3.1: AI in Relation to other fields of Computer Science[20]

As has already been mentioned, POLARIS makes use of the Case-Based Reasoning aspect of Artificial Intelligence, and in conjunction with the theory of Expert Systems (analysed in the following section) will aid in providing the users of POLARIS with the required and most accurate results.

[19] J. Batali, *Artificial Intelligence Modelling*
[20] Source of Figure: Bruce G. Buchanan, University of Pittsburg

3.2 Brief Description of Expert Systems

There are three definitions of Expert Systems that describe their use and function. These are:

> *"[Expert Systems] are man-machine systems with specialised problem-solving expertise. The* expertise *consists of knowledge about a particular domain, understanding of problems within that domain, and* skill *at solving some of these problems."*[21]

> *"[Expert Systems are] systems in which human expertise is held in the form of rules which enable the system to diagnose situations without the human expert being present."*[22]

> *"[An Expert System is a] software that behaves in much the same way as a human expert being present."*[23]

These definitions also describe the basic concepts behind POLARIS (as will be shown in Chapter 4 that follows). Fortunately, the three definitions also explain *exactly* what an Expert System is and/or does, and what it is used for. Basically, it is a system design to do the work of an expert in the field the software was designed for.

Unfortunately, the term "Expert Systems" refers to a very broad field of research. POLARIS only makes use of one part/aspect of Expert Systems, namely "Decision Support Systems" (DSS). A good definition of what is a Decision Support System is:

> *"Decision support systems are a class of computerised information systems that support decision making activities."*[24]

But what does that mean?

As the first researchers of the field found (Gorry & Scott-Morton in 1971), Decision Support Systems are systems designed to aid decision makers in making decisions for non-standard problems. Later, in 1978, Keen & Scott-Morton defined a Decision Support System as:

> *"A system that combines human intellect with computers, in order to improve the quality of decision making."*

Gorry & Scott-Morton distinguished three levels of management that a decision support system can be used in. These are:

[21] dssresources.com/glossary/dssglossary1999.html
[22] www.dti.gov.uk/actt/technology/glossary.html
[23] www.ballyclarehigh.co.uk/garden91/Glossary_finalRW.htm
[24] en.wikipedia.org/wiki/Decision_support_systems

1. Operational Control,
2. Management Control, and
3. Strategic Planning.

In addition, there are three types of problem structures that these systems are designed to solve. These are:

1. Structured problems,
2. Semi-structured problems, and
3. Unstructured problems.

Using this information, the Decision Support Systems are categorized around these two "groupings" (or axes), namely:

1. The "Management Levels", and
2. The "Degree of Problem Structure".

Steven Alter developed a matrix that places the various decision support systems on the matrix, based on the management level they are used in and the degree of problem structure they are designed to solve:

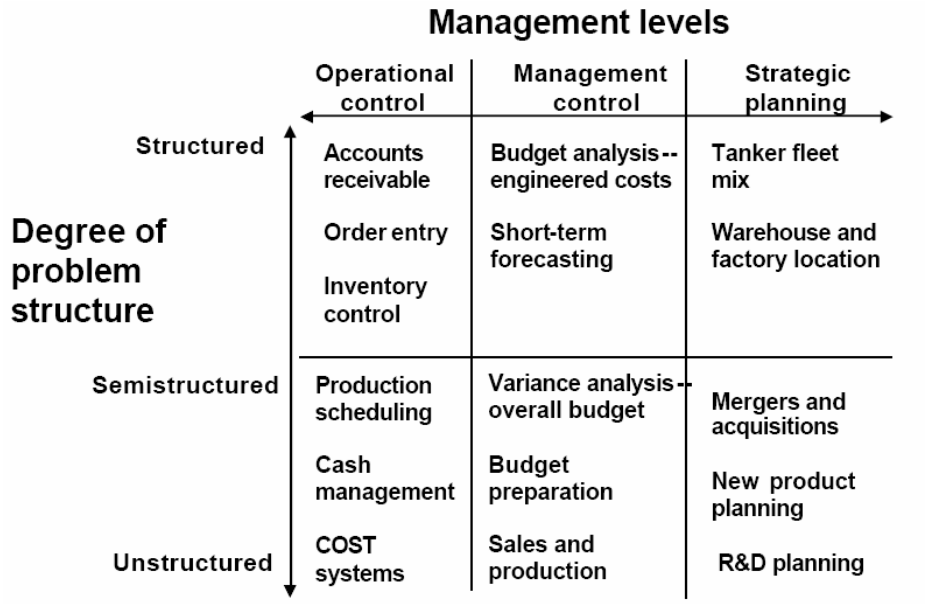


Figure 3.2: Gorry & Scott-Morton DSS Grid[25]

From this matrix one can see that a Decision Support System designed for a shipping company would be placed in the column of "Strategic Planning" due to

[25] Source of Figure: University of Stirling, 2004

the nature of the industry, and the degree of problem structure would be based according to the policy of the problem the user is trying to solve. Needless to say, that is where POLARIS would be placed, as it is a system designed for Shipping Companies.

Steven Alter developed a second matrix, one that places a Decision Support System in the matrix based on the Degree of Problem Solving Support and the Degree of Complexity of the Problem-solving System:

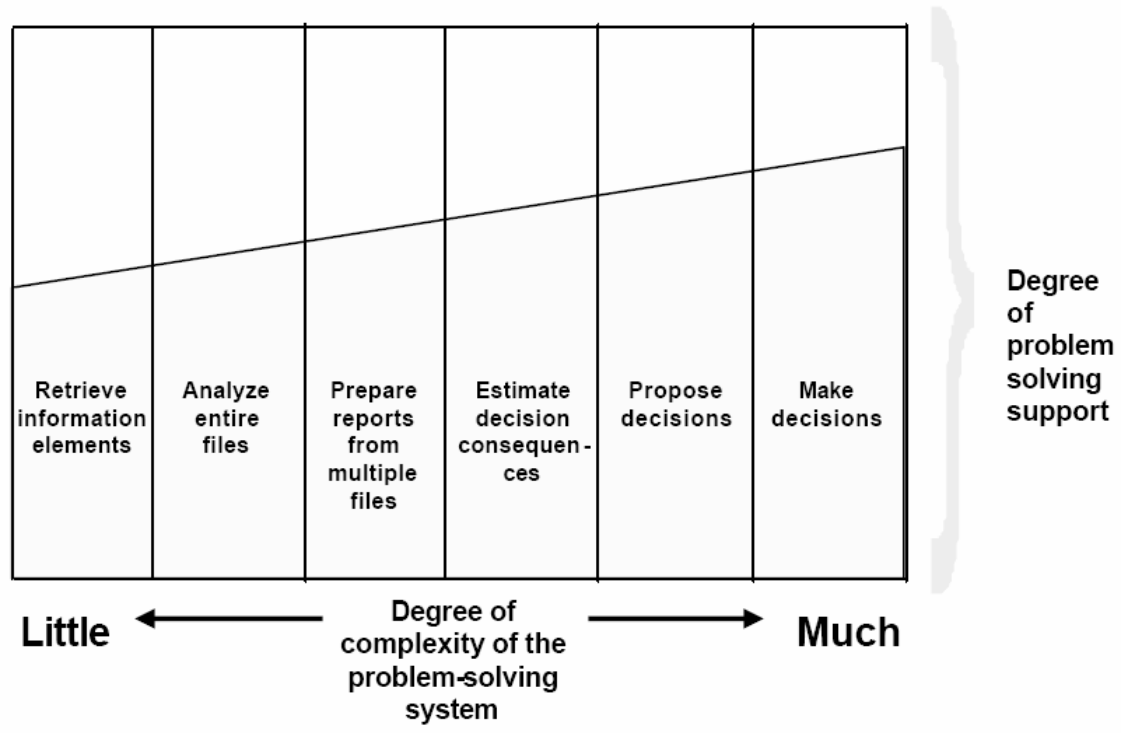


Figure 3.3: Steven Alter's DSS Types[26]

In this matrix, a Decision Support System designed for a shipping company (in the case of this thesis, POLARIS) would be placed on a HIGH degree of complexity as well as a HIGH degree of decision support to the user. This is due to the nature of shipping companies (the explanation of which is not in the scope of this thesis).

Alter believes that Decision Support Systems need to concentrate on solving semi-structured problems, as that is where they provide the best results in relation to the time it takes to develop the system. In addition, Keen states that the characteristics that a Decision Support System needs to have are:

1. Abetting the use of semi-structured and unstructured problems,
2. The support of but NOT the replacement of the decision maker,

[26] *Ibid*

3. Contribution to the effectiveness of the system,
4. A collection of software that provide the decision makers with tools, facts and cases in order to aid them in making a correct decision, and
5. To provide relevant information to the decision makers in order to aid them in making a correct decision.

The University of Stirling (where a large portion of the research in this field has taken place), states that a Decision Support System can also have a supporting database that aids in the processing of information. It also states that a Decision Support System that acts solely on a Strategic Planning level is characterised by "a low level of accuracy, a collection of information, long-term use, non-frequent use, the existence of foreign sources of information, while providing results of high quality." A Figure that best describes what has just been said is shown below:

Characteristics	Operational	Managerial	Strategic
Accuracy	High	⟷	Low
Level of detail	Detailed	⟷	Aggregate
Time horizon	Present	⟷	Future
Use	Frequent	⟷	Infrequent
Source	Internal	⟷	External
Scope	Narrow	⟷	Wide
Nature	Quantitative	⟷	Qualitative
Age	Current	⟷	Current/old

Figure 3.4: Information Characteristics for Different Types of Decisions[27]

Decision Support Systems are placed on a hierarchy based on their complexity during their development. This classification is also in relation with the broader category in which they belong to. As can be seen form Figure 3.5, POLARIS is placed at position 1, as it is a "Suggestion system" with a "High complexity" during its development phase:

[27] *Ibid*

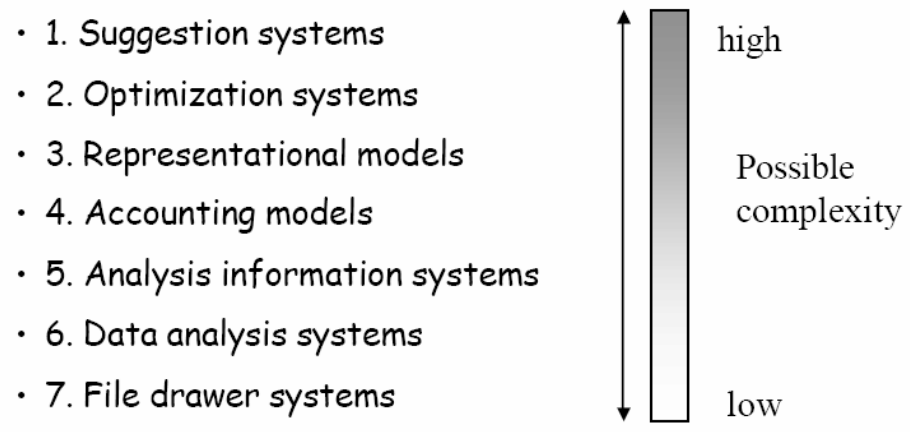


Figure 3.5: The DSS Hierarchy[28]

It would be useful at this point to give a description on what types of Decision Support Systems each of these categories include:

- **File Drawer Systems:** File Drawer Systems are systems that provide information that is used directly in the making of simple decisions. An example would be an ATM machine, where the user views his/her balance and decides whether or not he/she still wants to draw money.
- **Data Analysis Systems:** Data Analysis Systems provide some sort of data processing. An example would be a system that makes ticket reservations for users.
- **Analysis Information Systems:** Analysis Information Systems provide access to multiple databases while combining and analysing data for corporate use. An example would be a performance comparison system using various values as measuring criteria.
- **Accounting Models:** Accounting Models use internal data and are able to create accounting "cases" without needing to take uncertainty into account. An example would be an Invoicing System.
- **Representational Models:** Representational Models use forecasting models when solving problems, and reinforce the Accounting Models by being able to take uncertainty into account. An example would be a future demand forecasting program.
- **Optimisation Systems:** Optimisation Systems evaluate the impact of a decision, using optimisation models and taking uncertainty into account.

[28] *Ibid*

- **Suggestion Systems:** Suggestion Systems are able to describe or suggest a "recipe" to the user, in order to achieve the goal the user is aiming for. These systems can also include other expert systems in their analysis (like a Personal Loan Evaluation System). An example would be POLARIS.

It is important to note that Decision Support Systems that deal with Planning refer to the future (as in decisions for the future) and make use of a number of models, cases and facts. On the other hand, Decision Support Systems that deal with Implementation refer to the present (as in decisions for the present) and make use of information for the implementation of various plans, problem manipulation, error correction and information gathering. Decision Support Systems that deal with Control refer to the past and make use of plan changing (i.e. the changing of past plans), incentives and information gathering for guidance (Poulymenakou, 2000). Knowing that any good policy makes use of Planning, Implementation and Control means that a "decent" Decision Support System needs to make use of all the characteristics that are required for the three phases (Planning, Implementation and Control).

The contribution that a Decision Support System makes to a user according to its type is related to the type of questions it answers:

DSS Provides	Answers to Questions
Raw data and status access	What is...?
General analysis capabilities	What is/why...?
Representation models (financial statements Causal models (forecasting, diagnosis)	What will be...? Why...?
Solutions, suggestions, evaluation	What if...?
Solution selection	What is best/good enough..?

Figure 3.6: Questions that DSSystems Answer[29]

The decision policy that the Decision Support System will implement arises mainly from the first two questions shown in 3.6. It is also definite that the

[29] Source of Figure: Poulymenakou, 2000

questions about the future, questions about the evaluation of the consequences of the decision, as well as questions on evaluations on alternative decision will also need to be answered. The last two questions of Figure 3.6 are the most important ones (*"What if"* and *"What is best/good enough"*), whereas the first three (*"What is"*, *"What is/why"* and *"What will be/Why"*) are secondary.

The most important Decision Support Systems that exist are of two types:

1. Model Driven Systems, and
2. Data Driven Systems.[30]

The Model Driven Systems are self-contained systems that are based on a solid model and defined rules that perform various "what-if" analyses. The Data Driven Systems allow the user to find and make use of information that is retrieved from large databases.

It is important to note that POLARIS is a mix of the two systems, as it has the characteristics of the Model Driven Systems (the issues that a user will be trying to find an answer to are based on rules), as well as characteristics of Data Driven Systems (data on various countries and organizations will be retrieved from large databases, like Clarkson's). How POLARIS is a mix of the two systems will become clearer in the chapters to follow.

[30] Dhar & Stein, 1997

4. DESCRIPTION OF POLARIS

Now that the basic theoretical principles behind POLARIS have been covered, it is time to answer another key question for the thesis:

"What is POLARIS and what does it do?"

This question is not to be confused with "*How* POLARIS works". That is the subject of discussion in Chapter 7. This chapter concentrates on the *what*. Its aim is to explain to the reader what POLARIS will be doing, why it will be doing it and what type of results a user of POLARIS should expect. It will describe POLARIS on a more theoretical point of view, rather than a technical one (as will be done in Chapter 7).

As has already been mentioned a number of times, POLARIS is a Case-Based Reasoning Expert System. Expert System means is that it is designed to perform a task similar to the way an expert would perform it in the field that it has been created for (in this case, the Shipping Industry). Again it should be mentioned that this Expert System is **NOT** supposed to replace the expert; it is only supposed to **AID** the expert. The fact that a human will be making the final decision rather than a computer will take away the obvious problem of the computer not being able to see the whole picture (it only "sees" the data it is

provided) and of the computer not being able to take the human "random" factor into account (the fact that a human being might perform an action that is contrary to logic – i.e. based on his/her "gut feeling"). Case-Based Reasoning means that POLARIS will use past data (cases) of countries and organisations in order to retrieve the results desired by the user.

So what is it that POLARIS does and what is it that makes it useful to the expert using it? A good way to answer this question would be to explain what results POLARIS provides. This way its use in an organisation will become easily identifiable:

A user first selects the country and organisation that he/she wants to perform a "Test"[31] for. The user first selects a country for which he/she wants to set as the organisation's country of operation. He/she then selects either the name of the organisation for which he/she wants to run a "Test" for, or selects the organisation named "Government" in order to run a "Test" related to government policies (e.g. Increase GDP). As a working example for this chapter, it will be assumed that the user has selected "Greece" as the country and "Danaos" as the organisation.

(**Note:** From this point onwards in this chapter, and in order to help the user better understand the concepts, the working example will be used directly. In other words, instead of saying "organisation selected by the user" the author will say "Danaos" and instead of saying "country selected by the user" the author will say "Greece".)

The user then enters certain constraints or goals that he/she wants Danaos to achieve. For example, the user would set a constraint/goal "Increase of Turnover by 10%"[32].

POLARIS then performs the following three steps:

- **Step 1:** It finds all the organisations in the database that achieved this goal (or rather met this constraint). In other words it will find and return all the organisations in the database that achieved an Increase in Turnover of 10%[33].

- **Step 2:** From the organisations that achieved the Increase in Turnover of 10%, it will highlight the ones that are similar in nature to Danaos (There is no point in implementing a strategy that a grocery store implemented to achieve this increase when Danaos is a shipping company).

[31] From this point on in the thesis, the phrase "running a Test will be used to describe the action that a user performs in order to get the results he/she is looking for. This action's step-by-step process will be described in detail in Section 7.3

[32] How this is done is explained in detail in Section 7.3

[33] The working example is continued in order to aid the reader

- **Step 3:** From the organisations that achieved the Increase of 10% AND are similar to Danaos, POLARIS will highlight the ones that operate in a country similar to Greece (One cannot expect a strategy implemented in a Chinese company to have the same results in a Greek company, as, among other issues, the Human Resources differ in nature and culture).

Now that the user knows which organisations achieved the goal that Danaos wants to achieve, and knows which of those are similar to Danaos and operate in a country similar to Greece, the user can implement the strategy used by any of those companies in order to achieve the same result. As those organisations are similar in nature to Danaos and operate in a country similar to Greece, the strategy that they used will have a higher likelihood of success in Danaos, as it has been tried and tested.

Before any of this can be done, though, certain information needs to have been entered in POLARIS. This task is given to the administrators of POLARIS, who will have the access rights to change that data of the database. The tasks that an administrator will have to have performed, before a user can run a "Test", are the following:

He/she will have to have:

- Entered country names in the database (e.g. Greece)
- Entered data for each country in the database (e.g. GDP for each country for a number of years)
- Entered organisation names in the database (e.g. Danaos)
- Entered data for each organisation in the database (e.g. Turnover for each organisation for a number of years)
- Entered Categories (e.g. "Management")
- Entered Subcategories, each of which will belong to a Category (e.g. "Human Resource Management", which belongs to the "Management" Category)
- Entered Issues, each of which will belong to a Subcategory (e.g. "Recruitment", which belongs to the "Human Resource Management" Subcategory)
- Set Rules for each Issue.

The Rules that will have been entered for each Issue are the core of the "Tests". A user wanting to run a "Test" will first select a Category, then a Subcategory of the selected Category and finally an Issue of the selected

Subcategory. Using the Rules set out by an administrator the user will then be able to set his/her constraints/goals in order to run the “Test” and get the desired results.

There is only one more thing left to discuss: the Rules that are set for each Issue. These Rules are not set randomly, nor does an administrator set them as he/she pleases. The administrator has the RIGHT to set the Rules (as in he/she is authorised to alter the data of the database) but the Rules are created by an expert of the field that POLARIS is being used in, namely they Shipping Industry[34].

[34] In the case of this thesis, the Rules are set by Mr. Fykaris as part of his Doctoral thesis, and subsequently used by the author

PART II: DESIGN

5. THE WAMP PHILOSOPHY

In order for POLARIS to be developed as a Web Application four things are needed:

1. An Operating System,
2. A Web Server,
3. A Database Management System, and
4. A Server-side scripting language.

Having the above things in mind, the author chose to develop POLARIS on a Windows Operating System, with an Apache server. The “front-end” of the Web Application was created using the C-based Web scripting language, PHP (Php Hypertext Protocol), and the chosen Database Management System was MySQL. Thus, the philosophy that is being implemented is the WAMP philosophy (Windows, Apache, MySQL and PHP).

It is important to remember that the WAMP is a modification of the LAMP philosophy (Linux, Apache, MySQL and PHP) that is used by a large part of the developer community. This chapter aims to discuss the reasons that WAMP was used.

5.1 The Operating System

As POLARIS will be operated by "average" employees (i.e. employees that might not, necessarily, have expert computer knowledge) it needed to be implemented in an Operating System that will be most familiar with most users. This, according to statistics, is Windows[35].

The other two commonly available operating systems are:

- MacOS (Macintosh Operating System), and
- Linux (more accurately, derivations of Linux, e.g. Ubunty Feisty Fawn)

MacOS is an Operating System developed by Apple (formerly named "Apple Macintosh" – hence the name MacOS) and is designed to work ideally in an Apple PC, which has Apple hardware installed. As Apple PCs are priced higher than the PCs for the common user (e.g. Acer, IBM, etc.) companies do not often acquire them as company PCs. They usually acquire cheaper entry level PCs with equal "business" capabilities.

Linux is an Operating System based on UNIX. Its strongest selling point is that it is Open Source and thus free for any user to do with as he/she pleases (some Linux releases charge a license fee for the use of the Operating System by companies). Unfortunately, even though it is a very powerful Operating System it is not very user friendly, and thus not very widely used by the common user.

Due to Window's user friendliness, along with the fact that most users know and use Windows, the author chose to implement POLARIS on a Windows Operating System, even though it is more expensive than a Linux one.

[35] Statistics South Africa, 2007

5.2 The Server

Unlike a static website, where the user simply views information displayed on a page, in a Web Application the user can interact with the site. In other words, he/she can send and request information from that site. This processing of the users' requests happens on the Web Server that the site is hosted on, and not on the users' browser.

Unlike with the Operating System, the user does not interact directly with the Web Server. The interaction usually happens through a Web browser and the use of a Server-side scripting language, like PHP, ASP (Active Server Pages) and JSP (Java Server Pages). Thus the author was free to choose the web server that best suited the application.

The three Web Servers that are most commonly used are:

- Apache,
- IIS (Internet Information Services), and
- Apache Tomcat

IIS is the Web Server developed by Microsoft. Its first shortcoming is that it is not free as Microsoft charges a license fee for its installation on any PC. In addition, it is tailored to work with a Windows Operating system, which causes compatibility problems, especially when not used with ASP.

Apache Tomcat is the Web Server that was developed by the Apache Software Foundation for use with JSP. It is free, but as the author will be using PHP (for reasons explained later in the chapter) this web server was not chosen.

Apache is the Web Server that was created by the Apache Software Foundation to work with any scripting language and on any platform (Windows included). As it is Open Source (and thus free), and the fact that it takes care of a large number of cross-platform compatibility issues, the author chose to use the Apache Web Server.

5.3 Database Management System

In order to store the data of POLARIS, a Database will need to be created. In order to manage this database (i.e. to run queries on it), a Database Management System will need to be implemented. When choosing a DBMS (Database Management System), one does not look for user friendliness like they do in an Operating System. One looks for a short query response time and a large number of connections to the Database the DBMS can support at any one time.

There are three main DBMSs that are commonly used. These are:

- SQLServer,
- Oracle, and
- MySQL

SQLServer is the DBMS that was developed by Microsoft. Again, a license fee is required for its use on any PC, and it is tailored to work best with Microsoft products.

Oracle is one of the best Database Management Systems that has been developed (if not *the* best). However, Oracle charges a highly priced license fee for its use on a PC, and the "free" release (i.e. the one with which no license fee is required) has very limited capabilities.

MySQL is the DBMS that is favoured by a large number of developers. In addition to its more that acceptable functionality (i.e. it does more than is expected, and then some) it is Open Source and thus charges no license fee, except if the user plans to make a profit from its use. It is important to mention that one of its greater attributes is its excellent cross-platform compatibility. Needless to say, the author chose MySQL as POLARIS' DBMS.

5.4 Scripting Language

When a user interacts with a website or a Web Application, he/she views the page in a web browser (e.g. Internet Explorer, Mozilla Firefox, etc.). All browsers process HTML, however, when the user sends or requests information from the Server, the processing of that information takes place using a server-side scripting language. A Server-side scripting language does exactly what it says; it performs its processes on the "server side", as opposed to the "client side" (which HTML does).

When choosing a server-side scripting language, the developer chooses one that is most compatible with the other software he/she has chosen to use (e.g. Web Server and Operating System). A major factor is also the familiarity the developer has with the scripting language, i.e. the developer will probably choose the language he/she knows best.

The three most common server-side scripting languages are:

- ASP (Active Server Pages),
- JSP (Java Server Pages), and
- PHP (Php Hypertext Protocol)

ASP is the scripting language developed by Microsoft. In addition to not being free, it requires the use of an IIS web server, which the author has not chosen to implement.

JSP is the scripting language that was developed by Sun Microsystems (the company that developed Java). Even though it is free, just like PHP, the author chose to develop POLARIS in PHP as he is more familiar with the language. In addition, PHP is based on the programming language called C, which give it a lot of programming language capabilities, in addition to its scripting language capabilities.

(**Note:** The author will not make use a Client-side scripting language, like JavaScript, for the implementation of POLARIS. Client-side scripting languages process data on the user's PC rather than on the Web Server.)

6. THE DATABASE DESIGN

When using POLARIS, a user enters certain constraints that are compared to data on other organisations and countries, which will in turn give the user the answers he/she is looking for. In order for these comparisons to take place, key data on organisations and countries needs to be stored “somewhere”. This “somewhere” is a Database that will be created in MySQL, for the reasons mentioned in Chapter 5.

Unfortunately, storing data in a database is not as simple as sending commands like “store” or “retrieve”. If the database is not designed properly, not only the data will not make sense, but comparisons with and interpretation of the data will be meaningless, if not pointless. Therefore, a sound and structured Database Design needs to be drawn up.

In accordance with the description of POLARIS that was given in Chapter 4, the following information will need to be stored:

- The Country names,
- Data about each country for a number of years,
- The Organisation names,
- Data about each organisation for a number of years,
- The Categories that POLARIS will be working with,

- The Subcategories of each Category that POLARIS will be working with,
- The Issues that each Subcategory will have,
- The Rules that each Issue will have to adhere by,
- The User Types that will exist in POLARIS (e.g. administrator),
- The Users that will be using POLARIS, and
- A relationship between each User and User Type (i.e. what user type each user is)

Having the above in mind, the Database that was designed had the form shown of the Database Table Relational Diagram in Figure 6.1. In order to understand the data that is stored in each table, as well as the relationship among the tables, a detailed analysis of each "part" of the Diagram will follow.

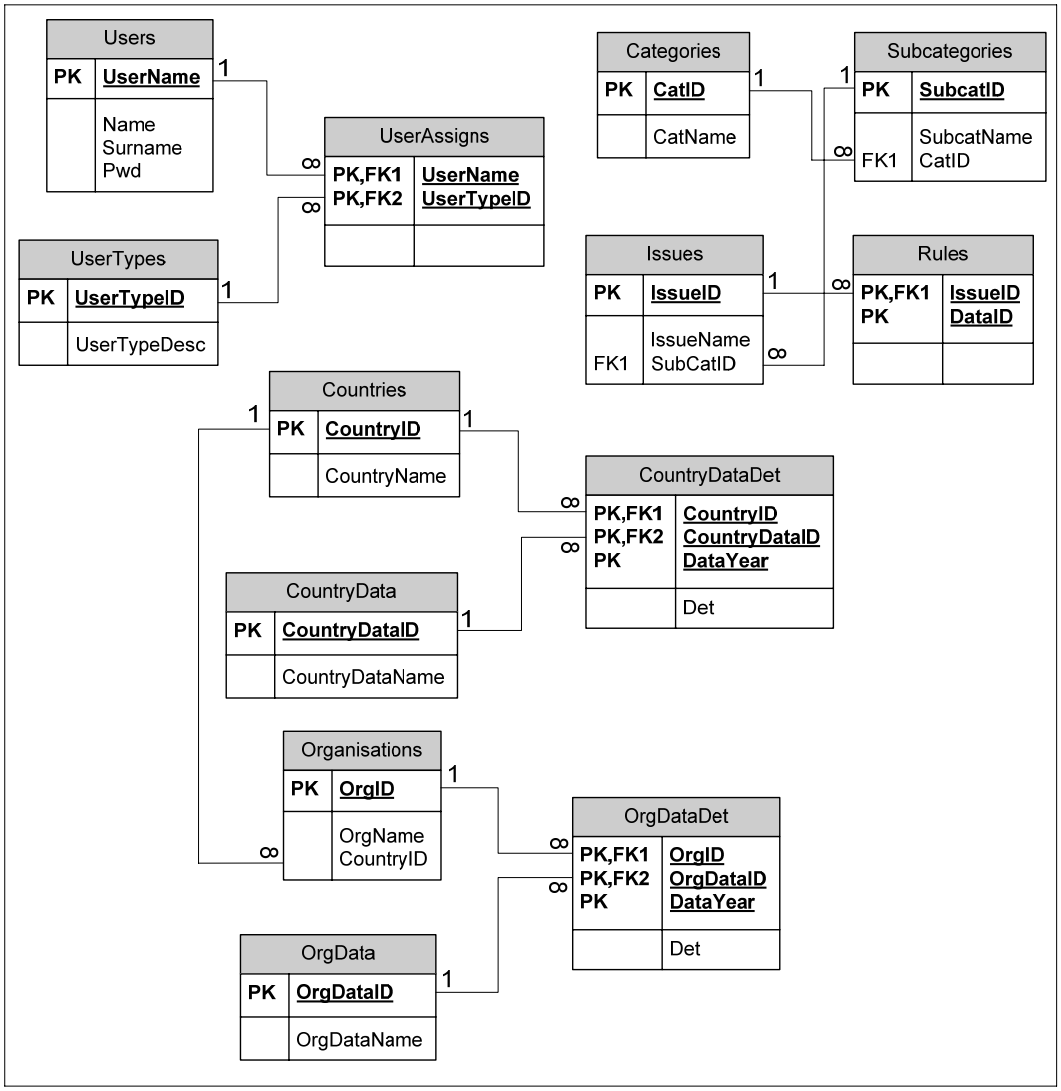


Figure 6.1: Database Table Relational Diagram

6.1 Users Part

In POLARIS, various users will be able to log in. That means that information about each user will need to be stored in the Database. This is done in the Users part. The DBTRD (Database Table Relational Diagram) for the Users part is shown in Figure 6.2 below:

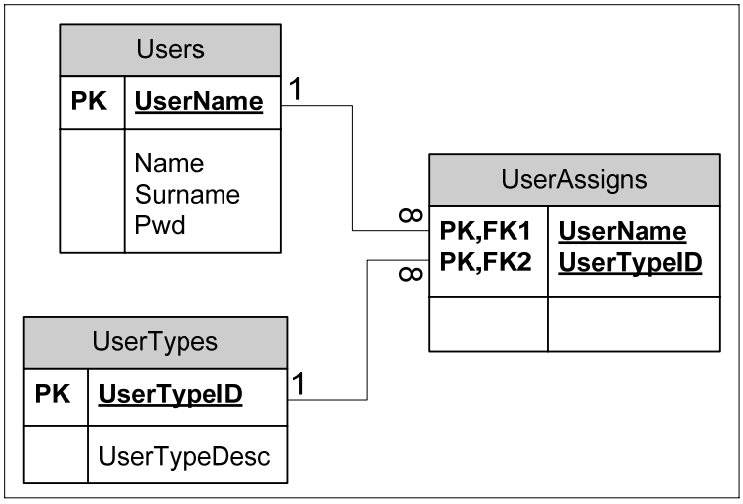


Figure 6.2: DBTRD for the Users part

The schemas for the tables are as follows:

- Users (UserName, Name, Surname, Pwd)
- UserTypes (UserTypeID, UserTypeDesc)
- UserAssigns (UserName, UserTypeID)

The "Users" table stores information about each user that is able to access POLARIS. The username of the user is stored (as the primary key of the table), along with the name, surname and password that the user uses to login to POLARIS. The password is encrypted using the SHA1[36] encryption method so that a third party cannot find out what a users password is, by simply looking in the database.

The "UserTypes" table stores the user types that are available in POLARIS. For example, it stores user types, like "administrator", which when linked to a username (e.g. admin) imply that the user ("admin") is of type "administrator".

[36] SHA1 is an encryption method that encrypts a word but cannot decrypt it. The login comparison takes place by encrypting the entered password and checking if the encrypted entered password is the same as the encrypted one stored in the database (Welling and Thomson, 2005).

The UserTypeID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table.

The final table in the Users part of the Database is the “UserAssigns” table. This is the table where the relationship between the username of each user is associated with a user type. Its two fields are UserName and UserTypeID, which are its primary keys. They are also foreign keys, as their names are the primary keys of the “Users” and “UserTypes” tables respectively.

The connection lines show that for every one UserName and for every one UserType there are many UserAssigns. This makes sense, as a user is able to be many user types (for example, username “bob” can be both “viewer” and “surveyor”[37]).

[37] The user types “viewer” and “surveyor” do not exist in POLARIS. There are simply used for the example.

6.2 Countries Part

In order for POLARIS to check whether a country is similar to another country, or for it to run comparisons with the data provided by a user, information about each country will need to be stored in the Database. This is done in the Countries part. The DBTRD for the Countries part is shown in Figure 6.3 below:

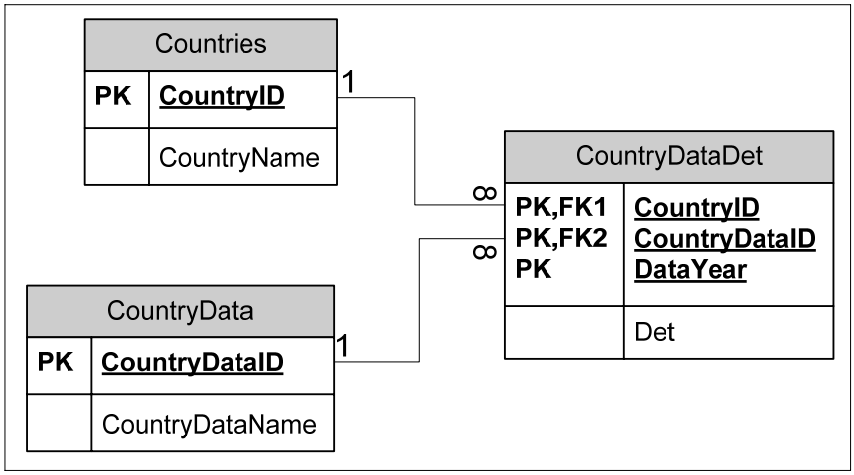


Figure 6.3: DBTRD for the Countries part

The schemas for the tables are as follows:

- Countries (CountryID, CountryName)
- CountryData (CountryDataID, CountryDataName)
- CountryDataDet (CountryID, CountryDataID, DataYear, Det)

The “Countries” table stores the names of the countries that will be used in POLARIS. The CountryID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table.

The “CountryData” table stores the names of the data that will be required for each country (e.g. “GDP”, etc.). The CountryDataID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table.

The “CountryDataDet” is the table that stores the information for each country’s country data for a single year. For example, it stores “5” as the percentage for Country Data “GDP” for the Country “Greece”, for the year “2007”. The field names CountryID, CountryDataID and DataYear are the primary keys for this table, as all three are needed to store and reference the information, i.e.

the field “Det”. CountryID and CountryDataID are also foreign keys, as their names are the primary keys of the “Countries” and “CountryData” tables respectively.

The connection lines show that for every one Country and for every one Country Data Name there are many Details. For example, there can be a country and a country data (e.g. “Greece” and “GDP”) that can have information stored for a number of years (e.g. 2005, 2006, 2007, etc.).

6.3 Organisations Part

In order for POLARIS to check whether an organisation is similar to another organisation, or for it to run comparisons with the data provided by a user, information about each organisation will need to be stored in the Database. This is done in the Organisations part. The DBTRD for the Organisations part is shown in Figure 6.4 below:

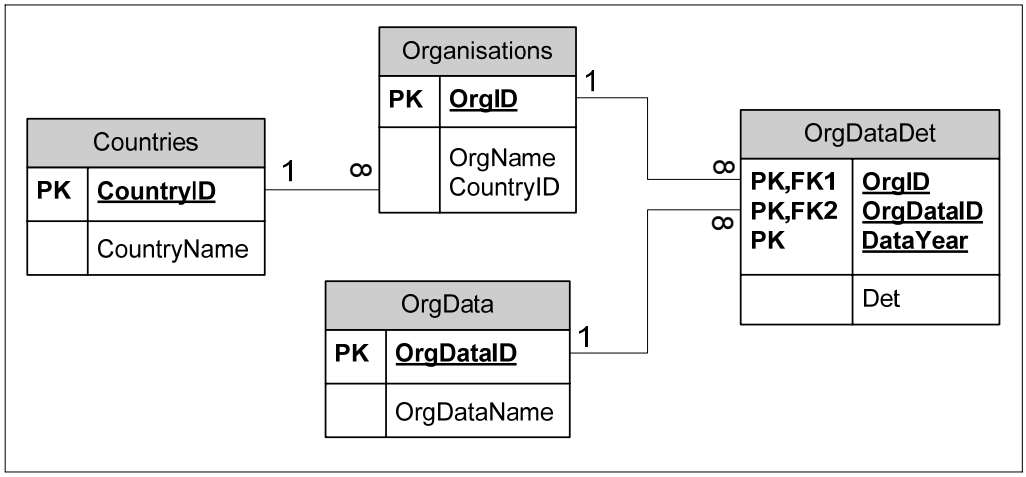


Figure 6.4: DBTRD for the Organisation part

The schemas for the tables are as follows:

- Organisations (OrgID, OrgName, CountryID)
- OrgData (OrgDataID, OrgDataName)
- OrgDataDet (OrgID, OrgDataID, DataYear, Det)

The “Organisations” table stores the names of the organisations that will be used in POLARIS. The OrgID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table. The CountryID field is the foreign key that links an organisation to a country (e.g. organisation “Danaos” is located in the country “Greece”). The “Country – Organisation” connection line shows that there can be many organizations in one country.

The “OrgData” table stores the names of the data that will be required for each organisation (e.g. “Turnover”, etc.). The OrgDataID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table.

The “OrgDataDet” is the table that stores the information for each organisation’s organisation data for a single year. For example, it stores “500 000” as the value for Organisation Data “Turnover” for the Organisation “Danoas”, for the year “2007”. The field names OrgID, OrgDataID and DataYear are the primary keys for this table, as all three are needed to store and reference the information, i.e. the field “Det”. OrgID and OrgDataID are also foreign keys, as their names are the primary keys of the “Organisations” and “OrgData” tables respectively.

The connection lines “Organisations – OrgDataDet” and “OrgData – OrgDataDet” show that for every one Organisation and for every one Organisation Data Name there are many Details. For example, there can be an organisation and an organisation data (e.g. “Danaos” and “Turnover”) that can have information stored for a number of years (e.g. 2005, 2006, 2007, etc.).

6.4 Rules Part

As was described in Chapter 4, POLARIS will have a number of Categories. Each Category will have a number of Subcategories and each Subcategory will have a number of Issues. These Issues will include Rules. These Rules are a combination of CountryData and/or OrganisationData. All this data will also be stored in the Database, more specifically, in the Rules part. The DBTRD for the Rules part is shown in Figure 6.5 below:

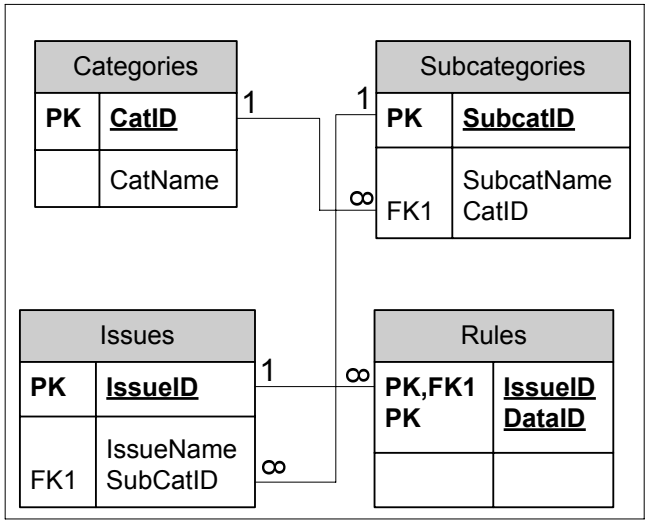


Figure 6.5: DBTRD for the Rules part

The schemas for the tables are as follows:

- Categories (CatID, CatName)
- Subcategories (SubcatID, SubcatName, CatID)
- Issues (IssueID, IssueName, SubCatID)
- Rules (IssueID, DatID)

The “Categories” table stores the names of the Categories that will be used in POLARIS (e.g. “Management”, etc.). The field CatID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table.

The “Subcategories” table stores the names of the subcategories that will be used in POLARIS. The SubcatID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table. The CatID field is the foreign key that links a subcategory to a category (e.g. subcategory “Human Resource Management” belongs to the

category “Management”). The “Category – Subcategory” connection line shows that there can be many subcategories in one category.

The “Issues” table stores the names of the issues that will be used in POLARIS. The IssueID is the primary key of the table, and is used to ensure that the primary key is unique, and that it can be referenced as a foreign key from another table. The SubCatID field is the foreign key that links an issue to a subcategory (e.g. issue “Recruitment” belongs to the subcategory “Human Resource Management”). The “Subcategory – Issue” connection line shows that there can be many issues in one subcategory.

The Rules table stores the rules that apply to each Issue. The IssueID and DataID are the primary keys for the table. The field IssueID is also a foreign key, as it is the primary key of the Issues table. Each issue has a number of Country Data and/or Organisation Data that make up its rules. The IDs of that data are stored in the DataID field. However, the DataID field is NOT a foreign key, as it does not use the primary key of the Country and Organisation Data tables. When an Organisation Data (e.g. with an ID of “3”) is to be stored as part of a rule, it is stored in the following form in the DataID field: “o-3” (“o” for “organization”). The same applies for the Country Data, only the letter “c” (for “country”) is used (e.g. “c-3”). This way, the application will know if the ID that is stored refers to the Organisation Data or the Country Data.

6.5 Overall Remarks

All in all, the Database was designed in such a way so that the storing and retrieving of information will be as efficient as possible. “Efficient”, though, is a very broad term. One needs to explain what is meant by the term “efficient”.

POLARIS is designed to have a fast response time. This means that when a user makes a query or wants to run a “Test” he/she will not be waiting for an unreasonable amount of time. This system has been designed to perform the administrative tasks in a few seconds. The running of the “Tests” has been designed to take as little time as possible, however, it is important to note that the larger the database, the longer the time it will take to run a “Test”.

POLARIS’ database is designed to be logical. This means that the data is stored in a logical way (not just placing random data in random tables), as has been explained in the previous sections of Chapter 6. This was done in a fashion that contributes to POLARIS’ fast response time, in other words the database was designed to have a fast response time.

Finally, POLARIS’ database is reliable. The use of a good Database Management System (DBMS), namely MySQL, and a good database design contribute to the reliability of POLARIS. For example, data will not be lost if a power failure occurs, and data will not be confused when more than one user accesses the database.

Now the term “efficient” has a meaning and one can safely say that POLARIS is an efficient system.

7. THE POLARIS "FLOW"

Up to this point, the only things that have been discussed about POLARIS are:

- What it is and what it does (Chapter 4),
- What data it stores and how one piece of data relates to another (Chapter 6)

What is missing is:

"How POLARIS works"

Answering the above question is the focus of this chapter. It is not to be mistaken for a manual, as this chapter does not explain what each page of the Web Application does, but rather explains the "flow" of POLARIS. In other words, it will focus on giving explanations on why the specific design was chosen and in what order the tasks of the application should be completed in, so that the user will get the desired results.

The design of POLARIS was divided into three main modules, which will be used as a basis for the explanations:

- The User module,
- The Admin module, and
- The “Test” module.

This design was chosen as it logically separates the application into three manageable parts. One that deals with the users’ personal information (User module), one that deals with the administration of POLARIS (Admin module) and one that deals with the running of the “Test” the user wishes to perform. Each of these modules will be discussed in detail in the subsections that follow.

(**Note:** The rules that will be used in the examples in this chapter, and that will be shown in the screenshots that follow, are not necessarily correct. The same applies for the values that will be used for the country and organisation data. Their use is merely exemplary, and serves to better explain the subjects under discussion. It is up to the final users of the system to populate the database correctly.)

7.1 The User Module

What differentiates this module from the Admin and "Test" modules is the fact that this contains all the functionality for the editing of the users' personal information. Through this module the user is able to change his/her personal information and change the password that he/she uses to log into POLARIS. More importantly, this is the module from which the user selects the country and organisation that he/she wants to operate as, for the "Testing" ("Testing" being the comparison process that the user performs in order to find the organisations that achieved the same goals as the ones the user is testing for).

7.1.1 The "My Account" page

When the user logs into POLARIS, he/she sees the "My Account" page:

Figure 7.1: "My Account" page - startup

First of all, from this page the user can see what user type(s) he/she is. Better phrased, the user will be able to see what user rights he/she has (i.e. what he/she can do in the POLARIS application):

Figure 7.2: "My Account" page – User Types

The second (and probably more important) thing that the user can do in this page is that he/she can select the Country and Organisation that will be used during “Testing”:

Figure 7.3: “My Account” page – Country / Organisation selection

7.1.2 The “Change My Password” page

The second page that the user can go into that belongs to this module, is the “Change My Password” page:

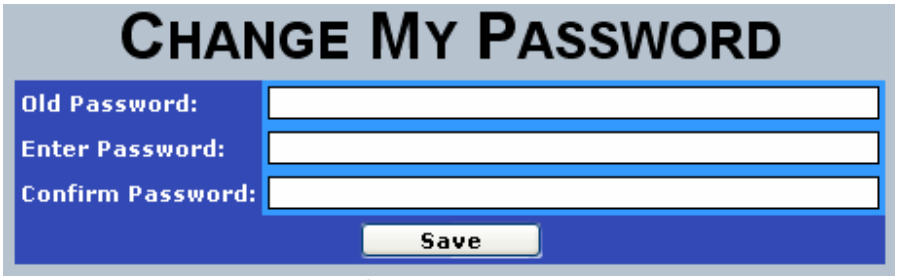


Figure 7.4: “Change My Password” page

Through this page the user can change the password that he/she uses to log in to POLARIS.

7.1.3 The "Edit My Details" page

The third and final page that the user can go into that belongs to this module, is the "Edit My Details" page:

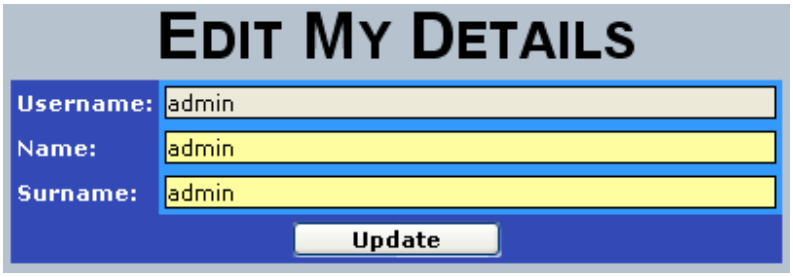


Figure 7.5: "Edit My Details" page

In this page the user can edit his/her personal details (but not the username) by simply clicking the "Update" button after changing the other fields in the form. This page might seem meaningless, as a user will not often change his/her name and surname, but it was created so that future modifications of POLARIS can include entries like a home telephone number or a cell-phone number, that might change more often.

Finally, it is important to note that all users will be able to access the pages of this module, at any point during their use of POLARIS. However, it is imperative to select a Country and Organisation *before* attempting to perform a "Test", as the results will be meaningless (to prohibit the user from running the "Test" without the aforementioned selection, the application performs a check – when the user enters the "Test" page – that sends the user back to the "My Account" page to make his/her selection).

7.2 The Admin Module

The Admin module is the module that contains all the functionality for the background work of POLARIS. In other words, it is responsible for the adding of Categories, Subcategories, Issues, Countries and Organisations, as well as the adding and editing of the Issue Rules, Country Data and Organisation Data. Finally, through this module, an administrator user can add and edit Users of the POLARIS application. It is important to note that the only users that have access to the pages of this module are the ones with administrator rights, i.e. the ones that have the “administrator” user type.

7.2.1 The “Add User” page

Before anyone can make use of POLARIS, the default administrator needs to add some users to the system. Without them, POLARIS cannot function, as there will be no one to populate the database and no one to run to the “Tests”.

When an administrator wants to add a new User to POLARIS he/she goes to the “Add User” page:

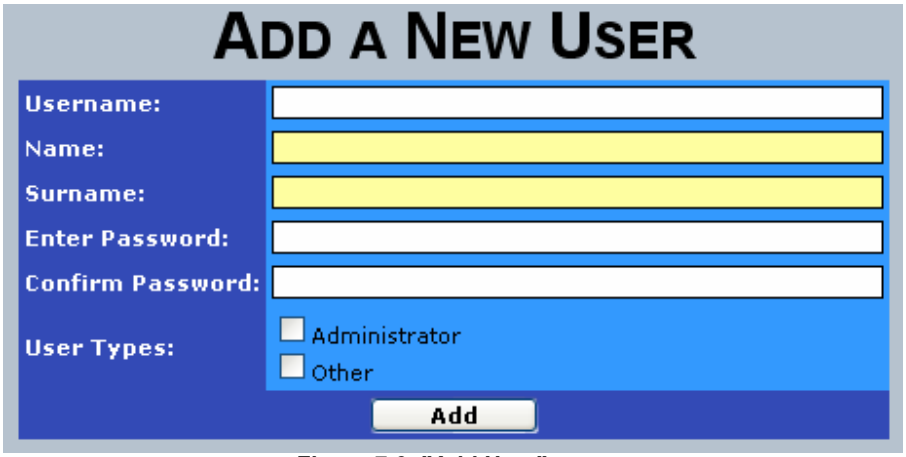


Figure 7.6: "Add User" page

By clicking the “Add” button after completing the fields and checking the appropriate user types, the new user (with the unique username) is added to POLARIS.

7.2.2 The "Edit User" page

Similarly to when a user wants to edit his/her own personal details, the administrator can perform the task on behalf of the user. When the administrator goes to the "Edit User" page, he/she sees what is shown in Figure 7.7:

Figure 7.7: "Edit User" page

By selecting to edit a User, the administrator can edit the selected user's details. However, there is one more piece of information he/she can edit about the user that the user cannot change him/herself: the user type. For security purposes, an administrator is the only user that can change the user type of another user (compare Figure 7.8 to Figure 7.5):

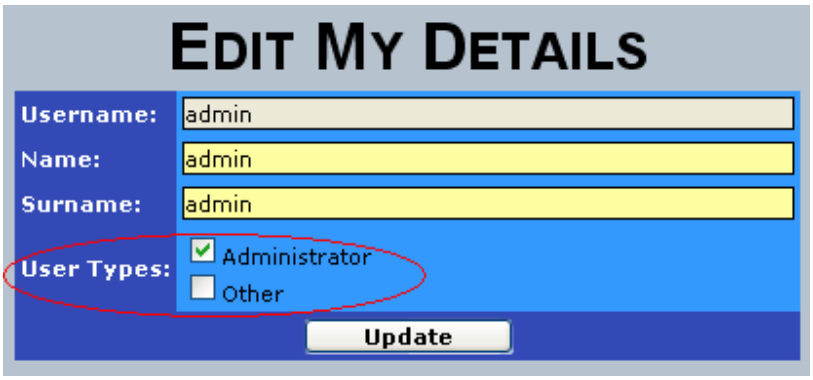


Figure 7.8: User Edit by Administrator

7.2.3 The "Add Category" page

In order for a user to run a "Test", he/she will need to select a Category. In order for the user to do that, an administrator will need to have populated the database with the appropriate categories. Without them, no "Tests" can be run.

When an administrator wants to add a new Category to POLARIS, he/she goes to the "Add Category" page and adds Categories by entering their names and clicking on the "Add" button:

Figure 7.9: "Add Category" page

7.2.4 The "Add Subcategory" page

In order for a user to run a "Test", he/she will need to select a Subcategory (after selecting a Category). In order for the user to do that, an administrator will need to have populated the database with the appropriate subcategories. Without them, no "Tests" can be run. Each Category has its own, relevant Subcategories, and, therefore, each Subcategory belongs to one Category.

When an administrator wants to add a new Subcategory to POLARIS, he/she goes to the "Add Subcategory" page and adds Subcategories by entering their names, selecting the Category they belong to, and clicking on the "Add" button:

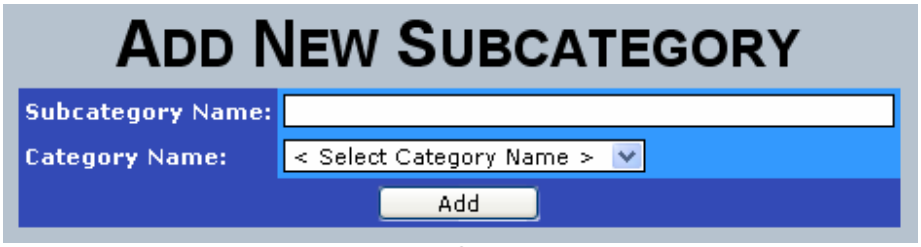


Figure 7.10: "Add Subcategory" page

7.2.5 The "Add Issue" page

In order for a user to run a "Test", he/she will need to select an Issue to run the "Test" on (after selecting a Subcategory). In order for the user to do that, an administrator will need to have populated the database with the appropriate issues. Without them, no "Tests" can be run. Each Subcategory has its own, relevant Issues, and, therefore, each Issue belongs to one Subcategory.

When an administrator wants to add a new Issue to POLARIS, he/she goes to the "Add Issue" page and adds Issues by entering their names, selecting the Subcategory they belong to, and clicking on the "Add" button:

Figure 7.11: "Add Issue" page

7.2.6 The "Add Countries" page

When a user wants to run "Tests" that pertain to government issues, or when POLARIS needs to compare countries to one another, the database needs to have been populated with Countries and their Data.

To add new Countries to the Database, an administrator goes to the "Add Country" page, and adds Countries by entering their names and clicking the "Add" button:

Figure 7.12: "Add Country" page

7.2.7 The "Add Country Data" page

The Country comparisons that POLARIS performs take place by comparing one country's data with the country data of every other country in the database. In order for this to happen, the database needs to have been populated with the data names (e.g. "GDP") that each country will have data for.

When an administrator wants to add a new Country Data Name, he/she goes to the "Add Country Data" page, and adds a Country Data Name by entering its name and clicking on the "Add" button:

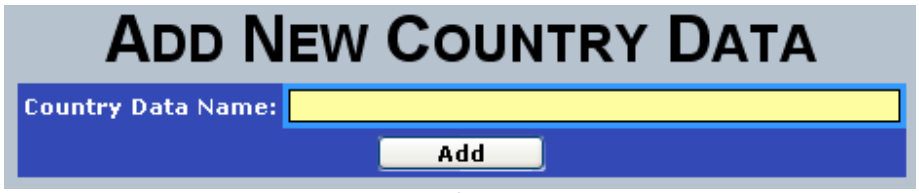


Figure 7.13: "Add Country Data" Page

7.2.8 The "Edit Country Data" page

For the comparisons to take place, data needs to have been provided for each data name, for each country. This takes place in the "Edit Country Data" page:

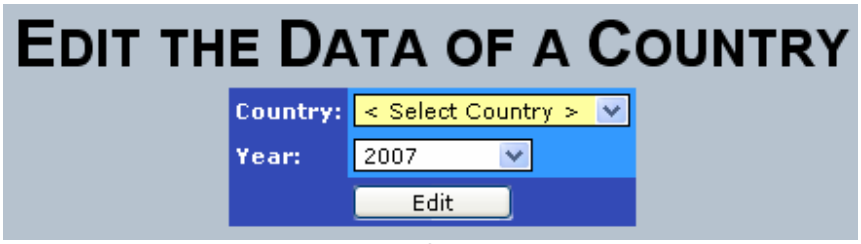


Figure 7.14: "Edit Country Data" page

By selecting a country and a year and clicking on the "Edit" button, the user sees what is shown in Figure 7.15 and is then able to edit the data of the Country for the selected year, by simply completing/editing the fields and clicking the "Save" button:

Figure 7.15: Editing Data for "Greece" for the year "2007"

7.2.9 The "Add Organisations" page

When a user wants to run "Tests" that pertain to organisation issues, or when POLARIS needs to compare organisations to one another, the database needs to have been populated with Organisations and their Data.

To add new Organisations to the Database, an administrator goes to the "Add Organisation" page, and adds Organisations by entering their names and clicking the "Add" button:

Figure 7.16: "Add Organisation" page

7.2.10 The "Add Organisation Data" page

The Organisation comparisons that POLARIS performs take place by comparing one organisation's data with the organisation data of every other organisation in the database. In order for this to happen, the database needs to have been populated with the data names (e.g. "Turnover") that each organisation will have data for.

When an administrator wants to add a new Organisation Data Name, he/she goes to the "Add Org. Data" page, and adds an Organisation Data Name by entering its name and clicking on the "Add" button:

Figure 7.17: "Add Org. Data" page

7.2.11 The "Edit Country Data" page

For the comparisons to take place, data needs to have been provided for each data name, for each organisation. This takes place in the "Edit Org. Data" page:

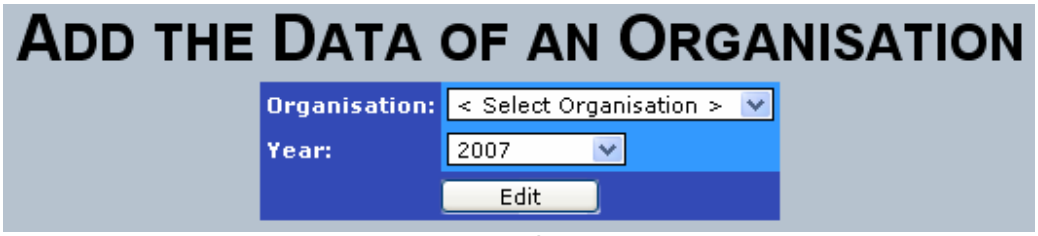


Figure 7.18: "Edit Org. Data" page

By selecting an organization and a year and clicking on the "Edit" button, the user sees what is shown in Figure 7.19 and is then able to edit the data of the Organisation for the selected year by simply completing/editing the fields and clicking the "Save" button:

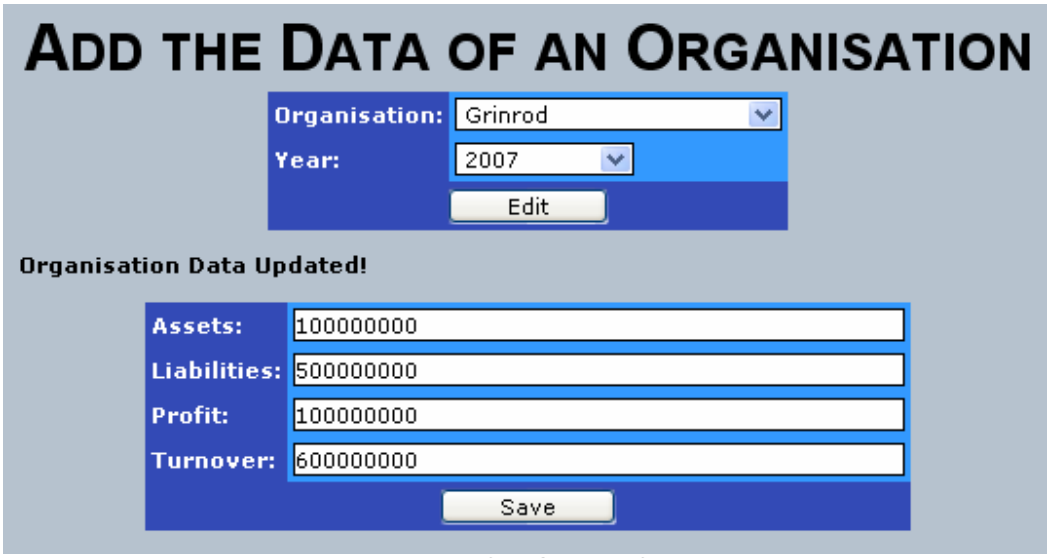


Figure 7.19: Editing Data for "Grinrod" for the year "2007"

7.2.12 The "Set Rules" page

For a user to be able to run a "Test", rules need to be set (or assigned) for each Issue. Without the rules, the user will not be able to enter the "constraints" he/she wishes to apply to the test (this will become clear in Section 7.3 that follows).

In order to apply the rules, the administrator needs to go to the "Set Rules" page:

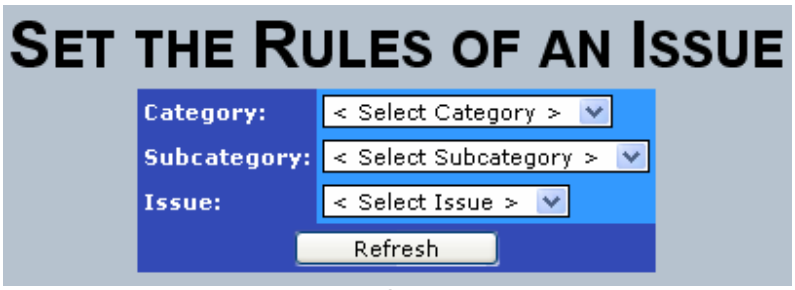


Figure 7.20: "Set Rules" page

The user selects a Category, Subcategory and Issue, and after clicking on the "Refresh" button for the third (and last) time Figure 7.21 is shown, where the user can select the Country and/or Organisation Data that make up the Rule:

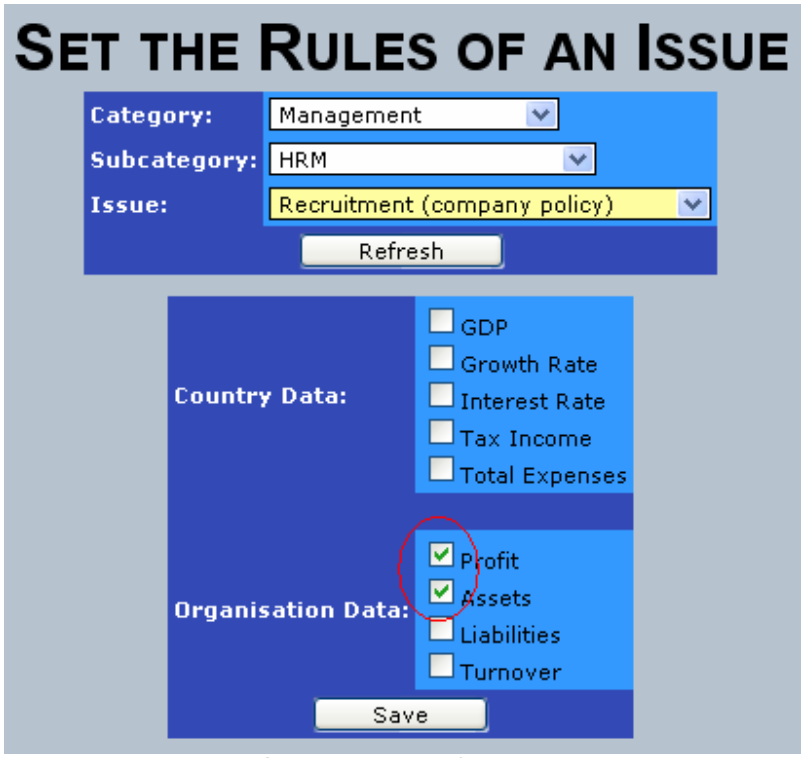


Figure 7.21: "Setting" the Rule for the selected Issue

(**Note:** At first, only the Categories combo box is populated, and the Subcategories and Issues combo boxes are empty. After every selection the user

needs to click on the “Refresh” button in order to populate the next combo box. For example, the user selects a category, and then clicks on the “refresh” button (NOT the browser’s refresh) in order to populate the subcategories combo box with the selected category’s subcategories. The same applies for the Issues combo box. This is done as the developer is only using PHP, which is a server-side scripting language (as explained in section 5.4). As no client-side scripting language is being used (e.g. JavaScript), the information needs to be sent to the Web Server in order to process the request of populating the next combo box.)

7.3 The "Test" Module

The "Test" module is the module where a user goes to run a "Test". Before the user can enter this page, he/she will have to have selected a Country and an Organisation that he/she will be performing the "Test" for (see section 7.1.1). In addition, the database will have to have been populated with the relevant data (i.e. Categories, Subcategories, Issues, Countries, Country Data, Organisations, Organisation Data and Issue Rules), as the "Test" will otherwise return meaningless results.

When the user goes to the "Test" page, he sees what is shown in Figure 7.22:

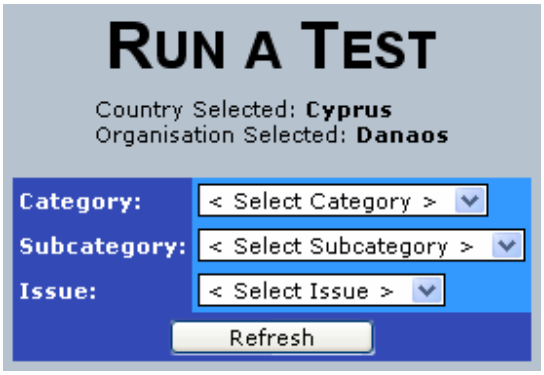


Figure 7.22: "Test" page

Here the user can see the Country and Organisation that he/she has chosen to perform the test for, and can select the Category, Subcategory and Issue that he/she wants to "Test". Once the user clicks the "Refresh" button for the third (and last) time[38], what is seen in Figure 7.23 is shown:

[38] See note at the end of Section 7.2

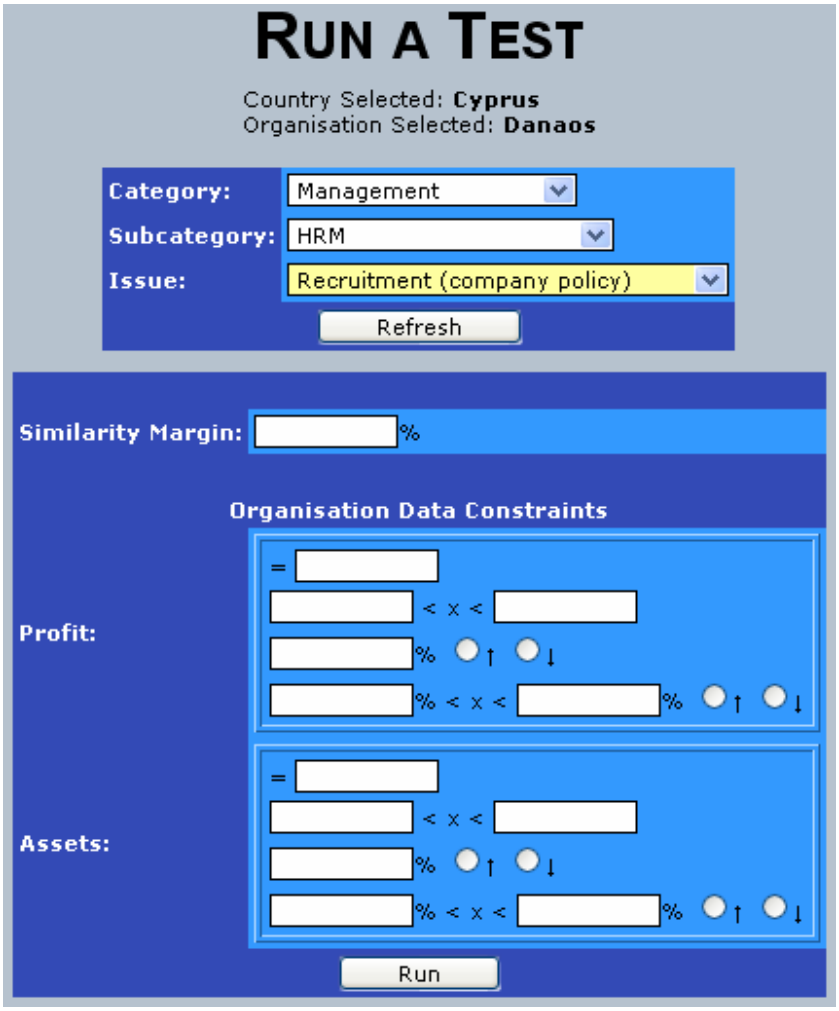


Figure 7.23: Issue's Rules (constraints)

These are Country and/or Organisation Data (in Figure 7.23 only Organisation Data was chosen as the "rules" for the selected Issue), with various boxes that perform different functions[39].

As was explained in Chapter 4, the result set of the "Test" will be names of organisations from the Database that met the constraints set by the user (this relates to the other boxes, which will be explained shortly). From that result set of organisations that met the criteria, a new result set of organisation names will be retrieved, with organisations that met the constraints AND are similar to the organisation that the "Test" is being performed for (in this example, similar to "Danaos"). After that, from the result set with the organisations that meet the constraints and are similar to the selected organisation, a third result set will be retrieved, with the organisations that are located in a country similar to the selected country (in this example, "Cyprus"). Figure 7.24 shows an example of the result sets:

[39] The example continues from the selected rules of the example in Figure 26

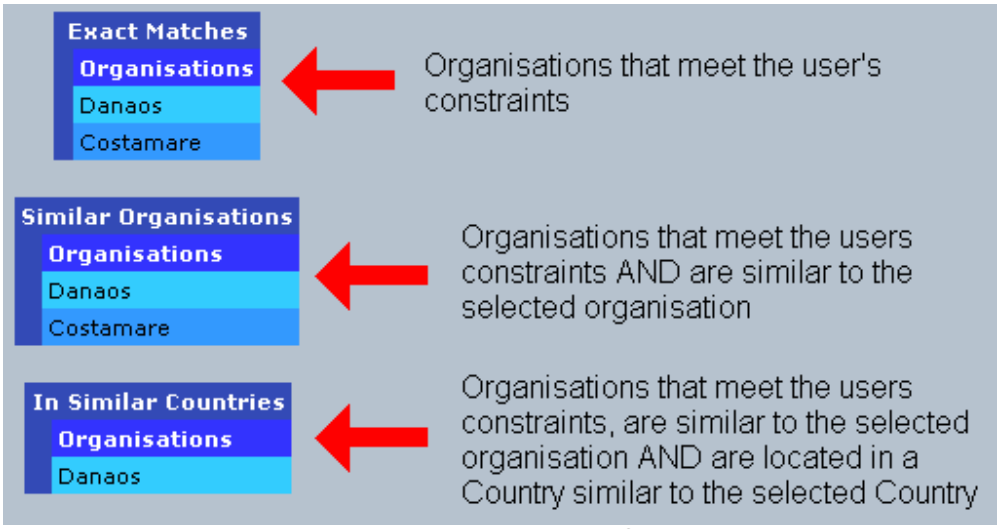


Figure 7.24: "Test" Result Sets

When comparing one organisation to another, or when comparing one country to another, POLARIS compares the selected country's / organisation's Data to every other country's / organisation's Data. Unfortunately, the data will almost never be the same from country to country or organisation to organisation. For that reason, a "Similarity Margin" is needed. For example, if a margin of 5% is entered by the user, a match will occur if the comparing country's / organisation's data is within 5% of the selected country's / organisation's data (for example, if one piece of organisation data is 100, then a match will be found if the comparing organisation's equivalent piece of data is between 95 and 100).

As far as what each of the "many" boxes mean for a piece of Data, explanations are given bellow (the "Profit" organisation Data will be used as an example and Figure 7.25 will be used as a guide):

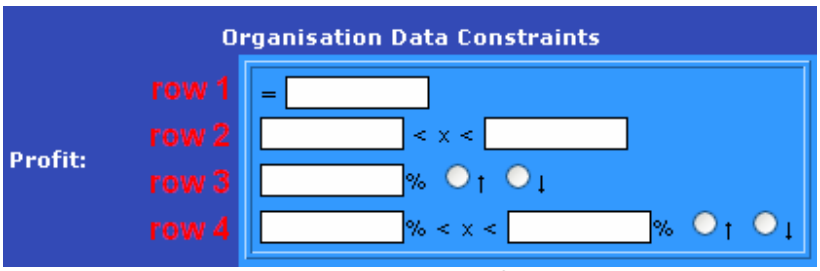


Figure 7.25: "Piece" of Data

POLARIS does not allow the user to enter a constraint in more than one row in any one piece of data (e.g. he/she cannot enter values in both Row 1 and Row 2). In addition, it performs checks that each entry is numeric. As for what each row is for:

- **Row 1:** The user enters a value if he/she wants an EXACT match (for example, all organizations where “Profit” equals exactly “1 000”)

- **Row 2:** The user enters values for which he wants the match to be between (for example, find all organisations where the “Profit” is “between 500 and 1 500”)

- **Row 3:** The user enters a percentage and selects if he/she wants it to be an increase or a decrease (for example, the user enters “5” and select the “up arrow” in order to find all organisations which had “a Profit increase of 5%)

- **Row 4:** The user enters a percentage range he wants the constraint to meet and selects an increase or decrease (for example, he enters a “5” and a “10”, and selects the up arrow to find all organisations with a “Profit increase of 5% to 10%”)

The “Testing” described above applies when the user wishes to perform a “Test” that relates to Issues regarding organisations. If the user selects “Government” as an organisation, it relates to Issues relating for the Government. Everything already mentioned in “The Test Module” still applies, except for the Result Sets, which look like the example in Figure 7.26, below:

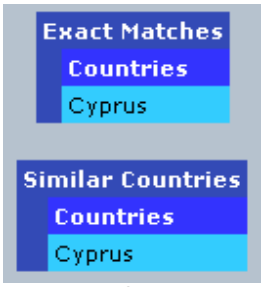


Figure 7.26: The "Country" Result Set

When it comes to “Government” related Issues the first result set contains all the Countries that meet the user’s constraints. From the result set of the Countries that meet the user’s constraints, a second set is retrieved, containing all the Countries that meet the user’s constraints AND are similar to the country selected by the user.

PART III: FINAL REMARKS

8. FUTURE IMPLEMENTATIONS

As was explained in Chapter 4, POLARIS was design to be a Cased-based Reasoning Expert System. The original idea was that a user would enter his/her constraints and POLARIS would return a viable strategy to achieve the desired goals.

Unfortunately, the time available for the development of POLARIS and writing of this thesis was fairly limited, and, therefore, the functions to be developed needed to be prioritised. This does not, in any way, mean that POLARIS is not useful. All it means is that there is great room for improvement in the future.

In Chapter 7, the thesis covered the functions that POLARIS can perform at present. These are:

- Adding Users,
- Adding Countries,
- Adding Country Data,
- Adding Organisations,
- Adding Organisation Data,

- Adding Categories,
- Adding Subcategories,
- Adding Issues,
- Setting Issue Rules, and
- Running a “Test”.

The “Test” function that has been developed does not return the exact result that the user might be looking for, but it does return a result that can help him/her to *find* the answer he/she is looking for. More precisely, the user prefers an exact solution to his/her problem (or to achieve the desired goal), but this version of POLARIS only provides the user with the Countries/Organisations that achieved the desired goal (or rather met the user’s constraints).

The future implementations (future versions) of POLARIS will include functions that make the process more automated and return results that will provide the user with better answers to his/her problems. In addition, they will make POLARIS more flexible.

The functions that can still be implemented include, among others, the following:

- Automated database population,
- Similarity Margins for each Country/Organisation Data,
- Situation Similarity,
- Strategic Solutions for each Issue, and
- Graded Result Sets.

Again, this list is by no means exhaustive, but if the above functions are implemented, it will make POLARIS a better Cased-based Reasoning Expert System, and will give its users better results to their problems.

8.1 Automated Database Population

As was shown in Section 7.2.8 and Section 7.2.11, when an administrator needs to edit the Data of a Country or an Organisation, he/she needs to do it *manually*, i.e. he/she will need to find the piece of data and change it, as shown again in Figures 8.1 and 8.2:

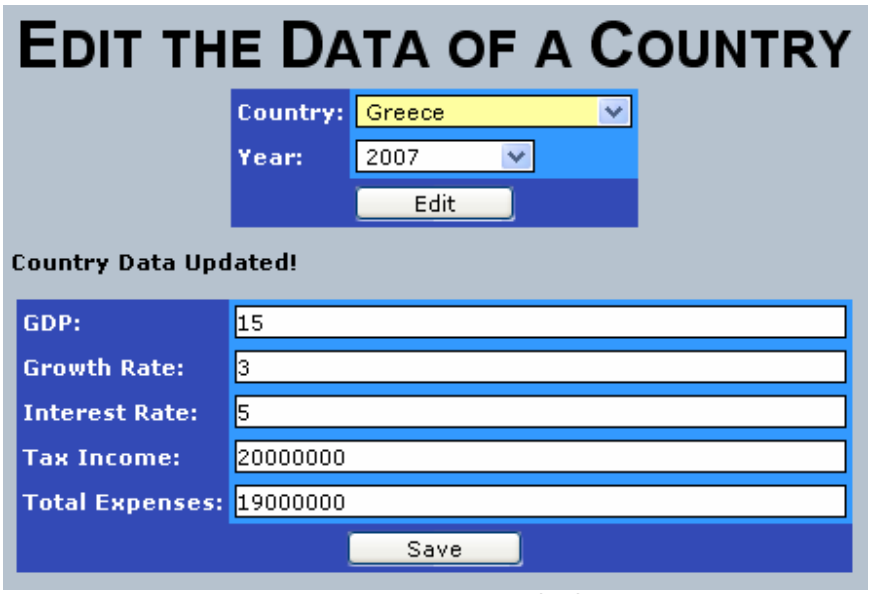


Figure 8.1: Editing the Data of a Country

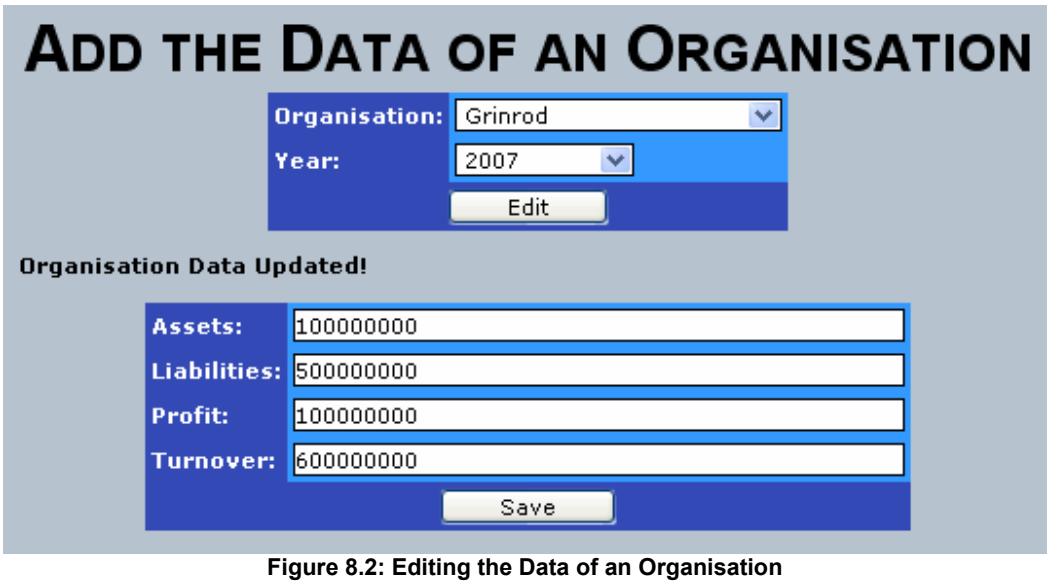


Figure 8.2: Editing the Data of an Organisation

This process becomes even more of a problem when data for a new year needs to be entered. When, for example, the year passes into 2008 the user will have to go to EVERY Country AND Organisation in the database and edit their Data. This is better explained with an example:

If there are 200 countries and organisations in the database, each with 25 required pieces of Data, then the total number of entries, that the user will have to make, equals 5 000. If each entry takes 2 minutes (checking the data from the source, and then typing it in the correct place) then the total amount of time to enter all 5 000 pieces of data equals 10 000 minutes, which is around 167 hours. With an 8 hour shift, this is around 21 days. This means that it will take the better part of a month to update the database during which no user will be able to make use of POLARIS, as data will be inconsistent (certain data will have 2008 information while other data will still have 2007 information). As the database grows, one can imagine the impact this would have on the efficiency POLARIS.

To circumvent this problem, 2 solutions were provided:

1. Uploading Data from a file, and
2. Retrieving Data from another database.

8.1.1 Uploading Data from a File

In order to decrease the amount of time that a user takes to enter data in POLARIS, a module can be created where the user can select to upload the data from a file directly into the database, for a specific country or organisation and for a selected year.

This method is only better than the existing one ONLY if the user receives the file ready. If he/she still has to create the file, then the amount of time taken will be almost the same as if the user entered the information directly into the database.

8.1.2 Retrieving Data from another Database

Very often, and especially in the Shipping industry (which POLARIS was designed for) companies like Clarksosns and InterTanko, have databases with various information on countries, companies and markets. With a module that “reads” XML files, POLARIS can be populated using information from the other databases.

This is far better than the current way of editing the data in POLARIS, and from uploading a file, as the data is ready for retrieval. All the administrator needs to do is point POLARIS to the right XML file and the updating will happen automatically. The only downside to this method is that a subscription fee might need to be paid for access to the foreign database (or for the foreign database to send the required XML file).

8.2 Similarity Margins for Country/Organisation Data

The current version of POLARIS incorporates one Similarity Margin for all pieces of Data, as shown in Figure 8.3:

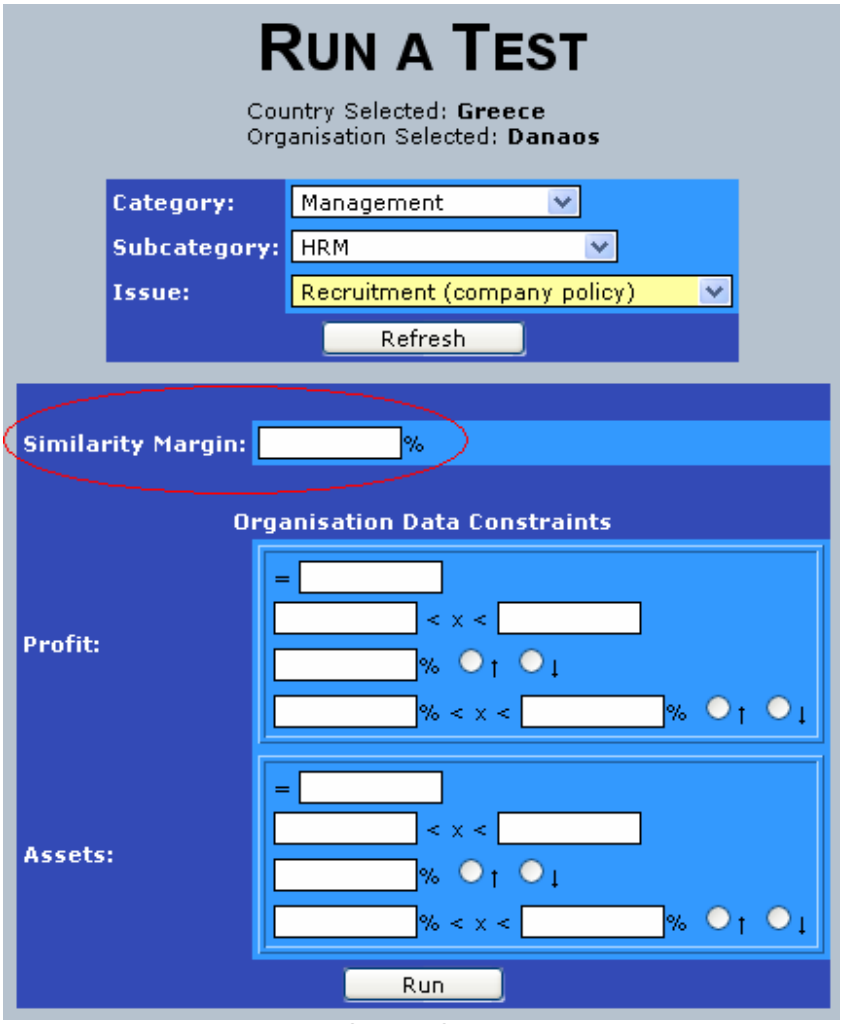


Figure 8.3: General Similarity Margin

In other words, when POLARIS performs the comparisons between countries or between organisations, it uses the same margin. Whether the data is "Profit" or "Liabilities" it will use the same margin, supplied by the user, to find a match. This is a reasonable solution for the moment, however, it poses one problem: a margin of 5% that is acceptable (if not reasonable) for one piece of data, might not be acceptable for another. For example, an organisation might be found similar to another organisation if their assets are within 5% of each other. The same two might be found similar if their turnovers are with 15% of each other.

The problem that occurs with this is the following: It excludes organisations that would otherwise be accepted if some data had a higher similarity margin, and includes organisations that might not have been accepted with a lower, more reasonable similarity margin for other pieces of data.

To circumvent this problem, a viable solution has been found: A module can be created where the user running the "Test" can select what similarity margins he or she wishes to place on every piece of country or organisation data. This way, a match will only be found among countries/organisations if their data's margins are reasonable. For example, now POLARIS will be able to allow the user to have a similarity margin for "Turnover" at 15% and for "Assets" at "5%".

The downside to this solution is that the user wanting to run the "Test" will have to enter the similarity margins for EVERY country and organisation data EVERY time he/she wishes to run a "Test". This is not only a tedious task, but also will take up a lot of the user's valuable time.

In order to solve this new problem that arose with the new solution, the new module can be modified to have "default similarity margins" for every user. The user will be able to "save" his/her similarity margins (assuming he/she picked them using thorough techniques) and be able to use them over and over again. This way, if the user wishes to change only one margin, he/she will only change that single one without needing to reenter all the margins for every run of a new "Test". A screenshot of what the module could look like s show in Figure 8.4:

Figure 8.4: Similarity Margins for every Data

8.3 Situation Similarity

It is all good and well that POLARIS returns a result set containing all the countries/organisations that meet the user's constraints. Unfortunately, this version of POLARIS does only *that*: shows the countries/organisations that met the constraints. It doesn't say which year the constraints were met, and more importantly, it doesn't say if the situation[40] in which the country/organisation was operating in, when it met the specified goal, is similar to today's.

In order to cater for this, two things need to be done:

1. Modify the "Test" module to cater for years, and
2. Create a new module where the administrator can set rules about what is "Situation Similarity" for each Issue (i.e. what rules need to hold in order to show that two situations are similar for a specific Issue).

8.3.1 Modifying the "Test" module

The first change that needs to be made is in the result sets. When POLARIS returns the countries/organisations that meet the criteria of the "Test", it must also return the "Year" of the data that met the user's constraints.

What is returned at the moment is what is shown in Figure 8.5 below:

Figure 8.5: Current "Test" Result Set

What needs to be returned is what is shown in Figure 8.6 below:

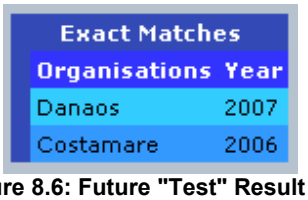


Figure 8.6: Future "Test" Result Set

[40] "Situation" is market conditions and the government regime that the country/organisation was operating in. For example, an organisation that met certain goals during World War II might not, necessarily, do so today, as the "situation" was different.

Having the Year of the organisation that met the user's constraints, allows POLARIS to compare that year's organisation data with the selected organisation's current year's data. That way POLARIS will find if the selected organisation is similar to the "returned" organisation in that year. In addition, it will compare the country data of that year, with the current year's data of the selected country, in order to find if the organisations are operating under a similar government regime.

All in all, it will check to see if the situation that existed when the "matched" organisation attained the user's desired goals is similar to the situation that the selected organisation is operating in, in the selected country. Hence the term *"Situation Similarity"*.

8.3.2 The New Module: Setting Situation Similarity Rules

Although the modifications to the "Test" module described in section 8.3.1 will suffice to give an adequate "Situation Similarity" analysis, the process can be taken one step further. What this new module will do, is set the exact rules that will define the term "similarity" between two situations.

An administrator will be able to select an Issue in the form described previously in section 7.2.12 and 7.3, as shown in Figure 8.7 below:

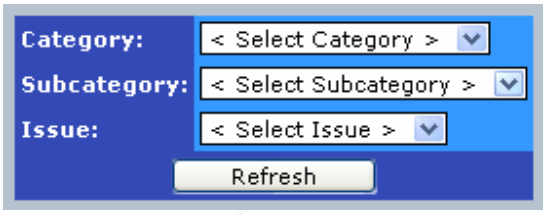


Figure 8.7: Selecting an Issue

He/she will then be able to select the country and/or organisation data in the form described in section 7.2.12 for the setting of the Issue Rules, shown in Figure 8.8 below:

Figure 8.8: Possible way of Setting "Situation Similarity" Rules

With this module, when comparing if countries/organisations are similar to one another, POLARIS will be able to check for Situation Similarity for the Issue that the user is running the "Test" on. This way, two situations might be similar for one Issue and not similar for another. All in all, the module aims to add to the flexibility of POLARIS.

8.4 Strategic Solutions for each Issue

As has already been explained a number of times throughout this thesis, POLARIS returns the names of the countries/organisations that met the constraints that the user placed on the "Test", or more accurately, attained the goals that the user is trying to achieve. In this version of POLARIS it is up to the user to find out what that country/organisation did to attain that goal.

Of course it would be much easier and simpler if POLARIS could suggest a strategy for attaining that goal, in addition to showing the user which countries/organisations attained the goal. For POLARIS to incorporate this functionality, two things can be done:

1. Implement a temporary solution by editing the "Admin" module, and
2. Creating a new module that suggests strategies according to the goal that the country/organisation is trying to achieve and the data that the country/organisation has.

8.4.1 Editing the "Admin" module

Editing the "Admin" module might not return the desired result, but it does provide a temporary solution to the problem.

The first thing that needs to be done is that for every country and organisation in POLARIS a user will need to be able to enter the strategy that that country/organisation used to counter a specific Issue. For example, the user will need to be able to select an organisation (e.g. "Danaos"), a year for the organisation (e.g. "2007") and an Issue that the selected organisation solved in the selected year (e.g. "Recruitment"). So, in our example, the user will need to enter (in text) the strategy that "Danaos" implemented in "2007" in order to counter a problem in "Recruitment".

A possible screenshot for a page that performs the described function is shown in Figure 8.9 below:

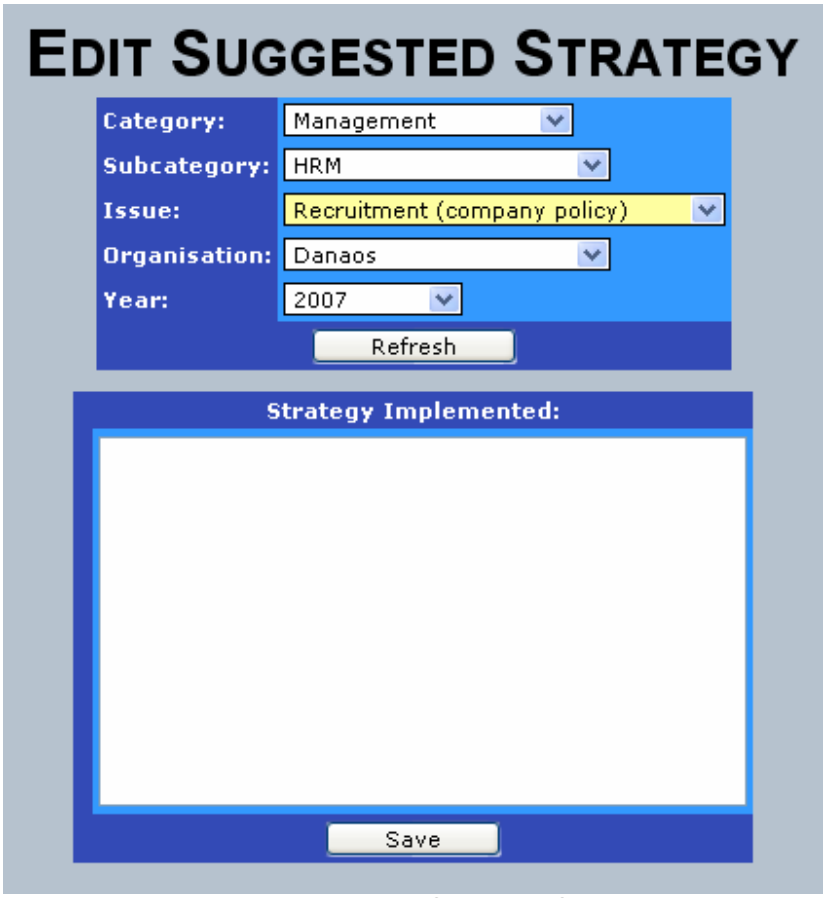


Figure 8.9: Editing the Suggested Strategy

The next and final step required to complete this method is to alter the "Result sets" that the "Test" module returns. At present it returns what is shown in Figure 8.10 below:

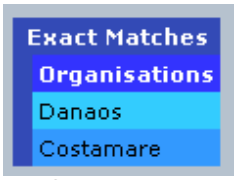


Figure 8.10: Current "Test" Result Set

What it should return is the above result set, where the only change is that each country/organisation name that is returned is, is returned as a link. Clicking on the country/organisation name will then open a new window where the strategy that was entered (as described with Figure 8.9) for this Issue and for this organisation is displayed. This way, the user will not have to look for the implemented strategy, as another user will already have entered it in POLARIS.

Needless to say, the same would apply if the modifications described in section 8.3 were implemented, and the result set would look like what is shown in Figure 8.11 below:

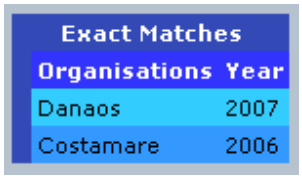


Figure 8.11: Possible Future "Test" Result Set

The difference is that the strategy that will be displayed will not only be for the country/organisation which solved the Issue, but it will also be for the year it was solved in. In other words, if this problem with the selected Issue was solved in more than one year for this organisation, it will display the strategy they used in the year that matched the user's constraints that he/she entered in the "Test".

8.4.2 The New Module: Strategy Suggestion

Unfortunately, the description of the creation of this module is not as simple as the other "future implementations" that have been described so far. This is not because the implementation is difficult "development" wise, but rather because a method to suggest strategies has not been devised by either the author of this thesis nor Mr. Fykaris. A description of this module's logic will, however, be given.

The method that will be devised in the future will need to take into account the data of the country/organisation that wants to achieve the goals described in the "Test". In addition, it will also need to take into account the constraints that the user entered in the "Test". Then, after finding the countries/organisations that met the goals, it will find the changes in the data of each of those countries/organisations. An administrator will have to have set the data that affects each issue (similar to what was described in section 8.3, but instead of "Situation Similarity" the data will refer to "Strategy Suggestion"), and then POLARIS can inform the user which of the data that was set for the Issue changed in the returned countries/organisations. For example, if the "Recruitment" Issue had "Turnover" set as important data for the Issue, then it would look for changes in the returned countries/organisations in the "Turnover" data, and inform the user of the direction and value of that change.

This way, the user will know indirectly what the other countries/organisations did in order to achieve the desired goal, and he/she will be able to implement his/her own strategy to increase or decrease the relevant data values accordingly.

8.5 Graded Result Sets

It is all good and well that the “Test” module returns countries/organisations that met the user’s constraints, but it would be a lot more useful if the result set was “Graded”. What is meant by “Graded” is that it would be more useful to the user if he/she knew which of the returned countries/organisations met the constraints more closely than others.

At present, the countries/organisations that match the user’s constraints are displayed in alphabetical order, as shown in Figure 8.12 below:

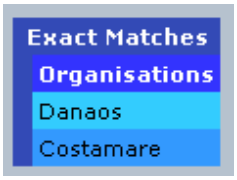


Figure 8.12: "Test" Result set in Alphabetical order

This result set does not indicate which country/organisation better matches the user’s constraints. For example, if the constraint was “Turnover” between “100 000 and 200 000”, and “Danaos’” “Turnover” was “150 000” and “Costamare’s” was “101 000”, it will show no difference, even though Danaos’ result better matches the constraint (“Costamare”, in the example shown, is situated on the border of the constraint).

What would be more useful to the user is if the result set returned the names of the countries/organisations along with a percentage value that signifies its match to the constraints. Again, the calculation for the percentage of the match has not been devised by either the author of the thesis nor Mr. Fykaris, and will need to be done so for this “Graded Result Set” to work.

An example of what the result set could look like if the “Graded Result Set” was implemented, is shown in Figure 8.13 below:

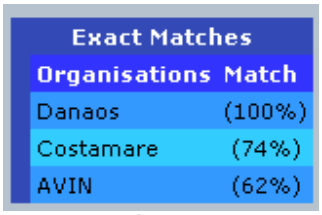
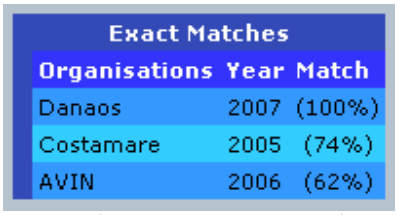


Figure 8.13: Future Graded "Test" Result Set

As one can see, the result set includes a percentage, showing how closely the returned organisations match the constraints, and are displayed in the order

of that percentage. If the percentages of one or more countries/organisations are equal, then the “tied” results are displayed in alphabetical order.

Needless to say, the same would apply if the modifications described in section 8.3 were implemented, and the result set would look like what is shown in Figure 8.14 below:



Exact Matches		
Organisations	Year	Match
Danaos	2007	(100%)
Costamare	2005	(74%)
AVIN	2006	(62%)

Figure 8.14: Future Graded "Test" Result Sets (with Years)

9. CONCLUSIONS

During the course of this thesis the author has covered a number of concepts and topics, ranging from theoretical to technical.

In Part I of the thesis, the author described some basic principles of Artificial Intelligence, and gave basic descriptions of some of the key aspects researched in the field. He then went on to explain what Expert Systems are, and more specifically he described the aspect of Expert Systems know as Decision Support Systems. These were required in order to define POLARIS as a Case-Based Reasoning Expert System. Finally, he went on to describe what it is that POLARIS does, what results it returns, and more importantly, how its results can be used in the business world.

In Part II of the thesis, the author went on to explain some of the more technical aspects of the application. He explained the reasons behind using the WAMP philosophy and why it was more beneficial in the scope of the thesis. He explained the design that was created for the database of POLARIS, as well as the reasons that the selected design was chosen. Finally, the author explained the design that was used to create the flow of POLARIS, as well as the reasons for the selected design.

To complete the description of the application, Chapter 8 of Part III provided a list of detailed descriptions of possible improvements and developments to POLARIS, that could make it a more efficient and useful application.

All in all, the following can be said about the POLARIS Case-Based Reasoning Expert System: It is a useful application, flexible and adaptable to any industry. If used correctly it can be proven to be a reliable assistant to any expert of the shipping industry, and possibly any industry in the future. This is not to say that it is the best application around, but with the recommended modifications described in Chapter 8, it will definitely have the potential to be among the most competitive.

10. BIBLIOGRAPHY

1. Aamodt, A., *"Towards robust expert systems that learn from experience - an architectural framework",* Third European Knowledge Acquisition for Knowledge-Based Systems Workshop, 1989

2. Aamodt, A., Plaza, E., *"Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches"*. AI Communications, 1994

3. Ashley, K.D., *"Arguing by Analogy in Law: A Case-Based Model"*, 1988

4. Barletta, R., *"An introduction to case-based reasoning"*, AI Expert, 1991

5. Batali, J., *"Artificial Intelligence Modelling",* University of California, San Diego's Department of Cognitive Science

6. Bergsten, H., *"Java Server Pages"*, O' Reilly, 2002, 2nd Edition

7. David B.S., *"Principles for case representation in a case-based aiding system for lesson planning"*, 1991

8. Dean, T., Kambhampati, S., *"Planning and Scheduling"*, CRC Handbook of Computer Science and Engineering, 1996

9. Farrel, R., *"Intelligent case selection and presentation"*, 1987

10. Fykaris, G., Doctorate Student at the University of the Aegean

11. Gallaire, H., Minker, J., Nicolas, J.M., "*Advances in Database Theory"*, Vol.1, Plenum, New York, 1981

12. Grant, T., Borcherds, R., *"Communicating @ Work"*, Oxford University Press (South Africa), 2002

13. Hall, M., *"Core Servlets and JSP"*, Prentice Hall and Sun Microsystems, 2002

14. Hayes-Roth, F., Waterman, D., Lenat D., *"Building expert systems"*. Addison Wesley, Reading, MA, US, 1983

15. Kolodner, J. L., *Case-Based Reasoning*. Morgan Kaufmann, 1993

16. Leake, D., *"Artificial Intelligence"*, University of Indiana, 2002

17. Lewis, C., Rieman, J., *"Task-Centered User Interface Design"*, 1993

18. Luger, G., Stubblefield, W., *"Artificial Intelligence and the Design of Expert Systems"*, Benjamin/Cummings Publishing Company, 1989

19. Nilsson, N., *"What is Machine Learning"*, 1996

20. Michie, D., *"A prototype knowledge refinery"*

21. PHP.net, *"PHP Official Website"*, Available: www.php.net, Accessed: Throughout Development

22. Pollice, G., *"Using the Rational Unified Process for Small Projects"*, Rational Software White Paper

23. Potts, J., *"Carnegie Melon Today"*, 2005

24. Schank, R.C., Abelson, R.P., *"Scripts, Plans, Goals and Understanding'*, Erlbaum, Hillsdale, New Jersey, US, 1977

25. Slack, J.M., *"Programming and Problem Solving with Java"*, Brooks/Cole, 2000

26. Slade. S., *"Case-based reasoning: A research paradigm"*, AI Magazine, 1991

27. Statistics South Africa (Communicated via telephone)

28. *"Van Nostrand Scientific Encyclopedia"*, Ninth Edition, Wiley, New York, 2002

29. Wielinga, B.J., Schreiber, A.Th., Breuker, J.A., *"KADS: A modeling approach to knowledge engineering"*, Knowledge Acquisition, 1992

30. Welling, L., Thomson, L., *"PHP and MySQL Web Development"*, Developer's Library, Sams Publishing, 2005, Third Edition

31. WWW Consortium, *"W3Schools"*, Available: www.w3schools.com, Accessed: Throughout Development

APPENDIX A:
SOURCE CODE LISTINGS

Appendix A will contain the Listings of the source code used to develop the POLARIS Web Application. It will be separated into sections each of which will contain the code of a specific “PHP page”. The sections will be ordered in the “PHP pages’” alphabetical order.

(**Note:** Some key functions will be named but their code will be excluded in order to protect the developer and designer of the system, i.e. the University of the Aegean, from Intellectual Property theft.)

A.1 “addCat.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Category Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Category",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add a New Category");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      $catName = $_POST['catName'];
      if ($catName != "")
        addNewCat($dbCon, $catName);

      displayBasicAddForm("addCat.php", "Category Name", "catName");
      displayBasicTable($dbCon, "Categories", "Categories", "CatName",
"There are no Categories in the Database.", "Could not Retrieve
Categories Names from the Database.");

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.2 “addCountry.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Country Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Country",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add a New Country");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      $countryName = $_POST['countryName'];
      if ($countryName != "")
        addNewCountry($dbCon, $countryName);

      displayBasicAddForm("addCountry.php", "Country Name",
"countryName");
      displayBasicTable($dbCon, "Countries", "Countries",
"CountryName", "There are no Countries in the Database.", "Could not
Retrieve Country Names from the Database.");

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.3 “addCountryData.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Country Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Country Data",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add New Country Data");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      $cdName = $_POST['cdName'];
      if ($cdName != "")
        addNewCountryData($dbCon, $cdName);

      displayBasicAddForm("addCountryData.php", "Country Data Name",
"cdName");
      displayBasicTable($dbCon, "Country Data", "CountryData",
"CountryDataName", "There are no Country Data in the Database.", "Could
not Retrieve Country Data Names from the Database.");

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.4 “addIssue.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Issues Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Issue",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add New Issue");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      if (isset($_POST['add']))
      {
        $issueName = $_POST['issueName'];
        $subCatID = $_POST['subCatID'];

        if ($issueName != "" && $subCatID != "-1")
        {
          addNewIssue($dbCon, $issueName, $subCatID);

            $issueName = "";
          $subCatID = "-1";
        }
        else
        {
          if ($issueName == "" || $subCatID == "-1")
          {
            echo 'One or More fields were left blank.<br />';
            echo 'Please try again!<br /><br />';
          }//end if
        }//end if-else
      }//end if

      displayAddIssueForm($dbCon, $issueName, $subCatID);
      displayIssuesTable($dbCon);

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
```

```
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.5 “addOrg.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Organisation Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Organisation",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add a New Organisation");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      if (isset($_POST['add']))
      {
        $orgName = $_POST['orgName'];
        $countryID = $_POST['countryID'];

        if ($orgName != "" && $countryID != "-1")
        {
          addNewOrg($dbCon, $orgName, $countryID);

            $orgName = "";
            $countryID = "-1";
        }
        else
        {
          if ($orgName == "" || $countryID == "-1")
            {
            echo 'One or More fields were left blank.<br />';
            echo 'Please try again!<br /><br />';
            }//end if
        }//end if-else
      }//end if

      displayAddOrgForm($dbCon, $orgName, $countryID);
      displayOrgTable($dbCon);

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
```

```
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.6 “addOrgData.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Organisation Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Organisation Data",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add New Organisation Data");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      $odName = $_POST['odName'];
      if ($odName != "")
        addNewOrgData($dbCon, $odName);

      displayBasicAddForm("addOrgData.php", "Organisation Data Name",
"odName");
      displayBasicTable($dbCon, "Organisation Data", "OrgData",
"OrgDataName", "There are no Organisation Data in the Database.",
"Could not Retrieve Organisation Data Names from the Database.");

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.7 “addSubCat.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add Subcategory Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Add Subcategory",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add New Subcategory");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      if (isset($_POST['add']))
      {
        $subCatName = $_POST['subCatName'];
        $catID = $_POST['catID'];

        if ($subCatName != "" && $catID != "-1")
        {
          addNewSubCat($dbCon, $subCatName, $catID);

            $subCatName = "";
          $catID = "-1";
        }
        else
        {
          if ($subCatName == "" || $catID == "-1")
          {
            echo 'One or More fields were left blank.<br />';
            echo 'Please try again!<br /><br />';
          }//end if
        }//end if-else
      }//end if

      displayAddSubCatForm($dbCon, $subCatName, $catID);
      displaySubCatTable($dbCon);

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
```

```
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.8 “addUser.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Add User Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $userName = $_POST['userName'];
  $name = $_POST['name'];
  $surname = $_POST['surname'];
  $pwd = $_POST['pwd'];
  $pwdRepeat = $_POST['pwdRepeat'];

  $dbCon = connectToDB();

  $userTypes = array();

  $utMaxID = getMaxUserTypeID($dbCon);

  //Adds all the checked checkboxes into an array
  for ($i = 1; $i <= $utMaxID; $i++)
  {
      if (isset($_POST["userTypeID$i"]))
      {
        $userTypes[$i] = $i;
      }//end if
  }//end for

  try
  {
    checkValidUser();

      addHeader("Add User",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add a New User");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      if (isset($_POST['add']))
      {
        if ($userName != "" && $name != "" && $surname != "" && $pwd !=
"" && $pwdRepeat && sizeof($userTypes) > 0)
        {
          if (!userExists($dbCon, $userName))
            {
              if ($pwd == $pwdRepeat)
              {
```

```
                addNewUser($dbCon, $userName, $name, $surname,
$userTypes);

                  $userName = "";
                  $name = "";
                  $surname = "";
                  $pwd = "";
                  $pwdRepeat = "";

                  $userTypes = "";
              }
              else
              {
                echo 'The Password Confrimation was Incorrect!<br />';
                echo 'Please try again!<br /><br />';
              }//end if-else
            }
            else
            {
              echo 'This User Name already exists.<br />';
              echo 'Please try Another one!<br /><br />';
            }//end if-else
        }
        else
        {
          echo 'One or More fields were left blank.<br />';
            echo 'Please try Again!<br /><br />';
        }//end if-else
      }//end if

      displayAddUserForm($dbCon, $userName, $name, $surname,
$userTypes, "Add");
      displayAllUsers($dbCon);

    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
  $dbCon->close();
?>
```

A.9 “adminFuncs.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains the functions for the Admin
//module of the POLARIS Web App

//---------------------------------------------------------------------
//Adds the new Category in the Database
function addNewCat($dbCon, $catName)
{
  if (itemExists($dbCon, $catName, "Categories", "CatName"))
  {
    echo 'This Category Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO Categories(CatName)
                VALUES ('$catName') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new Category
into the Database.');

      echo '<b>New Category Added!<b><br /><br />';
  }//end if-else
}//end addNewCat Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Adds the new Country in the Database
function addNewCountry($dbCon, $countryName)
{
  if (itemExists($dbCon, $countryName, "Countries", "CountryName"))
  {
    echo 'This Country Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO Countries(CountryName)
                VALUES ('$countryName') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new Country
into the Database.');

      echo '<b>New Country Added!<b><br /><br />';
  }//end if-else
}//end addNewCountry Function
//---------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
//Adds the new Country Data in the Database
function addNewCountryData($dbCon, $cdName)
{
  if (itemExists($dbCon, $cdName, "CountryData", "CountryDataName"))
  {
    echo 'This Country Data Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO CountryData(CountryDataName)
                VALUES ('$cdName') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new Country
Data into the Database.');

      echo '<b>New Country Data Added!<b><br /><br />';
  }//end if-else
}//end addNewCountryData Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Add the new Issue in the Database
function addNewIssue($dbCon, $issueName, $subCatID)
{
  if (itemExists($dbCon, $issueName, "Issues", "IssueName"))
  {
    echo 'This Issue Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO Issues(IssueName, SubcatID)
                VALUES ('$issueName', '$subCatID') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new Issue
into the Database.');

      echo '<b>New Issue Added!<b><br /><br />';
  }//end if-else
}//end addNewIssue Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Adds the new Organisation in the Database
function addNewOrg($dbCon, $orgName, $countryID)
{
  if (itemExists($dbCon, $orgName, "Organisations", "OrgName"))
  {
    echo 'This Organisation Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO Organisations(OrgName, CountryID)
```

```
                VALUES ('$orgName', '$countryID') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new
Organistaion into the Database.');

      echo '<b>New Organisation Added!<b><br /><br />';
  }//end if-else
}//end addNewOrg Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Adds the new Organisation Data in the Database
function addNewOrgData($dbCon, $odName)
{
  if (itemExists($dbCon, $odName, "OrgData", "OrgDataName"))
  {
    echo 'This Organisation Data Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO OrgData(OrgDataName)
                VALUES ('$odName') ";

      $resultSet = runQuery($dbCon, $sql, 'Counld not Add new
Organistaion Data into the Database.');

      echo '<b>New Organisation Data Added!<b><br /><br />';
  }//end if-else
}//end addNewOrgData Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Adds the new Subcategory in the Database
function addNewSubCat($dbCon, $subCatName, $catID)
{
  if (itemExists($dbCon, $subCatName, "Subcategories", "SubcatName"))
  {
    echo 'This Subcategory Name already exists.<br />';
      echo 'Please try another one!<br /><br />';
  }
  else
  {
    $sql = "INSERT INTO Subcategories(SubCatName, CatID)
                VALUES ('$subCatName', '$catID') ";

      $resultSet = runQuery($dbCon, $sql, 'Could not Add a new
Subcategory into the Database.');

      echo '<b>New Subcategory Added!<b><br /><br />';
  }//end if-else
}//end addNewSubCat Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
function dataExists($dbCon, $sourceID, $yr, $source)
```

```
{
  if ($source == "Country")
  {
    $table = 'CountryDataDet';
      $tableID = 'CountryID';
  }
  elseif ($source == "Organisation")
  {
    $table = 'OrgDataDet';
      $tableID = 'OrgID';
  }//end if-else

  $sql = "SELECT *
              FROM $table
              WHERE $tableID = $sourceID
              AND DataYear = $yr ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve the Detail
Contents from the Database.');

  if ($resultSet->num_rows)
  {
    return true;
  }
  else
  {
    return false;
  }//end if-else
}//end dataExists Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Gets the details of the Data, given the Table of the Data (Country or
Org),
//the DataID and the Coutry ro Org ID.
//It return the Detail of the Data, or "nithing" if it has not been
provided
function getDataDetails($dbCon, $dataID, $sourceID, $yr, $source)
{
  if ($source == "Country")
  {
    $table = 'CountryDataDet';
      $tableID1 = 'CountryID';
      $tableID2 = 'CountryDataID';
  }
  elseif ($source == "Organisation")
  {
    $table = 'OrgDataDet';
      $tableID1 = 'OrgID';
      $tableID2 = 'OrgDataID';
  }//end if-else

  $sql = "SELECT Det
              FROM $table
              WHERE $tableID1 = '$sourceID'
              AND $tableID2 = '$dataID'
```

```
              AND DataYear = $yr ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve the Detail
Contents from the Database 1.');

  if ($resultSet->num_rows > 0)
  {
    $row = $resultSet->fetch_assoc();

      return $row['Det'];
  }
  else
  {
    return "";
  }//end if-else
}//end getDataDetails Function
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
//Updates the Data of the Source (Country or Org) with the given data
function updateData($dbCon, $data, $source, $sourceID, $dataMaxID, $yr)
{
  if ($source == "Country")
  {
    $table = "CountryDataDet";
      $tableID1 = "CountryID";
      $tableID2 = "CountryDataID";
  }
  elseif ($source == "Organisation")
  {
    $table = "OrgDataDet";
      $tableID1 = "OrgID";
      $tableID2 = "OrgDataID";
  }//end if-else

  if (dataExists($dbCon, $sourceID, $yr, $source))
  {
    $sql = "DELETE FROM $table
                WHERE $tableID1 = '$sourceID'
                  AND DataYear = $yr ";

    $resultSet = runQuery($dbCon, $sql, 'Could not Delete Detail
Contents from the Database.');
  }//end if

  for ($i = 1; $i <= $dataMaxID; $i++)
    {
      if ($data[$i] != "")
        {
          $sql = "INSERT INTO $table
                    VALUES ('$sourceID', '$i', '$data[$i]', $yr) ";

          $resultSet = runQuery($dbCon, $sql, 'Could not Insert Detail
Contents into the Database.');
        }//end if
    }//end for
```

```
  echo '<b>'.$source.' Data Updated!</b><br /><br />';
}//end updateData Function
//---------------------------------------------------------------------
?>
```

A.10 “changePwd.php”

```
<?php
  session_start();

//DataMax
//Programmer: T. Scholiadis

//This is the Change Password Page of the DataMax Web App

  require("polarisFuncs.php"); //Including Function Files

  $userName = $_SESSION['validUser'];

  $oldPwd = $_POST['oldPwd'];
  $pwd = $_POST['pwd'];
  $pwdRepeat = $_POST['pwdRepeat'];

  try
  {
    checkValidUser();

      addHeader("Change Password",$_SESSION['validUser']);
    displayMenu();
    addHeading("Change My Password");

      $dbCon = connectToDB();

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      if (isset($_POST['save']))
      {
        if ($oldPwd == "" || $pwd == "" || $pwdRepeat == "")
        {
          echo 'One or More fields were left blank!<br /><br />';
        }
        else
        {
          if (oldPwdConfirmed($dbCon, $oldPwd))
            {
            if ($pwd == $pwdRepeat)
              {
              updatePassword($dbCon, $userName, $pwd);
              }
              else
              {
                echo 'The Password Confrimation was Incorrect!<br />';
                echo 'Please try again!<br /><br />';
              }//end if-else
            }
            else
            {
              echo 'The Old Password is Incorrect!<br />';
```

```
              echo 'Please try again!<br /><br />';
            }//end if-else
          }//end if-else
        }//end if

        displayChangePwdForm($dbCon);

      $dbCon->close();
        addFooter();
    }
    catch(Exception $e)
    {
      echo $e->getMessage();
    }//end try-catch
?>
```

A.11 “dbFuncs.php”

```
<?php
//POLARIS
//Developer: T. Scholiadis

//This file contains all the Database related functions for the POLARIS
Web App

//-------------------------------------------------------------------
---
//Function that tries to connect to the Database
function connectToDB()
{
   $result = new mysqli('localhost','polaris','polaris','polaris');
   if (!$result)
     throw new Exception('Could not connect to database server');
   else
     return $result;
}//end connectToDB Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Function that runs a query
function runQuery($dbCon, $sql, $errMsg)
{
  $result = $dbCon->query($sql);

  if (!$result)
    throw new Exception($errMsg);
  else
    return $result;
}//end runQuery Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
--------------------------------------
//-------------------------------------------------------------------
--------------------------------------

//-------------------------------------------------------------------
//Gets the Data name, given the Data ID
function getDataName($dbCon, $dataID, $source)
{
  if ($source == "Country")
  {
    $table = "CountryData";
      $tableID = "CountryDataID";
      $fieldName = "CountryDataName";
  }
  elseif ($source == "Organisation")
```

```
  {
    $table = "OrgData";
      $tableID = "OrgDataID";
      $fieldName = "OrgDataName";
  }//end if-ladder

  $sql = "SELECT $fieldName
              FROM $table
              WHERE $tableID = '$dataID' ";

  $resultSet = runQuery($dbCon, $sql, 'Counld not Retrieve '.$source.'
Data Name from Database.');

  $row = $resultSet->fetch_assoc();

  return $row[$fieldName];
}//end getDataName Function
//------------------------------------------------------------------------

//------------------------------------------------------------------------
---
//Returns Max ID of the given table
function getMaxID($dbCon, $table, $tableID)
{
  $sql = "SELECT Max($tableID) as maxID
              FROM $table ";

  $resultSet = runQuery($dbCon, $sql, 'Counld not Retrieve Max ID from
Database.');

  if ($resultSet->num_rows > 0)
  {
    $row = $resultSet->fetch_assoc();

      return $row['maxID'];
  }
  else
  {
    return "";
  }//end if-else
}//end gatMaxID Function
//------------------------------------------------------------------------
---

//------------------------------------------------------------------------
---
//Gets the name of the Country or Organisation given
//the ID number
function getName($dbCon, $sourceID, $source)
{
  if ($source == "Country")
  {
    $table = "Countries";
      $tableID = "CountryID";
      $tableDet = "CountryName";
  }
```

```
  elseif ($source == "Organisation")
  {
    $table = "Organisations";
      $tableID = "OrgID";
      $tableDet = "OrgName";
  }//end if-else

  $sql = "SELECT $tableDet
              FROM $table
              WHERE $tableID = '$sourceID' ";

  $resultSet = runQuery($dbCon, $sql, 'Counld not Retrieve '.$source.'
Details from Database.');

  $row = $resultSet->fetch_assoc();

  return $row[$tableDet];
}//end getName Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Checks if the item exists in the Database
//Returns true if it exists and false if it doesn't
function itemExists($dbCon, $fieldData, $table, $field)
{
  $sql = "SELECT *
              FROM $table
              WHERE $field = '$fieldData' ";

  $resultSet = runQuery($dbCon, $sql, 'Counld not Retrieve Item from
Database.');

  if ($resultSet->num_rows > 0)
    return true;
  else
    return false;
}//end itemExists FUnction
//-------------------------------------------------------------------------
---
?>
```

A.12 “editAdminUser.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Edit User Details Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  if (isset($_POST['edit']) && $_POST['username'] != "-1")
  {
    $_SESSION['editUser'] = $_POST['username'];

      header("Location: editUser.php");
  }//end if

  try
  {
    checkValidUser();

      addHeader("Edit User",$_SESSION['validUser']);
    displayMenu();
    addHeading("Edit User");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      displayEditForm($dbCon, "Username", "username", "Users",
"UserName", "UserName", "Could not Retrieve Users from the Database.",
"editAdminUser.php");
      displayAllUsers($dbCon);

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.13 “editCountryData.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Edit Country Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $countryID = $_POST['countryID'];
  $yr = $_POST['yr'];

  try
  {
    checkValidUser();

      addHeader("Edit Country Data",$_SESSION['validUser']);
    displayMenu();
    addHeading("Edit the Data of a Country");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      displayEditCountryOrgForm($dbCon, "Country", "countryID",
"Countries", "CountryID", "CountryName", "Could not Retrieve Country
Names from the Database.", "editCountryData.php", $yr, $countryID);

      if (isset($_POST['save']))
      {
        $dataMaxID = getMaxID($dbCon, "CountryData", "CountryDataID");

        updateData($dbCon, $_POST, "Country", $countryID, $dataMaxID,
$yr);

        $_POST['edit'] = true;
      }//end if

      if (isset($_POST['edit']))
      {
        if ($countryID != "-1" && $yr != "-1")
        {
          displayEditDataForm($dbCon, $countryID, $yr,
"editCountryData.php", "Country");
        }
        else
        {
          echo 'One or More Fields were left blank.<br />';
            echo 'Please try again!<br /><br />';
        }//end if-else
```

```
      }//end if

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.14 “editOrgData.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Edit Organisation Data Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $orgID = $_POST['orgID'];
  $yr = $_POST['yr'];

  try
  {
    checkValidUser();

      addHeader("Edit Organisation Data",$_SESSION['validUser']);
    displayMenu();
    addHeading("Add the Data of an Organisation");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      displayEditCountryOrgForm($dbCon, "Organisation", "orgID",
"Organisations", "OrgID", "OrgName", "Could not Retrieve Organisation
Names from the Database.", "editOrgData.php", $yr, $orgID);

      if (isset($_POST['save']))
      {
        $dataMaxID = getMaxID($dbCon, "OrgData", "OrgDataID");

        updateData($dbCon, $_POST, "Organisation", $orgID, $dataMaxID,
$yr);

        $_POST['edit'] = true;
      }//end if

      if (isset($_POST['edit']))
      {
        if ($orgID != "-1" && $yr != "-1")
        {
          displayEditDataForm($dbCon, $orgID, $yr, "editOrgData.php",
"Organisation");
        }
        else
        {
          echo 'One or More Fields were left blank.<br />';
            echo 'Please try again!<br /><br />';
        }//end if-else
```

```
      }//end if

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.15 “editUser.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Edit User Details Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $dbCon = connectToDB();

  if (isset($_SESSION['editUser']))
  {
    $userName = $_SESSION['editUser'];

      unset($_SESSION['editUser']);

      $adminEdit = true;
  }
  else
  {
    $userName = $_SESSION['validUser'];
  }//end if-else

  if (isset($_POST['update']))
  {
    $name = $_POST['name'];
    $surname = $_POST['surname'];

      if ($adminEdit)
      {
        $userTypes = array();

      $utMaxID = getMaxUserTypeID($dbCon);

      //Adds all the checked checkboxes into an array
      for ($i = 1; $i <= $utMaxID; $i++)
      {
          if (isset($_POST["userTypeID$i"]))
          {
            $userTypes[$i] = $i;
          }//end if
      }//end for
      }//end if
  }
  else
  {
    $userDetails = getUserDetails($dbCon, $userName);

    $name = $userDetails['Name'];
    $surname = $userDetails['Surname'];
```

```
      if ($adminEdit)
      {
        $utMaxID = getMaxUserTypeID($dbCon);

        $userTypes = getUserTypes($dbCon, $userName);
      }//end if
  }//end if-else

  try
  {
    checkValidUser();

      addHeader("Edit Details",$_SESSION['validUser']);
    displayMenu();
    addHeading("Edit My Details");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      if (isset($_POST['update']))
      {
        if ($userName != "" && $name != "" && $surname != "")
        {
          if (!$adminEdit)
              updateUserDetails($dbCon, $userName, $name, $surname);
            else
              updateUserDetails($dbCon, $userName, $name, $surname,
$userTypes, $utMaxID);
        }
        else
        {
          echo 'One or More fields were left blank.<br />';
            echo 'Please try Again!<br /><br />';
        }//end if-else
      }//end if

      displayAddUserForm($dbCon, $userName, $name, $surname,
$userTypes, "Edit", $adminEdit);

    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch

  $dbCon->close();
?>
```

A.16 “exmplmenu_var.js”

```
/*********************************************************************
************
*     (c) Ger Versluis 2000 version 5.411 24 December 2001 (updated Jan
31st, 2003 by Dynamic Drive for Opera7)
*     For info write to menus@burmees.nl                 *
*     You may remove all comments for faster loading           *
*********************************************************************
************/

      var NoOffFirstLineMenus=5;               // Number of first
level items
      var LowBgColor='003399';                 // Background color
when mouse is not over
      var LowSubBgColor='white';               // Background color
when mouse is not over on subs
      var HighBgColor='003399';                // Background color
when mouse is over
      var HighSubBgColor='003399';             // Background color
when mouse is over on subs
      var FontLowColor='white';                // Font color when
mouse is not over
      var FontSubLowColor='black';             // Font color subs when
mouse is not over
      var FontHighColor='white';               // Font color when
mouse is over
      var FontSubHighColor='white';            // Font color subs when
mouse is over
      var BorderColor='white';                 // Border color
      var BorderSubColor='cccccc';             // Border color for
subs
      var BorderWidth=1;                       // Border width
      var BorderBtwnElmnts=1;             // Border between elements 1
or 0
      var FontFamily="Verdana, Arial, Helvetica, sans-serif"      //
Font family menu items
      var FontSize=8;                     // Font size menu items
      var FontBold=1;                     // Bold menu items 1 or 0
      var FontItalic=0;                   // Italic menu items 1 or 0
      var MenuTextCentered='left';             // Item text position
'left', 'center' or 'right'
      var MenuCentered='center';               // Menu horizontal
position 'left', 'center' or 'right'
      var MenuVerticalCentered='top';          // Menu vertical
position 'top', 'middle','bottom' or static
      var ChildOverlap=.2;                     // horizontal overlap
child/ parent
      var ChildVerticalOverlap=.2;             // vertical overlap
child/ parent
      var StartTop=65;                    // Menu offset x coordinate
      var StartLeft=1;                    // Menu offset y coordinate
      var VerCorrect=0;                   // Multiple frames y
correction
```

```
      var HorCorrect=0;                    // Multiple frames x
correction
      var LeftPaddng=3;                    // Left padding
      var TopPaddng=2;                     // Top padding
      var FirstLineHorizontal=1;                 // SET TO 1 FOR
HORIZONTAL MENU, 0 FOR VERTICAL
      var MenuFramesVertical=1;                  // Frames in cols or
rows 1 or 0
      var DissapearDelay=1000;                   // delay before menu
folds in
      var TakeOverBgColor=1;               // Menu frame takes over
background color subitem frame
      var FirstLineFrame='navig';                // Frame where first
level appears
      var SecLineFrame='space';                  // Frame where sub
levels appear
      var DocTargetFrame='space';                // Frame where target
documents appear
      var TargetLoc='';                    // span id for relative
positioning
      var HideTop=0;                       // Hide first level when
loading new document 1 or 0
      var MenuWrap=1;                      // enables/ disables menu
wrap 1 or 0
      var RightToLeft=0;                         // enables/ disables
right to left unfold 1 or 0
      var UnfoldsOnClick=0;                // Level 1 unfolds onclick/
onmouseover
      var WebMasterCheck=0;                // menu tree checking on or
off 1 or 0
      var ShowArrow=1;                     // Uses arrow gifs when 1
      var KeepHilite=1;                    // Keep selected path
highligthed
      var
Arrws=['images/tri.gif',5,10,'images/tridown.gif',10,5,'images/trileft.
gif',5,10]; // Arrow source, width and height

function BeforeStart(){return}
function AfterBuild(){return}
function BeforeFirstOpen(){return}
function AfterCloseAll(){return}


// Menu tree
//    MenuX=new Array(Text to show, Link, background image (optional),
number of sub elements, height, width);
//    For rollover images set "Text to show" to:
"rollover:Image1.jpg:Image2.jpg"

      Menu1=new Array("TEST","runTest.php","",0,20,80);

      Menu2=new Array("ADMIN","","",6,20,80);
            Menu2_1=new Array("Users","","",2,20,120);
                  Menu2_1_1=new Array("Add
User","addUser.php","",0,20,130);
```

```
                  Menu2_1_2=new Array("Edit
User","editAdminUser.php","");
            Menu2_2=new Array("Countries","","",3,20,120);
                  Menu2_2_1=new Array("Edit Country
Data","editCountryData.php","",0,20,130);
                  Menu2_2_2=new Array("Add Country
Data","addCountryData.php","");
                  Menu2_2_3=new Array("Add
Country","addCountry.php","");
            Menu2_3=new Array("Organisations","","",3,20,120);
                  Menu2_3_1=new Array("Edit Org.
Data","editOrgData.php","",0,20,130);
                  Menu2_3_2=new Array("Add Org.
Data","addOrgData.php","");
                  Menu2_3_3=new Array("Add
Organisation","addOrg.php","");
            Menu2_4=new Array("Issues","","",2,20,120);
                  Menu2_4_1=new Array("Add
Issue","addIssue.php","",0,20,130);
                  Menu2_4_2=new Array("Set Rules","setRules.php","");
            Menu2_5=new Array("Add
Subcategory","addSubCat.php","",0,20,120);
            Menu2_6=new Array("Add Category","addCat.php","",0,20,120);

      Menu3=new Array("MY ACCOUNT","myAccount.php","",2,20,100);
            Menu3_1=new Array("Edit My
Details","editUser.php","",0,20,140);
            Menu3_2=new Array("Change Password","changePwd.php","",0);

      Menu4=new Array("--","underConstruction.php","",0,20,80);

      Menu5=new Array("LOGOUT","logout.php","",0);
```

A.17 “general.css”

```
/* POLARIS
Developer: T. Scholiadis

This is a CSS file containing less specific Style Editing*/

/* ------------------------------------------------------------*/
/*Section that edits the Header of every page*/
#header table
{
  width: 100%;
  text-align: center;
}
/* ------------------------------------------------------------*/

/* ------------------------------------------------------------*/
/*Section that edits the Login page*/
#login table
{
  width: 100%
}

#login td.left
{
  width: 50%;
  text-align: right;
}

#login td.right
{
  width: 50%;
  text-align: left;
}
/* ------------------------------------------------------------*/

/* ------------------------------------------------------------*/
#heading td.head
{
  font-family: sans-serif;
  font-variant: small-caps;
  font-weight: bold;
  font-size: 300%;
}
/* ------------------------------------------------------------*/
```

A.18 “login.php”

```
<?php
//POLARIS
//Developer: T. Scholiadis

//This is the Login Page of the POLARIS Web App

  require_once('polarisFuncs.php'); //Including Function Files
  addHeader('','');

  displayLoginForm();

  addFooter();
?>
```

A.19 “logout.php”

```
<?php
//POLARIS
//Developer: T. Scholiadis

//This is the Logout Page of the POLARIS Web App

  session_start();
  require_once('polarisFuncs.php');

  //Storing the "old User" in order to check if the user was logged in
  $oldUser = $_SESSION['validUser'];

  unset($_SESSION['validUser']);
  unset($_SESSION['userType']);
  unset($_SESSION['countryID']);
  unset($_SESSION['orgID']);
  $resultDestroy = session_destroy();

  //-------------------------------------------------------------------
--------
  //HMTL Ouptut
  addHeader("Logging Out", "");
  echo '<table align="center"><tr><td>';
  addHeading("Logging Out");
  if (!empty($oldUser))
  {
    if (resultDestroy)
      {
        //If they were logged in and are now logged out
        echo 'You have been successfully Logged Out! <br />';
        addURL("login.php", "Login");
      }
      else
      {
        //If they were logged in and could not be logged out
        echo 'Could not Log you out! <br />';
      }//end if-else
  }
  else
  {
    //If the user wasn't logged in but managed to get to this page
      echo 'You were not logged in, and so have not been logged out!
<br />';
    addURL("login.php", "Login");
  }//end if-else

  echo '</td></tr></table>';
  addFooter();
  //-------------------------------------------------------------------
--------
?>
```

A.20 “menu_com.js”

```
/**********************************************************************
**************
      (c) Ger Versluis 2000 version 5.5 24 December 2001 (updated Jan
31st, 2003 by Dynamic Drive for Opera7)
      Updated 19 July, 2003 by GV for CSS CompatMode
      HV Menu found on Dynamic Drive ONLY may be used on both
commercial and non commerical sites
      For info write to menus@burmees.nl

      This script featured on Dynamic Drive DHTML code library:
http://www.dynamicdrive.com
**********************************************************************
**************/
      var AgntUsr=navigator.userAgent.toLowerCase();
      var AppVer=navigator.appVersion.toLowerCase();
      var DomYes=document.getElementById?1:0;
      var NavYes=AgntUsr.indexOf('mozilla')!=-
1&&AgntUsr.indexOf('compatible')==-1?1:0;
      var ExpYes=AgntUsr.indexOf('msie')!=-1?1:0;
      var Opr=AgntUsr.indexOf('opera')!=-1?1:0;
      var Opr6orless=window.opera &&
navigator.userAgent.search(/opera.[1-6]/i)!=-1 //DynamicDrive.com added
code
      if(Opr){NavYes=1;ExpYes=0;}
      var DomNav=DomYes&&NavYes?1:0;
      var DomExp=DomYes&&ExpYes?1:0;
      var Nav4=NavYes&&!DomYes&&document.layers?1:0;
      var Exp4=ExpYes&&!DomYes&&document.all?1:0;
      var Exp6Plus=(AppVer.indexOf("msie 6")!= -1||AppVer.indexOf("msie
7")!= -1)?1:0
      var PosStrt=(NavYes||ExpYes||Opr)&&!Opr6orless?1:0;
      var
P_X=DomYes?"px":"",FHtml=null,ScHtml=null,FCmplnt=0,SCmplnt=0;
      var FrstLoc,ScLoc,DcLoc;
      var ScWinWdth,ScWinHght,FrstWinWdth,FrstWinHght;
      var ScLdAgainWin;
      var FirstColPos,SecColPos,DocColPos;
      var RcrsLvl=0;
      var FrstCreat=1,Loadd=0,Creatd=0,IniFlg,AcrssFrms=1;
      var FrstCntnr=null,CurrntOvr=null,CloseTmr=null;
      var CntrTxt,TxtClose,ImgStr;
      var Ztop=100;
      var ShwFlg=0;
      var M_StrtTp=StartTop,M_StrtLft=StartLeft;
      var StaticPos=0;
      var M_Hide=Nav4?'hide':'hidden';
      var M_Show=Nav4?'show':'visible';
      var
Par=parent.frames[0]&&FirstLineFrame!=SecLineFrame?parent:window;
      var Doc=Par.document;
      var Bod=Doc.body;
```

```
      var Trigger=NavYes&&!Opr?Par:Bod;

      MenuTextCentered=MenuTextCentered==1||MenuTextCentered=='center'?
'center':MenuTextCentered==0||MenuTextCentered!='right'?'left':'right';

      WbMstrAlrts=["Item not defined: ","Item needs height: ","Item
needs width: "];

      if(Trigger.onload)Dummy=Trigger.onload;
      Trigger.onload=Go;

function Dummy(){return}

function CnclSlct(){return false}

function RePos(){
      FrstWinWdth=ExpYes?FCmplnt?FHtml.clientWidth:FrstLoc.document.bod
y.clientWidth:FrstLoc.innerWidth;
      FrstWinHght=ExpYes?FCmplnt?FHtml.clientHeight:FrstLoc.document.bo
dy.clientHeight:FrstLoc.innerHeight;
      ScWinWdth=ExpYes?SCmplnt?ScHtml.clientWidth:ScLoc.document.body.c
lientWidth:ScLoc.innerWidth;
      ScWinHght=ExpYes?SCmplnt?ScHtml.clientHeight:ScLoc.document.body.
clientHeight:ScLoc.innerHeight;
      if(MenuCentered=='justify'&&FirstLineHorizontal){
            FrstCntnr.style.width=FrstWinWdth+P_X;
            var LftXtra=(DomNav&&!Opr)||FCmplnt?LeftPaddng:0;
            ClcJus();
            var P=FrstCntnr.FrstMbr,W=Menu1[5],i;

      for(i=0;i<NoOffFirstLineMenus;i++){P.style.width=W+P_X;P=P.PrvMbr
}}
      StaticPos=-1;
      if(TargetLoc)ClcTrgt();
      if(MenuCentered)ClcLft();
      if(MenuVerticalCentered)ClcTp();
      PosMenu(FrstCntnr,StartTop,StartLeft)}

function UnLoaded(){
      if(CloseTmr)clearTimeout(CloseTmr);
      Loadd=0; Creatd=0;
      if(HideTop){
            var FCStyle=Nav4?FrstCntnr:FrstCntnr.style;
            FCStyle.visibility=M_Hide}}

function ReDoWhole(){
      if(ScWinWdth!=ScLoc.innerWidth||ScWinHght!=ScLoc.innerHeight||Frs
tWinWdth!=FrstLoc.innerWidth||FrstWinHght!=FrstLoc.innerHeight)Doc.loca
tion.reload()}

function Check(WMnu,NoOf){
      var i,array,ArrayLoc;
      ArrayLoc=parent.frames[0]?parent.frames[FirstLineFrame]:self;
      for(i=0;i<NoOf;i++){
            array=WMnu+eval(i+1);
            if(!ArrayLoc[array]){WbMstrAlrt(0,array); return false}
```

```
            if(i==0){   if(!ArrayLoc[array][4]){WbMstrAlrt(1,array);
return false}
                  if(!ArrayLoc[array][5]){WbMstrAlrt(2,array); return
false}}

      if(ArrayLoc[array][3])if(!Check(array+'_',ArrayLoc[array][3]))
return false}
      return true}

function WbMstrAlrt(No,Xtra){
      return confirm(WbMstrAlrts[No]+Xtra+'   ')}

function Go(){
      Dummy();
      if(Loadd||!PosStrt)return;
      BeforeStart();
      Creatd=0; Loadd=1;
      status='Building menu';
      if(FirstLineFrame =="" || !parent.frames[FirstLineFrame]){
            FirstLineFrame=SecLineFrame;
            if(FirstLineFrame =="" || !parent.frames[FirstLineFrame]){
                  FirstLineFrame=SecLineFrame=DocTargetFrame;
                  if(FirstLineFrame =="" ||
!parent.frames[FirstLineFrame])FirstLineFrame=SecLineFrame=DocTargetFra
me=''}}
      if(SecLineFrame =="" || !parent.frames[SecLineFrame]){
            SecLineFrame=DocTargetFrame;
            if(SecLineFrame =="" ||
!parent.frames[SecLineFrame])SecLineFrame=DocTargetFrame=FirstLineFrame
}
      if(DocTargetFrame =="" ||
!parent.frames[DocTargetFrame])DocTargetFrame=SecLineFrame;
      if(WebMasterCheck){
      if(!Check('Menu',NoOffFirstLineMenus)){status='build
aborted';return}}
      FrstLoc=FirstLineFrame!=""?parent.frames[FirstLineFrame]:window;
      ScLoc=SecLineFrame!=""?parent.frames[SecLineFrame]:window;
      DcLoc=DocTargetFrame!=""?parent.frames[DocTargetFrame]:window;
      if (FrstLoc==ScLoc) AcrssFrms=0;
      if (AcrssFrms)FirstLineHorizontal=MenuFramesVertical?0:1;
      if(Exp6Plus||Opr){

      FHtml=FrstLoc.document.getElementsByTagName("HTML")[0];ScHtml=ScL
oc.document.getElementsByTagName("HTML")[0];
            FCmplnt=FrstLoc.document.compatMode.indexOf("CSS")==-
1?0:1;SCmplnt=ScLoc.document.compatMode.indexOf("CSS")==-1?0:1}
      FrstWinWdth=ExpYes?FCmplnt?FHtml.clientWidth:FrstLoc.document.bod
y.clientWidth:FrstLoc.innerWidth;
      FrstWinHght=ExpYes?FCmplnt?FHtml.clientHeight:FrstLoc.document.bo
dy.clientHeight:FrstLoc.innerHeight;
      ScWinWdth=ExpYes?SCmplnt?ScHtml.clientWidth:ScLoc.document.body.c
lientWidth:ScLoc.innerWidth;
      ScWinHght=ExpYes?SCmplnt?ScHtml.clientHeight:ScLoc.document.body.
clientHeight:ScLoc.innerHeight;
      if(Nav4){   CntrTxt=MenuTextCentered!='left'?"<div
align='"+MenuTextCentered+"'>":"";
```

```
            TxtClose="</font>"+MenuTextCentered!='left'?"</div>":""}
      FirstColPos=Nav4?FrstLoc.document:FrstLoc.document.body;
      SecColPos=Nav4?ScLoc.document:ScLoc.document.body;
      DocColPos=Nav4?DcLoc.document:ScLoc.document.body;
      if
(TakeOverBgColor)FirstColPos.bgColor=AcrssFrms?SecColPos.bgColor:DocCol
Pos.bgColor;
      if(MenuCentered=='justify'&&FirstLineHorizontal)ClcJus();
      if(FrstCreat){
            FrstCntnr=CreateMenuStructure('Menu',NoOffFirstLineMenus);
            FrstCreat=AcrssFrms?0:1}
      else CreateMenuStructureAgain('Menu',NoOffFirstLineMenus);
      if(TargetLoc)ClcTrgt();
      if(MenuCentered)ClcLft();
      if(MenuVerticalCentered)ClcTp();
      PosMenu(FrstCntnr,StartTop,StartLeft);
      IniFlg=1;
      Initiate();
      Creatd=1;
      if (AcrssFrms){
            ScLdAgainWin=ExpYes?ScLoc.document.body:ScLoc;
            ScLdAgainWin.onunload=UnLoaded}
      Trigger.onresize=Nav4?ReDoWhole:RePos;
      AfterBuild();
      if(MenuVerticalCentered=='static'&&!AcrssFrms)setInterval('KeepPo
s()',250);
      status='Menu ready for use'}

function KeepPos(){
      var
TS=ExpYes?SCmplnt?ScHtml.scrollTop:FrstLoc.document.body.scrollTop:Frst
Loc.pageYOffset;
      if(TS!=StaticPos){
            var FCStyle=Nav4?FrstCntnr:FrstCntnr.style;
            FrstCntnr.OrgTop=StartTop+TS;StaticPos=TS;
            FCStyle.top=FrstCntnr.OrgTop+P_X}}

function ClcJus(){
      var a=BorderBtwnElmnts?1:2,b=BorderBtwnElmnts?BorderWidth:0;
      var Size=Math.round(((FrstWinWdth-
a*BorderWidth)/NoOffFirstLineMenus)-b),i,j;
      for(i=1;i<NoOffFirstLineMenus+1;i++){j=eval('Menu'+i);j[5]=Size}
      StartLeft=0}

function ClcTrgt(){
      var
TLoc=Nav4?FrstLoc.document.layers[TargetLoc]:DomYes?FrstLoc.document.ge
tElementById(TargetLoc):FrstLoc.document.all[TargetLoc];
      StartTop=M_StrtTp;
      StartLeft=M_StrtLft;
      if(DomYes){

      while(TLoc){StartTop+=TLoc.offsetTop;StartLeft+=TLoc.offsetLeft;T
Loc=TLoc.offsetParent}}
```

```
      else{
      StartTop+=Nav4?TLoc.pageY:TLoc.offsetTop;StartLeft+=Nav4?TLoc.pag
eX:TLoc.offsetLeft}}

function ClcLft(){
      if(MenuCentered!='left'&&MenuCentered!='justify'){
            var Size=FrstWinWdth-
(!Nav4?parseInt(FrstCntnr.style.width):FrstCntnr.clip.width);
            StartLeft=M_StrtLft;
            StartLeft+=MenuCentered=='right'?Size:Size/2}}

function ClcTp(){
      if(MenuVerticalCentered!='top'&&MenuVerticalCentered!='static'){
            var Size=FrstWinHght-
(!Nav4?parseInt(FrstCntnr.style.height):FrstCntnr.clip.height);
            StartTop=M_StrtTp;
            StartTop+=MenuVerticalCentered=='bottom'?Size:Size/2}}

function PosMenu(CntnrPntr,Tp,Lt){
      RcrsLvl++;
      var Cmplnt=RcrsLvl==1?FCmplnt:SCmplnt;
      var LftXtra=(DomNav&&!Opr)||Cmplnt?LeftPaddng:0;
      var TpXtra=(DomNav&&!Opr)||Cmplnt?TopPaddng:0;
      var Topi,Lefti,Hori;
      var Cntnr=CntnrPntr;
      var Mmbr=Cntnr.FrstMbr;
      var CntnrStyle=!Nav4?Cntnr.style:Cntnr;
      var MmbrStyle=!Nav4?Mmbr.style:Mmbr;
      var PadL=Mmbr.value.indexOf('<')==-1?LftXtra:0;
      var PadT=Mmbr.value.indexOf('<')==-1?TpXtra:0;
      var
MmbrWt=!Nav4?parseInt(MmbrStyle.width)+PadL:MmbrStyle.clip.width;
      var
MmbrHt=!Nav4?parseInt(MmbrStyle.height)+PadT:MmbrStyle.clip.height;
      var
CntnrWt=!Nav4?parseInt(CntnrStyle.width):CntnrStyle.clip.width;
      var
CntnrHt=!Nav4?parseInt(CntnrStyle.height):CntnrStyle.clip.height;
      var SubTp,SubLt;
      if (RcrsLvl==1 && AcrssFrms)!MenuFramesVertical?Tp=FrstWinHght-
CntnrHt+(Nav4?4:0):Lt=RightToLeft?0:FrstWinWdth-CntnrWt+(Nav4?4:0);
      if (RcrsLvl==2 &&
AcrssFrms)!MenuFramesVertical?Tp=0:Lt=RightToLeft?ScWinWdth-CntnrWt:0;
      if (RcrsLvl==2 && AcrssFrms){Tp+=VerCorrect;Lt+=HorCorrect}
      CntnrStyle.top=RcrsLvl==1?Tp+P_X:0;
      Cntnr.OrgTop=Tp;
      CntnrStyle.left=RcrsLvl==1?Lt+P_X:0;
      Cntnr.OrgLeft=Lt;
      if (RcrsLvl==1 && FirstLineHorizontal){
            Hori=1;Lefti=CntnrWt-MmbrWt-2*BorderWidth;Topi=0}
      else{ Hori=Lefti=0;Topi=CntnrHt-MmbrHt-2*BorderWidth}
      while(Mmbr!=null){
            MmbrStyle.left=Lefti+BorderWidth+P_X;
            MmbrStyle.top=Topi+BorderWidth+P_X;

      if(Nav4)Mmbr.CmdLyr.moveTo(Lefti+BorderWidth,Topi+BorderWidth);
```

```
            if(Mmbr.ChildCntnr){

      if(RightToLeft)ChldCntnrWdth=Nav4?Mmbr.ChildCntnr.clip.width:pars
eInt(Mmbr.ChildCntnr.style.width);
                  if(Hori){   SubTp=Topi+MmbrHt+2*BorderWidth;
                        SubLt=RightToLeft?Lefti+MmbrWt-
ChldCntnrWdth:Lefti}
                  else{ SubLt=RightToLeft?Lefti-
ChldCntnrWdth+ChildOverlap*MmbrWt+BorderWidth:Lefti+(1-
ChildOverlap)*MmbrWt+BorderWidth;

      SubTp=RcrsLvl==1&&AcrssFrms?Topi:Topi+ChildVerticalOverlap*MmbrHt
}
                  PosMenu(Mmbr.ChildCntnr,SubTp,SubLt)}
            Mmbr=Mmbr.PrvMbr;
            if(Mmbr){   MmbrStyle=!Nav4?Mmbr.style:Mmbr;
                  PadL=Mmbr.value.indexOf('<')==-1?LftXtra:0;
                  PadT=Mmbr.value.indexOf('<')==-1?TpXtra:0;

      MmbrWt=!Nav4?parseInt(MmbrStyle.width)+PadL:MmbrStyle.clip.width;

      MmbrHt=!Nav4?parseInt(MmbrStyle.height)+PadT:MmbrStyle.clip.heigh
t;
                  Hori?Lefti-
=BorderBtwnElmnts?(MmbrWt+BorderWidth):(MmbrWt):Topi-
=BorderBtwnElmnts?(MmbrHt+BorderWidth):(MmbrHt)}}
      RcrsLvl--}

function Initiate(){
      if(IniFlg){ Init(FrstCntnr);IniFlg=0;
            if(ShwFlg)AfterCloseAll();ShwFlg=0}}

function Init(CntnrPntr){
      var Mmbr=CntnrPntr.FrstMbr;
      var MCStyle=Nav4?CntnrPntr:CntnrPntr.style;
      RcrsLvl++;
      MCStyle.visibility=RcrsLvl==1?M_Show:M_Hide;
      while(Mmbr!=null){
            if(Mmbr.Hilite){Mmbr.Hilite=0;if(KeepHilite)LowItem(Mmbr)}
            if(Mmbr.ChildCntnr) Init(Mmbr.ChildCntnr);
            Mmbr=Mmbr.PrvMbr}
      RcrsLvl--}

function ClearAllChilds(Pntr){
      var CPCCStyle;
      while (Pntr){
            if(Pntr.Hilite){
                  Pntr.Hilite=0;
                  if(KeepHilite)LowItem(Pntr);
                  if(Pntr.ChildCntnr){

      CPCCStyle=Nav4?Pntr.ChildCntnr:Pntr.ChildCntnr.style;
                        CPCCStyle.visibility=M_Hide;
                        ClearAllChilds(Pntr.ChildCntnr.FrstMbr)}
                  break}
            Pntr=Pntr.PrvMbr}}
```

```
function GoTo(){
      if(this.LinkTxt){
            status='';
            var HP=Nav4?this.LowLyr:this;
            LowItem(HP);
            this.LinkTxt.indexOf('javascript:')!=-
1?eval(this.LinkTxt):DcLoc.location.href=this.LinkTxt}}

function HiliteItem(P){
      if(Nav4){
            if(P.ro)P.document.images[P.rid].src=P.ri2;
            else{ if(P.HiBck)P.bgColor=P.HiBck;
                  if(P.value.indexOf('<img')==-1){
                        P.document.write(P.Ovalue);
                        P.document.close()}}}
      else{ if(P.ro){   var Lc=P.Level==1?FrstLoc:ScLoc;
                  Lc.document.images[P.rid].src=P.ri2}
            else{ if(P.HiBck)P.style.backgroundColor=P.HiBck;
                  if(P.HiFntClr)P.style.color=P.HiFntClr}}
      P.Hilite=1}

function LowItem(P){
      if(P.ro){   if(Nav4)P.document.images[P.rid].src=P.ri1;
            else{ var Lc=P.Level==1?FrstLoc:ScLoc;
                  Lc.document.images[P.rid].src=P.ri1}}
      else{ if(Nav4){   if(P.LoBck)P.bgColor=P.LoBck;
                  if(P.value.indexOf('<img')==-1){
                        P.document.write(P.value);
                        P.document.close()}}
            else{ if(P.LoBck)P.style.backgroundColor=P.LoBck;
                  if(P.LwFntClr)P.style.color=P.LwFntClr}}}

function OpenMenu(){
      if(!Loadd||!Creatd) return;
      var
TpScrlld=ExpYes?SCmplnt?ScHtml.scrollTop:ScLoc.document.body.scrollTop:
ScLoc.pageYOffset;
      var
LScrlld=ExpYes?SCmplnt?ScHtml.scrollLeft:ScLoc.document.body.scrollLeft
:ScLoc.pageXOffset;
      var CCnt=Nav4?this.LowLyr.ChildCntnr:this.ChildCntnr;
      var ThisHt=Nav4?this.clip.height:parseInt(this.style.height);
      var ThisWt=Nav4?this.clip.width:parseInt(this.style.width);
      var
ThisLft=AcrssFrms&&this.Level==1&&!FirstLineHorizontal?0:Nav4?this.Cont
ainer.left:parseInt(this.Container.style.left);
      var
ThisTp=AcrssFrms&&this.Level==1&&FirstLineHorizontal?0:Nav4?this.Contai
ner.top:parseInt(this.Container.style.top);
      var HP=Nav4?this.LowLyr:this;
      CurrntOvr=this;
      IniFlg=0;
      ClearAllChilds(this.Container.FrstMbr);
      HiliteItem(HP);
      if(CCnt!=null){
```

```
            if(!ShwFlg){ShwFlg=1;   BeforeFirstOpen()}
            var
CCW=Nav4?this.LowLyr.ChildCntnr.clip.width:parseInt(this.ChildCntnr.sty
le.width);
            var
CCH=Nav4?this.LowLyr.ChildCntnr.clip.height:parseInt(this.ChildCntnr.st
yle.height);
            var
ChCntTL=Nav4?this.LowLyr.ChildCntnr:this.ChildCntnr.style;
            var
SubLt=AcrssFrms&&this.Level==1?CCnt.OrgLeft+ThisLft+LScrlld:CCnt.OrgLef
t+ThisLft;
            var
SubTp=AcrssFrms&&this.Level==1?CCnt.OrgTop+ThisTp+TpScrlld:CCnt.OrgTop+
ThisTp;
            if(MenuWrap){
                  if(RightToLeft){

      if(SubLt<LScrlld)SubLt=this.Level==1?LScrlld:SubLt+(CCW+(1-
2*ChildOverlap)*ThisWt);

      if(SubLt+CCW>ScWinWdth+LScrlld)SubLt=ScWinWdth+LScrlld-CCW}
                  else{
      if(SubLt+CCW>ScWinWdth+LScrlld)SubLt=this.Level==1?ScWinWdth+LScr
lld-CCW:SubLt-(CCW+(1-2*ChildOverlap)*ThisWt);
                        if(SubLt<LScrlld)SubLt=LScrlld}

      if(SubTp+CCH>TpScrlld+ScWinHght)SubTp=this.Level==1?SubTp=TpScrll
d+ScWinHght-CCH:SubTp-CCH+(1-2*ChildVerticalOverlap)*ThisHt;
                  if(SubTp<TpScrlld)SubTp=TpScrlld}

      ChCntTL.top=SubTp+P_X;ChCntTL.left=SubLt+P_X;ChCntTL.visibility=M
_Show}
      status=this.LinkTxt}

function OpenMenuClick(){
      if(!Loadd||!Creatd) return;
      var HP=Nav4?this.LowLyr:this;
      CurrntOvr=this;
      IniFlg=0;
      ClearAllChilds(this.Container.FrstMbr);
      HiliteItem(HP);
      status=this.LinkTxt}

function CloseMenu(){
      if(!Loadd||!Creatd) return;
      if(!KeepHilite){
            var HP=Nav4?this.LowLyr:this;
            LowItem(HP)}
      status='';
      if(this==CurrntOvr){
            IniFlg=1;
            if(CloseTmr)clearTimeout(CloseTmr);
            CloseTmr=setTimeout('Initiate(CurrntOvr)',DissapearDelay)}}

function CntnrSetUp(Wdth,Hght,NoOff){
```

```
      var x=RcrsLvl==1?BorderColor:BorderSubColor;
      this.FrstMbr=null;
      this.OrgLeft=this.OrgTop=0;
      if(x)this.bgColor=x;
      if(Nav4){   this.visibility='hide';
            this.resizeTo(Wdth,Hght)}
      else{ if(x)this.style.backgroundColor=x;
            this.style.width=Wdth+P_X;
            this.style.height=Hght+P_X;
            this.style.fontFamily=FontFamily;
            this.style.fontWeight=FontBold?'bold':'normal';
            this.style.fontStyle=FontItalic?'italic':'normal';
            this.style.fontSize=FontSize+'pt';
            this.style.zIndex=RcrsLvl+Ztop}}

function MbrSetUp(MmbrCntnr,PrMmbr,WhatMenu,Wdth,Hght){
      var Location=RcrsLvl==1?FrstLoc:ScLoc;
      var MemVal=eval(WhatMenu+'[0]');
      var t,T,L,W,H,S;
      var a,b,c,d;
      var Cmplnt=RcrsLvl==1?FCmplnt:SCmplnt;
      var LftXtra=(DomNav&&!Opr)||Cmplnt?LeftPaddng:0;
      var TpXtra=(DomNav&&!Opr)||Cmplnt?TopPaddng:0;
      this.PrvMbr=PrMmbr;
      this.Level=RcrsLvl;
      this.LinkTxt=eval(WhatMenu+'[1]');
      this.Container=MmbrCntnr;
      this.ChildCntnr=null;
      this.Hilite=0;
      this.style.overflow='hidden';
      this.style.cursor=ExpYes&&(this.LinkTxt||(RcrsLvl==1&&UnfoldsOnCl
ick))?'hand':'default';
      this.ro=0;
      if(MemVal.indexOf('rollover')!=-1){
            this.ro=1;

      this.ri1=MemVal.substring(MemVal.indexOf(':')+1,MemVal.lastIndexO
f(':'));

      this.ri2=MemVal.substring(MemVal.lastIndexOf(':')+1,MemVal.length
);
            this.rid=WhatMenu+'i';
            MemVal="<img src=\""+this.ri1+"\" name=\""+this.rid+"\"
width=\""+Wdth+"\" height=\""+Hght+"\">"}
      this.value=MemVal;
      if(RcrsLvl==1){
            a=LowBgColor;
            b=HighBgColor;
            c=FontLowColor;
            d=FontHighColor}
      else{ a=LowSubBgColor;
            b=HighSubBgColor;
            c=FontSubLowColor;
            d=FontSubHighColor}
      this.LoBck=a;
      this.LwFntClr=c;
```

```
      this.HiBck=b;
      this.HiFntClr=d;
      this.style.color=this.LwFntClr;
      if(this.LoBck)this.style.backgroundColor=this.LoBck;
      this.style.textAlign=MenuTextCentered;
      if(eval(WhatMenu+'[2]'))this.style.backgroundImage="url(\'"+eval(
WhatMenu+'[2]')+"\')";
      if(MemVal.indexOf('<')==-1){
            this.style.width=Wdth-LftXtra+P_X;
            this.style.height=Hght-TpXtra+P_X;
            this.style.paddingLeft=LeftPaddng+P_X;
            this.style.paddingTop=TopPaddng+P_X}
      else{ this.style.width=Wdth+P_X;
            this.style.height=Hght+P_X}
      if(MemVal.indexOf('<')==-1&&DomYes){
            t=Location.document.createTextNode(MemVal);
            this.appendChild(t)}
      else this.innerHTML=MemVal;
      if(eval(WhatMenu+'[3]')&&ShowArrow){
            a=RcrsLvl==1&&FirstLineHorizontal?3:RightToLeft?6:0;
            S=Arrws[a];
            W=Arrws[a+1];
            H=Arrws[a+2];
            T=RcrsLvl==1&&FirstLineHorizontal?Hght-H-2:(Hght-H)/2;
            L=RightToLeft?2:Wdth-W-2;
            if(DomYes){

                  t=Location.document.createElement('img');
                  this.appendChild(t);
                  t.style.position='absolute';
                  t.src=S;

                  t.style.width=W+P_X;
                  t.style.height=H+P_X;
                  t.style.top=T+P_X;
                  t.style.left=L+P_X}
            else{ MemVal+="<div style='position:absolute; top:"+T+";
left:"+L+"; width:"+W+"; height:"+H+";visibility:inherit'><img
src='"+S+"'></div>";
                  this.innerHTML=MemVal}}
      if(ExpYes){this.onselectstart=CnclSlct;

      this.onmouseover=RcrsLvl==1&&UnfoldsOnClick?OpenMenuClick:OpenMen
u;
            this.onmouseout=CloseMenu;

      this.onclick=RcrsLvl==1&&UnfoldsOnClick&&eval(WhatMenu+'[3]')?Ope
nMenu:GoTo  }
      else{
      RcrsLvl==1&&UnfoldsOnClick?this.addEventListener('mouseover',Open
MenuClick,false):this.addEventListener('mouseover',OpenMenu,false);
            this.addEventListener('mouseout',CloseMenu,false);

      RcrsLvl==1&&UnfoldsOnClick&&eval(WhatMenu+'[3]')?this.addEventLis
tener('click',OpenMenu,false):this.addEventListener('click',GoTo,false)
}}
```

```
function NavMbrSetUp(MmbrCntnr,PrMmbr,WhatMenu,Wdth,Hght){
      var a,b,c,d;
      if(RcrsLvl==1){
            a=LowBgColor;
            b=HighBgColor;
            c=FontLowColor;
            d=FontHighColor}
      else {      a=LowSubBgColor;
            b=HighSubBgColor;
            c=FontSubLowColor;
            d=FontSubHighColor      }
      this.value=eval(WhatMenu+'[0]');
      this.ro=0;
      if(this.value.indexOf('rollover')!=-1){
            this.ro=1;

      this.ri1=this.value.substring(this.value.indexOf(':')+1,this.valu
e.lastIndexOf(':'));

      this.ri2=this.value.substring(this.value.lastIndexOf(':')+1,this.
value.length);
            this.rid=WhatMenu+'i';this.value="<img src='"+this.ri1+"'
name='"+this.rid+"'>"}
      if(LeftPaddng&&this.value.indexOf('<')==-
1&&MenuTextCentered=='left')this.value=' \;'+this.value;
      if(FontBold)this.value=this.value.bold();
      if(FontItalic)this.value=this.value.italics();
      this.Ovalue=this.value;
      this.value=this.value.fontcolor(c);
      this.Ovalue=this.Ovalue.fontcolor(d);
      this.value=CntrTxt+"<font face='"+FontFamily+"' point-
size='"+FontSize+"'>"+this.value+TxtClose;
      this.Ovalue=CntrTxt+"<font face='"+FontFamily+"' point-
size='"+FontSize+"'>"+this.Ovalue+TxtClose;
      this.LoBck=a;
      this.HiBck=b;
      this.ChildCntnr=null;
      this.PrvMbr=PrMmbr;
      this.Hilite=0;
      this.visibility='inherit';
      if(this.LoBck)this.bgColor=this.LoBck;
      this.resizeTo(Wdth,Hght);
      if(!AcrssFrms&&eval(WhatMenu+'[2]'))this.background.src=eval(What
Menu+'[2]');
      this.document.write(this.value);
      this.document.close();
      this.CmdLyr=new Layer(Wdth,MmbrCntnr);
      this.CmdLyr.Level=RcrsLvl;
      this.CmdLyr.LinkTxt=eval(WhatMenu+'[1]');
      this.CmdLyr.visibility='inherit';
      this.CmdLyr.onmouseover=RcrsLvl==1&&UnfoldsOnClick?OpenMenuClick:
OpenMenu;
      this.CmdLyr.onmouseout=CloseMenu;
      this.CmdLyr.captureEvents(Event.MOUSEUP);
```

```
      this.CmdLyr.onmouseup=RcrsLvl==1&&UnfoldsOnClick&&eval(WhatMenu+'
[3]')?OpenMenu:GoTo;
      this.CmdLyr.LowLyr=this;
      this.CmdLyr.resizeTo(Wdth,Hght);
      this.CmdLyr.Container=MmbrCntnr;
      if(eval(WhatMenu+'[3]')&&ShowArrow){
            a=RcrsLvl==1&&FirstLineHorizontal?3:RightToLeft?6:0;
            this.CmdLyr.ImgLyr=new Layer(Arrws[a+1],this.CmdLyr);
            this.CmdLyr.ImgLyr.visibility='inherit';

      this.CmdLyr.ImgLyr.top=RcrsLvl==1&&FirstLineHorizontal?Hght-
Arrws[a+2]-2:(Hght-Arrws[a+2])/2;
            this.CmdLyr.ImgLyr.left=RightToLeft?2:Wdth-Arrws[a+1]-2;
            this.CmdLyr.ImgLyr.width=Arrws[a+1];
            this.CmdLyr.ImgLyr.height=Arrws[a+2];
            ImgStr="<img src='"+Arrws[a]+"' width='"+Arrws[a+1]+"'
height='"+Arrws[a+2]+"'>";
            this.CmdLyr.ImgLyr.document.write(ImgStr);
            this.CmdLyr.ImgLyr.document.close()}}

function CreateMenuStructure(MName,NumberOf){
      RcrsLvl++;
      var i,NoOffSubs,Mbr,Wdth=0,Hght=0;
      var PrvMmbr=null;
      var WMnu=MName+'1';
      var MenuWidth=eval(WMnu+'[5]');
      var MenuHeight=eval(WMnu+'[4]');
      var Location=RcrsLvl==1?FrstLoc:ScLoc;
      if (RcrsLvl==1&&FirstLineHorizontal){
            for(i=1;i<NumberOf+1;i++){
                  WMnu=MName+eval(i);

      Wdth=eval(WMnu+'[5]')?Wdth+eval(WMnu+'[5]'):Wdth+MenuWidth}

      Wdth=BorderBtwnElmnts?Wdth+(NumberOf+1)*BorderWidth:Wdth+2*Border
Width;Hght=MenuHeight+2*BorderWidth}
      else{ for(i=1;i<NumberOf+1;i++){
                  WMnu=MName+eval(i);

      Hght=eval(WMnu+'[4]')?Hght+eval(WMnu+'[4]'):Hght+MenuHeight}

      Hght=BorderBtwnElmnts?Hght+(NumberOf+1)*BorderWidth:Hght+2*Border
Width;Wdth=MenuWidth+2*BorderWidth}
      if(DomYes){
            var MmbrCntnr=Location.document.createElement("div");
            MmbrCntnr.style.position='absolute';
            MmbrCntnr.style.visibility='hidden';
            Location.document.body.appendChild(MmbrCntnr)}
      else{ if(Nav4) var MmbrCntnr=new Layer(Wdth,Location)
            else{ WMnu+='c';

      Location.document.body.insertAdjacentHTML("AfterBegin","<div
id='"+WMnu+"' style='visibility:hidden; position:absolute;'><\/div>");
                  var MmbrCntnr=Location.document.all[WMnu]}}
      MmbrCntnr.SetUp=CntnrSetUp;
      MmbrCntnr.SetUp(Wdth,Hght,NumberOf);
```

```
      if(Exp4){   MmbrCntnr.InnerString='';
            for(i=1;i<NumberOf+1;i++){
                  WMnu=MName+eval(i);
                  MmbrCntnr.InnerString+="<div id='"+WMnu+"'
style='position:absolute;'><\/div>"}
            MmbrCntnr.innerHTML=MmbrCntnr.InnerString}
      for(i=1;i<NumberOf+1;i++){
            WMnu=MName+eval(i);
            NoOffSubs=eval(WMnu+'[3]');

      Wdth=RcrsLvl==1&&FirstLineHorizontal?eval(WMnu+'[5]')?eval(WMnu+'
[5]'):MenuWidth:MenuWidth;

      Hght=RcrsLvl==1&&FirstLineHorizontal?MenuHeight:eval(WMnu+'[4]')?
eval(WMnu+'[4]'):MenuHeight;
            if(DomYes){
                  Mbr=Location.document.createElement("div");
                  Mbr.style.position='absolute';
                  Mbr.style.visibility='inherit';
                  MmbrCntnr.appendChild(Mbr)}
            else Mbr=Nav4?new
Layer(Wdth,MmbrCntnr):Location.document.all[WMnu];
            Mbr.SetUp=Nav4?NavMbrSetUp:MbrSetUp;
            Mbr.SetUp(MmbrCntnr,PrvMmbr,WMnu,Wdth,Hght);
            if(NoOffSubs)
Mbr.ChildCntnr=CreateMenuStructure(WMnu+'_',NoOffSubs);
            PrvMmbr=Mbr}
      MmbrCntnr.FrstMbr=Mbr;
      RcrsLvl--;
      return(MmbrCntnr)}

function CreateMenuStructureAgain(MName,NumberOf){
      var i,WMnu,NoOffSubs,PrvMmbr,Mbr=FrstCntnr.FrstMbr;
      RcrsLvl++;
      for(i=NumberOf;i>0;i--){
            WMnu=MName+eval(i);
            NoOffSubs=eval(WMnu+'[3]');
            PrvMmbr=Mbr;

      if(NoOffSubs)Mbr.ChildCntnr=CreateMenuStructure(WMnu+'_',NoOffSub
s);
            Mbr=Mbr.PrvMbr}
      RcrsLvl--}
```

A.21 “myAccount.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the My Accounts Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("My Account",$_SESSION['validUser']);
    displayMenu();
    addHeading("My Account");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

?>
      You have the following rights:
      <ul>
        <li>
      <?php echo $_SESSION['userType']; ?>
        </li>
      </ul>
      <br />
      <br />
<?php
      $dbCon = connectToDB();

    if (isset($_POST['select']))
    {
      $countryID = $_POST['countryID'];
        $orgID = $_POST['orgID'];

        if ($countryID != "-1" && $orgID != "-1")
        {
          $_SESSION['countryID'] = $countryID;
            $_SESSION['orgID'] = $orgID;

          echo 'Country Selected: <b>'.getName($dbCon,
$_SESSION['countryID'], "Country").'</b><br />';
          echo 'Organisation Selected: <b>'.getName($dbCon,
$_SESSION['orgID'], "Organisation").'</b><br /><br />';
        }
        else
        {
          echo 'One or More fields were left blank.<br />';
          echo 'Please try again!<br /><br />';
```

```
        }//end if-else
    }//end if

      displayCountryOrgSelectForm($dbCon);

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.22 “outputFuncs.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains the General output functions for the POLARIS Web
App

//---------------------------------------------------------------------
//Function to add a Basic ComboBox
function addBasicComboBox($dbCon, $label, $varName, $table, $tableID,
$tableData, $errorMsg, $defaultVal="")
{
  $sql = "SELECT *
          FROM $table
              ORDER BY $tableData ";

  $resultSet = runQuery($dbCon, $sql, $errorMsg);
?>
  <tr>
      <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue">
        <select class="border" name="<?php echo $varName; ?>">
            <option value="-1">&lt Select <?php echo $label; ?>
&gt</option>
        <?php
            for ($i = 0; $i < $resultSet->num_rows; $i++)
          {
              $row = $resultSet->fetch_assoc();
            echo '  <option value="'.$row[$tableID].'"';
                  if ($defaultVal != "" && $defaultVal ==
$row[$tableID])
                    echo ' selected="selected" ';
                  echo '>'.$row[$tableData].'</option>';
          }//end for
        ?>
        </select>
      </td>
  </tr>
<?php
}//end addBasicComboBox Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to add a Button
function addButton($name, $colSpan, $id="")
{
?>
  <tr>
      <th class="blue" colspan=<?php echo $colSpan; ?>><input
type="submit" value="      <?php echo $name; ?>      "
      <?php
```

```
      if ($id != "")
        echo ' name="'.$id.'" ';
      ?>
      /></th>
  </tr>
<?php
}//end addButtonFunction
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to add the Data as a list of Checkboxes
function addCheckboxList($dbCon, $label, $varName, $table, $tableID,
$tableData, $errorMsg, $checkedArray="")
{
  $sql = "SELECT *
          FROM $table ";

  $resultSet = runQuery($dbCon, $sql, $errorMsg);
?>
  <tr>
      <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue">
    <?php
        for ($i = 0; $i < $resultSet->num_rows; $i++)
        {
          $row = $resultSet->fetch_assoc();

            echo '<input type="checkbox"
name="'.$varName.$row[$tableID].'"';
            if ($checkedArray != "" && isset($checkedArray[($i+1)]))
              echo ' checked="checked" ';
            echo '>'.$row[$tableData].'</input><br />';
        }//end for
      ?>
      </td>
  </tr>
<?php
}//end addCheckboxList
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function that Adds the HTML Footer
function addFooter()
{//Adds the HTML Footer
?>
  </body>
  </html>
<?php
}//end addFooter Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function that Adds the HTML Header
function addHeader($title,$loggedInUser)
{
  //VERSION 1.0.0
```

```
  define("VERSION", "1.0");
?>
  <html>
  <head>
      <title>POLARIS <?php echo VERSION; ?>
        <?php
          if ($title != '')
            echo ' - '.$title;
        ?>
      </title>
      <style type="text/css">
        <?php
            require_once('general.css');
            require_once('style.css');
        ?>
      </style>
  </head>
<body marginheight="0" marginwidth="0" leftmargin="0" topmargin="0">
<BR>
<table cellpadding="0" cellspacing="0" border="0" width="720"
align="center">
<tr>
            <?php
              if ($title != '' and $title != 'Logging Out') { ?>
      <td>You are Logged in as <b><?php echo $loggedInUser?></b></td>
              <?php } ?>
      <td align="right"><!--<object classid="clsid:D27CDB6E-AE6D-11cf-
96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swfla
sh.cab#version=7,0,19,0" width="209" height="50">
      <param name="movie" value="images/logo.swf">
      <param name="quality" value="high">
      <embed src="images/logo.swf" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" width="209" height="50"></embed>
        </object>--><img src="images/logo.gif" width="209"
height="50"><img src="images/left_logo.gif" width="66"
height="50"></td>
</tr>
</table>

<?php
}//end addHeader Function
//--------------------------------------------------------------------

//--------------------------------------------------------------------
//Function that Adds the Page Heading
function addHeading($heading)
{
?>
  <div id="heading">
    <table align="center">
        <tr>
          <td class="head"><?php echo $heading; ?></td>
            <?php/*
            if ($helpLink != "logout")
```

```
              echo '<td class="helpHead"><a
href="help.php#'.$helpLink.'" target="_blank">Help</a></td>';
            */?>
          </tr>
        </table>
  </div>
<?php
}//end addHeading Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to add a Password box
function addPwdBox($label, $varName, $txtContents="")
{
?>
  <tr>
    <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue"><input class="border" type="password"
name="<?php echo $varName; ?>" value="<?php echo $txtContents; ?>"
size=50 /></td>
  </tr>
<?php
}//end addTextBox function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to add a Text box
function addTextBox($label, $varName, $txtContents="")
{
?>
  <tr>
    <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue"><input class="border" type="text"
name="<?php echo $varName; ?>" value="<?php echo $txtContents; ?>"
size=50
      <?php
      if ($varName == "userName" && $txtContents ==
$_SESSION['validUser'])
        echo ' disabled="disabled" ';
      ?>
      /></td>
  </tr>
<?php
}//end addTextBox function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function that Adds URLs
function addURL($url, $name)
{
?>
  <br /><a href="<?php echo $url;?>"><?php echo $name;?></a><br />
<?php
}//end addURL Function
//---------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
//Adds a Combo Box for Years
function addYearsBox($yr)
{
?>
  <tr>
    <td class="blue">Year:</td>
    <td class="lightblue">
        <select class="border" name="yr">
            <option value="-1">&lt Select <?php echo $label; ?>
&gt</option>
        <?php
            for ($i = 2100; $i >= 1950; $i--)
          {
              echo '  <option value="'.$i.'"';
                  if ($yr != "" && $i == $yr)
                    echo ' selected="selected" ';
                  echo '>'.$i.'</option>';
          }//end for
        ?>
        </select>
      </td>
  </tr>
<?php
}//end addYearsBox Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays a basic Add form with which the user adds a new data in the
system
function displayBasicAddForm($actionForm, $txtBoxLbl, $txtBoxVar)
{
?>
  <form method="POST" action="<?php echo $actionForm; ?>">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
<?php
    addTextBox($txtBoxLbl, $txtBoxVar);
      addButton("Add", "2", "add");
?>
  </table>
  </form>
<?php
}//end displayBasicAddForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to display a Basic Talble
function displayBasicTable($dbCon, $label, $table, $field,
$emptyTableMsg, $errorMsg)
{
  $sql = "SELECT $field
          FROM $table
              ORDER by $field";

  $resultSet = runQuery($dbCon, $sql, $errorMsg);
```

```
  if ($resultSet->num_rows > 0)
  {
?>
     
      <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr>
          <th class="blue"><?php echo $label; ?></th>
        </tr>
<?php
    for ($i = 0; $i < $resultSet->num_rows; $i++)
    {
        $row = $resultSet->fetch_assoc();

        if ($i % 2 == 0)
          $styleClass = 'lightblue';
        else
          $styleClass = 'lightblue2';

      echo '<tr>';
        echo '  <td class="'.$styleClass.'">'.$row[$field].'</td>';
        echo '</tr>';
    }//end for
?>
    </table>
<?php
  }
  else
  {
      echo $emptyTableMsg;
  }//end if-else
}//end displayBasicTable Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the Form from which the user selects the Country
//and Organisation the he wants to test
function displayCountryOrgSelectForm($dbCon)
{
?>
  <form method="POST" action="myAccount.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
    addBasicComboBox($dbCon, "Country", "countryID", "Countries",
"CountryID", "CountryName", "Could not Retrieve Country Names form the
Database.", $_SESSION['countryID']);
      addBasicComboBox($dbCon, "Organisation", "orgID",
"Organisations", "OrgID", "OrgName", "Could not Retrieve Organisation
Names form the Database.", $_SESSION['orgID']);
      addButton("Select", "2", "select");
  ?>
  </table>
  </form>
<?php
}//end displayCountryOrgSelectForm Function
//---------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
//Function to display a basic Edit Form
function displayEditForm($dbCon, $label, $varName, $table, $tableID,
$tableData, $errorMsg, $actionForm, $selected="")
{
?>
  <form method="POST" action="<?php echo $actionForm; ?>">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
    addBasicComboBox($dbCon, $label, $varName, $table, $tableID,
$tableData, $errorMsg, $selected);
      addButton("Edit", "2", "edit");
  ?>
  </table>
  </form>
<?php
}//end displayEditForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the Login Form
function displayLoginForm()
{
?>
  <form method="POST" action="processLogin.php">
    <br><br>
      <table border=0 cellpadding="3" cellspacing="0" align="center">
      <tr>
        <td class="blue">Username:</td>
          <td class="lightblue"><input type="text" name="username"
size="20" /></td>
      </tr>
      <tr>
        <td class="blue">Password:</td>
          <td class="lightblue"><input type="password" name="pwd"
size="20" /></td>
      </tr>
      <tr>
        <th colspan=2 class="blue"><input type="submit" value="Log In"
/></th>
      </tr>
    </table>
  </form>
<?php
}//end DisplaysLoginForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to Display the Top Menu of the Web App
function displayMenu()
{

?>

<script type='text/javascript'>
```

```
//HV Menu- by Ger Versluis (http://www.burmees.nl/)
//Submitted to Dynamic Drive (http://www.dynamicdrive.com)
//Visit http://www.dynamicdrive.com for this script and more

function Go(){return}

</script>
<script type='text/javascript' src='js/exmplmenu_var.js'></script>
<script type='text/javascript' src='js/menu_com.js'></script>
<br><Br>
<table cellpadding="5" cellspacing="0" border="0" bordercolor="#000000"
align="center">
<tr><Td>
<br><br>
<?php
//  require_once('topMenu.php');
}//end displayMenu Function
//--------------------------------------------------------------------
?>
```

A.23 “outputFuncsAdmin.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains the Admin output functions for the Developer Web
App

//---------------------------------------------------------------------
//Function to add the Subcategory ComboBox
function addSubCatCombo($dbCon, $subCatID="")
{
  $sql = "SELECT s.SubcatID as sID, s.SubcatName as sn, c.CatName as cn
          FROM Subcategories as s, Categories as c
              WHERE c.CatID = s.CatID
              ORDER BY s.SubCatName ";

  $resultSet = runQuery($dbCon, $sql, "Could not Retrieve Subcategries
from the Database.");
?>
  <tr>
      <td class="blue">Subcategory:</td>
      <td class="lightblue">
        <select class="border" name="subCatID">
            <option value="-1">&lt Select Subcategory &gt</option>
        <?php
            for ($i = 0; $i < $resultSet->num_rows; $i++)
          {
              $row = $resultSet->fetch_assoc();
            echo '  <option value="'.$row['sID'].'"';
                  if ($subCatID != "" && $subCatID == $row['sID'])
                    echo ' selected="selected" ';
                  echo '>'.$row['sn'].' - '.$row['cn'].'</option>';
          }//end for
        ?>
        </select>
      </td>
  </tr>
<?php
}//end addSubCatCombo Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Display the Add Issue Form
function displayAddIssueForm($dbCon, $issueName, $subCatID)
{
?>
  <form method="POST" action="addIssue.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
<?php
      addTextBox("Issue Name", "issueName", $issueName);
      addSubCatCombo($dbCon, $subCatID);
```

```
    addButton("Add", "2", "add");
?>
  </table>
  </form>
<?php
}//end displayAddIssueForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the Add Form for a new Subcategory
function displayAddSubCatForm($dbCon, $subCatName, $catID)
{
?>
  <form method="POST" action="addSubCat.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
<?php
      addTextBox("Subcategory Name", "subCatName", $subCatName);
      addBasicComboBox($dbCon, "Category Name", "catID", "Categories",
"CatID", "CatName", "Cound not Retrieve Category Name from the
Database.", $catID);
    addButton("Add", "2", "add");
?>
  </table>
  </form>
<?php
}//end displayAddSubCatForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the Add Form for a new Organisation
function displayAddOrgForm($dbCon, $orgName, $countryID)
{
?>
  <form method="POST" action="addOrg.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
<?php
      addTextBox("Organisation Name", "orgName", $orgName);
      addBasicComboBox($dbCon, "Country Name", "countryID",
"Countries", "CountryID", "CountryName", "Cound not Retrieve Country
Names from the Database.", $countryID);
    addButton("Add", "2", "add");
?>
  </table>
  </form>
<?php
}//end displayAddOrgForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the form that will allow the user to edit the data
//of either a Country or an Organisation
function displayEditDataForm($dbCon, $sourceID, $yr, $actionForm,
$source)
{
  if ($source == "Country")
  {
```

```
    $dataTable = "CountryData";
      $dataTableID = "CountryDataID";
      $dataTableDet = "CountryDataName";

      $detTableName = "CountryDataDet";

      $sourceIDVarName = "countryID";
  }
  elseif ($source == "Organisation")
  {
    $dataTable = "OrgData";
      $dataTableID = "OrgDataID";
      $dataTableDet = "OrgDataName";

      $detTableName = "OrgDataDet";

      $sourceIDVarName = "orgID";
  }//end if-else

  $sql = "SELECT *
          FROM $dataTable
              ORDER BY $dataTableDet ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve '.$source.'
Details from the Database.');

  if ($resultSet->num_rows > 0)
  {
  ?>
    <form method="POST" action="<?php echo $actionForm; ?>">
    <table border=0 cellpadding="3" cellspacing="0" align="center">
      <?php
        for ($i = 0; $i < $resultSet->num_rows; $i++)
        {
          $row = $resultSet->fetch_assoc();

          addTextBox($row[$dataTableDet], $row[$dataTableID],
getDataDetails($dbCon, $row[$dataTableID], $sourceID, $yr, $source));
        }//end for
        addButton("Save", "2", "save");
        echo '<input type="hidden" name="'.$sourceIDVarName.'"
value="'.$sourceID.'" />';
        echo '<input type="hidden" name="yr" value="'.$yr.'" />';
      ?>
      </table>
      </form>
  <?php
  }
  else
  {
    echo 'No '.$source.' Data have been entered into the system.<br
/>';
      echo 'Please enter the Data first!';
      if ($source == "Country")
        addURL('addCountryData.php', 'Add Country Data');
      else if ($source == "Organisation")
```

```
        addURL('addOrgData.php', 'Add Organisation Data');
  }//end if-else
}//end displayEditDataForm Function
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
//Function to display Edit Country/Org Form
function displayEditCountryOrgForm($dbCon, $label, $varName, $table,
$tableID, $tableData, $errorMsg, $actionForm, $yr, $selected="")
{
  if ($yr == '')
    $yr = date("Y");
?>
  <form method="POST" action="<?php echo $actionForm; ?>">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
    addBasicComboBox($dbCon, $label, $varName, $table, $tableID,
$tableData, $errorMsg, $selected);
      addYearsBox($yr);
      addButton("Edit", "2", "edit");
  ?>
  </table>
  </form>
<?php
}//end displayEditForm Function
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
//Function to display the Issues Talble
function displayIssuesTable($dbCon)
{
  $sql = "SELECT i.IssueName as iName, s.SubcatName as sn, c.CatName as
cn
          FROM Issues as i, Subcategories as s, Categories as c
              WHERE c.CatID = s.CatID
              AND s.SubcatID = i.SubcatID
              ORDER by c.CatName, s.SubCatName, i.IssueName ";

  $resultSet = runQuery($dbCon, $sql, "Could not Retrieve Issue Names
from the Database.");

  if ($resultSet->num_rows > 0)
  {
?>
     
      <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr align="center">
            <th class="blue">Issue</th>
          <th class="blue">Subcategory</th>
            <th class="blue">Category</th>
        </tr>
<?php
    for ($i = 0; $i < $resultSet->num_rows; $i++)
    {
        $row = $resultSet->fetch_assoc();
```

```
        if ($i % 2 == 0)
          $styleClass = 'lightblue';
        else
          $styleClass = 'lightblue2';

      echo '<tr align="center">';
        echo '  <td class="'.$styleClass.'">'.$row['iName'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['sn'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['cn'].'</td>';
        echo '</tr>';
    }//end for
?>
    </table>
<?php
  }
  else
  {
      echo 'There are No Subcategories in the Database.';
  }//end if-else
}//end displayIssuesTable Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to display the Organisations Talble
function displayOrgTable($dbCon)
{
  $sql = "SELECT o.OrgName as orgN, c.CountryName as cn
          FROM Organisations as o, Countries as c
              WHERE c.CountryID = o.CountryID
              ORDER by o.OrgName ";

  $resultSet = runQuery($dbCon, $sql, "Could not Retrieve Organisation
Names from the Database.");

  if ($resultSet->num_rows > 0)
  {
?>
     
      <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr align="center">
          <th class="blue">Organisation</th>
            <th class="blue">Country</th>
        </tr>
<?php
    for ($i = 0; $i < $resultSet->num_rows; $i++)
    {
        $row = $resultSet->fetch_assoc();

        if ($i % 2 == 0)
          $styleClass = 'lightblue';
        else
          $styleClass = 'lightblue2';

      echo '<tr align="center">';
        echo '  <td class="'.$styleClass.'">'.$row['orgN'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['cn'].'</td>';
```

```
        echo '</tr>';
    }//end for
?>
    </table>
<?php
  }
  else
  {
      echo 'There are No Subcategories in the Database.';
  }//end if-else
}//end displayOrgTable Function
//------------------------------------------------------------------------

//------------------------------------------------------------------------
//Function to display the Subcategories Talble
function displaySubCatTable($dbCon)
{
  $sql = "SELECT s.SubcatName as sn, c.CatName as cn
          FROM Subcategories as s, Categories as c
              WHERE c.CatID = s.CatID
              ORDER by c.CatName, s.SubCatName ";

  $resultSet = runQuery($dbCon, $sql, "Could not Retrieve Subcategory
Names from the Database.");

  if ($resultSet->num_rows > 0)
  {
?>
     
      <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr align="center">
          <th class="blue">Subcategory</th>
            <th class="blue">Category</th>
        </tr>
<?php
    for ($i = 0; $i < $resultSet->num_rows; $i++)
    {
        $row = $resultSet->fetch_assoc();

        if ($i % 2 == 0)
          $styleClass = 'lightblue';
        else
          $styleClass = 'lightblue2';

      echo '<tr align="center">';
        echo '  <td class="'.$styleClass.'">'.$row['sn'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['cn'].'</td>';
        echo '</tr>';
    }//end for
?>
    </table>
<?php
  }
  else
  {
      echo 'There are No Subcategories in the Database.';
```

```
  }//end if-else
}//end displaySubCatTable Function
//---------------------------------------------------------------------
?>
```

A.24 “outputFuncsRules.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains the Set Rules output functions for the POLARIS Web
App

//---------------------------------------------------------------------
//Function to add the Tools used for the Constraints
function addConstraintTools($label, $varName, $equal, $lessVal,
$greaterVal, $perVal, $perIncDec, $lessPerVal, $greaterPerVal,
$perIncDecLessGr)
{
?>
  <tr>
    <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue">
        <table border=1>
        <tr><td>
        <table>
          <tr>
              <td> = <input class="border" type="text" name="<?php echo
$varName.'-equal'; ?>" value="<?php echo $equal; ?>" size=10 />
              </td>
            </tr>
            <tr>
              <td> <input class="border" type="text" name="<?php echo
$varName.'-lessVal'; ?>" value="<?php echo $lessVal; ?>" size=10 />
                < x <
                  <input class="border" type="text" name="<?php echo
$varName.'-greaterVal'; ?>" value="<?php echo $greaterVal; ?>" size=10
/>
              </td>
            </tr>
            <tr>
              <td>
                <input class="border" type="text" name="<?php echo
$varName.'-perVal'; ?>" value="<?php echo $perVal; ?>" size=10 />%
                <input type="radio" name="<?php echo $varName.'-
perIncDec'; ?>" value="Inc"
                  <?php
                  if ($perIncDec == "Inc")
                    echo ' checked="checked" ';
                  ?>
                  />&uarr;
                  <input type="radio" name="<?php echo $varName.'-
perIncDec'; ?>" value="Dec"
                  <?php
                  if ($perIncDec == "Dec")
                    echo ' checked="checked" ';
                  ?>
```

```
                />&darr;
              </td>
            </tr>
            <tr>
              <td>
                <input class="border" type="text" name="<?php echo
$varName.'-lessPerVal'; ?>" value="<?php echo $lessPerVal; ?>" size=10
/>%
                < x <
                <input class="border" type="text" name="<?php echo
$varName.'-greaterPerVal'; ?>" value="<?php echo $greaterPerVal; ?>"
size=10 />%
                <input type="radio" name="<?php echo $varName.'-
perIncDecLessGr'; ?>" value="Inc"
                <?php
                if ($perIncDecLessGr == "Inc")
                  echo ' checked="checked" ';
                ?>
                />&uarr;
                <input type="radio" name="<?php echo $varName.'-
perIncDecLessGr'; ?>" value="Dec"
                <?php
                if ($perIncDecLessGr == "Dec")
                  echo ' checked="checked" ';
                ?>
                />&darr;
              </td>
            </tr>
        </table>
        </td></tr>
        </table>
      </td>
  </tr>
<?php
}//end addConstraintTools function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to add a Dependent ComboBox. I.e. its contents
//are dependednt on another peice of data
function addDependentComboBox($dbCon, $label, $varName, $table,
$tableID, $tableData, $errorMsg, $otherIDName, $otherID,
$defaultVal="")
{
  $sql = "SELECT *
          FROM $table
              WHERE $otherIDName = '$otherID'
              ORDER BY $tableData ";

  $resultSet = runQuery($dbCon, $sql, $errorMsg);
?>
  <tr>
      <td class="blue"><?php echo $label; ?>:</td>
      <td class="lightblue">
        <select class="border" name="<?php echo $varName; ?>">
```

```
            <option value="-1">&lt Select <?php echo $label; ?>
&gt</option>
        <?php
            for ($i = 0; $i < $resultSet->num_rows; $i++)
          {
              $row = $resultSet->fetch_assoc();
            echo '  <option value="'.$row[$tableID].'"';
                  if ($defaultVal != "" && $defaultVal ==
$row[$tableID])
                    echo ' selected="selected" ';
                  echo '>'.$row[$tableData].'</option>';
          }//end for
        ?>
        </select>
      </td>
  </tr>
<?php
}//end addBasicComboBox Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays all the Data (Country & Organisation) so that the user
//can select which ones he wants in the rules for the selected Issue
function displayDataSelectionForm($dbCon, $catID, $subCatID, $issueID)
{
?>
  <form method="POST" action="setRules.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
    addCheckboxList($dbCon, "Country Data", "countryData",
"CountryData", "CountryDataID", "CountryDataName", "Could not Retrive
Country Data from the Database.", getRulesArray($dbCon, $issueID,
"Country"));
  ?>
    <tr>
        <td class="blue" colspan="2"> </td>
      </tr>
  <?php
      addCheckboxList($dbCon, "Organisation Data", "orgData",
"OrgData", "OrgDataID", "OrgDataName", "Could not Retrive Organisation
Data from the Database.", getRulesArray($dbCon, $issueID,
"Organisation"));
      addButton("Save", "2", "save");
  ?>
    <input type="hidden" name="catID" value="<?php echo $catID; ?>" />
      <input type="hidden" name="subCatID" value="<?php echo $subCatID;
?>" />
      <input type="hidden" name="issueID" value="<?php echo $issueID;
?>" />
  </table>
  </form>
<?php
}//end displayDataSelectionForm Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
```

```
//Displays the form that will allow the user to enter Constraints
//for the Test
function displayEnterConstraintsForm($dbCon, $issueID, $catID,
$subCatID, $constraints, $simMar)
{
  $countryRules = getRulesArray($dbCon, $issueID, "Country");
  $orgRules = getRulesArray($dbCon, $issueID, "Organisation");

  $maxCountryDataID = getMaxID($dbCon, "CountryData", "CountryDataID");
  $maxOrgDataID = getMaxID($dbCon, "OrgData", "OrgDataID");

  if (sizeof($countryRules) > 0 || sizeof($orgRules) > 0)
  {
?>
  <form method="POST" action="runTest.php">
    <table border=0 cellpadding="3" cellspacing="0" align="center">
      <tr>
        <td class="blue" colspan=2 align="center">   </td>
      </tr>
      <tr>
        <td class="blue">Similarity Margin:</td>
        <td class="lightblue"><input class="border" type="text"
name="simMar" value="<?php echo $simMar; ?>" size=10>%</td>
      </tr>
      <?php
        if (sizeof($countryRules) > 0)
        {
          ?>
            <tr>
              <td class="blue" colspan=2 align="center">   </td>
            </tr>
            <tr>
              <td class="blue" colspan=2 align="center"> Country Data
Constraints </td>
            </tr>
            <?php
          for ($i = 1; $i <= $maxCountryDataID; $i++)
          {
            if (isset($countryRules[$i]))
              {
                addConstraintTools(getDataName($dbCon,
$countryRules[$i], "Country"), 'c-'.$countryRules[$i], $constraints['c-
'.$i.'-equal'], $constraints['c-'.$i.'-lessVal'], $constraints['c-
'.$i.'-greaterVal'], $constraints['c-'.$i.'-perVal'], $constraints['c-
'.$i.'-perIncDec'], $constraints['c-'.$i.'-lessPerVal'],
$constraints['c-'.$i.'-greaterPerVal'], $constraints['c-'.$i.'-
perIncDecLessGr']);
              }//end if
          }//end for
        }//end if

        if (sizeof($orgRules) > 0)
        {
          ?>
            <tr>
              <td class="blue" colspan=2 align="center">   </td>
```

```
            </tr>
            <tr>
              <td class="blue" colspan=2 align="center"> Organisation
Data Constraints </td>
            </tr>
            <?php
          for ($i = 1; $i <= $maxOrgDataID; $i++)
          {
            if (isset($orgRules[$i]))
              {
                addConstraintTools(getDataName($dbCon, $orgRules[$i],
"Organisation"), 'o-'.$orgRules[$i], $constraints['o-'.$i.'-equal'],
$constraints['o-'.$i.'-lessVal'], $constraints['o-'.$i.'-greaterVal'],
$constraints['o-'.$i.'-perVal'], $constraints['o-'.$i.'-perIncDec'],
$constraints['o-'.$i.'-lessPerVal'], $constraints['o-'.$i.'-
greaterPerVal'], $constraints['o-'.$i.'-perIncDecLessGr']);
              }//end if
          }//end for
        }//end if

        echo '<input type="hidden" name="issueID" value="'.$issueID.'"
/>';
        echo '<input type="hidden" name="catID" value="'.$catID.'" />';
        echo '<input type="hidden" name="subCatID"
value="'.$subCatID.'" />';

        addButton("Run", "2", "run");
      ?>
      </table>
      </form>
<?php
  }
  else
  {
    echo 'No Rules have been set for this Issue.<br />';
      echo 'Please Set the Rules first!';
      addURL('setRules.php', 'Set Rules');
  }//end if-else
}//end displayEnterConstraintsForm Function
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
//Displays the Exact matches of the Test
function displayMatches($dbCon, $sourceLet, $exactMatchesArray, $head,
$emptySetMsg)
{
  if (sizeof($exactMatchesArray) > 0)
  {
    if ($sourceLet == 'c')
    {
      $maxID = getMaxID($dbCon, "Countries", "CountryID");
        $source = "Country";
        $sourceText = "Countries";
    }
    elseif ($sourceLet == 'o')
    {
```

```
      $maxID = getMaxID($dbCon, "Organisations", "OrgID");
        $source = "Organisation";
        $sourceText = "Organisations";
    }//end if-else

    ?>
    <table border=0 cellpadding="3" cellspacing="0" align="center">
    <tr>
      <td class="blue" colspan=2 align="center"><?php echo $head;
?></td>
    </tr>
    <tr>
      <td class="blue"> </td>
      <td class="blue2"><?php echo $sourceText; ?></td>
    </tr>
    <?php
    $counter = 0;
    for ($i = 1; $i <= $maxID; $i++)
    {//A different counter is used as the "$i" is the ID. If an ID is
skipped, then
     //the 2 different rows will have the same background colour
      if ($counter % 2 == 0)
          $className = "lightblue";
        else
          $className = "lightblue2";

        if (isset($exactMatchesArray[$i]))
        {
        ?>
          <tr>
            <td class="blue"> </td>
            <td class="<?php echo $className; ?>"><?php echo
getName($dbCon, $exactMatchesArray[$i], $source); ?></td>
          </tr>
          <?php
        }//end if

        $counter++;
    }//end for
    ?>
      <tr>
        <td> </tr>
      </tr>
    </table>
    <?php
  }
  else
  {
    ?>
      <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr>
          <td> </td>
        </tr>
        <tr>
          <td align="center">
              <?php echo $emptySetMsg; ?>
```

```
            </td>
          </tr>
        </table>
        <?php
  }//end if-else
}//end displayExactMatches Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Displays the Set Rules Form
function displaySetRulesForm($dbCon, $catID, $subCatID, $issueID,
$actionForm)
{
?>
  <form method="POST" action="<?php echo $actionForm; ?>">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
     addBasicComboBox($dbCon, "Category", "catID", "Categories",
"CatID", "CatName", "Could not Retrieve Categories from the Database.",
$catID);
      addDependentComboBox($dbCon, "Subcategory", "subCatID",
"Subcategories", "SubcatID", "SubcatName", "Could not Retrieve
Subcategories from the Database.", "CatID", $catID, $subCatID);
      addDependentComboBox($dbCon, "Issue", "issueID", "Issues",
"IssueID", "IssueName", "Could not Retrieve Issues from the Database.",
"SubcatID", $subCatID, $issueID);
      addButton("Refresh", "2", "refresh");
  ?>
  </table>
  </form>
<?php
}//end displaySetRulesForm Function
//---------------------------------------------------------------------
?>
```

A.25 “outputFuncsUser.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains all the output functions that are related
//to the User Data, for the POLARIS Web App

//---------------------------------------------------------------------
//Display the Form to Add a New User
function displayAddUserForm($dbCon, $userName, $name, $surname,
$userTypes, $source, $adminEdit="")
{
  if ($source == "Add")
    $actionForm = "addUser.php";
  elseif ($source == "Edit")
    $actionForm = "editUser.php";
?>
  <form method="POST" action="<?php echo $actionForm; ?>">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
    addTextBox("Username", "userName", $userName);
      addTextBox("Name", "name", $name);
      addTextBox("Surname", "surname", $surname);

      if ($source == "Add")
      {
        addPwdBox("Enter Password", "pwd");
        addPwdBox("Confirm Password", "pwdRepeat");
        addCheckboxList($dbCon, "User Types", "userTypeID",
"UserTypes", "UserTypeID", "UserTypeDesc", "Could not Retrieve User
Types.", $userTypes);
        addButton("Add", "2", "add");
      }
      elseif ($source == "Edit")
      {
        if ($adminEdit)
        {
          addCheckboxList($dbCon, "User Types", "userTypeID",
"UserTypes", "UserTypeID", "UserTypeDesc", "Could not Retrieve User
Types.", $userTypes);

            $_SESSION['editUser'] = $userName;
        }//end if
        addButton("Update", "2", "update");
      }//end if-else
  ?>
  </table>
  </form>
<?php
}//end displayAddUserForm Function
//---------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
//Displays a List of all the Users
function displayAllUsers($dbCon)
{
  $sql = "SELECT *
          FROM Users ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Users form
the Database.');

  if ($resultSet->num_rows > 0)
  {
?>
    <table border=0 cellpadding="3" cellspacing="0" align="center">
        <tr>
          <td width="25%"> </td>
          <td width="25%"> </td>
          <td width="25%"> </td>
            <td width="25%"> </td>
        </tr>
        <tr>
          <th class="blue">Username</th>
            <th class="blue">Surname</th>
            <th class="blue">Name</th>
            <th class="blue">User Types</th>
        </tr>
<?php
      for ($i == 0; $i < $resultSet->num_rows; $i++)
      {
        if (($i % 2) == 0)
            $styleClass = "lightblue";
        else
          $styleClass = "lightblue2";

      $row = $resultSet->fetch_assoc();

        echo '<tr align="center">';
        echo '  <td class="'.$styleClass.'">'.$row['UserName'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['Surname'].'</td>';
        echo '  <td class="'.$styleClass.'">'.$row['Name'].'</td>';
        echo '  <td class="'.$styleClass.'">'.userTypesToStr($dbCon,
$row['UserName']).'</td>';
        echo '</tr>';
       }//end for
?>
    </table>
<?php
  }
  else
  {
    echo 'There are No Users in the Database.';
  }//end if-else
}//end displayAllUsers Function
//---------------------------------------------------------------------
```

```
//-----------------------------------------------------------------------
//Display the Form to change a User's Password
function displayChangePwdForm($dbCon)
{
?>
  <form method="POST" action="changePwd.php">
  <table border=0 cellpadding="3" cellspacing="0" align="center">
  <?php
      addPwdBox("Old Password", "oldPwd");
      addPwdBox("Enter Password", "pwd");
      addPwdBox("Confirm Password", "pwdRepeat");
      addButton("Save", "2", "save");
  ?>
  </table>
  </form>
<?php
}//end displayChangePwdForm Function
//-----------------------------------------------------------------------
?>
```

A.26 “polarisFuncs.php”

```
<?php
//POLARIS
//Developer: T. Scholiadis

//This file contains all the function requires for the POLARIS Web App

  //General POLARIS functions
  require_once('dbFuncs.php');
  require_once('outputFuncs.php');
  require_once('userAuthFuncs.php');

  //Rules functions
  require_once('outputFuncsRules.php');
  require_once('rulesFuncs.php');

  //User functions
  require_once('outputFuncsUser.php');
  require_once('userFuncs.php');

  //Other Admin function
  require_once('outputFuncsAdmin.php');
  require_once('adminFuncs.php');
?>
```

A.27 “processLogin.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis -->

//This is the Page that processes the login of the POLARIS Web App -->

  require("polarisFuncs.php"); //Including Function Files

  //create short varaible names
  $username = $_POST['username'];
  $pwd = $_POST['pwd'];

  //connect to DB
  $dbCon = connectToDB();

  //-----------------------------------------------------------------
-----
  //If User tried logging in
  if (isset($username) && isset($pwd))
  {
    try
      {
      login($dbCon, $username, $pwd);

        //If the user is in the DB, then register the user ID to the
session
        $_SESSION['validUser'] = $username;

        $_SESSION['userType'] = getUserType($dbCon);

        header("Location: myAccount.php");
        exit;
      }
      catch(Exception $e)
      {//Failed Login
        addHeader('','');
      echo '<table align="center"><tr><td>';
        addHeading('Problem:');
        echo 'You could not be logged in. <br />
                 You must be logged in to view this page!';
        addURL('login.php','Go back to Login page');
      echo '</td></tr></table>';
        addFooter();
        exit;
      }//end try-catch
  }
  else
  {
    addHeader('','');
      addHeading('Problem:');
```

```
      echo 'You could not be logged in. <br />
               You need to fill in both the Username and Password!';
    addURL('login.php','Go back to Login page');
      addFooter();
      exit;
  }//end if-else
  //-------------------------------------------------------------------
-----
?>
```

A.28 “rulesFuncs.php”

```
<?php
//POLARIS
//Developer: T. Scholiadis

//This file contains all the Database related functions for the POLARIS
Web App

//-------------------------------------------------------------------
---
//Compares the Organisations or Countries to see which
//ones meet the entered constraints
function compareExact($dbCon, $constraints, $sourceLet, $issueID)
{
  //Code Removed for Intellectual Property Protection
}//end compareExact Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Finds theSimilar Orgs in Similar Countries
function compareOrgsInSimilarCountries($dbCon, $simMar,
$exactAndSimilar)
{
  //Code Removed for Intellectual Property Protection
}//end compareOrgsInSimilarCountries Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Finds the Similar Countries/Organisations
function compareSimilar($dbCon, $simMar, $exactMatchesArray,
$sourceLet)
{
  //Code Removed for Intellectual Property Protection
}//end compareSimilarCountries Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Returns the Country ID of an Organisation
function getCountryID($dbCon, $orgID)
{
  $sql = "SELECT CountryID
              FROM Organisations
              WHERE OrgID = $orgID ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Country ID
from the Database.');
```

```
  $row = $resultSet->fetch_assoc();

  return $row['CountryID'];
}//end getCountryID Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Returns an array with the Entered Constraints
function getEnteredConstraints($dbCon, $dataArray)
{
  $tempArray = array();

  $countryMaxID = getMaxID($dbCon, "CountryData", "CountryDataID");
  $orgMaxID = getMaxID($dbCon, "OrgData", "OrgDataID");

  //-----------------------------------------
  //Gets the Country Constraints
  for ($i = 1; $i <= $countryMaxID; $i++)
  {
    if (isset($dataArray['c-'.$i.'-equal']) && $dataArray['c-'.$i.'-
equal'] != "")
      {
        $tempArray['c-'.$i.'-equal'] = $dataArray['c-'.$i.'-equal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-lessVal']) && $dataArray['c-
'.$i.'-lessVal'] != "")
      {
        $tempArray['c-'.$i.'-lessVal'] = $dataArray['c-'.$i.'-
lessVal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-greaterVal']) && $dataArray['c-
'.$i.'-greaterVal'] != "")
      {
        $tempArray['c-'.$i.'-greaterVal'] = $dataArray['c-'.$i.'-
greaterVal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-perVal']) && $dataArray['c-'.$i.'-
perVal'] != "")
      {
        $tempArray['c-'.$i.'-perVal'] = $dataArray['c-'.$i.'-perVal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-perIncDec']) && $dataArray['c-
'.$i.'-perIncDec'] != "")
      {
        $tempArray['c-'.$i.'-perIncDec'] = $dataArray['c-'.$i.'-
perIncDec'];
      }//end if

      if (isset($dataArray['c-'.$i.'-lessPerVal']) && $dataArray['c-
'.$i.'-lessPerVal'] != "")
```

```
      {
        $tempArray['c-'.$i.'-lessPerVal'] = $dataArray['c-'.$i.'-
lessPerVal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-greaterPerVal']) && $dataArray['c-
'.$i.'-greaterPerVal'] != "")
      {
        $tempArray['c-'.$i.'-greaterPerVal'] = $dataArray['c-'.$i.'-
greaterPerVal'];
      }//end if

      if (isset($dataArray['c-'.$i.'-perIncDecLessGr']) &&
$dataArray['c-'.$i.'-perIncDecLessGr'] != "")
      {
        $tempArray['c-'.$i.'-perIncDecLessGr'] = $dataArray['c-'.$i.'-
perIncDecLessGr'];
      }//end if
  }//end for
  //-----------------------------------------

  //-----------------------------------------
  //Gets the Organisation Constraints
  for ($i = 1; $i <= $orgMaxID; $i++)
  {
    if (isset($dataArray['o-'.$i.'-equal']) && $dataArray['o-'.$i.'-
equal'] != "")
      {
        $tempArray['o-'.$i.'-equal'] = $dataArray['o-'.$i.'-equal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-lessVal']) && $dataArray['o-
'.$i.'-lessVal'] != "")
      {
        $tempArray['o-'.$i.'-lessVal'] = $dataArray['o-'.$i.'-
lessVal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-greaterVal']) && $dataArray['o-
'.$i.'-greaterVal'] != "")
      {
        $tempArray['o-'.$i.'-greaterVal'] = $dataArray['o-'.$i.'-
greaterVal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-perVal']) && $dataArray['o-'.$i.'-
perVal'] != "")
      {
        $tempArray['o-'.$i.'-perVal'] = $dataArray['o-'.$i.'-perVal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-perIncDec']) && $dataArray['o-
'.$i.'-perIncDec'] != "")
      {
        $tempArray['o-'.$i.'-perIncDec'] = $dataArray['o-'.$i.'-
perIncDec'];
```

```
      }//end if

      if (isset($dataArray['o-'.$i.'-lessPerVal']) && $dataArray['o-
'.$i.'-lessPerVal'] != "")
      {
        $tempArray['o-'.$i.'-lessPerVal'] = $dataArray['o-'.$i.'-
lessPerVal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-greaterPerVal']) && $dataArray['o-
'.$i.'-greaterPerVal'] != "")
      {
        $tempArray['o-'.$i.'-greaterPerVal'] = $dataArray['o-'.$i.'-
greaterPerVal'];
      }//end if

      if (isset($dataArray['o-'.$i.'-perIncDecLessGr']) &&
$dataArray['o-'.$i.'-perIncDecLessGr'] != "")
      {
        $tempArray['o-'.$i.'-perIncDecLessGr'] = $dataArray['o-'.$i.'-
perIncDecLessGr'];
      }//end if-ladder
  }//end for
  //-----------------------------------------

  return $tempArray;
}//end getEnteredConstraints Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Gets the Selected Country/Org Latest entered year
function getLatestYear($dbCon, $sourceLet, $coOrgID)
{
  if ($sourceLet == 'c')
  {
    $table = 'CountryDataDet';
      $tableID = 'CountryID';
  }
  elseif ($sourceLet == 'o')
  {
    $table = 'OrgDataDet';
      $tableID = 'OrgID';
  }//end if-else

  $sql = "SELECT Max(DataYear) maxY
              FROM $table
              WHERE $tableID = $coOrgID ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Latest
DataYear from the Database.');

  $row = $resultSet->fetch_assoc();

  return $row['maxY'];
```

```
}//end getLatestYear Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Return the rules array for the Country Data and the Organisation Data
function getRulesArray($dbCon, $issueID, $source)
{
  if ($source == "Country")
  {
    $table = "Countries";
      $tableID = "CountryID";

      $prefix = "c";
      $prefix2 = "countryData";
  }
  else
  {
    $table = "Organisations";
      $tableID = "OrgID";

      $prefix = "o";
      $prefix2 = "orgData";
  }//end if-else

  //$dataMaxID = getMaxID($dbCon, $table, $tableID);

  $sql = "SELECT *
              FROM Rules
              WHERE IssueID = '$issueID' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Rules from
the Database.');

  $tempArray = array();

  if ($resultSet->num_rows > 0)
  {
      for ($i = 1; $i <= $resultSet->num_rows; $i++)
      {
        $row = $resultSet->fetch_assoc();

        $textArray = explode('-', $row['DataID']);

        if ($textArray[0] == $prefix)
        {
          $tempArray[$textArray[1]] = $textArray[1];
        }
      }//end for
  }//end if

  return $tempArray;
}//end getRulesArray Function
//-------------------------------------------------------------------------
---
```

```
//-------------------------------------------------------------------
---
//Check if ALL the constraints match for the specified Coutry/OrgID
function isMatch($dbCon, $sourceLet, $constraints, $issueID, $coOrgID)
{
  if ($sourceLet == 'c')
  {
      $table = 'CountryDataDet';
      $tableID1 = 'CountryID';
      $tableID2 = 'CountryDataID';
  }
  elseif ($sourceLet == 'o')
  {
      $table = 'OrgDataDet';
      $tableID1 = 'OrgID';
      $tableID2 = 'OrgDataID';
  }//end if-ladder

  $sql = "SELECT DataID
              FROM Rules
              WHERE IssueID = $issueID ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Rules from
the Database.');

  $counter = 0;
  for ($i = 0; $i < $resultSet->num_rows; $i++)
  {
    $row = $resultSet->fetch_assoc();

      $anID = explode('-', $row['DataID']);

      if (isset($constraints[$row['DataID'].'-equal']) &&
$constraints[$row['DataID'].'-equal'] != "")
      {
        $compData = $constraints[$row['DataID'].'-equal'];

        $sql2 = "SELECT Det
                    FROM $table
                    WHERE $tableID1 = $coOrgID
                    AND $tableID2 = $anID[1]
                    AND Det = $compData ";

        $rs2 = runQuery($dbCon, $sql2, 'Could not Retrieve EQUAL DATA
from the Database.');

        if ($rs2->num_rows == 0)
          return false;

        $counter++;
      }
      elseif ((isset($constraints[$row['DataID'].'-lessVal']) &&
$constraints[$row['DataID'].'-lessVal'] != "") ||
(isset($constraints[$row['DataID'].'-greaterVal']) &&
$constraints[$row['DataID'].'-greaterVal'] != ""))
```

```
      {
        $compData = $constraints[$row['DataID'].'-lessVal'];
        $compDataGr = $constraints[$row['DataID'].'-greaterVal'];

        $sql2 = "SELECT Det
                    FROM $table
                    WHERE $tableID1 = $coOrgID
                    AND $tableID2 = $anID[1]
                    AND Det > $compData
                    AND Det < $compDataGr ";

        $rs2 = runQuery($dbCon, $sql2, 'Could not Retrieve LESS GREATER
VAL DATA from the Database.');

        if ($rs2->num_rows == 0)
          return false;

        $counter++;
      }
      elseif (isset($constraints[$row['DataID'].'-perVal']) &&
$constraints[$sourceLet.'-'.$i.'-perVal'] != "")
      {
        $counter++;
      }
      elseif ((isset($constraints[$row['DataID'].'-lessPerVal']) &&
$constraints[$row['DataID'].'-lessPerVal'] != "") ||
(isset($constraints[$row['DataID'].'-greaterPerVal']) &&
$constraints[$row['DataID'].'-greaterPerVal'] != ""))
      {
        $counter++;
      }//end if-ladder
  }//end for

  if ($counter > 0)
    return true;
  else
    return false;
}//end isMatch Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Checks if the given Country/Org is similar to the selected one
function isSimilar($dbCon, $simMar, $matchID, $sourceLet, $selYear,
$matchYear)
{
  if ($sourceLet == 'c')
  {
    $maxID = getMaxID($dbCon, "CountryData", "CountryDataID");

      $table = 'CountryDataDet';
      $tableID1 = 'CountryID';
      $tableID2 = 'CountryDataID';
      $selID = $_SESSION['countryID'];
  }
```

```
  elseif ($sourceLet == 'o')
  {
    $maxID = getMaxID($dbCon, "OrgData", "OrgDataID");

      $table = 'OrgDataDet';
      $tableID1 = 'OrgID';
      $tableID2 = 'OrgDataID';
      $selID = $_SESSION['orgID'];
  }//end if-else

  for ($i = 1; $i <= $maxID; $i++)
  {
    $sql = "SELECT Det
                  FROM $table
                  WHERE $tableID1 = $selID
                  AND $tableID2 = $i
                  AND DataYear = $selYear ";

      $rs1 = runQuery($dbCon, $sql, 'Could not Retrieve Det Selected
from the Database.');

      if ($rs1->num_rows > 0)
      {
        $row1 = $rs1->fetch_assoc();

        $low = $row1['Det'] - ($row1['Det']*($simMar/100));
        $high = $row1['Det'] + ($row1['Det']*($simMar/100));

        $sql2 = "SELECT Det
                     FROM $table
                     WHERE $tableID1 = $matchID
                     AND $tableID2 = $i
                     AND DataYear = $matchYear
                     AND Det > $low
                     AND Det < $high ";

        $rs2 = runQuery($dbCon, $sql2, 'Could not Retrieve Det Match
from the Database.');

        if ($rs2->num_rows == 0)
          return false;
      }//end if
  }//end for

  return true;
}//end isSimilar Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Performs the Test:
//1. Compares the constraints with the data
//2. Checks which returned results are similar
//    to the selected Organisation
//3. Checks which organisations are situated in Countries
```

```
//    similar to the selected ones.
function performTest($dbCon, $constraints, $issueID, $simMar)
{
  //Code Removed for Intellectual Property Protection
}//end performTest Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Adds the Selected rules into the Database
function saveRules($dbCon, $issueID, $countryArray, $orgArray,
$countryDataMaxID, $orgDataMaxID)
{
  $sql = "DELETE FROM Rules
              WHERE IssueID = '$issueID' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Delete Rules from the
Database.');

  for ($i = 1; $i <= $countryDataMaxID; $i++)
  {
    if (isset($countryArray[$i]))
      {
        $sql = "INSERT INTO Rules
                VALUES ('$issueID', 'c-$countryArray[$i]') ";

        $resultSet = runQuery($dbCon, $sql, 'Could not Add Country Data
Assigment.');
      }//end if
  }//end for

  for ($i = 1; $i <= $orgDataMaxID; $i++)
  {
    if (isset($orgArray[$i]))
      {
        $sql = "INSERT INTO Rules
                VALUES ('$issueID', 'o-$orgArray[$i]') ";

        $resultSet = runQuery($dbCon, $sql, 'Could not Add Organisation
Data Assigment.');
      }//end if
  }//end for

  echo '<b>The Rules have been Saved!</b><br /><br />';
}//end saveRules Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Validates that the data has been entered correctly for each
constraint
//I.e. Only 1 row filled in and the Radio buttons for the appropriate
row
//Returns true if validation passes, false if it fails
```

```
function validateEnteredConstraints($dbCon, $constraints, $simMar)
{
  $errStr = "";

  $countryMaxID = getMaxID($dbCon, "CountryData", "CountryDataID");
  $orgMaxID = getMaxID($dbCon, "OrgData", "OrgDataID");

  //-------------------------------------------
  //Checks the Similarity Margin
  if ($simMar != '')
  {
      if (!is_numeric($simMar))
      $errStr = $errStr.'<b>Error in <i>Similarity Margin</i>:</b> You
need to enter a numeric value!<br />';
  }
  else
  {
    $errStr = $errStr.'<b>Error in <i>Similarity Margin</i>:</b> You
need to enter a value!<br />';
  }//end if-else

  //-------------------------------------------
  //Checks the Country Constraints
  for ($i = 1; $i <= $countryMaxID; $i++)
  {
    $tempArray = array();

    if (isset($constraints['c-'.$i.'-equal']) && $constraints['c-
'.$i.'-equal'] != "")
      {
        $tempArray['c-'.$i.'-equal'] = 1;

        if (!(is_numeric($constraints['c-'.$i.'-equal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to enter a numeric value!<br />';
      }//end if

      if ((isset($constraints['c-'.$i.'-lessVal']) && $constraints['c-
'.$i.'-lessVal'] != "") || (isset($constraints['c-'.$i.'-greaterVal'])
&& $constraints['c-'.$i.'-greaterVal'] != ""))
      {
        if (!(is_numeric($constraints['c-'.$i.'-lessVal'])) ||
!(is_numeric($constraints['c-'.$i.'-greaterVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to enter a numeric value!<br />';

        $tempArray['c-'.$i.'-lessVal'] = 1;
      }//end if

      if ((isset($constraints['c-'.$i.'-perVal']) && $constraints['c-
'.$i.'-perVal'] != "") || (isset($constraints['c-'.$i.'-perIncDec']) &&
$constraints['c-'.$i.'-perIncDec'] != ""))
      {
        if (!(isset($constraints['c-'.$i.'-perIncDec']) &&
$constraints['c-'.$i.'-perIncDec'] != ""))
```

```
            $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to select an Increase or Decrease!<br
/>';

        if (!(is_numeric($constraints['c-'.$i.'-perVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to enter a numeric value!<br />';

        $tempArray['c-'.$i.'-perVal'] = 1;
      }//end if

      if ((isset($constraints['c-'.$i.'-lessPerVal']) &&
$constraints['c-'.$i.'-lessPerVal'] != "") || (isset($constraints['c-
'.$i.'-greaterPerVal']) && $constraints['c-'.$i.'-greaterPerVal'] !=
"") || (isset($constraints['c-'.$i.'-perIncDecLessGr']) &&
$constraints['c-'.$i.'-perIncDecLessGr'] != ""))
      {
        if (!(isset($constraints['c-'.$i.'-perIncDecLessGr']) &&
$constraints['c-'.$i.'-perIncDecLessGr'] != ""))
            $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to select an Increase or Decrease!<br
/>';

        if (!(is_numeric($constraints['c-'.$i.'-lessPerVal'])) ||
!(is_numeric($constraints['c-'.$i.'-greaterPerVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> You need to enter a numeric value!<br />';

        $tempArray['c-'.$i.'-lessPerVal'] = 1;
      }//end if

      if (sizeof($tempArray) > 1)
        $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Country").'</i>:</b> Please enter only 1 row of constraints!<br
/>';
  }//end for
  //-----------------------------------------

  //-----------------------------------------
  //Checks the Organisation Constraints
  for ($i = 1; $i <= $orgMaxID; $i++)
  {
    $tempArray = array();

    if (isset($constraints['o-'.$i.'-equal']) && $constraints['o-
'.$i.'-equal'] != "")
      {
        if (!(is_numeric($constraints['o-'.$i.'-equal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to enter a numeric value!<br
/>';

        $tempArray['o-'.$i.'-equal'] = 1;
      }//end if
```

```
      if ((isset($constraints['o-'.$i.'-lessVal']) && $constraints['o-
'.$i.'-lessVal'] != "") || (isset($constraints['o-'.$i.'-greaterVal'])
&& $constraints['o-'.$i.'-greaterVal'] != ""))
      {
        if (!(is_numeric($constraints['o-'.$i.'-lessVal'])) ||
!(is_numeric($constraints['o-'.$i.'-greaterVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to enter a numeric value!<br
/>';

        $tempArray['o-'.$i.'-lessVal'] = 1;
      }//end if

      if ((isset($constraints['o-'.$i.'-perVal']) && $constraints['o-
'.$i.'-perVal'] != "") || (isset($constraints['o-'.$i.'-perIncDec']) &&
$constraints['o-'.$i.'-perIncDec'] != ""))
      {
        if (!(isset($constraints['o-'.$i.'-perIncDec']) &&
$constraints['o-'.$i.'-perIncDec'] != ""))
            $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to select an Increase or
Decrease!<br />';

        if (!(is_numeric($constraints['o-'.$i.'-perVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to enter a numeric value!<br
/>';

        $tempArray['o-'.$i.'-perVal'] = 1;
      }//end if

      if ((isset($constraints['o-'.$i.'-lessPerVal']) &&
$constraints['o-'.$i.'-lessPerVal'] != "") || (isset($constraints['o-
'.$i.'-greaterPerVal']) && $constraints['o-'.$i.'-greaterPerVal'] !=
"") || (isset($constraints['o-'.$i.'-perIncDecLessGr']) &&
$constraints['o-'.$i.'-perIncDecLessGr'] != ""))
      {
        if (!(isset($constraints['o-'.$i.'-perIncDecLessGr']) &&
$constraints['o-'.$i.'-perIncDecLessGr'] != ""))
            $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to select an Increase or
Decrease!<br />';

        if (!(is_numeric($constraints['o-'.$i.'-lessPerVal'])) ||
!(is_numeric($constraints['o-'.$i.'-greaterPerVal'])))
          $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> You need to enter a numeric value!<br
/>';

        $tempArray['o-'.$i.'-lessPerVal'] = 1;
      }//end if

      if (sizeof($tempArray) > 1)
        $errStr = $errStr.'<b>Error in <i>'.getDataName($dbCon,
$i,"Organisation").'</i>:</b> Please enter only 1 row of
constraints!<br />';
```

```
  }//end for
  //-----------------------------------------

  if ($errStr != "")
  {
    echo $errStr.'<br />';
      return false;
  }//end if

  return true;
}//end validateEnteredConstraints
//-------------------------------------------------------------------------
---
?>
```

A.29 “runTest.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Page where the User runs his test in the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $catID = $_POST['catID'];
  $subCatID = $_POST['subCatID'];
  $issueID = $_POST['issueID'];
  $simMar = $_POST['simMar'];

  try
  {
    checkValidUser();

      addHeader("Run Test",$_SESSION['validUser']);
    displayMenu();
    addHeading("Run a Test");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      if ((isset($_SESSION['countryID']) && $_SESSION['countryID'] !=
"-1") && (isset($_SESSION['orgID']) && $_SESSION['orgID'] != "-1"))
      {
        ?>
          <table align="center">
              <tr>
                <td>
                  <?php
                    echo 'Country Selected: <b>'.getName($dbCon,
$_SESSION['countryID'], "Country").'</b><br />';
                    echo 'Organisation Selected: <b>'.getName($dbCon,
$_SESSION['orgID'], "Organisation").'</b><br /><br />';
                  ?>
                  </td>
              </tr>
            </table>
        <?php

        displaySetRulesForm($dbCon, $catID, $subCatID, $issueID,
"runTest.php");

        if (isset($_POST['run']))
        {
          $constraints = getEnteredConstraints($dbCon, $_POST);
```

```
            $pass = validateEnteredConstraints($dbCon, $constraints,
$simMar);
        }//end if

        if (isset($issueID) && $issueID != "-1")
        {
          displayEnterConstraintsForm($dbCon, $issueID, $catID,
$subCatID, $constraints, $simMar);
        }//end if

        if (isset($_POST['run']))
        {
          if ($pass)
            {
              performTest($dbCon, $constraints, $issueID, $simMar);
            }//end
        }//end if
      }
      else
      {
        echo 'You have not selected a Country or an Organisation.<br
/>';
        echo 'Please make the Selections first!<br />';
        addURL("myAccount.php","Select Country and Organisation");
        echo '<br /><br />';
      }//end if-else

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.30 “setRules.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Set Issue Rules Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  $catID = $_POST['catID'];
  $subCatID = $_POST['subCatID'];
  $issueID = $_POST['issueID'];

  try
  {
    checkValidUser();

      addHeader("Set Rules",$_SESSION['validUser']);
    displayMenu();
    addHeading("Set the Rules of an Issue");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);

      $dbCon = connectToDB();

      displaySetRulesForm($dbCon, $catID, $subCatID, $issueID,
"setRules.php");

      if (isset($_POST['save']))
      {
        //-------------------------------------------------------------
-----------------
        //Adds all the checked Country checkboxes into an array
        $countryArray = array();

        $countryDataMaxID = getMaxID($dbCon, "Countries", "CountryID");

      for ($i = 1; $i <= $countryDataMaxID; $i++)
      {
          if (isset($_POST["countryData$i"]))
          {
            $countryArray[$i] = $i;
          }//end if
      }//end for
        //-------------------------------------------------------------
-----------------

        //-------------------------------------------------------------
-----------------
        //Adds all the checked Organisation checkboxes into an array
```

```
        $orgArray = array();

        $orgDataMaxID = getMaxID($dbCon, "Organisations", "OrgID");

      for ($i = 1; $i <= $orgDataMaxID; $i++)
      {
          if (isset($_POST["orgData$i"]))
          {
            $orgArray[$i] = $i;
          }//end if
      }//end for
        //-------------------------------------------------------------
-----------------

        saveRules($dbCon, $issueID, $countryArray, $orgArray,
$countryDataMaxID, $orgDataMaxID);
      }//end if

      if (isset($issueID) && $issueID != "-1")
      {
        displayDataSelectionForm($dbCon, $catID, $subCatID, $issueID);
      }//end if

      $dbCon->close();
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.31 “style.css”

```
.answered
{
	-background-color:lime;
}
.not_answered
{
	-background-color:orange;
}
h2
{
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 16px;
	font-weight:bolder;
}

h3
{
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 13px;
	font-weight:bold;
}

body
{
	background-color: #B6C2CE;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #000000;
	background-image: url(bg.gif);
}
table1
{
	width: 100%;
}
table, td, th
{
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
	color: #000000;
}
input, select, check, radio, textarea
{
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
	color: #000000;
}

.blue {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	//font-size: 11px;
	//background-color:#003399;
	background-color:#334AB6;
```

```
	color:#FFFFFF;
	font-weight:bold;
}

.blue2 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	//font-size: 11px;
	//background-color:#003399;
	background-color:#3333FF;
	color:#FFFFFF;
	font-weight:bold;
}

.lightblue {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	//font-size: 11px;
	background-color:#3399FF;
	color:#000000;
}

.lightblue2 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	//font-size: 11px;
	background-color:#33CCFF;
	color:#000000;
}

.blueSmall {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 9px;
	//background-color:#003399;
	background-color:#334AB6;
	color:#FFFFFF;
	font-weight:bold;
}
.border
{
	border-top-width: thin;
	border-right-width: thin;
	border-bottom-width: thin;
	border-left-width: thin;
	border-top-style: solid;
	border-right-style: solid;
	border-bottom-style: solid;
	border-left-style: solid;
	border-color:#000000;
}

a:link {
	color: #000000;
	text-decoration: none;
}
a:visited {
	text-decoration: none;
	color: #999999;
}
```

```
a:hover {
	text-decoration: underline;
	color: #333333;
}
a:active {
	text-decoration: none;
}
```

A.32 “underConstruction.php”

```
<?php
  session_start();

//POLARIS
//Developer: T. Scholiadis

//This is the Under Construction Page of the POLARIS Web App

  require("polarisFuncs.php"); //Including Function Files

  try
  {
    checkValidUser();

      addHeader("Under Construction",$_SESSION['validUser']);
    displayMenu();
    addHeading("Under Construction");

      $authorisedUserTypes = array('Administrator');
      checkUserTypeAuthorisation($authorisedUserTypes);
?>

      <h1> This Page is Under Construction </h1>

      <img src="page-under-construction.jpg" alt="Under Construction"
/>

<?php
    addFooter();
  }
  catch(Exception $e)
  {
    echo $e->getMessage();
  }//end try-catch
?>
```

A.33 “userAuthFuncs.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains all the User Authentication functions for the
POLARIS Web App

require_once('dbFuncs.php');

//-------------------------------------------------------------------
---
//Function to check that the user is Authorised to view the page
//If he is, the function returns true
function checkUserTypeAuthorisation($authorisedUserTypes)
{
  for ($i = 0; $i < sizeof($authorisedUserTypes); $i++)
  {
      if ($_SESSION['userType'] == $authorisedUserTypes[$i])
        return true;
  }//end for

  echo 'You are not Authorised to View this page!';
  addFooter();
  exit;
}//end checkUserTypeAuthorisation Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//Function to check if the user is logged in
function checkValidUser()
{
  if (!isset($_SESSION['validUser']))
  {//Warns if user is not logged in
    addHeader('','');
    echo '<table align="center"><tr><td>';
      addHeading("Problem:");
      addHeading("Problem:");
      echo 'You are not logged in!';
      addURL('login.php','Login');
      echo '</td></tr></table>';
      addFooter();
      exit;
  }//end if
}//end checkValidUser Function
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
```

```
//Function to retrieve the User Type of the User
function getUserType($dbCon)
{
  $sql = "SELECT UserTypeDesc
          FROM UserAssigns as ua, UserTypes as ut
              WHERE ua.UserTypeID = ut.UserTypeID
              AND ua.UserName = '$_SESSION[validUser]' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not retrieve USer
Types.');

  $row = $resultSet->fetch_assoc();

  $resultSet->free();

  return $row['UserTypeDesc'];
}//end getUserType Function
//-------------------------------------------------------------------------
---

//-------------------------------------------------------------------------
---
//Check the Username and Password with the database. If it exists
//it returns true. Otherwise it throws an exception
function login($dbCon, $username,$pwd)
{
  //Check if user exists and is Unique
  $sql = "SELECT * from users
            WHERE UserName='$username'
            AND Pwd=sha1('$pwd')";
            //AND Pwd='$pwd'";

  $resultSet = runQuery($dbCon, $sql, 'Could not Log you in.');

  if ($resultSet->num_rows > 0)
  {
    $resultSet->free();
      return true;
  }
  else
    throw new Exception('Could not Log you in.');
}//end login Function
//-------------------------------------------------------------------------
---
?>
```

A.34 “userFuncs.php”

```
<?php

//POLARIS
//Developer: T. Scholiadis

//This file contains the functions for the View
//module of the POLARIS Web App

//---------------------------------------------------------------------
//Add a new User
function addNewUser($dbCon, $userName, $name, $surname, $userTypes)
{
  $sql = "INSERT INTO Users
          VALUES ('$userName', '$name', '$surname', sha1('$pwd')) ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Add User.');

  $utMaxID = getMaxUserTypeID($dbCon);
  for ($i = 1; $i <= $utMaxID; $i++)
  {
    if (isset($userTypes[$i]))
      {
        $sql = "INSERT INTO UserAssigns
                VALUES ('$userName', '$userTypes[$i]') ";

        $resultSet = runQuery($dbCon, $sql, 'Could not Add User
Assigment.');
      }//end if
  }//end for

  echo '<b>New User Added!</b><br /><br />';
}//end addNewUser Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to get the number of User Types
function getMaxUserTypeID($dbCon)
{
  $sql = "SELECT MAX(UserTypeID) as utID
          FROM UserTypes ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Get Max User Type
ID.');

  $row = $resultSet->fetch_assoc();
  return $row['utID'];
}//end getNumVesselTypes Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Function to get the Details of a User and return them in an array
function getUserDetails($dbCon, $userName)
```

```
{
  $sql = "SELECT Name, Surname
          FROM Users
              WHERE Username = '$userName' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve User Deatails
from the Database.');

  $row = $resultSet->fetch_assoc();

  return $row;
}//end getUserDetails Function
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
//Function to Get the the User Types of the User
function getUserTypes($dbCon, $userName)
{
  $sql = "SELECT UserTypeID
          FROM UserAssigns
              WHERE UserName = '$userName' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve User Types
from Database.');

  $tempArray = array();

  for ($i = 0; $i < $resultSet->num_rows; $i++)
  {
    $row = $resultSet->fetch_assoc();

     $tempArray[$row['UserTypeID']] = $row['UserTypeID'];
  }//end for

  return $tempArray;
}//end getUserTypes Function
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
//Checka that the user entered the correct Old Password
function oldPwdConfirmed($dbCon, $oldPwd)
{
  //Check if user exists and is Unique
  $sql = "SELECT *
          FROM users
            WHERE UserName = '$_SESSION[validUser]'
            AND Pwd = sha1('$oldPwd')";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve Old Password
from Database.');

  if ($resultSet->num_rows > 0)
    return true;
  else
    return false;
}//end oldPwdConfirmed Function
```

```
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Update a User's Password
function updatePassword($dbCon, $userName, $pwd)
{
  $sql = "UPDATE Users
          SET Pwd = sha1('$pwd')
              WHERE UserName = '$userName' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Update User
Details.');

  echo '<b>User Password Changed!</b><br /><br />';
}//end updatePassword Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Checks if the user already exists
function userExists($dbCon, $userName)
{
  $sql = "SELECT *
          FROM Users
              WHERE UserName = '$userName' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve User Names
from the Database.');

  if ($resultSet->num_rows > 0) //if a user with that user name already
exists
    return true;
  else
    return false;
}//end userExists Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Update a User's Details
function updateUserDetails($dbCon, $userName, $name, $surname,
$userTypes="", $utMaxID="")
{
  $sql = "UPDATE Users
          SET Name = '$name', Surname = '$surname'
              WHERE UserName = '$userName' ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Update User
Details.');

  if ($utMaxID != "")
  {
    $sql = "DELETE FROM UserAssigns
                  WHERE UserName = '$userName' ";

      $resultSet = runQuery($dbCon, $sql, 'Could not Delete User from
the Database.');
```

```
      for ($i = 1; $i <= $utMaxID; $i++)
      {
        if (isset($userTypes[$i]))
        {
          $sql = "INSERT INTO UserAssigns
                      VALUES ('$userName', '$userTypes[$i]') ";

            $resultSet = runQuery($dbCon, $sql, 'Could not Add User
Assigns into the Database.');
        }//end if
      }//end for
  }//end if

  echo '<b>User Details Updated!</b><br /><br />';
}//end updateUserDetails Function
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//Puts the User Types of one user into a string
function userTypesToStr($dbCon, $userName)
{
  $sql = "SELECT ut.UserTypeDesc as utd
              FROM UserTypes as ut, UserAssigns as ua
              WHERE ua.UserName = '$userName'
              AND ua.UserTypeID = ut.UserTypeID ";

  $resultSet = runQuery($dbCon, $sql, 'Could not Retrieve User Types
from the Database.');

  $str = "";
  for ($i = 0; $i < $resultSet->num_rows; $i++)
  {
    $row = $resultSet->fetch_assoc();

      $str = $str.$row['utd'].', ';
  }//end for

  //removes the last character of the String (which is a comma)
  $str = substr($str, 0, strlen($str)-2);

  return $str;
}//end userTypeToStr Function
//---------------------------------------------------------------------
?>
```

APPENDIX B:
DATABASE ".SQL" File

Appendix B.2 will contain a Listing of a MySQL dump of the database schemas for the POLARIS Web Application. It will also contain some example data, namely the data used in the presentation of the thesis.

Even though the contents of Appendix B.2 the only ones needed to build the database, there is one more important thing that is required: a user needs to be created that has access to the MySQL database. This is the general user of the "POLARIS Database" that performs the actions on behalf of the users created using POLARIS. This is explained in Appendix B.1.

B.1 Creating the POLARIS Database

Before uploading the Database (or rather running the script of the Listing in Appendix B.2), a MySQL administrator will need to create the POLARIS database. This is done with the following line of code using MySQL:

```
create database POLARIS;
```

After the database has been created, the administrator will need to give access of POLARIS to a user. This user will be able to perform all the functions on behalf of the users that are created in POLARIS. This is done for two reasons:

1. The MySQL administrator will not be able to know which users will be using POLARIS,
2. Restricts the access to the database to only one user instead of all the users in POLARIS. It is up to the “POLARIS general user” to check if the user logging into POLARIS is an authorised user.

This user is created with the following code:

```
GRANT select, insert, update, delete
ON polaris.*
TO polaris IDENTIFIED BY ‘polaris’
```

Now the POLARIS database is ready to have the database uploaded from the Listing in Appendix B.2.

B.2 “POLARIS.sql”

```
-- MySQL dump 10.10
--
-- Host: localhost    Database: POLARIS
-- ------------------------------------------------------
-- Server version 5.0.27-community-nt

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `categories`
--

DROP TABLE IF EXISTS `categories`;
CREATE TABLE `categories` (
  `CatID` int(10) unsigned NOT NULL auto_increment,
  `CatName` varchar(45) NOT NULL,
  PRIMARY KEY  USING BTREE (`CatID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `categories`
--

LOCK TABLES `categories` WRITE;
/*!40000 ALTER TABLE `categories` DISABLE KEYS */;
INSERT INTO `categories` VALUES
(1,'Deck'),(2,'Engine'),(3,'Management'),(4,'Shipbuilding'),(5,'Environ
ment'),(6,'Finance'),(7,'Law (Ship)'),(8,'Law (Gov)'),(9,'Port
Operations'),(10,'Various');
/*!40000 ALTER TABLE `categories` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `countries`
--

DROP TABLE IF EXISTS `countries`;
CREATE TABLE `countries` (
  `CountryID` int(10) unsigned NOT NULL auto_increment,
  `CountryName` varchar(45) NOT NULL,
  PRIMARY KEY  (`CountryID`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `countries`
--

LOCK TABLES `countries` WRITE;
/*!40000 ALTER TABLE `countries` DISABLE KEYS */;
INSERT INTO `countries` VALUES (1,'Greece'),(2,'South
Africa'),(3,'England'),(4,'Cyprus'),(5,'Italy');
/*!40000 ALTER TABLE `countries` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `countrydata`
--

DROP TABLE IF EXISTS `countrydata`;
CREATE TABLE `countrydata` (
  `CountryDataID` int(10) unsigned NOT NULL auto_increment,
  `CountryDataName` varchar(45) NOT NULL,
  PRIMARY KEY  (`CountryDataID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `countrydata`
--

LOCK TABLES `countrydata` WRITE;
/*!40000 ALTER TABLE `countrydata` DISABLE KEYS */;
INSERT INTO `countrydata` VALUES (1,'GDP'),(2,'Growth
Rate'),(3,'Interest Rate'),(4,'Tax Income'),(5,'Total Expenses');
/*!40000 ALTER TABLE `countrydata` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `countrydatadet`
--

DROP TABLE IF EXISTS `countrydatadet`;
CREATE TABLE `countrydatadet` (
  `CountryID` int(10) unsigned NOT NULL,
  `CountryDataID` int(10) unsigned NOT NULL,
  `Det` varchar(100) NOT NULL,
  `DataYear` int(10) unsigned NOT NULL,
  PRIMARY KEY  USING BTREE (`CountryID`,`CountryDataID`,`DataYear`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `countrydatadet`
--

LOCK TABLES `countrydatadet` WRITE;
/*!40000 ALTER TABLE `countrydatadet` DISABLE KEYS */;
INSERT INTO `countrydatadet` VALUES
(1,1,'15',2007),(1,2,'3',2007),(1,3,'5',2007),(1,4,'20000000',2007),(1,
```

```
5,'19000000',2007),(4,1,'789',2006),(4,1,'123',2007),(4,2,'456',2006),(
4,2,'654',2007),(4,3,'123',2006),(4,4,'147',2006),(4,4,'789',2007),(4,5
,'258',2006),(4,5,'012',2007);
/*!40000 ALTER TABLE `countrydatadet` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `issues`
--

DROP TABLE IF EXISTS `issues`;
CREATE TABLE `issues` (
  `IssueID` int(10) unsigned NOT NULL auto_increment,
  `IssueName` varchar(45) NOT NULL,
  `SubcatID` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`IssueID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `issues`
--

LOCK TABLES `issues` WRITE;
/*!40000 ALTER TABLE `issues` DISABLE KEYS */;
INSERT INTO `issues` VALUES (1,'Recruitment (state
policy)',20),(2,'Education & Training (state policy)',20),(3,'Wage
Agreements (state policy)',20),(4,'Unemployment (state
policy)',20),(5,'Social Benefits (state policy)',20),(6,'Job
Descriptions (state policy)',20),(7,'Recruitment (company
policy)',20),(8,'Salaries (company policy)',20),(9,'Productivity
(company policy)',20),(10,'Training (company as a
provider)',20),(11,'Tugging',44),(12,'Pilotage',44),(13,'Manoeuvering',
44),(14,'Bunkering',44),(15,'Supplies &
Provisions',44),(16,'Mooring',44),(17,'Dredging',44),(18,'Watchkeeping'
,44),(19,'Free Pratique',44),(20,'Quarantine',44);
/*!40000 ALTER TABLE `issues` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `organisations`
--

DROP TABLE IF EXISTS `organisations`;
CREATE TABLE `organisations` (
  `OrgID` int(10) unsigned NOT NULL auto_increment,
  `OrgName` varchar(45) NOT NULL,
  `CountryID` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`OrgID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `organisations`
--

LOCK TABLES `organisations` WRITE;
/*!40000 ALTER TABLE `organisations` DISABLE KEYS */;
```

```
INSERT INTO `organisations` VALUES (1,'Government',0),(2,'Top
Tankers',1),(3,'Grinrod',2),(4,'Danaos',4),(5,'Costamare',3),(6,'Teo
Shipping',3),(7,'Maersk',5);
/*!40000 ALTER TABLE `organisations` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `orgdata`
--

DROP TABLE IF EXISTS `orgdata`;
CREATE TABLE `orgdata` (
  `OrgDataID` int(10) unsigned NOT NULL auto_increment,
  `OrgDataName` varchar(45) NOT NULL,
  PRIMARY KEY  (`OrgDataID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `orgdata`
--

LOCK TABLES `orgdata` WRITE;
/*!40000 ALTER TABLE `orgdata` DISABLE KEYS */;
INSERT INTO `orgdata` VALUES
(1,'Profit'),(2,'Assets'),(3,'Liabilities'),(4,'Turnover');
/*!40000 ALTER TABLE `orgdata` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `orgdatadet`
--

DROP TABLE IF EXISTS `orgdatadet`;
CREATE TABLE `orgdatadet` (
  `OrgID` int(10) unsigned NOT NULL,
  `OrgDataID` int(10) unsigned NOT NULL,
  `Det` varchar(100) NOT NULL,
  `DataYear` int(10) unsigned NOT NULL,
  PRIMARY KEY  USING BTREE (`OrgID`,`OrgDataID`,`DataYear`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `orgdatadet`
--

LOCK TABLES `orgdatadet` WRITE;
/*!40000 ALTER TABLE `orgdatadet` DISABLE KEYS */;
INSERT INTO `orgdatadet` VALUES
(3,1,'100000000',2007),(3,2,'100000000',2007),(3,3,'500000000',2007),(3
,4,'600000000',2007),(4,1,'369',2006),(4,1,'789',2007),(4,2,'147',2006)
,(4,2,'123',2007),(4,3,'258',2006),(4,3,'456',2007),(4,4,'159',2006),(5
,1,'360',2006),(5,1,'780',2007),(5,2,'987',2006),(5,2,'120',2007),(5,3,
'654',2006),(5,3,'450',2007),(5,4,'843',2006),(5,4,'753',2007);
/*!40000 ALTER TABLE `orgdatadet` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--
-- Table structure for table `rules`
--

DROP TABLE IF EXISTS `rules`;
CREATE TABLE `rules` (
  `IssueID` int(10) unsigned NOT NULL,
  `DataID` varchar(100) NOT NULL,
  PRIMARY KEY  (`IssueID`,`DataID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `rules`
--

LOCK TABLES `rules` WRITE;
/*!40000 ALTER TABLE `rules` DISABLE KEYS */;
INSERT INTO `rules` VALUES (1,'c-1'),(1,'c-2'),(1,'c-3'),(1,'c-
5'),(1,'o-1'),(1,'o-2'),(1,'o-4'),(7,'o-1'),(7,'o-2');
/*!40000 ALTER TABLE `rules` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `subcategories`
--

DROP TABLE IF EXISTS `subcategories`;
CREATE TABLE `subcategories` (
  `SubcatID` int(10) unsigned NOT NULL auto_increment,
  `SubcatName` varchar(45) NOT NULL,
  `CatID` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`SubcatID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `subcategories`
--

LOCK TABLES `subcategories` WRITE;
/*!40000 ALTER TABLE `subcategories` DISABLE KEYS */;
INSERT INTO `subcategories` VALUES (1,'Ship
Management',1),(2,'Navigation',1),(3,'Manoeuvering',1),(4,'Stability',1
),(5,'Safety',1),(6,'Cargo',1),(7,'Seamanship',1),(8,'Communications',1
),(9,'Equipment',1),(10,'Main
Engine',2),(11,'Generators',2),(12,'Auxiliaries',2),(13,'Automation',2)
,(14,'Propulsion',2),(15,'Dry
Docking',2),(16,'Operations',3),(17,'Chartering',3),(18,'Insurance',3),
(19,'Marketing',3),(20,'HRM',3),(21,'Auditing',3),(22,'Sales &
Purchases',3),(23,'Quality Assurance',3),(24,'Naval
Architecture',4),(25,'Ship
Construction',4),(26,'Inspections',4),(27,'Classifications',4),(28,'Pro
tection',5),(29,'Control',5),(30,'Planning',5),(31,'Env.
Management',5),(32,'Economic
Analysis',6),(33,'Loans',6),(34,'Pricing',6),(35,'Cost
Control',6),(36,'Investments',6),(37,'Cases',7),(38,'Claims',7),(39,'Ma
ritime Policy',8),(40,'Conventions',8),(41,'EU
```

```
Directives',8),(42,'Other',8),(43,'Port
Management',9),(44,'Procedures',9),(45,'Logistics',9),(46,'Cargo
Handling',9),(47,'Port Pricing',9),(48,'Technology',9),(49,'Port
Safety',9);
/*!40000 ALTER TABLE `subcategories` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `userassigns`
--

DROP TABLE IF EXISTS `userassigns`;
CREATE TABLE `userassigns` (
  `UserName` varchar(16) NOT NULL,
  `UserTypeID` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`UserName`,`UserTypeID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `userassigns`
--

LOCK TABLES `userassigns` WRITE;
/*!40000 ALTER TABLE `userassigns` DISABLE KEYS */;
INSERT INTO `userassigns` VALUES ('admin',1);
/*!40000 ALTER TABLE `userassigns` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `UserName` varchar(16) NOT NULL,
  `Name` varchar(25) NOT NULL,
  `Surname` varchar(25) NOT NULL,
  `Pwd` char(40) NOT NULL,
  PRIMARY KEY  (`UserName`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `users`
--

LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES
('admin','admin','admin','d033e22ae348aeb5660fc2140aec35850c4da997');
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `usertypes`
--
```

```
DROP TABLE IF EXISTS `usertypes`;
CREATE TABLE `usertypes` (
  `UserTypeID` int(10) unsigned NOT NULL auto_increment,
  `UserTypeDesc` varchar(45) NOT NULL,
  PRIMARY KEY  (`UserTypeID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `usertypes`
--

LOCK TABLES `usertypes` WRITE;
/*!40000 ALTER TABLE `usertypes` DISABLE KEYS */;
INSERT INTO `usertypes` VALUES (1,'Administrator'),(2,'Other');
/*!40000 ALTER TABLE `usertypes` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2007-10-07 13:33:48
```