Time Table Assistant

Design Project Report

Developers: Marit Sæther and Edward Calderon
Client: Ian Gomm
Supervisor: Tim Hunt
Co-ordinator: Dr. Hao Shi
Summary

The purpose of the *Time Table Assistant* project was to develop a software system that would support the creation and management of student timetables for the School of Computer Science and Mathematics at Victoria University of Technology. The previous system involved recording the school’s resources across four separate spreadsheet files. Distributing resource data in this way made timetable management complex and time inefficient.

The *Time Table Assistant* software system automates the timetable creation and management process. The complete system consists of a user interface that provides timetable creation and scheduling functions and a database that stores details of the school’s resources. By providing a fully featured timetable editing interface to a school resource database, the *Time Table Assistant* system allows for comprehensive timetable management from a centralised administration point.

The development process used for the *Time Table Assistant* system consisted of four essential phases; Requirements/Specification, Design, Implementation and Integration. Each aspect of this process has been documented in detail in four major technical documents. The Problem Statement was produced at the end of the Requirements phase and presented a description of the existing and proposed systems. The proposed system would be a timetable editor, which could perform automatic clash checking. Use case modelling was performed during the Specification phase to define the functionality of the proposed system. To ‘realise’ these use cases, the system’s interface would need to access a database containing records of the school’s resources. The interface would display a visual representation of a timetable complete with editing controls that would interact with the database to schedule school resources to weekly timeslots. A database schema, user interface design and software requirements analysis followed the use case analysis and these were presented in the Software Requirements Specification.

The Software Design Document was the result of the Design phase and presented the detailed and architectural design of the proposed system. The application was coded in Visual Basic .NET during the Implementation phase and details of the integration and testing plan were presented in the Test Plan document. Weekly team meetings and also meetings with the client and supervisor were recorded in the project log book and a task schedule was used to help monitor the team’s progress through the development process.

The biggest difficulty in the design and implementation of the application came in the attempt to make the program code as re-usable and generic as possible. While the database access coding has been based heavily on the school’s resource database (and is therefore not re-usable) much of the interface coding has been made adaptable. Developing the application’s database access code to be able to operate on any timetable-resource database could be a goal of future improvements and would make the *Time Table Assistant* application a powerful management support tool.

The development of this project from inception to completion has provided the project team with invaluable experience in the development of a medium-scale business system and an insight into real world software design standards and practices.
# Table of Contents

1. **Introduction** ................................................................. 5  
   1.1. **Problem Definition** .................................................. 5  
   1.2. **Project Description** .................................................. 5  
      1.2.1. *Timetable Creation and Resource Data Entry* ............... 5  
      1.2.2. *Allocation Clash Checking* ...................................... 6  
      1.2.3. *Reports, Saving and Printing* .................................... 6  
      1.2.4. *Email* ........................................................................ 6  

2. **Project Process** ............................................................... 7  
   2.1. **Requirements Analysis** .............................................. 7  
   2.2. **Design** ...................................................................... 7  
   2.3. **Test Planning** ............................................................ 8  
   2.4. **Implementation** .......................................................... 8  

3. **Project Management** ....................................................... 9  
4. **Evaluation** ......................................................................... 11  
   4.1. **Problems Encountered** .............................................. 11  
   4.2. **Strengths and Limitations** ........................................... 12  
   4.3. **Future Improvements** ............................................... 12  
   4.4. **Lessons Learned** ....................................................... 13  

5. **Conclusion** ....................................................................... 13  
6. **Bibliography** ..................................................................... 14  
7. **Glossary** .......................................................................... 15  

**Appendix A : Software Requirements Specification** 16  
**Appendix B : Software Design Document** ......................... 17  
**Appendix C : Test Plan** ..................................................... 18
Table of Figures

Figure 1 Table of Tasks with Estimated Time Spent.................................10
Figure 2 Total Hours per Semester ............................................................11
1. Introduction

*Time Table Assistant* is a software system that will support the task of creating and updating timetables. The client for this project is a lecturer at Victoria University of Technology, who is responsible for creating and maintaining the timetables for the School of Computer Science and Mathematics. *Time Table Assistant* will replace the current system in use, which consists of a set of four Microsoft Excel spreadsheets. Each spreadsheet contains its own representation of the school’s timetables based on a separate school resource. These representations include the student timetable, location timetable, subject timetable and staff timetable. The student timetable presents a schedule of the subject, class, location and staff resources. The location timetable presents the subjects scheduled for a particular location. The subject timetable presents the class, location and staff scheduled for a particular subject and the staff timetable presents the subject, class and location for a particular staff member.

1.1. Problem Definition

In the current system timetables are created and updated manually by typing in scheduled resources into the cells of the relevant spreadsheet. Before resources can be scheduled into a timetable, it is necessary to manually compare and check each of the other spreadsheets to confirm that the resource to be scheduled is available at the required day and time. After a resource has been scheduled, the spreadsheets are then manually updated to reflect the latest schedule. Typing and manually updating the spreadsheets causes errors and inconsistency. In addition, manually resolving allocation clashes is time consuming and tedious.

1.2. Project Description

The aim of the project was to develop a software system that would automate the process of maintaining the school’s timetables.

1.2.1. Timetable Creation and Resource Data Entry

*Time Table Assistant* is designed to assist in the task of creating timetables and making changes to already scheduled timetables. 

*Time Table Assistant* provides the user with the option to customise the layout of a new timetable by specifying values for day(s), time(s) and student group(s). A student group is specified by which course and level the timetable is created for. Levels are undergraduate and postgraduate, where undergraduate student groups are year 1, 2 or 3. Year 1 student groups are divided into sub groups that can go from A and up to M.

When a timetable has been created, a drag and drop interface provides a quick and easy way of filling the timetable with resources. This interface will limit the risk of errors, as no typing is required. Resources that are to be scheduled are dragged into the timetable from four different fill windows, one for each of the resources. The resources are Subject, Class, Location and Staff.
The resources are stored in the *Time Table Assistant* Database. *Time Table Assistant* provides an interface that enables the user to access the resources in the database in order to insert a new resource, delete and update existing resources.

### 1.2.2. Allocation Clash Checking

*Time Table Assistant* runs automatic allocation checks when a timetable is filled with resources, and when a change is made to a timetable. The allocation checks will make sure no allocation clashes are made. Allocation clashes are defined by obvious clashes defined for any scheduling problem involving a time factor and resources available at a particular day and time. If an allocation clash is detected during a check, a message will inform the timetable administrator of where the clash was found. Checks are run against allocations in the timetable currently being scheduled and in all other existing timetables. Allocations are stored in the *Time Table Assistant* Database. An update to a timetable will automatically update the database.

### 1.2.3. Reports, Saving and Printing

The Report function allows the user to view selected information from one or more timetables. Reports are the equivalent to the representations of the timetables in the current system. A report will show the timetable for a subject, a staff or one or more locations. *Time Table Assistant* allows the user to save timetables and reports. Timetables and reports can be saved under a name in the database for later retrieval. Timetables and reports can also be converted to and saved as an Excel file.

### 1.2.4. Email

*Time Table Assistant* provides the timetable administrator with an interface to log into a personal email account. The timetable administrator can specify the Uniform Resource Locator, URL for the email interface. This function requires access to the Internet. Once the timetable assistant is logged into the email account, an Excel file with a timetable or report can be attached to an email. Staff members will receive the email with the attachment and be able to open the timetable or report in Excel.

Section 2 of the Design Project Report contains a detailed description of each phase of the design process including requirements analysis, design, test planning and implementation.
Section 3 deals with the project management issues including time spent on various activities and log book analysis.
Section 4 provides an evaluation of the project including problems encountered, how they were dealt with and what they taught the developers. This section also includes future improvements that can be done to *Time Table Assistant* if it were to be upgraded.
For more information about the Software Requirements Specifications refer to Appendix A. Appendix B is a reference for the Software Design Document, while Appendix C contains the Test Plan for *Time Table Assistant*. 
2. Project Process

*Time Table Assistant* was developed following the conventional Software Life Cycle Model (SLCM) which characterised the activities and the interaction of the activities involved in the development process. The SLCM chosen for this project was the incremental model. The first stage of activities involved identifying the system’s requirements. Once the requirements had been defined, the team established a set of Use Cases that captured the essential functionality of the system. The following stages of activities were dedicated to developing these Use Cases further before implementing them one at a time, starting with the core Use Cases, and integrating them into a complete software system. An early mock up was developed followed by a series of early versions that were presented to the client at each stage for acceptance.

2.1. Requirements Analysis

The requirements analysis began with regular meetings with the client where the team could discuss the client’s needs in regards to the proposed system. At the conclusion of this analysis we had derived a comprehensive set of functional and non-functional requirements for *Time Table Assistant*. The full documentation of these requirements appears in the SRS found in Appendix A. In keeping with the conventions of the SLCM, the client signed the SRS as an agreement of its content. The SRS would now act as the contract between the developers and the client. The developers took the responsibility of developing the proposed system according to the requirements stated in the SRS within the given time. The client acknowledged that after the SRS was signed, no more features could be added to *Time Table Assistant*.

2.2. Design

During the design process the team focused on developing the architectural design of *Time Table Assistant*. The factors that shaped the architectural design were the system processes, which were identified using several modelling methods and the requirements stated in the SRS. With these factors in consideration, the team made the decision to design the system using the two-tier architecture. The first layer (tier one) in this design was the presentation layer which incorporated the system’s Graphical User Interface, GUI. The design of the GUI was critical to the effective ‘usability’ of the software application. A well designed GUI would allow the user to easily take advantage of the full functionality of the application. A significant amount of time and effort went into expressing the essential functionality of the system (as identified by the Use Cases specified in the SRS) in the most intuitive and easy to understand manner. Once a mock up GUI had been developed, screen shots were taken and presented to the client for acceptance.

The second layer, representing the storage layer, incorporated the database that would be used by the software application. The database schema was derived from the Natural language Information Analysis Method, NIAM, diagrams drawn in the SRS. A Context level Data Flow Diagram (DFD at level 0), was obtained to analyse the logical requirements and specify the flow of data between the two layers. In addition,
Level 1 DFDs were derived for each sub system in order to identify and model the various sub processes in system. The details of the design process are stated in the SDD (Refer to Appendix B)

2.3. Test Planning

The Test Plan documented the suite of testing procedures used to ensure the quality and correctness of the program application. Correctness was based on the ability of the test module to respond appropriately to all variations of input data (including expected, unexpected, invalid and boundary test data) and generate the desired output data, if any. Quality was assessed on the ability of the test module to provide the desired functionality as specified by the Use Case(s) that the test module was based on. These assessments were applied at three levels of testing; unit testing, integration testing and acceptance testing. (Refer to Appendix C for the Test Plan document.)

Unit testing involved taking each individual class of the program and running it in isolation while analysing its execution and performance. The test modules in this case were each and every Visual Basic .Net class that comprised the application.

The integration testing involved taking a group of two or more classes, running them together as a single unit and analysing their execution and performance. The test modules in this case were the ‘task-group’ of classes that together performed a specific function (such as, ‘load a database and edit a Staff Member’s details’). Black Box testing was used at both the Unit and Integration testing levels in order to gauge the quality and correctness of the given test module.

Informal acceptance testing was performed at various stages of the coding and implementation phase and was useful in gauging the level of satisfaction of the client and the success of the current-most version. These informal testing sessions would all lead to the final acceptance test where the client would apply the full functionality of the program application to ‘live’ data in actual working conditions.

2.4. Implementation

After completing the SDD and finalising the Design phase, the Implementation Phase was ready to proceed. The basic user interface of the program already existed at this point as it had been used as a prototype to illustrate to the client the essential functionality of the program. Much of this functionality was expressed through a set of task specific window forms where each form was generated by a separate Visual Basic .NET class. As such, a general class hierarchy was already in place and many of the program’s classes had already been created. This made it possible to insert programming logic directly into those class files under construction.

A prototype database was also being finalised at this point and the implementation strategy became a combination of the Top-Down and Bottom-Up approaches. The high level functionality of the user interface allowed the developers to see how the data would be manipulated while the low-level organisation of the database tables allowed the developers to see how the data would be stored and retrieved.
The coding and implementation agenda was based generally the order of the Use Cases specified in the SRS. The first Use Case to be fully implemented was ‘Connect Database’ since the functionality to specify and create a new timetable could only be implemented after a logical connection to the Time Table Assistant Database had been made available in the program. From there the implementation was carried out class by class, function by function and then Use Case by Use Case. That is, the programming logic required to perform a certain function (such as, ‘accept input data from user to use as specifications for a new timetable’) was firstly coded and inserted into the relevant class (or classes). A Use Case could then be realised by combining the functionality of two or more of these functions as they were implemented (such as, ‘take timetable specifications and then submit these to another class to create and display the timetable’). The goal of each successive version was to build upon the functionality of the previous version.

Informal ‘White Box’ testing was performed throughout the coding of each class in order to facilitate the rapid development of that class to the suitable level of accuracy where formal (unit) testing could be applied. (The ‘White Box’ testing technique proved to be a highly effective pre and post-test debugging tool). Once a class had passed its unit testing it could then be integrated with a ‘task-group’ of two or more other classes.

3. Project Management

The project Time Table Assistant was inaugurated in semester 2 July 2002. The team of developers had been established by the end of June. The developers arranged from the very beginning of the project to set up regular meeting times each week for three team meetings and for one supervisor meeting. Regular contact was kept with the client by through meetings and an open dialog was kept through emails. From the outset it was made known to the team that the problem would be challenging and that success would depend on diligence. As such, a logbook was kept and written by all team members as a way of informing, keeping track of and solving tasks. Once we established that a set of tasks needed to be done, each of those tasks was assigned a time-for-completion and then allocated to a team member with a deadline attached. Tasks were prioritised according to their estimated time-for-completion. Research and development efforts could then be focused on tasks with higher priority and enable the developer team to resolve potential problem areas before they grew too large to handle. In addition to the log book a project web site was designed and published containing the project meetings, supervisor meetings and client meetings, and the project schedule with deadlines, assessments and goals.

The project Time Table Assistant was conducted over two semesters with each semester focusing on a different set of tasks. The first semester involved deriving the problem statement, the SRS in Appendix A and the SDD in Appendix B. The problem statement was completed by week 4, the SRS by week 8 and the SDD by week 13. In addition to the documentation, 2 oral presentations were held, in week 7 and week 12 on the topics SRS and SDD respectively.
The team consisted in the first semester of four team members, Edward Calderon, the team leader, Anne Marte Bye, Astrid Mohan and Marit Sæther. Martin Schweizer was originally appointed as the project supervisor and was later replaced by Dr Jayanthi Kumar. The project coordinator was Dr Elizabeth Haywood.

The second semester involved the implementation of *Time Table Assistant*, deriving the Test Plan, TP in Appendix C and writing a user manual. 2 oral presentations were held in semester 2. The first were held in week 5 on the topics progress status and TP. The second presentation was the final conference presentation in week 13, presenting the finished product for the client, supervisor, coordinator and other developer teams. In addition a poster presentation was made also in week 13. (A detailed breakdown of the amount of time spent on each of these tasks is given in figure 1. The total time spent on the project during each semester is given in figure 2.)

The team was in the second semester reduced to two members. This was due to the graduation of one of the team members and due to another team member decision to pursue a mathematical project in semester 2. The team was aware of these factors from the very beginning of the project and they did not cause big change overs, only minor reorganisations. The team in the second semester consisted of Marit Sæther, the team leader and Edward Calderon. The new project supervisor was Tim Hunt and project coordinator Dr Hao Shi.

During the early stages of the implementation phase, the developers set standards for declarations, variable names and comments to be used in the code. This was performed to ease the task of understanding and interpreting each other’s lines of code as the total number grew to more than 15 000 lines.

<table>
<thead>
<tr>
<th>Task</th>
<th>Hours Spent Per Week</th>
<th>Per Team Member</th>
<th>No. of Weeks</th>
<th>Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 1 Project Meetings</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>156</td>
</tr>
<tr>
<td>Semester 1 Supervisor Meetings</td>
<td>0.5</td>
<td>4</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>Semester 1 Client Meetings</td>
<td>0.5</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Problem Statement</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Software Requirements Specifications</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>Software Design Document</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>Oral Presentations Preparations</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Semester 2 Project Meetings</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td>96</td>
</tr>
<tr>
<td>Semester 2 Supervisor Meetings</td>
<td>0.5</td>
<td>2</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Semester 2 Client Meetings</td>
<td>0.5</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Test Plan</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Design Project Report</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>Software Requirements Specifications Update</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>Software Design Document Update</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>User Manual</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>Implementation</td>
<td>8</td>
<td>2</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td>Testing</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Research</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>72</td>
</tr>
<tr>
<td>Oral Presentation Preparation</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Poster Presentation</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

*Figure 1 Table of Tasks with Estimated Time Spent*
4. Evaluation

The goal was to create an easy to use software system that would assist the timetable administrator in creating and maintaining timetables. As a result, the final product has an easy to use GUI that offers a comprehensive set of timetable development functions. The most important functions included in the GUI are timetable creation, timetable editing and timetable saving as these respond directly to the client’s requirements. By automating these essential functions, *Time Table Assistant* saves the client time in comparison to the current manual system. In particular, the automatic allocation clash check provides for a vast saving in time. In addition the system can create reports based on timetables which enables the user to view selected information. Furthermore, the system provides functionality to convert timetables and reports to an Excel file and to send these files over the Internet as email messages.

4.1. Problems Encountered

The first challenge the team met when starting to implement *Time Table Assistant* was to connect the GUI to the database. It was found that it was necessary to move from Visual Basic 6 to Visual Basic.NET to be able to connect to an Access 2000 database. The developers managed to adapt quickly to the new environment through researching resources on the Internet and books.

Designing the GUI for *Time Table Assistant* has also proven to be a demanding task. Design issues included; how to fit timetable information on one screen for different timetables, how to design fill windows so that they are detailed yet compact, and how to best design windows where the user selects information from either combo boxes, radio buttons or list boxes. The design issues have been solved through trial and error, detailed analyses and usability testing. Design Heuristics and ‘The 8 Golden Rules of Interface Design’ have also been used in decision making.

<table>
<thead>
<tr>
<th>Semester</th>
<th>Team Effort (In total hours)</th>
<th>Individual Effort (In total hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>605</td>
<td>302.5</td>
</tr>
</tbody>
</table>

*Figure 2 Total Hours per Semester*
4.2. **Strengths and Limitations**

*Time Table Assistant* is designed to give the time table administrator a system that saves time and maximises data integration. Errors made are limited by the drag and drop interface that does not require any typing. Time is also saved by this quick and easy way of filling a timetable. Time is also saved by the automatic allocation checks performed by *Time Table Assistant*. Another strength of the application is that the resources all are stored in one place, in a database that is accessible through *Time Table Assistant*. Save and email can also be done through the same system. Having all these features in one system allows the timetable administrator to perform all tasks without leaving the system.

*Time Table Assistant* is not designed to be networked. The system is also highly specific to the client and not fully generic and therefore not be used for any other timetabling situations, which require a different database.

4.3. **Future Improvements**

While the application is able to provide a comprehensive range of timetable development functions for the designated school resource database, it is unable do so for any other resource database. Much of the programming code that deals with database access depends on the structure and naming of the tables defined in the resource database created especially for the school. The restructuring of this database-dependent code and the reforming of the current data-storage procedure for the purposes of allowing the application to be used with any resource database would be a worthwhile objective for future improvements.

This generalisation of the application’s functionality however would be a major undertaking and would demand advanced programming skills. A requirement of this type of improvement task would include developing the application’s database editing functionality. The application would need to provide options that enable the user to set up and configure a resource database that they have just loaded into the program before allowing them to create and customize their own timetables. The programmer (or programming team) would need to know how to code nested logical loops (at least three levels deep) while dynamically creating database tables and assigning user-specified names to database table attributes.

If the application could be developed to the point where it could accept any given resource database, it could then be used by any number of small to medium sized organisations that require staff/resource scheduling and thereby establish itself as a truly flexible and powerful management tool.
4.4. Lessons Learned

Arguably the biggest lesson learned from the undertaking of this project was that of effective time management. In hindsight it appears that the coding and testing phases proceeded smoothly and without a problem. However it is acknowledged that this would not have been possible without the planning and preparation conducted in the first semester. The entire point of engaging in the specification and design phases was to save time during the implementation and testing phases. The excess time spent in the detailed documentation of these preliminary phases being more than justified during the latter phases, given that by this stage our team of four had been reduced to a team of two. The lessons of early preparation and planning also demonstrate their significance here.

5. Conclusion

_Time Table Assistant_ successfully implements a timetable editing and management system which meets all the software requirements specified by our client. With the completion of final stage testing and documentation, the delivery of the finished product ends two semesters of a dedicated software development endeavour.

The experience gained from this project is wide-ranging. Each stage of the development process called for organisation of team meetings, identification of required tasks, task allocation and scheduling, research and development of tasks, management of team progress and documentation of all results into a professional-level technical report. Added to this list of skills are conceptual and analytical modelling required during the specification and design phases and high quality code generation, testing and debugging during the implementation phase. Teamwork skills such as co-operation, negotiation and dependability were provided by all team members and supported by the time management and project planning skills of the team leaders.

From the initial description of the business problem to the delivery of the final product, the project has given our team an advanced introduction into real-world software development. By adhering to the conventions of an industry standard SLCM, the team members are now able to confidently apply these procedures in any future project they are assigned to or to otherwise adapt their knowledge to suit similar development models.
6. Bibliography


## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Box Testing</td>
<td>Method of testing programming code where the expected output of the test module is compared against the module’s actual output as based on a given input data set. See also, ‘White Box Testing’.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>Internet</td>
<td>The largest group of interconnected computers in the world.</td>
</tr>
<tr>
<td>Microsoft Access</td>
<td>Microsoft database software.</td>
</tr>
<tr>
<td>Microsoft Visual Basic.NET</td>
<td>A high-level programming language from Microsoft that is graphically oriented. Used to create database applications and software packages.</td>
</tr>
<tr>
<td>NIAM</td>
<td>Natural language Information Analysis Method.</td>
</tr>
<tr>
<td>Rational Rose</td>
<td>Model-driven development tool.</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>Use case</td>
<td>Describes a basic process in application area from user’s point of view.</td>
</tr>
<tr>
<td>White Box Testing</td>
<td>Method of testing programming code involving the analysis/inspection of the values of key variables within the test module at various stages of execution.</td>
</tr>
</tbody>
</table>
Appendix A :

Software Requirements Specification
Appendix B :

Software Design Document
Appendix C :

Test Plan