Development of a Web Based Timetable System

Hayley McCafferty

1711751

*April 2012*

Dissertation submitted in partial fulfilment for the degree of *BSc (Hons) in Computing Science*

Department of Computing Science and Mathematics

University of Stirling

Abstract

This dissertation involves creating a web-based Timetable system for the University of Stirling. The system is needed as it will help staff members prepare and plan timetables each semester. The system used personally by a member of staff does work but is not very user friendly due to having no interface to work with. The system developed was set out to help assist students, staff members and admin users and making their lives a little easier by having a piece of software work out completed problems that follow creating timetables. It will help unite multiple course co-ordinators across the university ensuring that not too many clashes will occur in the timetable system. This system was built by following a mixture of 2 software development techniques: waterfall and agile programming. With this a web application was developed that is basic but and although not all the objects were met it still has great potential for future development if someone chooses to do so.

Attestation

I understand the nature of plagiarism, and am aware of the University’s policy on this. I certify that this dissertation reports original work by me during my University project except for the following:

* The code discussed in Section 3.4.1.3 was written by my supervisor.
* The code to display a timetable as discussed in Section 3.4.1.4 was written by my supervisor.

**Signature** **Date**

Acknowledgements

First of all, I would like to express my sincere thanks to my supervisor, Dr. David Cairns for giving me support and support throughout the duration of the entire project. He helped greatly with providing suggestions, testing out my project as it has been written and providing valuable comments for the system that I have developed. He has also helped teach me skills that will prove to be handy in the future. Thank you sir!

Next I would like to say thank you to my family members, especially my parents and my sister for giving the support that I needed throughout the development. They have encouraged me to keep on fighting when the going got tough and I came to a halt. You three have been the rock when coding at home.

I would also like to thank all the staff at the Computing Science department at University of Stirling, without them I would not have learnt all the skills needed for this project and for future projects.

Lastly I would like to give my grateful thanks to my close friends, Miss Ezinne Nzewi and Miss Emma Couper. Without them I would have broken down a long time ago, they have helped provide advice and suggestions for my project. You guys have been my rock at university.

Table of Contents

You can automatically generate a table of contents from paragraphs of style *Heading N* and *Unnumbered N*. To update this after revisions, right-click in the table and choose *Update Field* for the entire table.

[Abstract 1](#_Toc322669531)

[Attestation 2](#_Toc322669532)

[Acknowledgements 3](#_Toc322669533)

[Table of Contents 4](#_Toc322669534)

[List of Figures 5](#_Toc322669535)

[Abbreviations 6](#_Toc322669536)

[1 Introduction 7](#_Toc322669537)

[1.1 Background and Context 7](#_Toc322669538)

[1.2 Scope and Objectives 7](#_Toc322669539)

[1.3 Achievements 7](#_Toc322669540)

[1.4 Overview of Dissertation 8](#_Toc322669541)

[2 State-of-The-Art 9](#_Toc322669542)

[2.1 UniTime 9](#_Toc322669543)

[2.2 UniTime in relation to project 10](#_Toc322669544)

[2.3 Timetabler 10](#_Toc322669545)

[2.4 Timetabler in relation to the project 11](#_Toc322669546)

[3 Development Process 12](#_Toc322669547)

[3.1 Requirements 13](#_Toc322669548)

[3.2 Design 14](#_Toc322669549)

[3.2.1 Database Design 14](#_Toc322669550)

[3.2.2 System Design 16](#_Toc322669551)

[3.2.3 User Interface Design 16](#_Toc322669552)

[3.2.4 Back End Design 17](#_Toc322669553)

[3.3 Implementation 21](#_Toc322669554)

[3.3.1 Technology 21](#_Toc322669555)

[3.4 System Functions 24](#_Toc322669556)

[3.4.1 Admin Functions 24](#_Toc322669557)

[3.4.2 Student Functions 28](#_Toc322669558)

[3.4.3 Staff Functions 30](#_Toc322669559)

[3.5 Testing 30](#_Toc322669560)

[3.5.1 Usability Testing 30](#_Toc322669561)

[4 Conclusion 32](#_Toc322669562)

[4.1 Evaluation 32](#_Toc322669563)

[4.2 Future Work 32](#_Toc322669564)

[References 35](#_Toc322669565)

[Appendix A Admin Sequence Diagram 36](#_Toc322669566)

[Appendix B Usability Testing Student Questionnaire 37](#_Toc322669567)

[Appendix C Usability Testing Staff Questionnaire 39](#_Toc322669568)

[Appendix D Usability Testing Admin Questionnaire 41](#_Toc322669569)

[Appendix E Student User Guide 45](#_Toc322669570)

[Appendix F Staff User Guide 47](#_Toc322669571)

[Appendix G Admin User Guide 49](#_Toc322669572)

List of Figures

You can automatically generate a list of figures from paragraphs of style *Figure*. To update this after revisions, right-click in the table and choose *Update Field* for the entire table.

[Figure 1. UniTime User Interface 9](#_Toc322669511)

[Figure 2. Timetabler User Interface 11](#_Toc322669512)

[Figure 3. Waterfall Development Cycle 12](#_Toc322669513)

[Figure 4. Agile Programming Development Cycle 12](#_Toc322669514)

[Figure 5. Student Use Case 13](#_Toc322669515)

[Figure 6. Staff Use Case 13](#_Toc322669516)

[Figure 7. Admin Use Case 14](#_Toc322669517)

[Figure 8. Classes Database Design 15](#_Toc322669518)

[Figure 9. Users Database Design 16](#_Toc322669519)

[Figure 10. Mock Up User Interfaces 17](#_Toc322669520)

[Figure 11. Student Flow Diagram 17](#_Toc322669521)

[Figure 12. Student Sequence Diagram 18](#_Toc322669522)

[Figure 13. Staff Flow Diagram 18](#_Toc322669523)

[Figure 14. Staff Sequence Diagram 19](#_Toc322669524)

[Figure 15. Admin Flow Diagram 20](#_Toc322669525)

[Figure 16. Model of Servlet and Servlet Container 22](#_Toc322669526)

[Figure 17. Life of a JSP file 23](#_Toc322669527)

[Figure 18. Admin User Interface 25](#_Toc322669528)

[Figure 19. Text File Format 26](#_Toc322669529)

[Figure 20. Timetable Output 29](#_Toc322669530)

# Abbreviations

JSP – Java Server Pages

CSV – Comma-Separated Values

PDF – Adobe Portable Document Format

Bot – (Internet robot) Software applications that run automated tasks over internet.

Java EE – Java Enterprise Edition. Platform provides API and runtime environment for developing and running enterprise software, including network and web services.

IDE – Integrated Development Environment

CSS – Cascading Style Sheets

# Introduction

## Background and Context

This project involves designing and implementing a timetable system that is web-based to be used by the teaching staff of the Computing Science and Mathematics department of University of Stirling. It is hoping that on final release it may be used by every department of the university. It will be used by the teaching staff to help in the preparation and checking of timetables. It will be used by students across the university to generate their class timetables which will display all information required for that semester.

## [Scope](http://www.cs.stir.ac.uk/~kjt/research/conformed.html) and Objectives

* The system aims to create a system which will be able to take a text file generated by the University’s central timetabling office and store in a database.
* The system will allow admin staff (e.g. Course Coordinator) to edit timetables through the use of a database.
* The system also aims to allow staff and students to generate timetables through a dynamic output by searching for particular modules.
* Will allow a user to export data to an appropriate file format (PDF or Microsoft Excel)
* If a particular module does not exist then the system will inform the staff member or student by displaying an error message with an appropriate message.
* The system will have several graphical user interfaces that will be web based.
  + An interface for students to generate a timetable for selected modules.
  + An interface for staff to generate a timetable for selected modules.
  + An interface for admin staff to edit and remove classes also to add and remove admin users.
* The system must be secure to ensure the database is not exploited and if so action can be taken appropriately.

## Achievements

The system that has been developed so far implements the main critical features that are required of this system:

* The system aims to create a system which will be able to take a text file generated by the University’s central timetabling office and store in a database.
* The system will allow admin staff (e.g. Course Coordinator) to edit timetables through the use of a database.
* The system also aims to allow staff and students to generate timetables through a dynamic output by searching for particular modules.
* The system will have several graphical user interfaces that will be web based.
  + An interface for students to generate a timetable for selected modules.
  + An interface for staff to generate a timetable for selected modules.
  + An interface for admin staff to edit and remove classes also to add and remove admin users.

The following features have been implemented but are not fully functional and therefore would require further work to implement them:

* The system must be secure to ensure the database is not exploited and if so action can be taken appropriately.
* If a particular module does not exist then the system will inform the staff member or student by displaying an error message with an appropriate message.

Unfortunately some features were not implemented during the building of this system due to time constraints and would be ideal to implement those in future development of the system if required:

* Will allow a user to export data to an appropriate file format (PDF or Microsoft Excel)

## Overview of Dissertation

This report will go through the different stages of building the Web-Based Timetable System. It will discuss the requirements of the system and how they were gathered. How these requirements were transcribed into ideas at the design stage leading to how these design ideas are implemented into code. Followed by how the code was tested and evaluating how well the system turned out.

# State-of-The-Art

There are many systems that exist in the world which are very similar to the system that is being developed through this project. Some examples are as follows:

* **UniTime**
* **Timetabler**

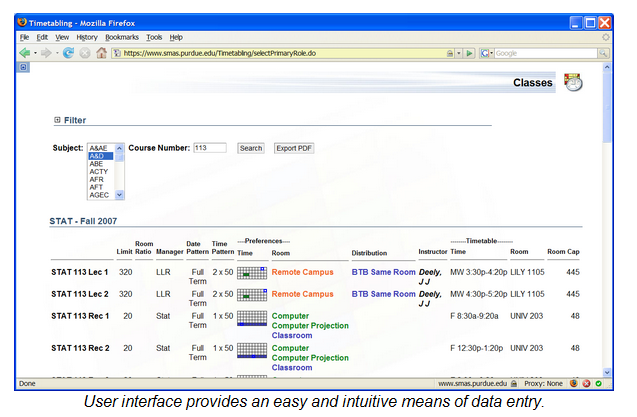
## UniTime

UniTime is an open source web-based enterprise system that uses a combination of Java Servlets and Java Server Pages to create the system. It is platform independent with nearly all development of the program being done in Java and SQL.

It is widely adaptable to fit the needs of every school or university that wishes to use the program. The system is currently in use successfully at Purdue University, Indiana, United States of America.

Users are able to automatically import data into the SQL database by the means of file execution very much similar to the system that is being developed for this project. Users are also able to manually input data if an extra class much be added to the curriculum. When data is input into the database, the system will consistently check and identify any inconsistencies in the information being passed in. This helps identify clashes at the very start rather than later on down the lines.

As it is a web-based system then many users are able to access the system and access the data held within the database. It also means that many staff and students are able to generate their personal timetables. [1]

1. UniTime User Interface

## UniTime in relation to project

In comparing UniTime (a system which is constantly in development) against the system that this dissertation is developing, it is clear to see the similarities between the two of them. The importance of a timetabling system to be web-based is critical when the system is in use at a large establishment (for example a high school or university). The system is used by many users and therefore if the system was not web-based then it would be a hindrance for the software to be installed on each computer that is required for use.

The project that has been developed through this dissertation is basic in comparison to the system of UniTime. It is hoping that future work on this project will include a lot more features which are similar to those in UniTime. These features include checking for inconsistencies between classes and exporting the data inside the database to an appropriate file format. The exporting of data would allow a user to use the data elsewhere in the process of building timetables.

UniTime could be seen as a fully extensive version of the system being developed and it may be possible to implement majority of the functions featured in their system to the system in production at a later date.

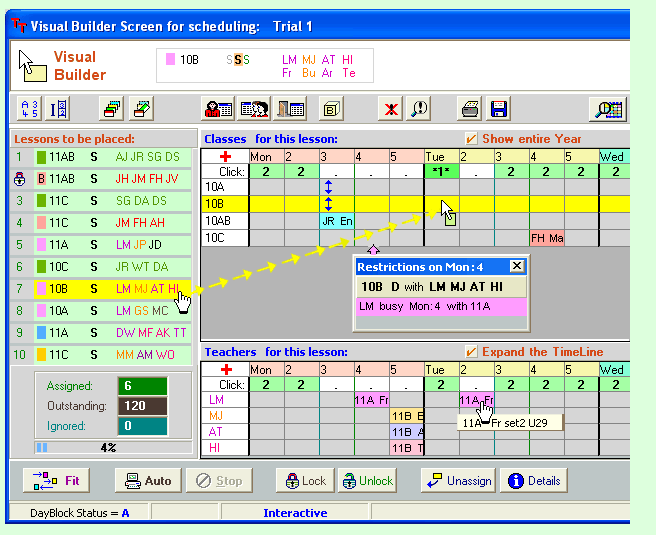
## Timetabler

Timetabler is an interactive web based time tabling software that is used by many schools and colleges throughout the world.

The system is very intelligent and easy to use for everybody and support is given all the time if a user has any questions on how to operate the software. It is also constantly being updated with new features added to the release every month it seems.

The system is used in a step by step process: Entering data about school or college, entering data you want to timetable, schedule the actual timetable and the printing and publishing the timetable. This also includes exporting it to various other pieces of software that can be used in conjunction with it. As data is passed inside the system, it will automatically check and ensure that no two entities are the same which would result in a clash.

If the user has a class that they must include in the timetable then the software will work out an appropriate slot by looking at patterns of already built timetables and calculate the outcome using internal algorithms. It will then present the user a suggested list of times and dates on where the class could be put. If the class is of high priority then the system can move lower priority classes in a ‘musical chair’ fashion. It will also display any problems that may occur if such class was to be entered into the timetable. See Figure 3 for a screenshot of the software in operation. [2]



1. Timetabler User Interface

## Timetabler in relation to the project

In the comparison of Timetabler (a system which is also constantly in development) against the system that this dissertation is developing, it is not as clear to see the similarities between the two of them. It was mentioned earlier that this system is web-based (as stated in 2.3) but this is not strictly true. It is a software package that has to be installed onto the server for it to be used via web browsers which causes implications at the beginning as the server will need to be set up in order to handle the many requests that will be passing through the server.

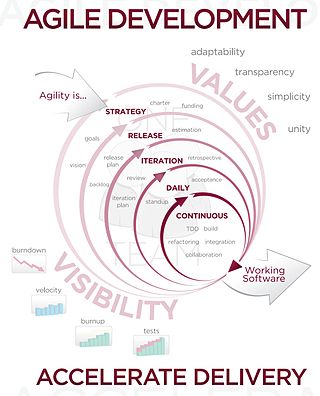
In relation to having features similar to this system in the project it will be the basic concept of creating a timetable. The software was used to allow teaching staff a much smoother process in creating timetables for a large organisation (such as a university) and it is the same reason why this dissertation is being carried out at University of Stirling.

# Development Process

On building this system it was important to follow a software development cycle to ensure that a high standard of system was built. At the beginning of the development of the timetable system the waterfall method was approached. However as progress happened throughout the development a mixture between the waterfall method and agile programming was adopted.



1. Waterfall Development Cycle



1. Agile Programming Development Cycle

## Requirements

To build the Timetable system it was required to lay out what the system fully required. To do so would involve looking at the previous system that was developed by Dr. Cairns for his own personal use. This system was fully functional but was not user friendly as there wasn’t a user interface to interact with the program. The old system also had no means of storage and therefore required a text file with data to be parsed each time a user wished to use the system.

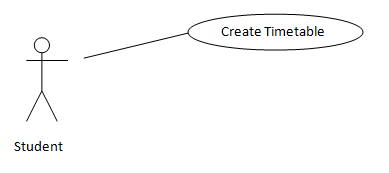
From looking at the previous system, requirements for the new system were thought up:

* System will incorporate a user interface.
* System will store data within program (e.g. a database).

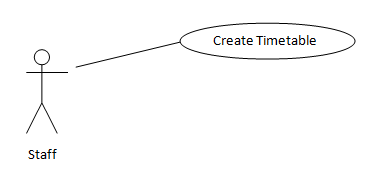
Upon asking potential users on what they would require in this sort of system the following requirements were drawn up:

* System will generate a timetable displaying a user’s classes for that semester through dynamic output.
* System will highlight if any clashes occur between classes.
* System will parse a text file containing data about classes and store in a database.
* System will allow an advanced user to edit timetables.

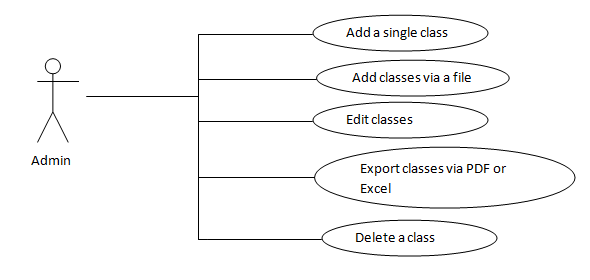
From these requirements, use cases for three different users are drafted up as seen below:



1. Student Use Case



1. Staff Use Case



1. Admin Use Case

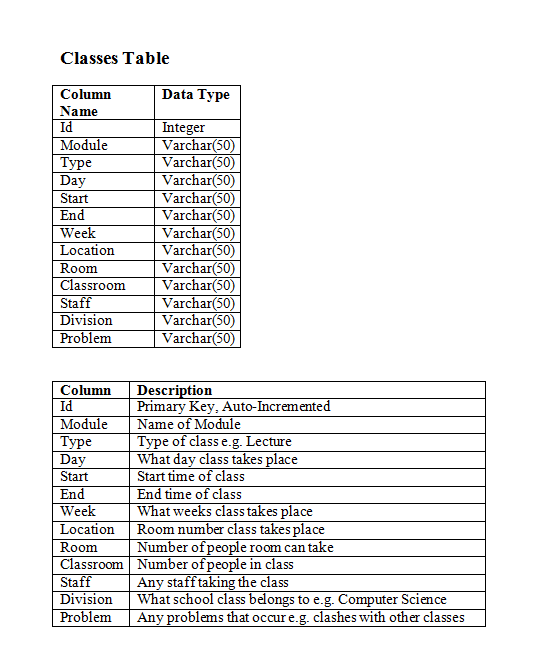
## Design

After the requirements were written up it was time to move onto the design stage of software development. As this system is mainly web-based there were some changes to an original software development process.

### Database Design

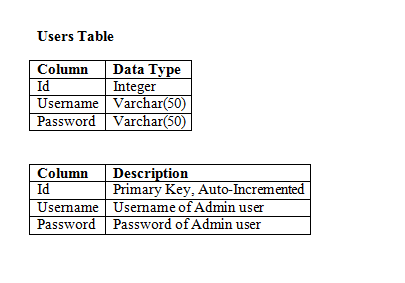
The inclusion of a database to store data gathered by the system is hoping to improve efficiency of the system. Parsing a text file can be slow and does not allow the system to be used on multiple machines unless the user has the required text file. By having a database it allows data to be accessed by multiple users and ensuring that the data is correct and up to date.

It was decided to have a single database as the project is still small but further development ay implement use of multiple databases. The use of two separate tables for both Admin user details and the classes for that particular semester as seen below:



1. Classes Database Design

Variable names were decided upon looking at an example text file that will be used to upload classes into the database. Not all these variables will be used when inserting a class manually as the Admin user may not know the information required.



1. Users Database Design

This table will hold information for Admin users to log in to the system to use the more advanced features such as adding and removing classes.

#### Professional Issues

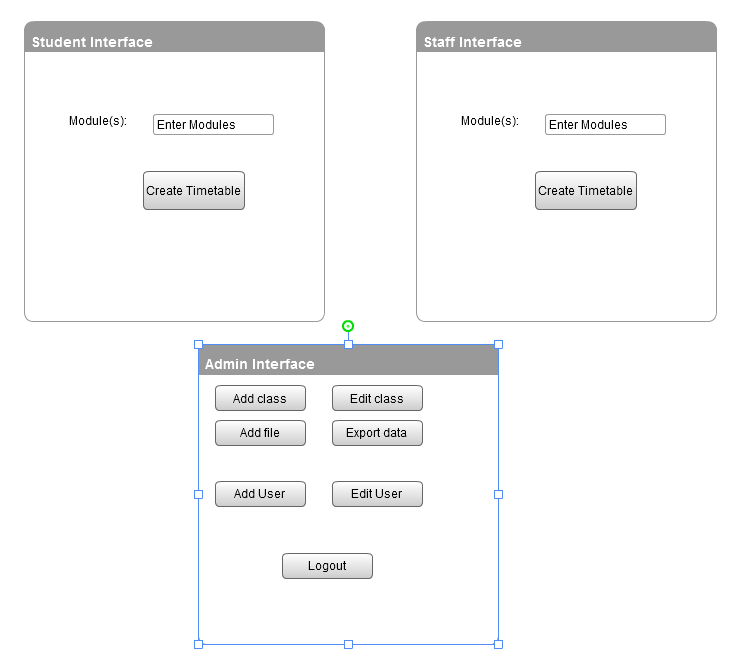
By the inclusion of databases in this system, Admin users must adhere to the Data Protection Act (1998). They must ensure that the data is only used for the required purpose of information about classes at University of Stirling. Admin users must also ensure that the data is up to date and correct.

### System Design

The overall system is required to be portable due to the possibility of many users using the system. It was decided that the system should be web-based to allow easier access to the system and not needing the requirements of installing the software onto individual systems, except the server.

### User Interface Design

One key but important difference about this system is the inclusion of a user interface. It was decided that due to having three types of users; Student, Staff and Admin, it would be required to have three separate user interfaces as seen below:

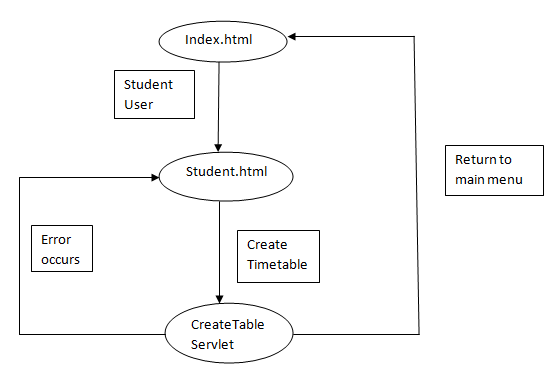


1. Mock Up User Interfaces

At this stage the design is very basic but will help create the final design at implementation stage.

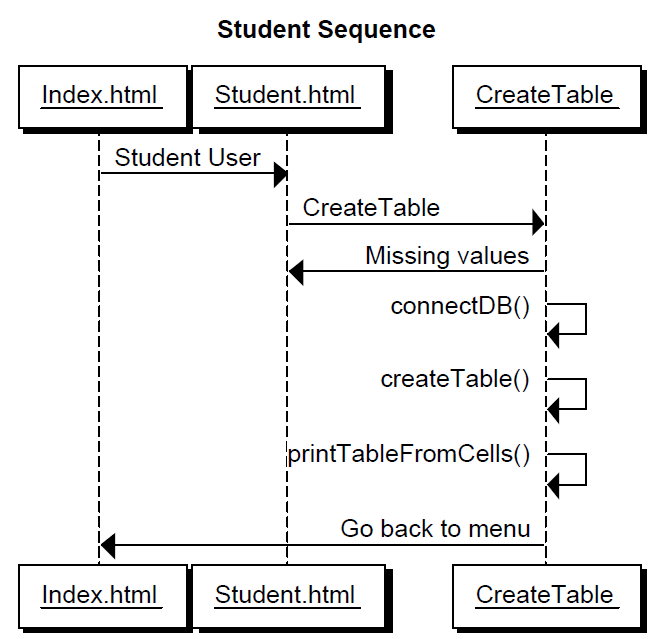
### Back End Design

For the main part it was decided to use a mixture of HTML, JSP and Servlets to build the system.



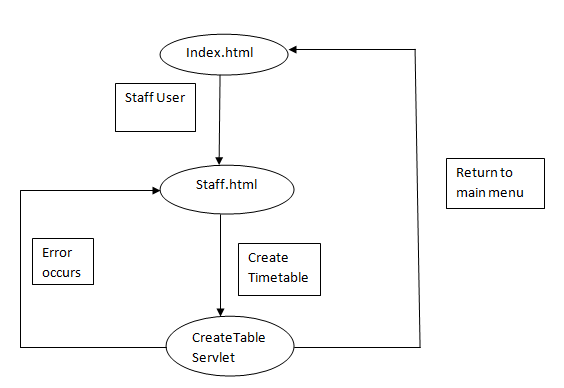
1. Student Flow Diagram

Figure 11 displays the basic flow if the system if a Student user was to use it. From this a sequence diagram is created that displays methods to be used in the system. The sequence diagram is shown in Figure 12.

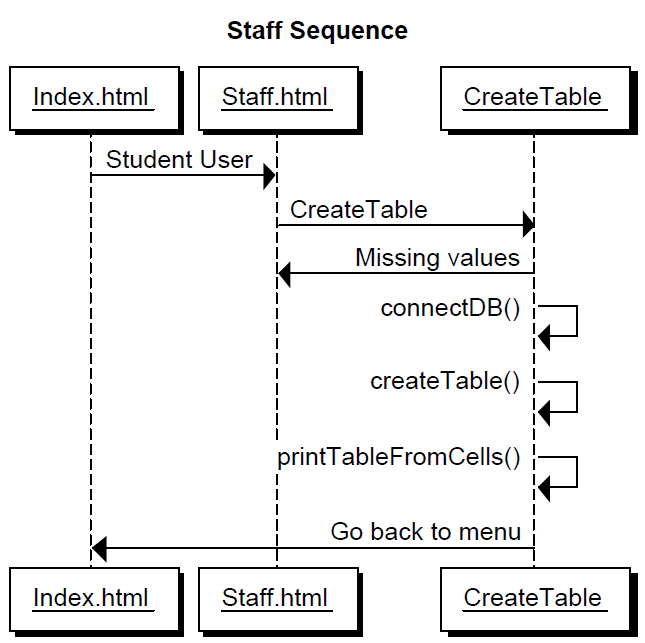


1. Student Sequence Diagram

Figure 13 displays the basic flow if the system if a Staff member was to use it. From this a sequence diagram is created that displays methods to be used in the system. The sequence diagram is shown in Figure 14.

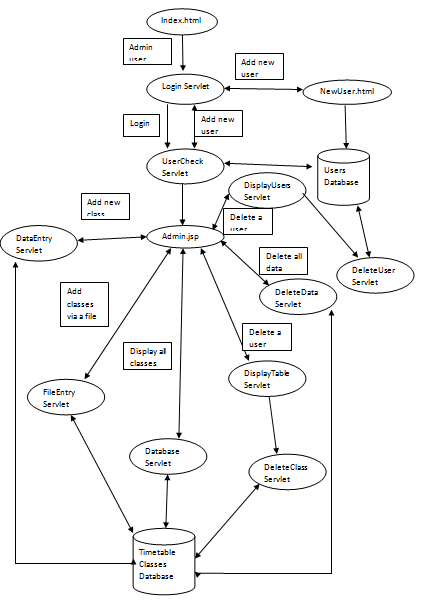


1. Staff Flow Diagram



1. Staff Sequence Diagram

Figure 15 shows a flow layout of how an Admin user may interact with the system. The Admin part of the system is more complicated as it involves more functions than what a Student and Staff user have access to. From this a sequence diagram is created that displays methods to be used in the system. The sequence diagram produced is very detailed at first glance can appear to be very complicated. The diagram is located in Appendix A.



1. Admin Flow Diagram

## Implementation

### Technology

It was decided that as the system would be web-based the following technology would be used to develop and run the system:

* Eclipse IDE for Java EE Developers
* SQLite
* Servlets 3.0
* JSPs
* Valid HTML
* Javascript
* Apache Tomcat 7
* CSS

#### Eclipse IDE for JAVA EE Developers

Eclipse was chosen as the IDE to develop this system due to the ability to code and test the system easily. It contains the plugins required to run an internal server from the computer without relying on an external server.

#### SQLite

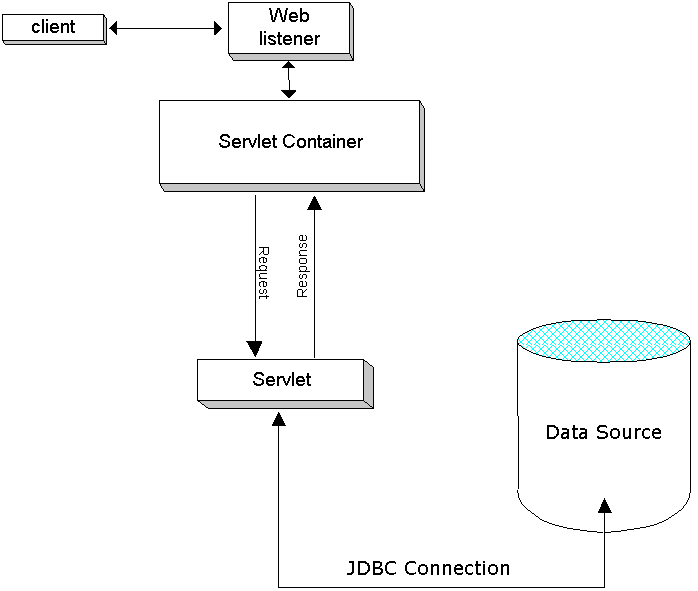
SQLite was chosen as the SQL database to be used with the system. The reasons for choosing SQLite are highlighted below:

* SQLite is open-sourced
  + All the code is publically available meaning that the code is free to use in any application that a developer may be producing.
  + There is no need to purchase a licence to use the code in software that is being developed.
* Self-contained
  + SQLite is very independent as it requires very little support from external libraries that exist or libraries contained within an operating system.
* Serverless
  + Unlike many other SQL databases (such as MySQL), programs do not require to make a separate fetch request to a server as SQLite is stored within the program. In the case of a web service, the database is uploaded alongside the system itself.
  + With the database being contained within the coding, the method that requires access to the database can read and write directly to it – therefore cutting out the middle man.
  + There are advantages and disadvantages to being Serverless.
  + Advantages include that there is not a separate process for setting up the database as it is all done internally in the code. Meaning that SQLite is a zero-configuration database engine.
  + SQLite can allow multiple systems access the same database at the exact same time due to the read/lock system implemented in the code.
  + The disadvantages are that there is no protection from any bugs that can occur in a client program. Meaning coding bugs can corrupt the database causing data to not be consistent.
  + If the database was stored on a separate server then it can control who has access to the database and when they can access it.
* SQLite is Transactional
  + A transactional database is a database of which all changes and queries that are carried out appear to be Atomic, Consistent, isolated and Durable (ACID)
  + This means that any changes that are made within a transaction in SQLite are either done completely or not at all – this can occur if the transaction is interrupted by a sudden crash or power failure to the computer in use.
* It is reported that SQLite is the most widely deployed SQL database
  + Although it is not possible to measure this it is known that major companies such as Oracle, Adobe, Apple and Mozilla all use SQLite in many of the products that they produce.
  + SQLite can be used on large systems such as servers and desktop PCs, to small devices such as mobile phones and MP3 players.
  + Mozilla use SQLite in Firefox to manage and store user’s bookmarks. Apple uses SQLite in their portable devices and software named ITunes.

#### Servlets 3.0

Servlets 3.0 was chosen to develop the web based system. A Servlet is Java based server side web technology that will take a client’s request and receive a response from the server, much like PHP. A Servlet is a class in Java EE that complies with the Java Servlet API.

Unlike PHP which is interpreted by the server which then generates a web page to be passed to the client, a Servlet is deployed inside a Web container and is ran from there. A container is a component of a web server that acts as a middle man in the interaction between a Servlet and the server itself.



1. Model of Servlet and Servlet Container

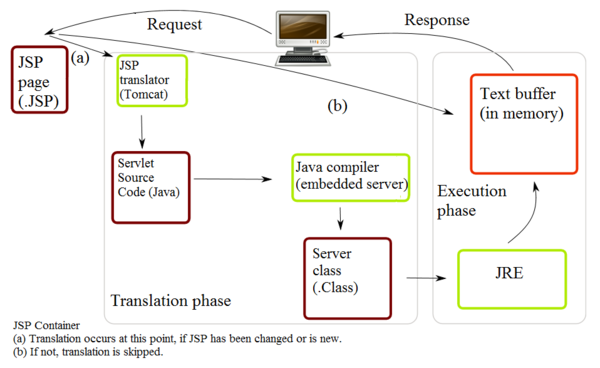
A Servlet is able to produce dynamic content on server-side much like PHP. The one difference is that a server will translate the whole Servlet at once and run it through the container and afterwards will cache it for future use. From this process a client who sends a request from their browser and gets a reply from the server is sped up, as PHP requires the server to interpret the response each time.

Servlets 3.0 was chosen in particular due to the easiness of building a web application. Before Servlets 3.0 it was required to have a Web.xml file. The Web.xml file is the Web Application deployment Descriptor for the web application. It is a XML document that defines everything about the web application on which the server is required to know – e.g. initialisation parameters, welcome pages and URL mappings. However with the introduction of Servlets 3.0 the Web.xml file can be omitted completely as all this information can be contained within the Servlet itself. This makes it easier to customise URLs for each Servlet and different parameters required by the Servlet itself.

Servlets 3.0 was also chosen due to the inclusion of a feature that allows a user to upload files to the web application. This is key feature as in previous versions of Servlets it was possibly with third party libraries but was very complex to code. Servlets 3.0 has reduced the number of lines of code drastically.

#### JSPs

Java Server Pages were chosen to be used where Java code is required but on a HTML page. Therefore it will be used when a user logs out of the system and for the Admin Interface. The diagram below show the typical life of a JSP file:



1. Life of a JSP file

A JSP file can be viewed as being a high-level abstraction of Java Servlet as they are translated into Servlets at run-time and much like Servlets they are cached and reused until the original JSP file is modified. JSPs are often easier to write than a Servlet as they are just ordinary HTML files that contain snippets of Servlet code. However for a more complicated function such as creating a timetable or adding a class then a Servlet may be a better option. JSPs are useful if majority of the output is just HTML.

#### Valid HTML

To comply with web standards, HTML 4.01 used throughout the system must be valid to ensure that no errors exist. It also allows the web pages to be displayed correctly on whatever browser the user wishes to use. W3Schools was used to validate each of the web pages created.

#### Javascript

Javascript is used to display the menus inside the Admin User Interface. Javascript is not an ideal solution as in the past users have used Javascript to exploit web pages meaning that some users will have turned off Javascript automatically. In the case for this system it will not allow any menus to be displayed rendering the Admin Interface useless.

Javascript is useful for error checking and validating form data before passing the information to the server. This saves the server sending out multiple send and fetch requests as the validation checking is all done on the client side of the web application.

#### Apache Tomcat 7

This version of Tomcat was used as Servlet container as it is the latest release of the Apache Tomcat server and it allows the use of Servlets 3.0 and JSP 2.2. It is hoping that a lot of companies will be upgrading to this version of Tomcat if they are currently using Tomcat 6.

#### CSS

CSS is used to help design the web pages and most importantly to hide elements for a user to print out their timetable. CSS has helped keep a distinct style constant throughout the system and allows all web pages to be changed from one file rather than entering multiple files to change different values. CSS is a very powerful tool and can help create intelligent looking web sites for people to see and use.

## System Functions

### Admin Functions

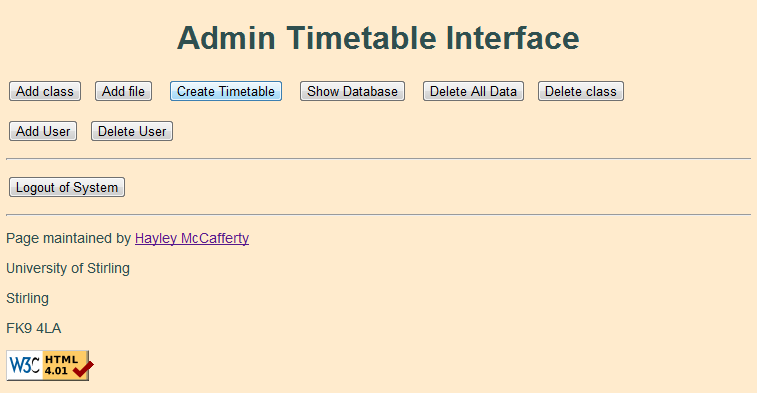
The functions that an Admin user will able to use are the following:

* Add a single class via interface
* Add classes via a text file uploaded to server
* Ability to edit classes
* Ability to export classes via PDF or Excel file format
* Ability to delete a class

All the above functions will require an Admin user to log into the system. An Admin user will enter the system from the main Index menu and will be passed to a web page asking the user to log in to the system. If the system is new or there are no known users in the database and a user attempts to log in then a message indicating that no users exist and that they must register to proceed. The process of registering as a new user will be discussed later. An Admin user will be passed to the Login Servlet and asked to enter their Username and password which will be checked in the UserCheck Servlet. At this stage a new session will be set up with a maximum interval of 10 minutes.

UserCheck will check which web page the request has came from; in this case it will ensure that a Username and Password has been passed over and if not the user will be prompted to try and log in again. UserCheck will take the Username and Password given and check it against the table Users inside the database. If the Username and Password both exist and are correct then the system will confirm they are a user and invite them to enter the system. If the Username and Password do not exist or they do not match then the system will prompt user to either try logging in again or register an account.

On success of entering the system the Admin user will be presented with the Admin User Interface as seen below:



1. Admin User Interface

From this interface the user will be able to carry out the functions highlighted at the start of Section 3.4.1.

#### Admin Interface

The user interface is basic but is very similar to the mock up interfaces as seen in Figure 8. The interface is built up from many HTML elements and the functions are operated through the use of Javascript. This is not an ideal situation as users have the ability to disable Javascript to run through their web browser. If the user has turned off Javascript, then the buttons will not display anything – this is a design fault and needs to be addressed. The buttons correspond to the functions as follows:

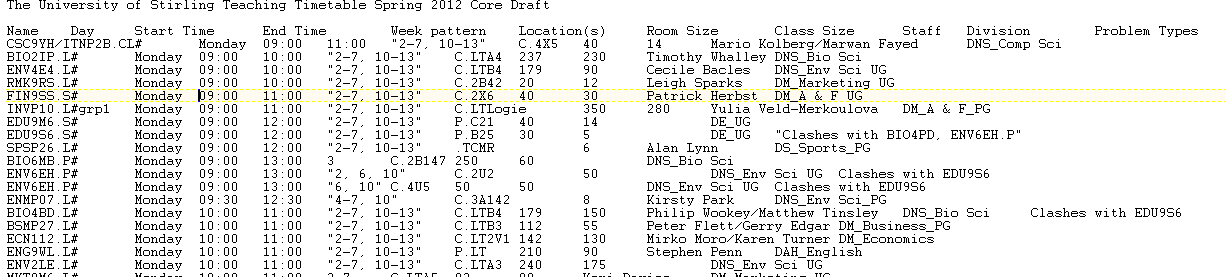
* Button: “Add Class” – Add a single class to the database
* Button: “Add file” – Add multiple classes via a text file to the database
* Button: “Create Timetable” – Create timetable showing classes for module(s)
* Button: “Show Database” – Show all classes currently stored in database
* Button: “Delete All Data” – Deletes all classes from the database
* Button: “Delete class” – Delete a single class from the database
* Button: “Add User” – Add new Admin user to the system
* Button: “Delete User” – Delete an Admin user from the system
* Button: “Logout of System” – Log user out of the system

#### Add a single class via interface

An Admin user has the ability to add a single class via the above interface. It does not ask for all details that are added when a user inserts classes via a text file. The thought behind this is that the user may not know the exact details required by these fields and are therefore filled with an empty string to indicate no data exists. The only field that can remain empty on submission is Staff; therefore all other fields must have some value stored inside. At this stage there is no error checking to ensure that a valid module has been entered. This could be a feature implemented in a further version of the system.

The values are passed to a Servlet named DataEntry. If there are values required that are not present then the user will be prompted to enter the missing values required. DataEntry will combine the teaching building and room location together as this is the format it is stored inside the database. The Servlet will create a new table if it does not exist and add a new class with the values given. Any missing values will have an empty string value allowing it not to be null. On success the user will be notified with a summary of the class they have added and invited to return to the Admin Interface.

#### Add classes via a text file uploaded to server

The Admin user has the ability to add multiple files via a text file which can be uploaded to the server. The text file is generated from a Microsoft Excel document that has been designed by the Central Timetable department at the University of Stirling. The only format that would be accepted is seen below, however there is no full check in place but any other format will cause the database to have errors:

1. Text File Format

Each value is separated by a tab and contains eleven variables: Name, Day, Start Time, End Time, Week Patterns, Locations(s), Room Size, Class Size, Staff, Division and Problem Types. Once the user has uploaded the file from the interface and submitted it to the server, the file is passed to a Servlet named FileEntry.

The text file is not necessarily uploaded but the data inside the file is passed via multiple parts. These parts are then passed into an InputStream which is used to get all the values required in the method dataFile. DataFile will take the InputStream and read in all string values and split them with a delimiter of a tab. The splitting of strings is a complicated process and Dr. Cairns has kindly given permission for code from his Slots class and the methods padTo and addClass to be used. The methods will split up the data correctly into their relevant variables. If a value does not exist then the variable is assigned a question mark to indicate a missing value.

Once all the variables have been assigned values they are passed to the database where they are added a class at a time. After all the classes have been added from the text file, the user is notified on how many classes have been added and is invited to return to the Admin Interface.

#### Create timetable showing classes for module(s)

A user is able to check for classes which clash together. The user will enter multiple modules split up by a single comma and no white space. The information will be passed to a Servlet named AdminTable. AdminTable will check that there are modules passed from the user’s input and if there aren’t then the Admin user is asked to enter values again. On obtaining the required elements the Servlet will capitalise all letters in the module variable if they are not already and pass it to the next method to be used.

CreateDB method helps set up a connection to the SQLite database named “timetable” that is stored alongside the program code of the Servlet. From this database the system will obtain all relevant information regarding classes to the particular module present. If any errors occur at this point, then the Admin user will be notified of what the error is and prompted to return to the Admin Interface.

Once a connection has been opened then the Servlet will split up all modules contained inside the module variable (as there is a possibility of multiple modules codes). This is why module codes have to be entered a certain way at the Admin interface; as if they are incorrect then the system will not recognise the module and will not display the timetable correctly. When the modules have been correctly split up then details of each class that is associated with every module is passed to the next method to set up each individual cell. This method is called getCellData.

GetCellData will set up the table cell with the data required about the class. The information that is displayed includes: name of module, type of class, the room location and what weeks the class will take place on. The next step is to build the table using printTableFromCells.

The method will set up the table to display days of the week (Monday – Friday) along the top and down the left hand side the time (09:00 – 18:00). The method will loop through the days and times that are available whilst building cells with classes in it at the appropriate day and time. The finished table will be built dynamically and displayed to the Admin user on their web browser with a legend indicating what each symbol means in the timetable. The Admin user will also be able to print the timetable with only certain elements being on display to the user.

#### Show all classes currently stored in database

This function will allow a user to display all the classes that are currently held within the table “Timetable”. The Servlet Database will connect to the database and iterate through all classes while displaying them in a table. The only columns that are displayed include: Module, Type, Day, Start Time, End Time, Week Pattern, Location, Room Size, Class Size and any Staff involved. The Servlet will display all classes in a HTML table. At the end of the table the user can go back to the Admin Interface.

#### Deletes all classes from the database

The Admin user will be able to delete all classes from the database on pressing a button. The user is warned on screen that all data will be deleted. The Servlet DeleteData will drop the table containing all the classes allowing classes to be added from scratch. This function is most likely to be used at the beginning of a new semester unless multiple tables are implemented for different semesters.

#### Delete a single class from the database

User has the ability to select a class to delete from the database if they do not wish for it to exist anymore. The user will enter a module code and submit the page to the server which will pass the module code to the Servlet DisplayTable. DisplayTable will search for the given module in the database and display all classes related to that module in a HTML table to the user with an additional “Delete” button alongside it.

When a class is chosen to be deleted the information is then passed on to a Servlet named DeleteClass. DeleteClass will take the class information and remove the class that matches those exact values from the database. On completion the user will be notified on what class has been deleted and if the action has been successful or not.

#### Add new Admin user to the system

This function will allow an Admin user to add another user to the system. The Servlet UserCheck will ensure that all details are present and add the user to the database. In a future release there would be a check that if a Username was chosen that already existed in the database; the user would be required to choose a new Username. It is also at this stage that the passwords would be encrypted to ensure that they are secure and accounts are not exploited.

This function is also applicable at the beginning of the system. If the user does not have an account to enter the system with then they can register for a new account.

#### Delete an Admin user from the system

An Admin user will be able to remove a user from the system if they are no longer required. The Servlet DisplayUsers will display a HTML table containing all the Usernames of the users in the system that exists. The user can then select a single user to be deleted which will be passed to the next Servlet DeleteUser. The Username and Password are passed onto DeleteUser with the Password value contained inside a Hidden field. DeleteUser will take the Username and Password values and remove them from the table “Users”. On success the user will be notified on what user has been removed from the system.

#### Log user out of the system

To exit the system the Admin user must Log out via the Admin Interface to ensure the system is closed properly. By the user clicking the “Log out” button it will invalidate the current session in place meaning that the next time the user will visit the system they will have to log back in.

If a user tries to access any page that requires an Admin account they will be redirected back to the Login Servlet indicating that it is a secure area which requires special permission.

### Student Functions

The main function that a Student user will be able to use is:

* Create a timetable displaying classes from given modules

This function works from the Student User Interface which consists of a HTML file. The Student user is required to enter their name and the modules that they wish to see classes from. The modules must be entered in a certain manner of being separated by a comma with no whitespace in between.

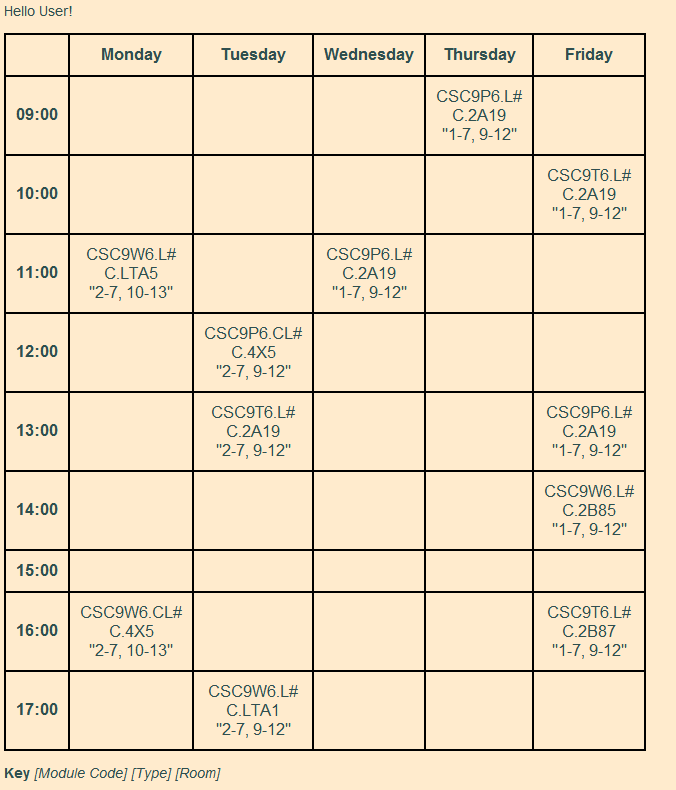
The modules are then passed to a Servlet named CreateTable. CreateTable will check that there are no missing values from the Student’s input and if there are the Student user is invited to try again. On obtaining the required elements the Servlet will capitalise all letters in the module variable if they are not already and pass it to the next method to be used.

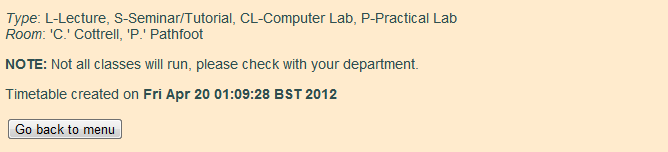
CreateDB method helps set up a connection to the SQLite database named “timetable” that is stored alongside the program code of the Servlet. From this database the system will obtain all relevant information regarding classes to the particular module present. If any errors occur at this point, then the Student user will be notified of what the error is and prompted to return to the Index web page.

Once a connection has been opened then the Servlet will split up all modules contained inside the module variable (as there is a possibility of multiple modules codes). This is why module codes have to be entered a certain way at the Student interface; as if they are incorrect then the system will not recognise the module and will not display the timetable correctly. When the modules have been correctly split up then details of each class that is associated with every module is passed to the next method to set up each individual cell. This method is called getCellData.

GetCellData will set up the table cell with the data required about the class. The information that is displayed includes: name of module, type of class, the room location and what weeks the class will take place on. The next step is to build the table using printTableFromCells.

The method will set up the table to display days of the week (Monday – Friday) along the top and down the left hand side the time (09:00 – 18:00). The method will loop through the days and times that are available whilst building cells with classes in it at the appropriate day and time. The finished table will be built dynamically and displayed to the Student user on their web browser with a legend indicating what each symbol means in the timetable. The Student user is also notified that not all classes will run each week and that it should be checked with the appropriate department. An example screenshot of a timetable is displayed below:





1. Timetable Output

Any classes that occur at the same time (e.g. clash) will appear in the cell after each other. It would be ideal for the clashing class to be highlighted in red text but this is not implemented in this version of the system.

If a user wishes to print out their timetable then the webpage has been set up to only include certain elements: the timetable, the legend and the note highlighting that all classes may not run. When printing the text colour will change to black to allow easier printing.

### Staff Functions

The main function that a Staff user will be able to use is:

* Create a timetable displaying classes from given modules

This function works from the Staff User Interface which consists of a HTML file. The Staff user is required to enter their name and the modules that they wish to see classes from. The modules must be entered in a certain manner of being separated by a comma with no whitespace in between. A Staff user can search for classes with their name – this is not implemented in this version of the system.

The modules are then passed to a Servlet named CreateTable. CreateTable will check that there are no missing values from the Staff’s input and if there are the Staff user is invited to try again. The Servlet will check what web page the user came from and will display a button to go back to the relevant interface. On obtaining the required elements the Servlet will capitalise all letters in the module variable if they are not already and pass it to the next method to be used.

CreateDB method helps set up a connection to the SQLite database named “timetable” that is stored alongside the program code of the Servlet. From this database the system will obtain all relevant information regarding classes to the particular module present. If any errors occur at this point, then the Staff user will be notified of what the error is and prompted to return to the Index web page.

Once a connection has been opened then the Servlet will split up all modules contained inside the module variable (as there is a possibility of multiple modules codes). This is why module codes have to be entered a certain way at the Staff interface; as if they are incorrect then the system will not recognise the module and will not display the timetable correctly. When the modules have been correctly split up then details of each class that is associated with every module is passed to the next method to set up each individual cell. This method is called getCellData.

GetCellData will set up the table cell with the data required about the class. The information that is displayed includes: name of module, type of class, the room location and what weeks the class will take place on. The next step is to build the table using printTableFromCells.

The method will set up the table to display days of the week (Monday – Friday) along the top and down the left hand side the time (09:00 – 18:00). The method will loop through the days and times that are available whilst building cells with classes in it at the appropriate day and time. The finished table will be built dynamically and displayed to the Staff user on their web browser with a legend indicating what each symbol means in the timetable. The Staff user will also be able to print the timetable similar to a Student user with only certain elements being on display to the user.

## Testing

Throughout the development of this system it was tested as a new major feature was added. The coding was thoroughly tested by the coder and members of the public as it was in development. No part was left untested as it was ensured that the function developed was nearly fully working before coding the next function.

### Usability Testing

Usability testing was done once the system was as fully developed as it could have been at this stage. The testing consisted of different members of the public who had various experience in computing. The testing questionnaires can be found from Appendix B to Appendix D with a different questionnaire for each interface. The testing was able to see how the system functions with end users and highlighted areas that required improvement and gave ideas for future possible developments.

All testing was done with 8 testers. 4 members have advanced experience in computing and 4 members who occasionally use a computer.

The Student Interface was marked with an average of 6.5 out of 10 for effectiveness. This is expected as the system is still very basic but does the correct job for the time being.

The Staff Interface was marked with an average of 6.5 out of 10 for effectiveness. Again this is expected as both the Student and Staff user functions are the exact same.

The Admin Interface was also marked with an average of 6.5 out of 10 for effectiveness. It is understandable as it does not quite live up to the requirements stated at the beginning of development but perhaps with further development that rating will rise further.

# Conclusion

## Evaluation

Overall I have found this dissertation an interesting concept. I feel that I have gone on a journey and discovered a lot about myself. I have been amazed at the knowledge I have retained over the years from studying at the University of Stirling.

Reflecting back on the dissertation and building the system from scratch, I feel I may have been over ambitious with the requirements I set myself. I did not expect some things to affect how I developed this system but they did. I am disappointed I was not able to fulfil the requirements stated in section 3.1. If I had more time then maybe I would have achieved the dream, however not all software is perfect the first time around.

I have learnt the importance of following a software development and why each stage is required, it helps in the long run understand what is required of the system and how it can be achieved. It is a case of careful planning is required at each stage and no just diving head first into coding and hoping for the best.

It has been a worthwhile task and has allowed myself to learn more about the software development process, learn new things to do with coding and learn that I should trust my gut instinct more often.

## Future Work

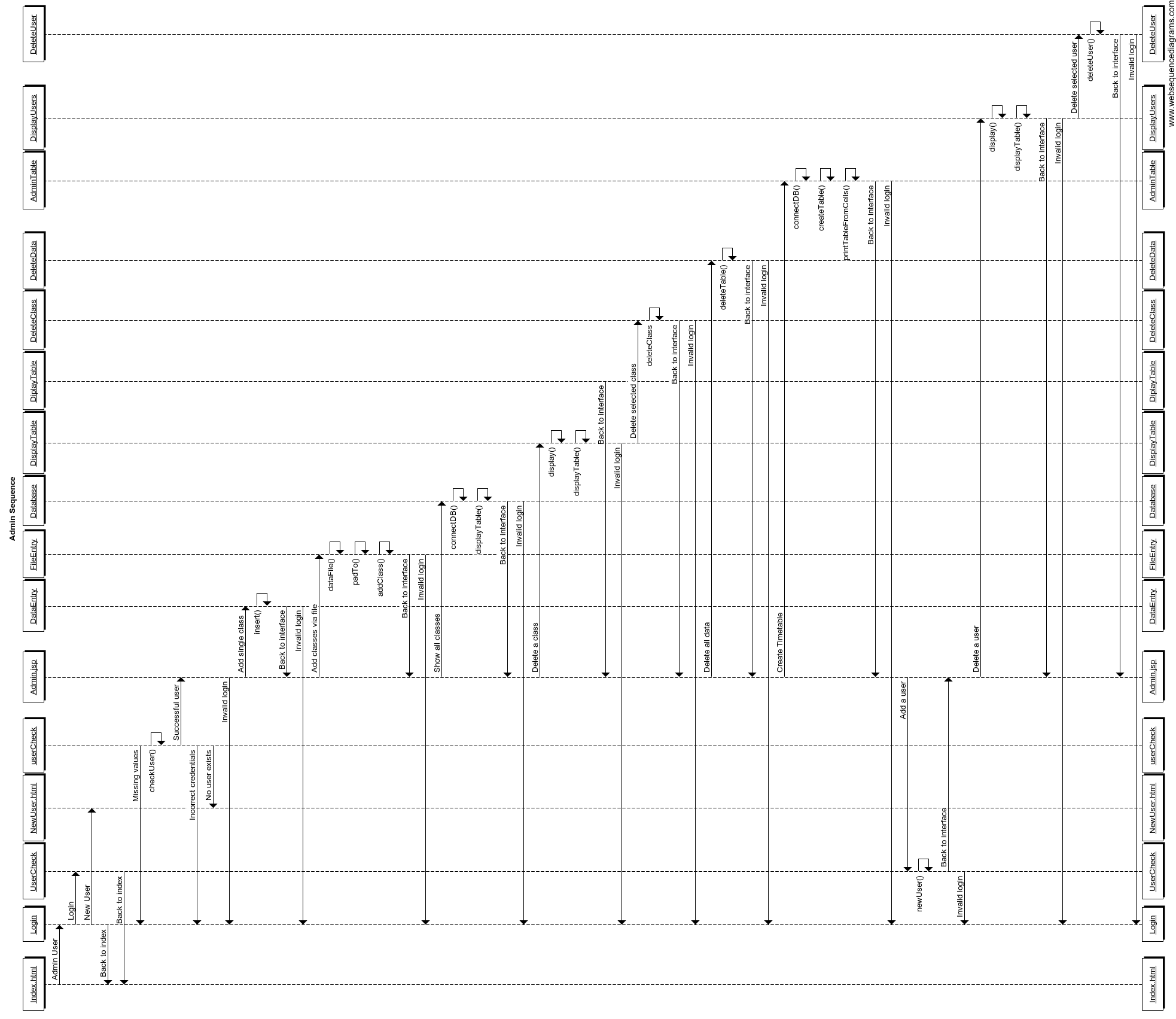
The system developed during this dissertation is very basic in comparison to the requirements produced. It has been disappointing not to be able to fully implement all the features intended due to time constraints and other factors that have occurred during this dissertation. The following will list possible future work to undertake in order of critical need:

1. System needs to be secure enough for full release.
   * Currently the system does store a Username and Password in the internal database but the passwords themselves are not encrypted. With this the system can be compromised and exploited leading to system failure.
   * Encrypt passwords internally possibly using the MD5 algorithm or another algorithm.
2. Logging of system activities.
   * The system will be able to take a note of the time and day on which the system is accessed by an Admin user and log exactly what changes occur during their visit.
   * If any suspicious activity is found; another Admin user will be able to trace the activity, identify the user who carried out the problem and rollback the database to avoid any permanent damage to the system.
3. Backing up of databases on a regular basis.
   * Ability to back up the database at regular intervals to ensure that the data is kept up to date and correct if anything was to ever go wrong.
4. New user verification.
   * To ensure that a user exists and is not a “bot” or that this particular person has permission to be an Admin user – send a verification e-mail to an overall member in charge of system.
   * They will ensure that user is who they say they are and have the ability to allow a user to continue registering or withdrawing their application.
   * This further improves security of the system ensuring that no exploits can happen.
5. Editable classes.
   * Currently this system will only allow an Admin user to add and remove a class. This is not an ideal situation.
   * Admin users should be able to edit details about individual classes such as changing the time or room location.
6. Editable User details.
   * Currently the system will only allow an Admin user to add or remove a user. This is not an ideal situation if a user forgets their password.
   * Admin users should be able to edit only their accounts if they wish to change their password.
   * A function if a user has forgotten their password would also be required if password has been forgotten at log in stage of entering the system. This would require additional security measures; such as a security question, memorable phrase or a memorable date.
7. Proper error checking throughout the system.
   * Currently the system does display error checking in certain areas but not all.
   * When creating a timetable to be printed, if a module is entered but is not found then the timetable may show up blank.
   * It would be ideal for the system to display an appropriate error message highlighting that the class the user is trying to find does not exist. The system will then provide helpful advice to contact their module co-ordinator for further help.
   * Multiple users with same Username are allowed to be added, this is not a great function – can cause confusion.
   * System will highlight that a username already exists in the database with a username that a user is attempting to register. An error message will highlight that a user should choose a different Username.
   * When adding a new class to database, multiple instances of the same class can be added.
   * If the class exists with exact same details then user should be prompted that a class already exists. System may offer user a choice if they wish to save these details and overwrite the class or not.
8. Error messages currently in place not very informative.
   * Error messages only indicate that a problem has occurred and that the user should try again later.
   * It would be appropriate for error codes to be thought up to allow a system manager to understand what sort of problem has occurred.
   * These error codes can be sent by the user via an e-mail indicating when, where and how the problem occurred - allowing a system manager to solve the problem.
9. System will notify Admin user if any clashes occur with a class they are adding.
   * System does not highlight if there are any major clashes between different classes.
   * It would show the Admin user an error message prompting that a clash could possibly occur.
   * In creating a timetable for a student or staff user – highlighting clashing classes in another colour would draw attention for them to contact an appropriate person to discuss the next course of action.
10. A suggestion that was made during development was to have an editable timetable.
    * This would allow an Admin user to click on a class in a timetable and edit the details.
    * The Admin user can make multiple changes and save all changes at the end.
    * Another version of this timetable can be a drag and drop method.
    * Admin users would be able to move classes around by clicking and dragging to the appropriate time and day.
11. On deletion of classes – multiple classes may be deleted at same time.
    * System only allows a single class to be removed each time. This can be time consuming for a user if they have multiple classes to be removed each time.
    * User interface will present a list/table of all classes involving a particular module. An Admin user will be able to select multiple classes and submit them for deletion.
12. Verification that text file input into system is in correct format.
    * System will check that the text file contains appropriate columns for classes to be added from.
    * Text file must contain: Module, Day, Start Time, End Time, Week Pattern, Location, Room Size, Class Size, Staff, Division, Problem Types
    * Realistically, it is unsure if this is achievable currently or not.
13. Staff users can create timetable from searching their name.
    * Staff users currently search for classes they are teaching by searching for modules.
    * In future it is hoping to be possible to search for classes by using their name.
    * This will help identify classes that they are only teaching and not fellow colleagues’ classes.
14. Allow data from database to be exported to an appropriate file format.
    * This will help keep the databases up to date and secure, as this could be a form of back up.
    * This will also allow the central timetabling department at the University of Stirling to have a format to work with as they original send out the files.
    * Appropriate file formats may include: Microsoft Excel or CSV.
15. Improving user interfaces
    * Currently user interfaces can use Javascript which a user can turn off.
    * Redesigning the user interfaces will improve the look and keep it fresh for users of the system.

References

1. UniTime LLC. UniTime Home Page: http://www.unitime.org, March 2012
2. Chris and Keith Johnson. TimeTabler Home Page: http://www.timetabler.com, March 2012
3. Wikipedia. Software Development Process: [http://www.wikipedia.org/wiki/ Software\_development\_process](http://www.wikipedia.org/wiki/Servlet), April 2012
4. Wikipedia. Agile Software Development: <http://www.wikipedia.org/wiki/> Agile\_software\_development, April 2012
5. SQLite. SQLite: <http://www.sqlite.org>, April 2012
6. Wikipedia. Java Servlet: <http://www.wikipedia.org/wiki/Servlet>, April 2012
7. Jones, S. CSC9YD JSP Lecture, March 2012
8. Oracle Docs. J2EE Servlet Developer’s Guide: [http://docs.oracle.com](http://docs.oracle.com/), April 2012
9. W3Schools: http://www.w3schools.com, March 2012
10. Wikipedia. Javascript: <http://en.wikipedia.org/wiki/JavaScript>, April 2012
11. Apache Tomcat: <http://tomcat.apache.org>, April 2012
12. W3Schools. CSS: <http://www.w3schools.com/css/>, April 2012
13. Perry, B: “Java Servlet and JSP Cookbook”, First Edition. O’Reilly, United States of America; 2004
14. Core Servlets & Java Server Pages: <http://pdf.coreservlets.com/>, March 2012
15. Jason Hunter, William Crawford: “Java Servlet Programming”, 2nd edition, O’Reilly UK; 2001

Appendix A Admin Sequence Diagram



Appendix B Usability Testing Student Questionnaire

**Student Interface Usability Test**

**Task – Create a Timetable**

1. Click “Student” button to enter Student Interface. Enter your name and the following modules: CSC9P6,CSC9N6,CSC9W6 (exactly as shown) and click Create Timetable. What happens? How many classes are shown in total?
2. Now right-click and click on “Print…”. Look at the picture displayed, are there any noticeable differences you can see from the previous screen?
3. Close the Print window with the little cross. Click “Go back to menu” and go back into the Student Interface. Click on “Create Timetable” without entering anything. What happens? Is the message clear on what has happened?
4. Click “Student Interface” and now enter just your name and press “Create Timetable”. What happens this time? Do you feel the error message is guiding you well enough?
5. Now go back to the Student Interface and enter your name and the following modules: BUAP25,BIO8ID,MKTP29 and create the timetable. What happens? Is the timetable displayed correctly?
6. Are there any improvements you can suggest? What other features would you like to see for Students?

What would you rate the system out of for effectiveness? (1 – Lowest, 10 – Highest)

**Thanks for taking the time to test out my program and filling out this questionnaire!**

Appendix C Usability Testing Staff Questionnaire

**Staff Interface Usability Test**

**Task – Create a Timetable**

1. Click “Staff” button to enter Staff Interface. Enter your name and the following modules: CSC9P6,CSC9N6,CSC9W6 (exactly as shown) and click Create Timetable. What happens? How many classes are shown in total?
2. Now right-click and click on “Print…”. Look at the picture displayed, are there any noticeable differences you can see from the previous screen?
3. Close the Print window with the little cross. Click “Go back to menu” and go back into the Student Interface. Click on “Create Timetable” without entering anything. What happens? Is the message clear on what has happened?
4. Click “Staff Interface” and now enter just your name and press “Create Timetable”. What happens this time? Do you feel the error message is guiding you well enough?
5. Now go back to the Staff Interface and enter your name and the following modules: BUAP25,BIO8ID,MKTP29 and create the timetable. What happens? Is the timetable displayed correctly?
6. Are there any improvements you can suggest? What other features would you like to see for Staff members?

What would you rate the system out of for effectiveness? (1 – Lowest, 10 – Highest)

**Thanks for taking the time to test out my program and filling out this questionnaire!**

Appendix D Usability Testing Admin Questionnaire

**Admin Interface Usability Test**

**Task 1 – Log in**

1. Click “Admin” button to enter Admin Interface. Login with the following details:

Username: user Password: abc

What happens? Is the error message appropriate for what has happened?

1. Let’s register you into the system. Click on “Register” and enter the above details again and click on “Register”. You are now registered!
2. Click on “Admin” to access the Admin Interface again and login in with the account you have just created.

**Task 2 – Add a class**

**\* AT ANY STAGE YOU ARE LOGGED OUT, LOG BACK IN WITH USER DETAILS PROVDIDED!\***

1. Now click on “Add class”. You are now going to add a class to the database. First without typing any data, click “Submit”. What happens? Do you feel the error message provided is helpful in any way, and can it be improved?
2. Now go back to the Admin Interface and click “Add class”. Now enter the data to the following fields (exactly as typed below!):

**Module:** csc9w6

**Type:** Computer Lab

**Day:** Wednesday

**Start Time:** 10:00

**End Time:** 11:00

**Building:** Cottrell

**Location:** 4x5

**Weeks:** 1-5

**Staff:** Mario Kolberg

Click “Submit” and explain what you see on the screen in front of you.

**Task 3 – Adding multiple classes via a file**

**\* AT ANY STAGE YOU ARE LOGGED OUT, LOG BACK IN WITH USER DETAILS PROVDIDED!\***

1. Go back to the Admin Interface if not done so already and click on “Add file”. Now click on browse and locate the file named “test1.txt” on the Desktop. Press “Submit” when found. How many classes have been added?
2. Go back to the Interface and go to add another file named “test2.txt” located in the same area. How many classes are added this time?
3. Now go back to the interface and press “Delete All Data” and proceed to press “Delete All Data” to remove all the classes in the database.

**Task 4 – Deleting a class**

**\* AT ANY STAGE YOU ARE LOGGED OUT, LOG BACK IN WITH USER DETAILS PROVDIDED!\***

1. Add “test1.txt” file to the system once again and proceed back to Admin Interface when done so. Click “Delete Class”. Enter CSC9W6 into the module and press “Show classes”. How many classes are shown? What do you think of the presentation?
2. Delete the class that contains the class type “CL”. Has the correct class been deleted? Indicate if it has not done so.
3. Go back to “Delete class” again and search for classes belonging to CSC9W6 again. How many classes are shown this time? Now delete the class that takes place on a Friday. Has the correct data been deleted? Indicate if it has not done so.
4. Go back to the Admin Interface and click “Logout of System”. Are you logged out? Then you have finished the test!
5. Are there any improvements you can suggest for the system? What other features would you like to see for Admin users?

What would you rate the system out of for effectiveness? (1 – Lowest, 10 – Highest)

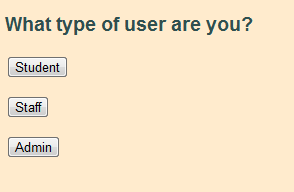
**Thanks for taking the time to test out my program and filling out this questionnaire!**

Appendix E Student User Guide

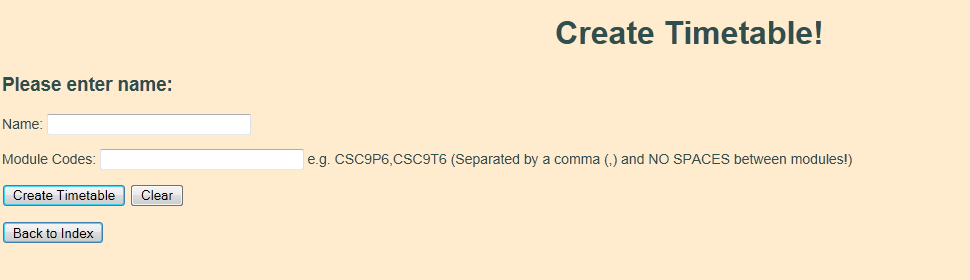
**Student User Guide**

The following is a guide for a Student to use the system.

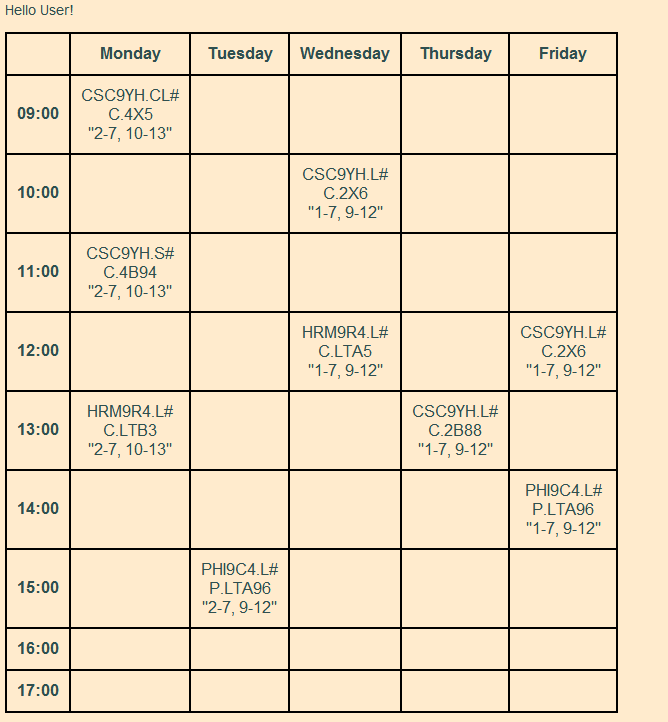
1. Click on button named “Student” to access Student Interface.



1. Enter your name and modules that you are taking during that semester.
   * Modules are entered with a comma separating each module.
   * There must be no spaces between each module entered into text field!

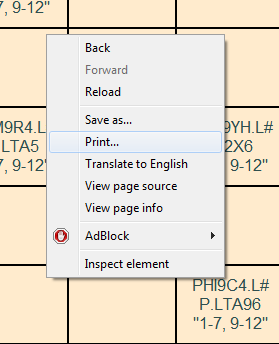
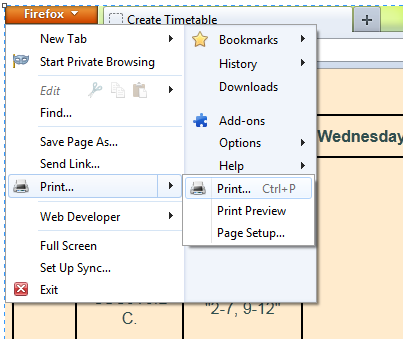
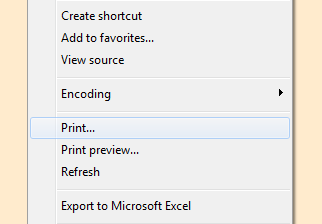


1. A timetable will be displayed with the modules you picked, displaying details of where and when your classes will take place.



1. You will be able to print out the timetable via the print function on your web browser.

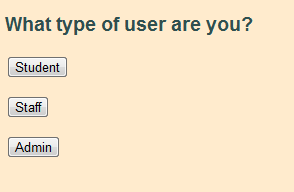
* Google Chrome: Right click on page and select “Print…”
* Mozilla Firefox 11: Click on Firefox tab located in top left and selecting “Print…”
* Internet Explorer 9: Right click on page and select “Print…”

**Google Chrome** **Mozilla Firefox 11** **Internet Explorer 9**

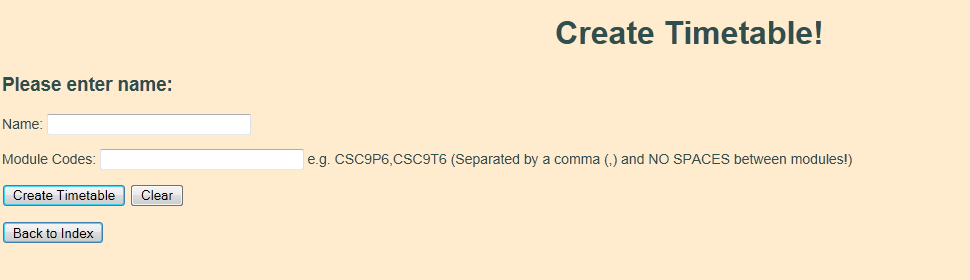
Appendix F Staff User Guide

The following is a guide for a Staff member to use the system.

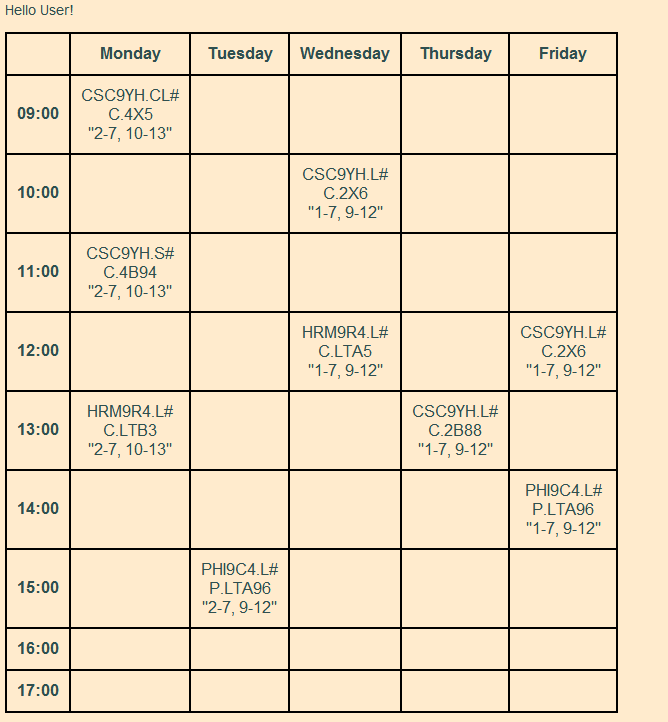
1. Click on button named “Staff” to access Staff Interface.



1. Enter your name and modules that you are taking during that semester.
   * Modules are entered with a comma separating each module.
   * There must be no spaces between each module entered into text field!

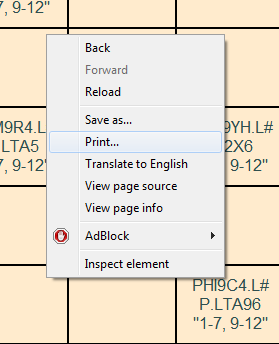
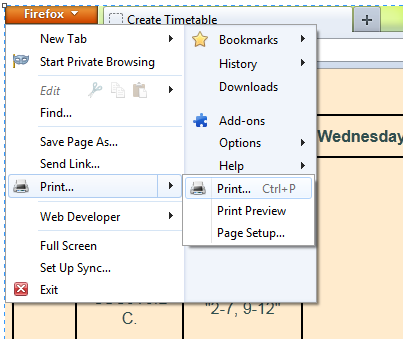
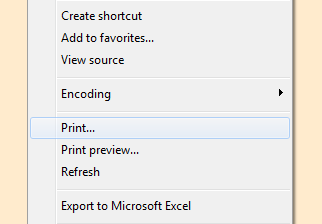


1. A timetable will be displayed with the modules you picked, displaying details of where and when your classes will take place.



1. You will be able to print out the timetable via the print function on your web browser.

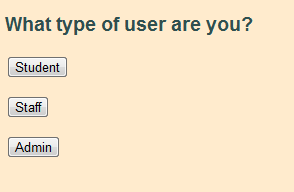
* Google Chrome: Right click on page and select “Print…”
* Mozilla Firefox 11: Click on Firefox tab located in top left and selecting “Print…”
* Internet Explorer 9: Right click on page and select “Print…”

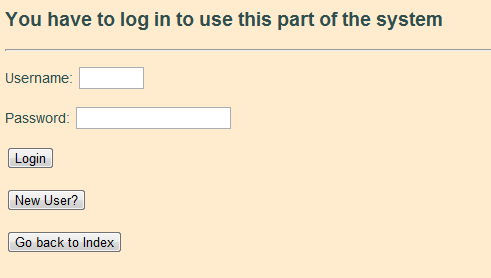
**Google Chrome** **Mozilla Firefox 11** **Internet Explorer 9**

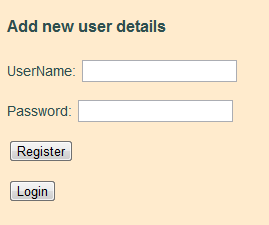
Appendix G Admin User Guide

The following is a guide for an Admin user to use the system.

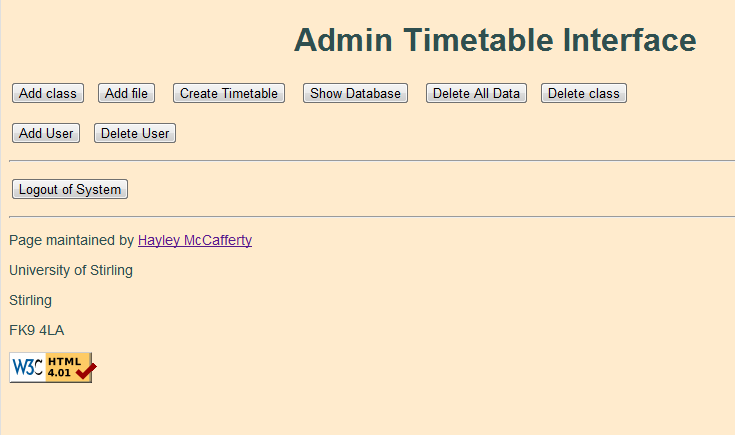
1. Click on button named “Admin” to access Admin Interface.



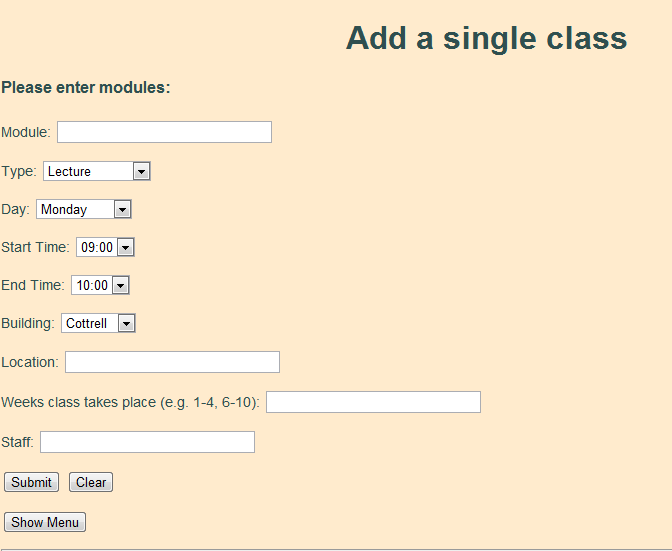
1. Admin users are required to be logged in to use the Timetable system.
   * You can login with username and password used with system – click “Login”.
   * If new to the system, an account is required to be registered - click “New User?”
2. If you are registering, enter an appropriate username and password and click “Register” – on success you will be required to use this information to login.



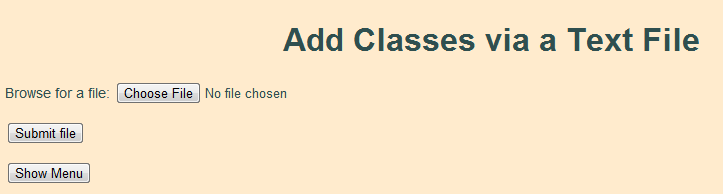
1. On success of logging in you will be presented with the Admin User Interface.

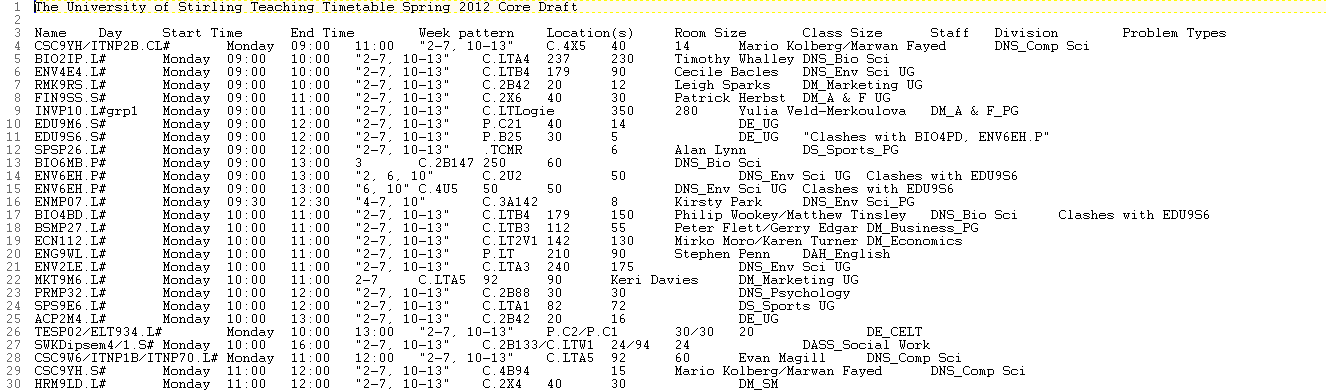


1. To add a class – click “Add Class”.
   * Fill out the details required and click “Submit” to add the class to the database.
   * You will be notified if addition was successful or not – click “Admin Interface” to go back to main interface.

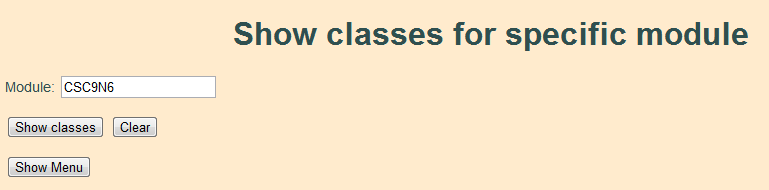


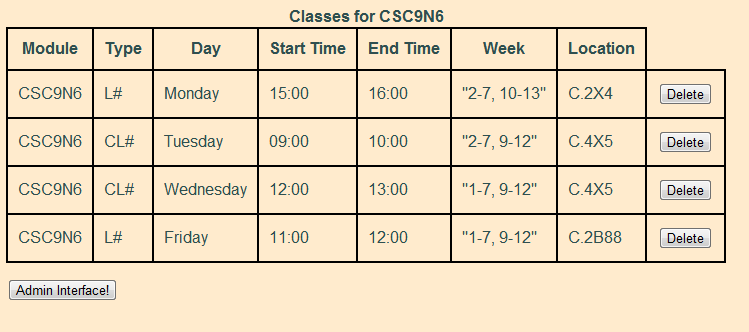
1. To add multiple classes via a text file – click “Add file”.
   * Browse for the text file that is required and submit it to the system. The system should notify you on how many classes have been added.
   * The text file **must** be in a certain format as seen below:



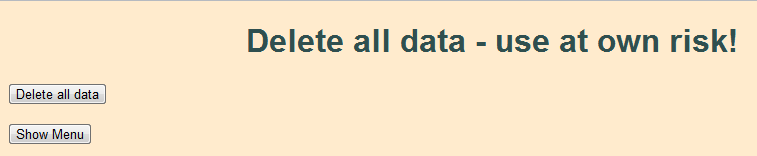


1. To delete a class from the system – click “Delete class”.
   * You will be asked to enter a single module on which a table will display all classes to do with that module.
   * Select a class to be deleted and the system will notify you on success.

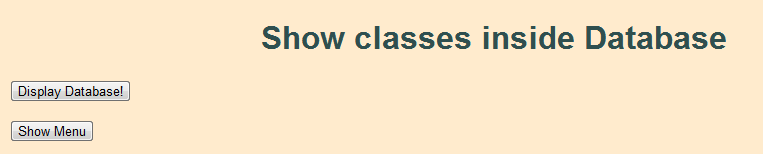




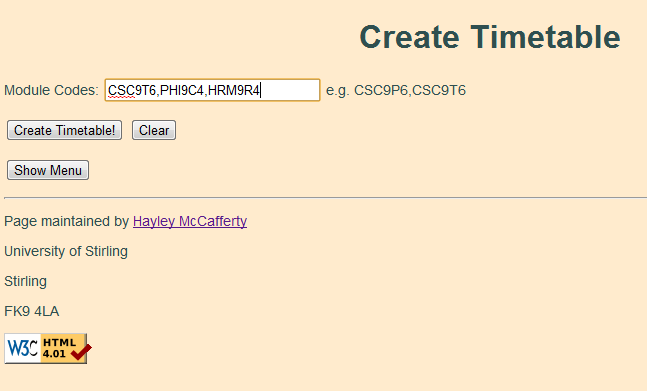
1. At the start of a new semester it is required to clear the database of previous classes – click “Delete All Data” to do this.
   * This must only be used at the start of a new semester as it will **delete all classes** in the database

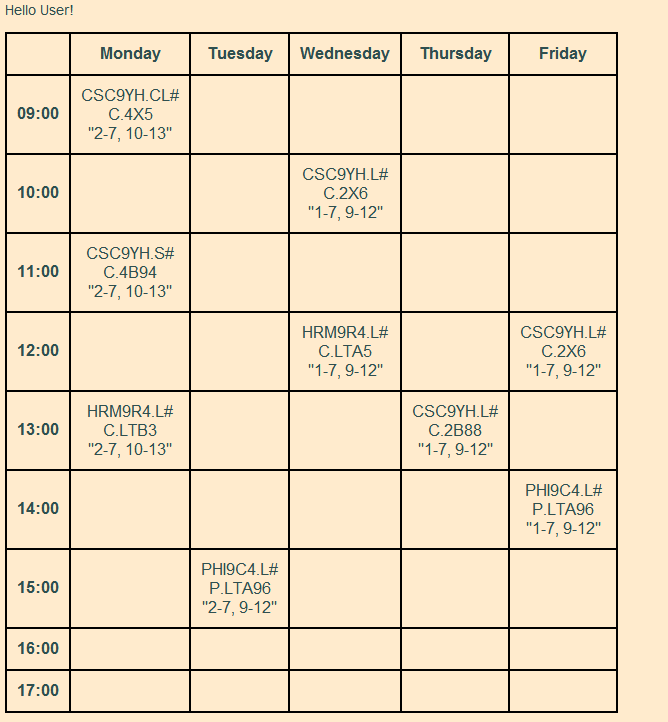


1. To see all the current classes held within the system – click “Show Database”.
   * Will show all the classes held within the module and any additional information held on them.

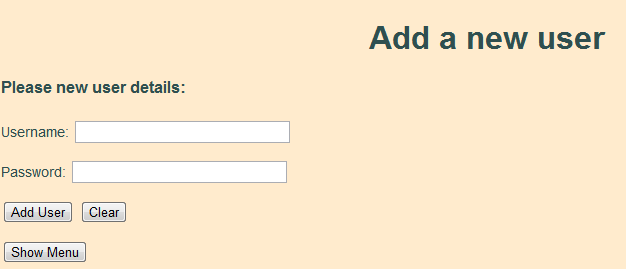


1. To check how classes are looking on a timetable – click “Create Timetable”.
   * Enter modules separated by a comma (,) and no spaces in between.
   * Click “Create Timetable!” to create a timetable and the system will produce a timetable for you to print out.

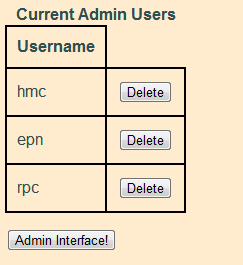




1. You can add additional Admin users – click “Add User”.
   * Enter new details consisting of a username and password and click “Add User”
   * The system will then notify you if you are successful or not.



1. You are also able to delete other Admin users if they are no longer required – click “Delete User”.
   * You will be presented with a table of all the current users in the system.
   * Click “Delete” to remove a user from the system.



1. At any time you are able to leave the system by logging out from the interface – click “Logout of System”
   * You will be informed you have now exited the system and will proceed back to the main index.