|  |
| --- |
| MIT Logo  Engineering  **115.611 DE6499 Engineering Project (Electrical)**  Report 2016  **Assessment Manager**  *By : Liam Mills*  *130008961*  *Supervisor : Snjezana Soltic* |

Abstract

A desktop application named Assessment Manager was designed to assist lecturers in writing, publishing and marking assessments electronically. The application consisted of two parts; Assessment Designer and Examinee. Assessment Designer is used by the examiner to write, publish and mark assessments and provides tools for creating PDF documents of those assessments and emailing them to students. The second part, Examinee, is used by students to participate in the assessment. It controls the timing of the assessment and ensures that only the allocated amount of time is given.

The project as a whole is distributed as an installer and is intended to work on Windows machines, from Windows 7 and up. The program targets the .NET 4.5.2 environment and is programmed in C# 6.0 using Visual Studio.

Acknowledgments

Snjezana Soltic is to be acknowledged for her help and guidance throughout development of the project. She provided design ideas and aimed the development towards a more user friendly result.

Tim Roberts is also to be acknowledged for his original program, ExamManager, which provided the inspiration and design template for Assessment Manager.

Contents

[Abstract ii](#_Toc468101234)

[Acknowledgments iii](#_Toc468101235)

[Contents iv](#_Toc468101236)

[List of Figures v](#_Toc468101237)

[1 Introduction 1](#_Toc468101238)

[2 History 2](#_Toc468101239)

[3 Specifications 3](#_Toc468101240)

[4 Key Elements 5](#_Toc468101241)

[4.1 Shared Code 5](#_Toc468101242)

[4.1.1 Class UMLs 5](#_Toc468101243)

[4.2 Assessment Designer Elements 7](#_Toc468101244)

[4.2.1 Assessment Designer GUI 7](#_Toc468101245)

[4.2.2 AssessmentWriter 8](#_Toc468101246)

[4.2.3 EmailHandler 10](#_Toc468101247)

[4.2.4 Course Manager 10](#_Toc468101248)

[4.3 Examinee Elements 12](#_Toc468101249)

[4.3.1 Examinee GUI 12](#_Toc468101250)

[4.3.2 ‘Submit’ Event 13](#_Toc468101251)

[4.3.3 Auto-save Feature 14](#_Toc468101252)

[4.4 Testing 15](#_Toc468101253)

[5 Conclusion 16](#_Toc468101254)

[6 References 17](#_Toc468101255)

[7 Appendices 18](#_Toc468101256)

[Appendix A: Semester 1 Time Plan 18](#_Toc468101257)

[Appendix B: Semester 2 Time Plan 18](#_Toc468101258)

[Appendix C: Project Budget 19](#_Toc468101259)

List of Figures

[Figure 1: Assessment, AssessmentScript and Question UML 5](file:///C:\Users\Liam\Documents\AssessmentManager\Liam%20Mills%20Dev%20Project%20Final%20Report.docx#_Toc468101200)

[Figure 2: Assessment Designer GUI 7](#_Toc468101201)

[Figure 3: AssessmentWriter UML 9](file:///C:\Users\Liam\Documents\AssessmentManager\Liam%20Mills%20Dev%20Project%20Final%20Report.docx#_Toc468101202)

[Figure 4: EmailHandler GUI 10](file:///C:\Users\Liam\Documents\AssessmentManager\Liam%20Mills%20Dev%20Project%20Final%20Report.docx#_Toc468101203)

[Figure 5: Examinee GUI 12](#_Toc468101204)

[Figure 6: Semester 1 Time Plan 18](file:///C:\Users\Liam\Documents\AssessmentManager\Liam%20Mills%20Dev%20Project%20Final%20Report.docx#_Toc468101205)

[Figure 7: Semester 2 Time Plan 18](file:///C:\Users\Liam\Documents\AssessmentManager\Liam%20Mills%20Dev%20Project%20Final%20Report.docx#_Toc468101206)

[Figure 8: Project Budget 19](#_Toc468101207)

# Introduction

This report details the development of the C# application Assessment Manager. This application was developed as a replacement to an older program called ExamManager. The objectives of the project were to create two fully functional programs, written in C#, the first of which, named Assessment Designer, would provide a GUI and functions for the use of lecturers to electronically write, publish and mark assessments. Assessment Designer would give lecturers tools to add questions of different types and allow text formatting of the question text. It would provide a method of creating PDF’s of the assessments, whether they be completed by the student or sent for moderation. Marking assessments would be automated for certain question types and the process would be designed to make it as quick and easy and automatic as possible for the user.

The second, named Examinee, would give students a GUI to complete these assessments on the computer. Examinee would incorporate appropriate timing measures to control the length and accessibility of the assessment to give a proper assessment environment and disallow cheating or taking extra time.

The project as a whole would be distributed as an installer and intended to work on Windows machines, from Windows 7 and up. The program would target the .NET 4.5.2 environment and be programmed in C# 6.0. Assessment Manager would not use any database and therefore be portable between systems.

# History

Before Assessment Manager was developed, an older program was in use called ExamManager. This program was developed over ten years before and was written in Delphi. Many portions of its functionality no longer worked correctly due to changing systems at MIT and newer operating systems being used. The need arose for a newer program to be developed.

ExamManager provided the basic objectives for Assessment Manager, those being a program which allowed a lecturer to write, publish and mark assessments, as well as provide an interface for students to complete those assessments on the computer. These objectives were to be improved upon to provide a better and more automatic experience for the lecturer to enable a faster and easier experience when running assessments.

# Specifications

The specifications for Assessment Manager were to create two programs, Assessment Designer and Examinee, which would give lecturers a tool to write, publish and mark assessments and give students a way of completing these assessments on the computer.

For Assessment Designer, there were a few objectives that were to be met in order to accomplish this goal:

Writing:

* Three different question types would be available: Short, Long and Multi-Choice
* Controls to format the question text, including colour, font, alignment etc.
* Add and remove questions and sub-questions
* Ability to save and reload assessment
* Create a PDF of the assessment, with option to include model answers

Publish:

* Automatically send assessments to student test accounts
* Set start date, time and assessment length
* Set list of students to participate in assessment, with each student start time and assessment length able to be customised individually
* Import student details and test account details from an MS Excel spreadsheet

Mark:

* Automatically retrieve completed assessments for marking
* Ability to mark assessments, give responses to questions and assign marks
* Automatically mark single answer and multi-choice questions
* Create PDF of the completed assessment, including student answers and option to include model answers
* Ability to email students with completed assessment PDF

For Examinee, the objectives for the program were:

* Provide a simple to use interface for students to complete assessment
* Provide timing mechanics which would run the assessment for as long as the lecturer sets
* Show information about the assessment to the student: time, course, name, weighting etc.
* Without approval from the lecturer, block re-entry to the assessment once it has been completed and submitted

The project was to use C# as the programming language and be made using Visual Studio. It would target version 4.5.2 of the .NET framework and would be intended to be used on MIT computers, which at the time of development were running Windows 7. Assessment Manager would be packaged in an installer to allow it to be easily installed to a computer.

Unlike the original ExamManager program, it would not use any databases, nor would it connect to the online Blackboard service used by MIT. This was to enable it to be self-contained and portable, not reliant on external services.

# Key Elements

There were several key elements that went into the development of Assessment Manager, split across the two programs, which went towards achieving the objectives.

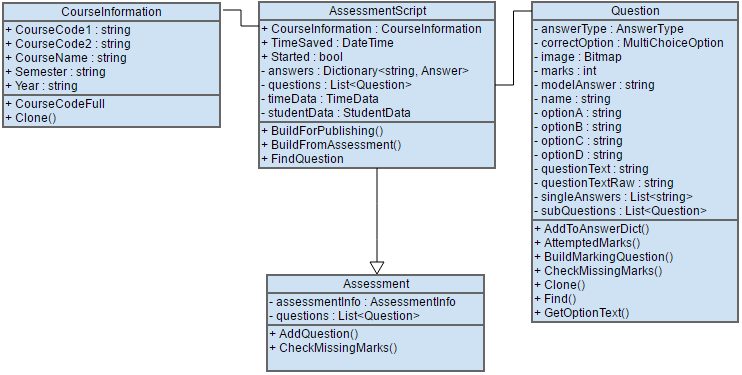
## Shared Code

Between Assessment Designer and Examinee, there is a lot of code that is shared. For this reason, a common DLL was developed to hold this code. This DLL, called ‘AssessmentManagerLib.dll’, is placed in the root folder of the application for the two executables, ‘Assessment Designer.exe’ and ‘Examinee.exe’, to access. AssessmentManagerLib (from here on referred to as ‘DLL’ or ‘the DLL’) contains all the classes that Assessment Designer and Examinee require to run. This was done to cut out duplicate code and allow the two programs to communicate with each other, so to speak.

### Class UMLs

Arguably the most important classes that were developed for the project were the Assessment and AssessmentScript classes and the Question class (Fig. 1). These classes were written to hold the data pertaining to the assessments which would be written and read by Assessment Manager.

Figure 1: Assessment, AssessmentScript and Question UML



The DLL holds the definitions for these classes so that they can be accessed and used by both programs. The Assessment class holds the information of an assessment that has been written. When an assessment is published, it is converted into an AssessmentScript class. This holds not only the questions of the assessment, but also a dictionary holding Answer objects which are used to record the students’ entered answers. This conversion happens when the assessment is deployed and is handled by the AssessmentScript’s ‘BuildFromAssessment()’ method.

## Assessment Designer Elements

Assessment Designer utilises a tab system to present the controls to the user. Four tabs were added: Designer, Publish, Courses and Mark. Each tab is responsible for a specific operation within the program and this allows it to present each part separately to the user which helps with usability and not overwhelming the user with controls.

### Assessment Designer GUI

The Assessment Designer GUI (Fig. 2) was designed to be simple to use and to provide an easy way to create assessments. It borrows from the original ExamManager program in the layout of the GUI to help previous users with moving across to the new program and the layout was efficient and clean.

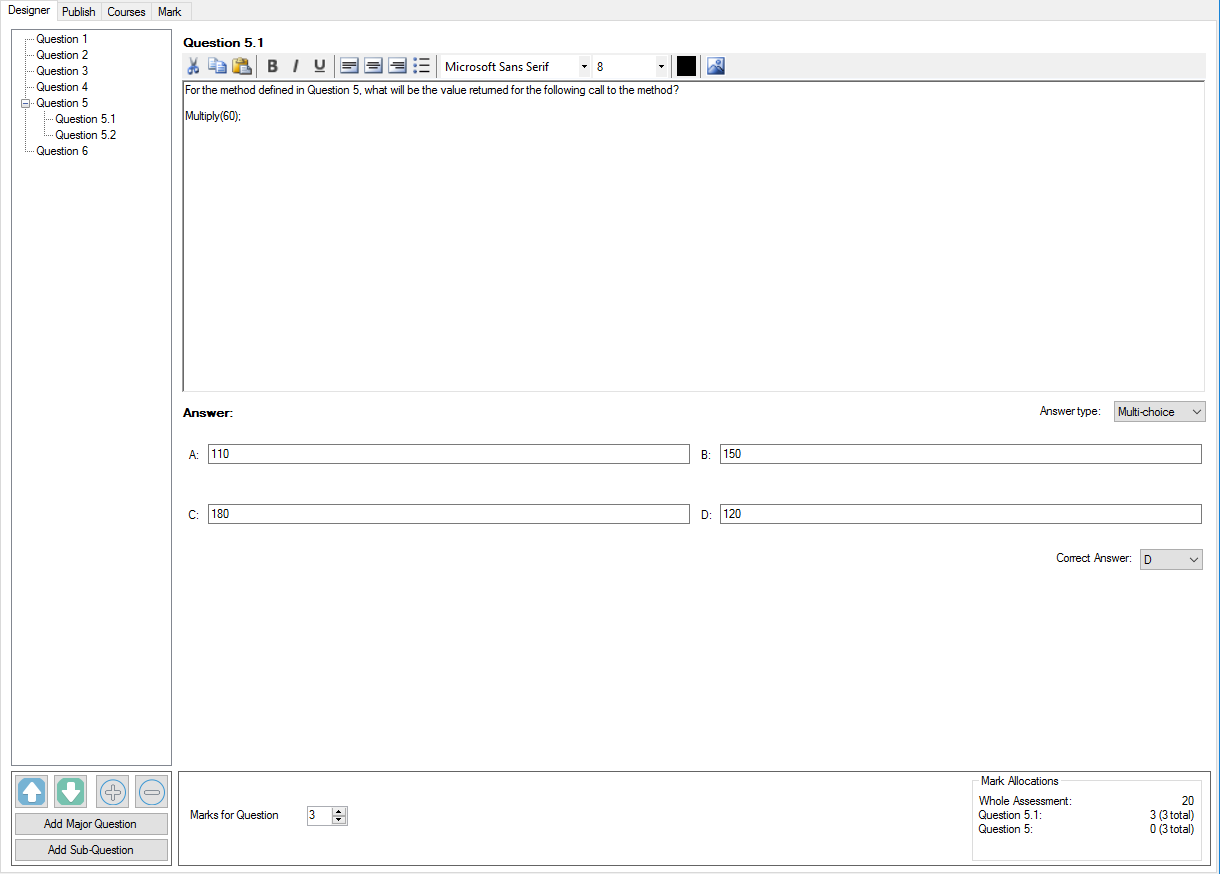


Figure 2: Assessment Designer GUI

The Designer tab makes use of textboxes to allow the user to input question text and model answers. Questions are displayed in a TreeView on the left, which is a standard WinForms control. To be able to access the question contained in each ‘node’ in the question tree, a custom node class, QuestionNode, was inherited from the base to hold and give access to a question. When the user adds or removes a node from this tree, the question list contained within the assessment instance is cleared, then rebuilt by recursively looping through all the nodes in the TreeView and recording the questions held by them. In this way, the program keeps the list of questions for the assessment mirrored with the TreeView; ultimate control of the list of questions is given to the user through the use of the TreeView.

As the user types in the textboxes, the selected question in the TreeView is updated by that textbox’s ‘TextChanged’ event. This keeps the question being edited always up-to-date with what the user is doing. As changes are made, a flag Boolean variable is flipped to indicate that changes were made. This causes an indicatory ‘\*’ symbol to appear in the window’s top frame and also makes sure that if the user attempts to close without saving, it prompts them to save or discard the changes.

Assessment Designer uses two file formats to save assessments. It uses ‘.exm’ for saving an assessment and ‘.exms’ for saving an assessment which has been published. Published assessments use the AssessmentScript class and so the ‘s’ in ‘.exms’ denotes that. When the program is installed, it writes entries to the Windows registry to associate these file extensions with itself. This allows users to open them with double click and gives them the appropriate icon.

### AssessmentWriter

A key element of Assessment Designer was the ability to create a PDF representation of the assessment that could be printed. This PDF was to have the option of being created with or without the model answers. To achieve this, a third party library was used called iTextSharp. This library provided many functions to generate a PDF document. A wrapper class was developed to simplify the PDF document generation process to a single method call. This class was called ‘AssessmentWriter’.

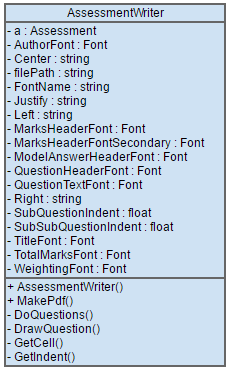


Figure 3: AssessmentWriter UML

The AssessmentWriter class (Fig. 3) takes the Assessment in the constructor and produces a PDF document when the MakePdf() method is called. Pseudo-code for this method shows how it did this:

//Create FileStream to given path for PDF

//Create new PDF document

//If AssessmentInfo present:

// Print author

// Print Title

// Print Weighting

//end

//Enter recursive loop for questions:

//Print question data

//If Question has sub-question(s):

// Enter recursive loop for sub-question

//end

//Repeat loop for all questions and sub-questions

//end

//Print the total marks for assessment

//Close document

//End method

### EmailHandler

Another key element for Assessment Designer was to send emails to students, once their assessments had been marked, containing their results. A wrapper Windows form was created to handle this process, called ‘EmailHandler’ (Fig. 4). Emails can be sent in C# by utilising the built in ‘SmtpClient’ class. The ‘EmailHandler’ form used this class to send emails to students.

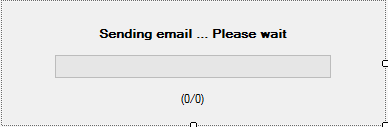


Figure 4: EmailHandler GUI

It builds the student’s email address by combining the given MIT username for that student with the MIT mail box domain name. It then builds an email message using the ‘SmtpClient’ class and attaches the PDF document. If the user chose to add a message in the email configuration menu, then this is also sent. Emails sent by the EmailHandler are sent asynchronously. This means that it is able to continuously update its GUI to present to the user the progress of sending the email.

The form includes a ‘time out’ mechanism that terminates it if it sits too long without doing anything. This mechanism is a timer which counts up to 60 seconds. The timer is reset every time the form performs an action and this ensures that it never hangs.

### Course Manager

To keep track of the students that were taking part in assessments and to make it easier to set up new assessments, a Course Manager system was developed for Assessment Designer. This system allows the user to create a Course and add student information into that Course. This information is then stored by the program and recalled each time it is started. In this way, the program is able to keep track of the assessments which have been published.

The Course Manager is a class which is in a single instantiation setup. This means that it can only ever have one instance of it instantiated at any one time. This is to ensure that all data saved and retrieved by it is handled only once and cannot be changed accidentally elsewhere within the program. This was achieved by making the constructor private, which means that only it can create an instance of it and this instance is recorded in a static variable. This happens inside a static method that is called when Assessment Designer is started by the user.

The Course Manager creates a file structure on the computer it was installed on and saves the different Courses in this place. Each Course created is given a unique 6-character ID code. This ensures that all Courses created have their own file storage area and that Courses can be created with the same names.

A specification for Assessment Designer was to be able to import students into the program from a spreadsheet. The course manager uses the ‘Microsoft.Office.Interop.Excel’ namespace, provided by the .NET framework, to import student data from spreadsheets. It expects the spreadsheet to be in a particular format, this being:

[Student ID] [Surname] [First Name] [MIT Username]

It loads this data into a DataGridView on a form and displays it in these columns. The user can check to make sure that it is correct and then select to import it to the Course.

## Examinee Elements

Examinee is in charge of running the assessments and as such requires reliable timing constraints to make sure that the assessments only run for the allotted time. To accomplish this, it uses the DateTime structure defined in .NET.

When a student opens Examinee, it displays an introduction form which gives information about the assessment and only allows the assessment to be started if the start time has been met.

It uses a timer with a 1000 millisecond tick rate to give an accurate clock and generates the time that it should stop the assessment by using the given start time and the length of the assessment, which is given in minutes.

### Examinee GUI

The Examinee GUI (Fig. 5) incorporates a TreeView on the left to list the questions contained in the assessment. This TreeView uses the same ‘QuestionNode’ system used by Assessment Designer to hold and give access to the questions. The TreeView nodes were built from the assessment question list and mirror its structure exactly.

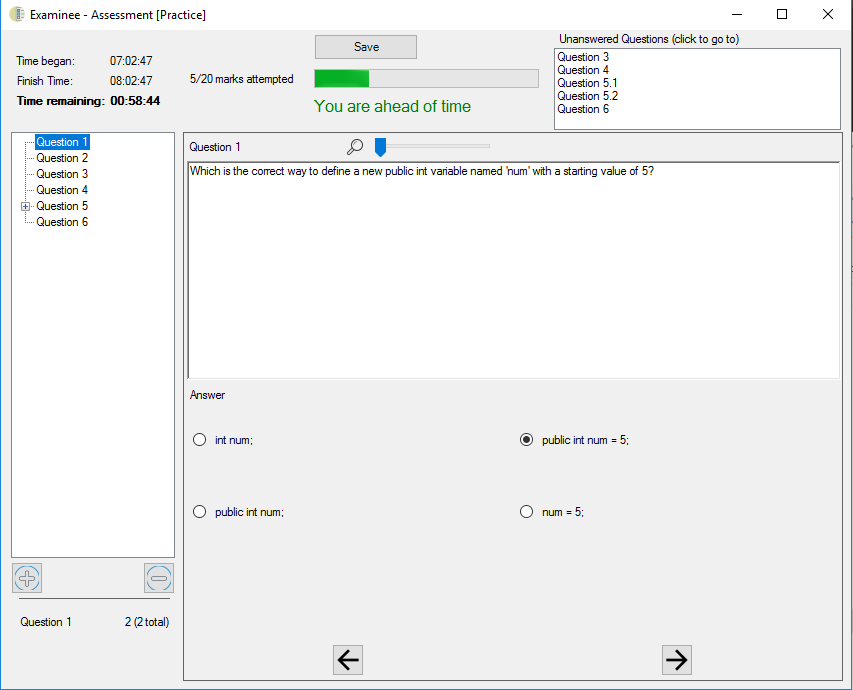


Figure : Examinee GUI

A timer runs continuously in the background with a 1000 millisecond tick rate. This is used to update the time information shown at the top of the GUI and to update the progress bar.

The answer area of the GUI was created with the use of three panels. When the user clicks on a question, the program reads the question’s answer type and displays the answer panel which holds the appropriate controls for that answer type.

Pseudo-code for the operation of the Examinee GUI follows:

//User selects a question

//Create autosave

//Update unanswered questions

//Update progress bar and marks attempted

//Read selected question data

//Display question text

//Show panel containing corresponding answer type (long, short, multi)

//Wait for answer entered event

//Answer entered:

// Find Answer object inside AssessmentScript’s ‘answers’

// dictionary using question name as the key

// Update answer with entered answer

//end

//Continue waiting for answer entered event

### ‘Submit’ Event

When the user has completed the assessment or the time has run out, they click the ‘Submit’ button to finish the assessment. This calls the ‘Submit’ event. The assessment is saved to the ‘.exms’ file that it was loaded from and a ‘Submitted’ Boolean flag is be set. This tells the program that the assessment has been completed already if the user tries to open it again. Examinee shows on the introduction form that the assessment has been completed. If there is time remaining on the assessment, it asks for a special password to give access to the assessment. This password was be generated when the assessment was published and the user can ask the lecturer for it.

### Auto-save Feature

Examinee contains an auto-save feature which automatically makes a backup of the assessment each time the user selects a different question. When the first backup is made, the program generates a ‘backups’ folder and saves the backups there. It does this by recording the file path of the assessment when it was opened and then combines the name of the assessment, after parsing it for any illegal folder path characters, with the suffix “\_autosaves”. It then uses the ‘File’ class from the .NET framework to create a folder where the assessment was opened from.

These autosaves are used by the lecturer during marking to see earlier versions of the students’ progress through the assessment. This is helpful in situations such as a student writing one answer and then later on changing it to something else.

## Testing

Assessment Manager was tested extensively throughout its development. Each week the most current build was given to Snjezana to test and give feedback on. Bugs and ideas for improvement were reported back and this feedback was used to assist the development.

In the month of November, a full test of the program was performed. Snjezana used Assessment Designer to deploy an assessment to student accounts and four MIT students partook in this test assessment. They used Examinee to complete the assessment under proper test conditions. The test was completed successfully and the program ran without problems. The purpose of this test run was to put the program into a real use environment and ensure that all parts of it worked according to design and specification.

# Conclusion

The development of Assessment Manager was a success and produced a fully functional toolset.

Assessment Designer fulfilled its objectives to provide lecturers a tool to write, publish and mark assessments electronically. It provides formatting tools for writing questions and defines three different answer types: short, long and multi-choice. It includes the ability to create PDF’s of those assessments, and to automatically email PDF’s of the completed and marked assessment scripts.

Examinee provides students an easy to use GUI to complete these assessments on the computer. It includes timing constraints to control the length of the assessment and shows the details of the assessment prior to beginning it.

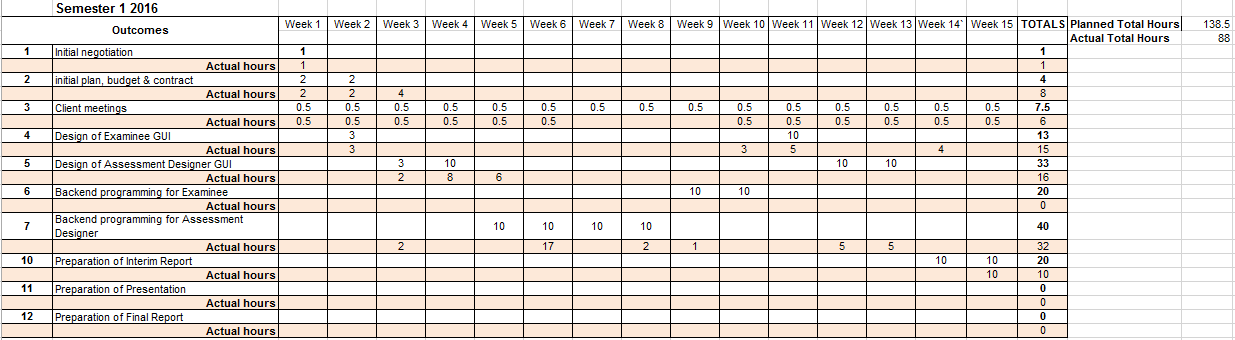
# References

1. T Roberts, Exam Manager, Private Communication, 2016.
2. SmtpClient Class [Online], Accessed: https://msdn.microsoft.com/, 18 November 2016.
3. File Class [Online], Accessed: https://msdn.microsoft.com/, 18 November 2016.
4. TreeView Class [Online], Accessed: https://msdn.microsoft.com/, 18 November 2016.

# Appendices

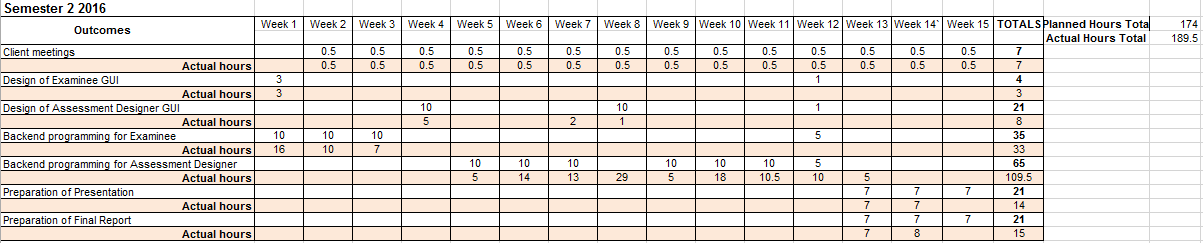
#### Semester 1 Time Plan

Figure 6: Semester 1 Time Plan



#### Semester 2 Time Plan

Figure 7: Semester 2 Time Plan



As is shown in Figs. 7 and 8, there was a 35 hour difference between the planned time for the project and the actual time used. This was caused by overestimation of the time it would take to complete some parts of the project and also by external events disrupting development of the project.

#### Project Budget

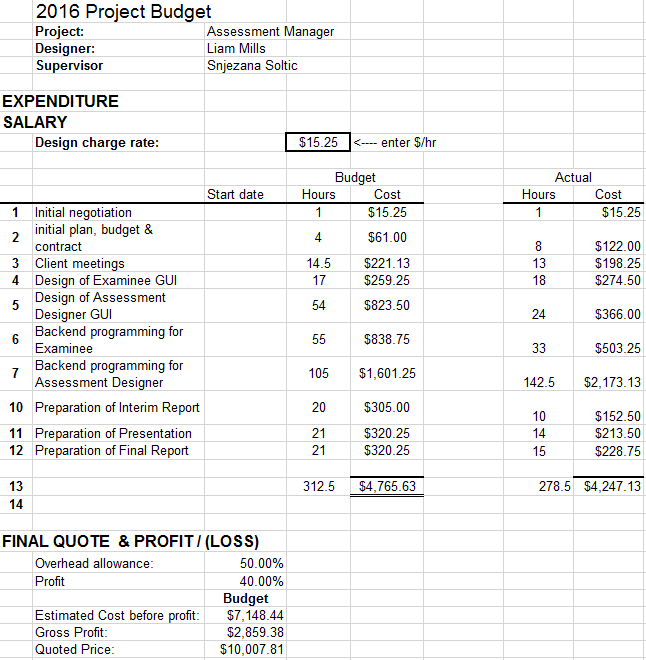


Figure 8: Project Budget