|  |  |  |
| --- | --- | --- |
| (The Easy Adjustable Administration System) | | |
| Student nameRichard Galvin | Student numberD11125535 | SupervisorEdina Hatunic Webster |

# Project statement

The main aim of this project is to provide small businesses and education centres with limited resources and budgets a simple and secure way to collect, store and manipulate user data as well as automating everyday tasks such as timetabling/ staff scheduling, attendance and statistical reporting as well as automating basic network administration tasks such as server account and folder creation as well as group and permissions management.

## Aims and Objectives

1. Simple and secure data collection and storage
2. Regulated access
3. Attack prevention mechanisms
4. Simple user friendly interfaces
5. Easy to use options for bulk import of data
6. Data integrity checks
7. Clear concise error messages and warnings
8. Automation of timetabling, scheduling and reporting
9. Easy to use tools
10. Intelligent option provision
11. Automated alerts
12. Background functionality to manage network resources
13. Simple data retrieval mechanisms and attractive data presentation
14. Filter and search options were needed
15. Clear and concise data presentation

Although this application is being designed with an adult/ community education centre in mind it is hoped that is designed in such a way that with minimal configuration and structural changes it could used by any small to medium organization.

# What research has been done and what are the outputs?

## Background research

Small education centres and businesses to store and manipulate data belonging to their users for various different reasons. These users may be staff or external users such as students or customers. The challenge these organisations face is to being able to store this data in a secure manner while maintaining a high level of availability. Although they may have some IT infrastructures in place many of these organisations lack the knowledge and skills in-house to utilise these structures properly and achieve their IT goals. Due to cutbacks in funding over the past few years such centres have been forced to severely trim back their staffing levels to a point were some staff are carrying out duties which had been carried out by multiple members of staff in the past. Many administrative duties such as timetabling, scheduling, reporting and student/ tutor participation are carried out manually as it is felt there is a lack of affordable integrated software on the market [1], [2]. Indeed this view is shared by other small to medium business owners. [3], [4].

Education centres need to store sensitive data regarding staff and students including PPS numbers. Any application developed will need to adhere to these rules in order to ensure the centre remains compliant. The following are the eight golden rules of data protection [5].

1. [Obtain and process information fairly](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#1)

2. [Keep it only for one or more specified, explicit and lawful purposes](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#2)

3. [Use and disclose it only in ways compatible with these purposes](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#3)

4. [Keep it safe and secure](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#4)

5. [Keep it accurate, complete and up-to-date](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#5)

6. [Ensure that it is adequate, relevant and not excessive](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#6)

7. [Retain it for no longer than is necessary for the purpose or purposes](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#7)

8. [Give a copy of his/her personal data to an individual, on request](http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm#8)

The centre chosen as a case study for this project has provided a useful insight to the struggles they face to in managing data and network resources. Due to financial constraints it has been left to tutoring/ administration staff and volunteers to carry out these duties. Considering the lack of time, resources and specialist skills available to the centre a gallant effort has been made in this respect. Having said that the collective feeling is that some new technology could help provide compliance with data protection rules, provide automation of existing tasks whilst improving the efficiency of carrying out others and providing automatic and manual network administration options[6].

## Alternative existing solutions to the problem you are solving

As TEAd@S is intended to be an all in one administration system that will incorporate functionality which might usually be provided in separate applications no one application reviewed provides all the functionality proposed in TEAd@S and vice versa. Instead applications that provide much of the functionality proposed and those that provide specific functionality have been chosen. There are many different software tools on the market targeting the different academic markets. Through research it has been found such tools are quite limited in this country particularly those intended for the adult and community education market. Most applications on the market provide functionality aimed more at delivering course material rather than providing the resources to set up and manage courses.

Where possible trial versions of the different applications have been tested using Nielson’s Heuristics as a guide. Jakob Nielson proposes 10 general principles concerning user interaction design which are outlined below [7].

1. Visibility of system status

User should know where they are in the system and should be kept informed of what is going on.

1. Match between system and real world

The system should match how the user communicates and approaches tasks as much as possible

1. User control and freedom

Should provide easy to find undo and exit options for the user

1. Consistency and standards

Language and wording should be consistent throughout the system and comparable with is already available in the same category on peer systems

1. Error prevention

Should prevent errors as much as possible providing users with confirmation messages before completing actions

1. Recognition rather than recall

Options, objects and actions should be visible to the user. Help should be visible or easy to find

1. Flexibility and efficiency of use

System should be useable from a novice and an expert’s point of view. Expert users should be provided with options to accelerate use

1. Aesthetic and minimalist design

Only provide information that is needed. Keep as uncluttered as possible.

1. Help users recognise, diagnose and recover from error

Provide clear concise error messages and solution suggestions

1. Help and documentation

Should be easy to find, short and concise

### Education Solutions

MIT’s suite of provides much of the functionality proposed in TEAd@S albeit in a suite rather than one application. Unfortunately a trial couldn’t be secured for this product so no review was possible [7].

## Sales Pulse

Adult education centres are required by funding bodies to use the SalesPulse Adult Education application. This is a web application which is cloud based with a client side written in HTML, ASP and JavaScript. It is mainly used by the various bodies to collect information in order to sanction funding and tutor payments. The application allows the creation of course categories and courses, the insertion of tutor and student data and the allocation of students and tutors to specific courses. Timetables and reports can also be generated.

Once logged into SalesPulse the user interface is aligned to the left hand-side and appears cluttered and cramped with a small font. Once adjusted to the appearance the user interface it is quiet intuitive and easy to use. SalesPulse Adult Education offers resources to users depending on which projects they are involved with. The main screen offers a choice of all projects across the top of the screen with access to those the user is offering courses in. On the main screen options for entering and retrieving data are provided depending on what project has been selected. IFrames provide forms for adding data and for presenting data retrieved from the database. A menu bar provides tabs for each sub category which when selected allows the user to tailor search queries and enter data depending on what the category is.

The format of each different section is consistent throughout the application. Easy to uses search options are provided as well as instructions on how they can be used. Although many search options are provided one particularly useful one is the provision of the letters of the alphabet.

In general the help provided by this application is sufficient. Instructions on the various screens are quite limited although user guides are available for most sections and tasks on the main screen. The application is a little inconsistent in informing the user exactly where they are in the system. In some categories the user is kept informed at the top of the screen to exactly what section and subsection they are in. Other sections either don’t provide this at all or only in part. After a period of use it became clear the most efficient way to get back to the main menu which can involve up to three steps depending what section the user is in.

Although SalesPulse contains functionality that is the most comparable to that proposed in TEAd@S and is designed for adult education centres it does present some shortcomings for centres that provide courses other than those that fall under the adult education umbrella. Data collected and stored by Sales pulse is relevant only to courses affiliated to funding bodies [9]. Timetables generated in this application are produced for tracking of course hours rather than as use as a reference for tutors and students. Reports generated in Sales Pulse are rendered in excel format only and do not represent all data required for end of year reports. Sales Pulse also does not allow for data integration and bulk data importation although these options have been explored in the past [10].

## Moodle

Whilst formulating a project idea providing functionality to facilitate the deliverance of course material in TEAd@S was considered. This led to research of applications that currently provide this type of functionality. One of these is Moodle and even tough this type of functionality will not be included it was decided to include the review in this report as the application does provide some of the functionality proposed for TEAd@S. Some of this functionality includes:

* adding users
* groups
* courses
* adding tutors and students to courses
* adding grades
* reporting
* user administration
* backups

Moodle uses HTML, PHP and JavaScript to present content to users. Its main page is uncluttered with large font and provides all options available to the user on the left hand-side and any relevant content is displayed in the centre of the screen. Along the top of the screen is a status bar which keeps the user informed exactly what part of the system they are in at any given time. This type of layout is used throughout the application giving the user the impression that they it is the page content that is changing and not the actual page. This provides for easy navigation and clear system status but it does result in cluttered displays at in particular when generating reports.

Users can be added to Moodle either individually through a form or by bulk upload using csv delimited files. Categories and course can be added and updated through a course and category screen. This screen offers options to sort and move categories as well as adding courses to categories. In truth this screen is slightly confusing and requires some trial and error before figuring out what each option is for. A nice feature in this section is the ability to restore courses and categories from backups. Backups can be run manually or be configured to run automatically. The user can choose to back-up the whole of the application specific parts.

As a demo version of Moodle was used to review Moodle some of the criticism aimed at it may be unfair but there did seem to be a lack of easy accessible help although plenty of documentation is available from the Moodle website. All icons used throught Moodle also provides information regarding their use if hovered over. Error prevention is lacking slightly in places particularly when adding users. As stated previously TEAd@S is not intended to be a Moodle type application but ideas on features and layouts have been provided by the application as well as how the application should perform overall. [11]

## E-Front

This application provides similar functionality to that available in Moodle and so only a short review has been carried out on it. This application provides a friendly user interface which incorporates images with text to present user options. Forms for adding content are also visually appealing and with a simple layout.

|  |
| --- |
|  |

Fig 2. Print screen of EFront demo home screen. (http://demoefrontlearning.com)

|  |
| --- |
|  |

Fig 3. Print screen of EFront Demo user information form (www.demoeforntlearning.com)

During testing this application scored badly in some of the heuristics being used as a testing guide. Error prevention and detection is inconsistent throughout the application. Once a task has been completed either successfully or not there is no clear exit/ return path or progression path. An interesting feature throughout the application is the use of type ahead lists instead of drop dropdown lists for certain fields. Depending on the amount of options available this can be significantly more efficient. There is little or no help available in this application but as this is a demo version it may not be available. [12]

|  |  |  |  |
| --- | --- | --- | --- |
| Heuristic | Moodle | SalesPulse | EFront |
| Visibility of system status | 10 | 8 | 7 |
| Match between system and real world | 7 | 8 | 8 |
| User control and freedom | 8 | 8 | 5.5 |
| Consistency and standards | 8 | 8 | 6 |
| Error prevention | 8 | 7 | 5 |
| Recognition rather than recall | 8 | 8 | 6.5 |
| Flexibility and efficiency of use | 8 | 8 | 6 |
| Aesthetic and minimalist design | 8 | 7.5 | 8.5 |
| Help users recognise, diagnose and recover from error | 8 | 7 | 5 |
| Help and documentation | 7 | 9 | 0 |

Table 1. Comparison of reviewed applications using Nilesons heuristics

As it is proposed that TEAd@S will provide automated network administration programs that already provide this functionality were investigated. Below is a table of applications available for Windows. No programs for Linux were found during this research through research of network administrator forums it seems scripts are the main method of carrying out such tasks [13], [14].

|  |  |  |
| --- | --- | --- |
|  | Adaxes from Softerra | [ADManager](http://www.manageengine.com/products/ad-manager/index.html) |
| Platform | Windows | Windows |
| User management | Yes | Yes |
| Group Management | N/A | Yes |
| Password management | Yes[1] | Yes |
| Bulk User/ group/ password management |  | Yes |
| Import /export files | No | Yes |
| Automation of user admin tasks | Yes | Yes |
| Reporting | Yes | Yes |
| Backup management | No | No |
| Server cleanup/maintenance | No | Yes |
| Cost | Per AD user license | Free for 100 systems |

Table 2. table of applications an associated features

Note

1.separate application

## Other relevant research done

## Brainstorming Session

An initial brainstorming session was held with the centre coordinator and two members of staff who carry out both administrative and teaching duties. Some ideas were discussed as to what the application may include. The main points that came from the meeting are as follows:

* The centres current database was developed using legacy software which is now outdated
* Problems with data storage such as duplication, corruption and loss have occurred increasingly over the past few months
* It is feared requirements for data protection are not being met
* Data access is limited is limited as is how it is presented
* Many tasks are being carried out manually taking up staff hours which could be better spent on other duties
* There is no structured way to record and monitor student and tutor participation
* The same data is required to be entered for different organisations
* There is no way of tracking student progression
* There is a lack of knowledge regarding how to create and manage user accounts and how to create and manage folders and their permissions
* The centres IT resources are either being underutilised or not used correctly including:
* Duplication of files on the file server
* Files saved in incorrect folders
* Files being saved on PCs rather than the file server

## Interview

The brainstorming session had provided an outline of what the user requirements for TEAd@S might be. A set of questions was sent to the centres director with the view of holding an interview a short time later. Through investigation and research a clear picture of what IT resources were available at the centre and how they were being utilised had been developed, clarification was now needed regarding data in particular the following:

* What data regarding what parties is being stored locally
* What data is required by third party organisations
* How was data was currently inputted and retrieved
* What reporting was required and how it was currently been carried out

Clarification was also sought on what type of learners attended the centre with a particular focus on their IT literacy skills.

## Surveys

The interview helped clarify issues surrounding data and eliminate some provisional requirement particularly those that were being considered for learners. Three separate surveys were released during the requirement gathering phase, the purpose of which was to further clarify what the user requirements should be. One survey was created for tutors, another for administration staff and a third designed for other centres that provide similar education services.

|  |
| --- |
| Fig 4. Screen shot of survey results for proposed requirements from tutor survey (survey monkey) |

Results from the tutors survey help clarify what features would be useful to them for course delivery. From the table above it is evident that there is support for many features laid out in a provisional list of functionality which was presented for consideration. Other suggestions were also provided through the survey such as the provision of individual learning plans for students and the ability for tutors to upload links to QA surveys.

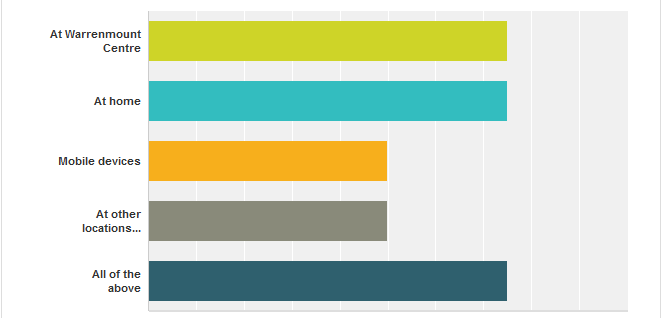
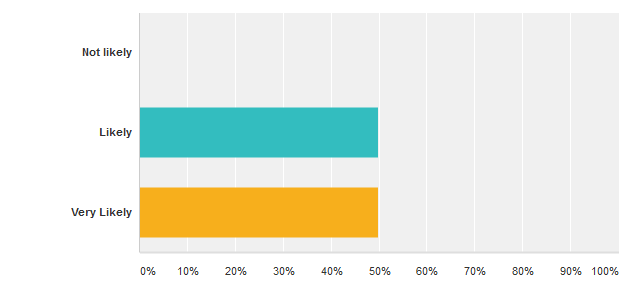


Fig 5. Screen shot of survey results for likelihood of use and location of use from tutor survey (survey monkey)

The survey also provided an insight into where tutors would like to be able to use the application. This will aid the developed in deciding which frameworks should be used and what platforms it should be aimed at. Other questions were designed to ascertain how tutors currently carry out tasks such as roll call and the recording and tracking of student performance.

The survey also provided an indication of the level of support for such an application and the willingness to utilise it if developed.

The second survey was designed to ascertain what features administration staff would find useful in an application. In hindsight due to the fact staff carries out different administrative tasks the survey could have been designed better to provide more informative results than what it did. The survey did provide useful feedback in relation to perceived strengths, weakness and difficulties associated with applications currently in use at the centre including SalesPulse which was reviewed previously.

|  |
| --- |
| Fig 6, Screen shot from survey results from survey designed for admin staff (survey monkey) |

The table above right represents the respondent’s opinion of the usefulness of timetabling functionality which will in turn provide the means for student and tutor participation tracking and reporting. The survey also confirmed that it is felt a student and tutors profiles would be made available which would contain personal information as well as any course participation records. It was suggested some sorted of unique identifier be assigned to users in order to eliminate duplicate eateries and the re-entry of data from archives. The table above left represents the opinion of the respondents when asked if the ability to upload data in bulk using .csv files would be a useful piece of functionality.

The third survey was designed to get a general idea of what other centres were using to carry out their own data storage and handling tasks and to try a general opinion if they felt the functionality proposed in TEAd@S would be useful. Although limited responses were received they served to confirm that most of applications used by other centres are bespoke owing to the fact that there are very few applications on the market which are affordable and provide the functionality required by such centres. Respondents also indicated that they felt a lot of functionality proposed would be useful or are already included in the applications they use.

As mentioned previously the survey didn’t provide all the necessary information needed to aid a final decision on requirements so a meeting was arranged a selection of administration staff and management. During this meeting the results of the different surveys were discussed and the level of support the proposed requirements received. The meeting also served to clear up what data was required by the centre itself and third parties such as funding bodies.

## Technologies researched

### Languages and Platforms

### PHP

PHP is a widely used language for the creation of dynamic web content. It can also be used for command-line scripting and client-side GUI applications. PHP can be used on all major operating systems such as Ubuntu, Debian, Mac OS and windows as well as the leading web servers such as Apache and IIS. PHP supports all of the major relational databases such as MySQL, Oracle, MS-SQL, PostgreSQL and NOSQL databases such as MongoDB. PHP provides a library of code which can be used for common tasks such as error handling and also provides support for many different output types such as PDF, Photograph files and Flash movies [15]. PHP has been noted as proving many advantages for web development including the fact that it is open source, less class structure than other languages allowing for easier interpretation of code, portability it that it works on all major operating systems, web servers and with most major database systems. Overall it is deemed as quiet an easy language to learn and use resulting in an increased speed of development [16], [17]. Some disadvantages of PHP have been described as an apparent lack of security due to its open source nature and also its lack of standardisation [18].

### Java

Java is an object orientated language for developing client server applications. Java is known to be an extremely portable language. Source program are compiled into bytecode which can be run on servers or clients using a Java Virtual Machine (JVM) providing Java is installed. As with PHP Java can also be used with all of the major databases types and its help resources are vast. Java is also known as a robust language which is quiet secure. Objects cannot contain references to external data and so addresses of data storage in other systems or operating systems cannot be passed in as an instruction. The JVM performs data integrity checks on each object in order to enforce this [19]. A major attraction of Java for developers is its automatic garbage collection which relieves the developer of the task of allocation and deallocation of memory. Although it is based as C++ as a whole Java is considered an easier language to learn. Some drawbacks of using Java are considered to its lack of performance at times and lack of support for low level programming. [20], [21]

Many different flavours of Java are available for web application including Java Servlet API, JavaServer Pages Technology, JavaServer Faces Technology and JDBC API to name a few [22].

### Active Server Pages (ASP)

Asp was developed by Microsoft as a server side scripting language to facilitate the development of dynamic interactive web pages. VBScript is the default scripting language but others can be used such as Jscript. ASP runs on IIS servers which can be run on any Windows machine from NT4 up [23]. Visual studio is also available from Microsoft which provides developers a platform to develop web applications integrating scripting languages with coding languages such as C#, C++, Java, JavaScript and VB. Some advantages of using ASP for web development are considered to be its ease of use, language options, help and documentation availability, development tools and extensibility via COM components. [24]. A substantial drawback of using ASP is the fact if an application is to be hosted locally it must be run on an IIS server which requires a Windows machine to run which will typically add expense to a project.

## Databases

## Oracle

Oracle DB is a relational databases management system. Oracle structure is split logically and physically making it extremely scalable and fault tolerant. Logical elements include table spaces which each create one or more physical datafiles, schema objects which include tables, views, store procedures and indexes to name a few, datablocks, extents and segments [25]. Physical structures comprise of datafiles, redo logfiles and control files [26]. Oracle comes in several different versions including enterprise, standard, express and lite. PL/SQL is an extension to SQL offered by Oracle in order to write triggers, procedures, functions, types and triggers [25]. Oracle is available in different version depending on user needs including Express which is a scaled down version with restrictions on CPU, storage and memory usage that is free for non commercial use [27]. Oracle is compatible with most major operating systems [25].

## Microsoft SQL Server

According to db-engines.com’s scoring system Microsoft SQL Server currently lies third behind MySQL and oracle in the database ranking table. [28] Like oracle MS SQL Server is a relational database which is accusable by using the SQL language either locally or via applications [29]. T-SQL is similar to Oracles PL/SQL in that it extends SQL to allow developers include procedural programming [30].[tsq l.info] MS SQL Server comes in many different flavours depending on an applications needs. Express is a down version which is free for non commercial use. It contains many features of a full version with certain limitations in place such as processor usage and database sizes. One feature in particular it does not contain is SQL Server agent which provides features such as job scheduling. A Microsoft operating system is required to run all versions of MS SQL Server which makes it a lot less attractive for applications which are required to be portable.

## MySQL

MySQL is an open source relational database system which is now ranked second behind Oracle in terms of use. MySQL is easily deployable and is compatible with all major operating systems. As with Oracle and MS Server SQL is the language of choice when interacting with MYSQL databases and it supports support procedures, triggers, functions and views. Some advantages of using MySQL have been identified as cost, portability, security, scalability and performance [31]. Some disadvantages which have been identified are the lack of good development and debugging tools, lack of transaction support, lack of check constraints, lack of efficiency when working with large databases and the lack of development on the open source version since taken over by Oracle [31],[32].

## User Interface

## Javascript

JavaScript is a programming language which makes websites interactive. It can be used to carry out a range of tasks such as but not limited to changing HTML content, attributes, styles and the likes of data validation [33], [34].

## jQuery

jQuery is a lightweight JavaScript library which wraps JavaScript into methods which can be called by one line of code. It makes JavaScript manipulation much easier as well as Ajax calls. Ajax (Asynchronous JavaScript and XML) is used to make calls behind the scenes to a server in order to update pages without the need for reloading. jQuery is very popular and widely used and there is a vast range of online help available [35].

## Dojo

Dojo is a toolkit which is similar to jQuery. It was designed to aid the rapid design of cross platform, JavaScript, Ajax based websites. dojo is split into several packages including dojoCore which is the most widely used, dijit, dojox, and util. again dojo is extremely popular and there are plenty of online resources [36].

## Conclusion

## Language

As a result from the research carried out it has been decided the Java Programming language will be used to develop the business logic layer of this project. Although there was very little to choose from between the different languages it is felt that Java will provide more flexibility and security for this project.

Bash scripting will be used to carry out the various file server administration tasks that will be required. Research has shown that such scripts can be run quiet easily from Java methods.

## Database

MySQL will be used for this project mainly due to its cost, scalability and cross platform support. Although free versions of Oracle and MS SQL Server provide many of the features available in full versions they do come with certain limitations in terms of performance and scalability. MS SQL Server was the first to be discounted as it can only be used with Microsoft products resulting in a lack of portability and added expense. Although Oracle ticked most boxes its lack of scalability would be a worry in the event of a major expansion or deployment of the application.

## User Interface

Along with HTML and CSS JavaScript and jQuey will be used to develop the user interface. Although there is not a lot to choose between jQuery and other toolkits such as Dojo it is felt there is more support available for the latter..

## Application Server

Apache Tomcat will be used as an application server mainly due to it cost nd ease of use.

The developer has some experience in Java, Bash and MySQL and has successfully tested communication between the three. A big reason the developer is keen to utilise JavaScript and jQuery is the total lack of experience in both. Steps have already been taken to become familiar with both and will continue until development starts. The same is true for MySql.

## Resultant findings/requirements

Carrying out research allowed the developer gain a clearer picture of the applications requirements and discount several requirements which had been proposed. From completed surveys student note provision and work collection gained favourable support. Through the meeting discussed previously and through discussions with a project supervisor it was decided that both these pieces of functionality would be left for future work. It was conveyed in the meeting that they wouldn’t be seen as a priority for the centre and that they would probably be underutilised due to the student’s low IT literacy levels. It was also suggested that they would probably form part of a separate application. Another proposed requirement was a reporting engine which would auto-generate the various reports required by the centre. Although it is felt that this would be a useful feature it was decided also to add this to future work or if time permits a final phase. It is felt it is more important to provide the correct data for the reports than concentrating on report design. Some suggestions for features were conveyed through the surveys one of which was an individual learning plan for students. Again it is felt although this would be useful it is not a priority and so has been added to future work and would more than likely be implemented in a separate application. The possibility of allowing students to request access to their student profiles for third parties such as potential employers or school registrars. Although this would probably be a good feature it has also been marked for future work.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phase | Req ID | Name of Req | Description | Priority | | User Contact | | |
| 1 (a) | **1** | **Secure Login** | **Login with provided user credentials. Log user visit details** | **High** | | **All Users** | |
| 1 (a) | **2** | **Manual insertion of user data** | **Enter data relating to students or tutors using an electronic form. If required accounts and or folders will be created for the user on the files server.** | **High** | | **Administration staff** | |
| 1 (b) | **3** | **Upload data from files** | **Bulk upload user data from .csv file. If required accounts and or folders will be created for the user on the files server.** | **Medium** | | **Administration staff** | |
| 1 (b) | **4** | **Restore user data from archive** | **Student and tutor data is archived at the end of each semester. This feature will allow the restoration of only required data. If required accounts and or folders will be created/ reactivated for the user on the file server.** | **High** | | **Administration staff** | |
| 2 (a) | **6** | **Add categories** | **Add new categories of courses for example ICT, literacy** | **High** | | **Administration staff** | |
| 2 (a) | **7** | **Add new course** | **A new course must be added to existing category. One course can belong to different categories** | **High** | | **Administration staff** | |
| 2 (a) | **8** | **Update, delete courses and categories** | **Delete or Update course or category data and resources on file server if required** | **Medium** | | **Administration staff** | |
| 2 (b) | **9** | **Add tutor to a course** | **Create a new instance of a course with a unique course code, tutor and running date range. Create folder on file server** | **High** | | **Administration staff** | |
| 2 (b) | **10** | **Add a student to an instance of a course** | **Add a student to an instance of a tutors course. If necessary create a folder for the student on the file server** | **High** | | **Administration and teaching staff** | |
| 2 (c) | **11** | **View user profile and Update, delete or archive user data** | **To present user data including personal and course details in a visually appealing manner allowing fields to be altered or user deleted** or archived | **High** | | **All users** | |
| 3 (a) | **12** | **Create timetable** | **Allow the user to define a specific term period including holidays and**  **to Select relevant data in order to create a timetable** | **High** | | **Administration staff** | |
| 3 (b) | **13** | **Select class list from schedule and fill in data** | **To allow teaching staff select class lists from their personal daily schedules and enter student attendance and performance data** | **Medium** | | **System/ teaching staff** | |
| 4 (a) | **14** | **Generate statistics for reports** | **Allow users select various criteria and generate statistics relating to them** | **Medium** | | **Administration staff** | |
| 4 (b) | **15** | **Manage Groups** | **To allow administration staff to manage groups and their membership** | | **Low** | | **Administration staff** | |
| 4 (b) | **16** | **Perform backups** | **Allow users perform backups of the database or file server** | | **Low** | | **Administration staff** | |

## Table 3 User requirements

## Non Functional Requirements

## Security and integrity

As sensitive data will be inputted and stored in the database it is imperative that security criteria are evaluated at every stage of development. Attacks such as SQL Injection, XSS and infernal due to elevated privileges must be secured against as well as unauthorised access to the system. All passwords and other critical will be encrypted to an acceptable standard as will any communications containing such data. Measure will also be included to ensure data is not replicated in the databases and that data cannot be accessed or affected by operations that are not authorized to do so.

## Usability

Ease of use is one of the most important requirements of this application. As the main purpose of the application is to attempt to encourage the correct use of IT resources plus recording and handling user data safely users must be comfortable using the system and be able to carry out tasks without assistance after minimal training. Nielsen’s guide on heuristics will be followed closely in order to achieve this goal.

## Availability and recovery

If users are to be encouraged to use this system to its full potential it will need to be available constantly. Also if an error or malfunction occurs the data must be recoverable in a very short space of time. In order to achieve this a fail over server will be utilized as will an automated backup plan. It is foreseen with time to spare users will be able to perform backups of data before carrying out operations to enable them to rollback to before they started in the event of an irreversible error. This will be available due to the automatic backups but the data may be older then desired.

## Bibliography (research sources)

[1] Interview with Pauline Mc Galey, Centre Director, Warrenmount Community Education and Development Centre, September 2014.

[2] Respondents to survey sent to Adult/ Community Education Centres

[3] Interview with Hugh Patchel, Business owner, (Printing, Hospitality)

[4]

[5] Data protection agency, A guide for data controllers, http://www.dataprotection.ie/docs/A-Guide-for-Data-Contollers/696.htm

[6] Brainstorming session at Warrenmount Centre

[7] Jacob Nielson,(1995), 10 Usability Heuristics for User Interface Design, <http://www.nngroup.com/articles/ten-usability-heuristics/>, Accessed October 2014

[8] MIT, http://www.mit.ie/, Accessed September 2014

[9] SalesPulse, Live application, https://www.success.ie

[10] Brian Kelly, Email correspondence, [brian@salespulse.com](mailto:brian@salespulse.com), 28th November 2014

[11] Moodle, (-), Demo, <http://school.demo.moodle.net/>, accessed September 2014

[12] Efront,(-), Demo, <http://demo.efrontlearning.net/educational/www/index.php?index_efront&bypass_language=english>, accessed September 2014

[13] Adexes, (-),http://www.adaxes.com/, accessed October 2014-12-11

[14] AdManager, (-),http://www.manageengine.com/products/ad-manager/, accessed October 2014

[15] Kevin Tatroe et al,(-), Programming PHP, Capter 1,<http://wtf.tw/ref/tatroe.pdf>] programming PHP, Accessed November 2014

[16] vandelaydesign.com, (-), Advantages of PHP, <http://www.vandelaydesign.com/advantages-of-php/>, Accessed November 2014

[17]Paul Mah,(-), Advantages of PHP

http://www.itbusinessedge.com/cm/blogs/mah/advantages-of-web-development-using-php/?cs=33357, Accessed November 2014

[18] Paul Mah,(-), Advantages of PHP, <http://www.itbusinessedge.com/cm/blogs/mah/disadvantages-of-web-development-using-php/?cs=33397>, Accessed November 2014

[19] [Margaret Rouse](http://www.techtarget.com/contributor/Margaret-Rouse),(-), Java, <http://searchsoa.techtarget.com/definition/Java>, Accessed December 2014

[20] freejavaguide.com, History of Java Programming, http://www.freejavaguide.com/history.html, Accessed December 2014

[21] O’Reilly and Associates, Java in a Nutshell 3rd Edition, O’Reilly and Associates, Java in a Nutshell 3rd Edition, Chapter 1, http://docstore.mik.ua/orelly/java-ent/jnut/ch01\_01.htm , Accessed November 2014

[22] [*Dana Nourie*](http://www.oracle.com/technetwork/java/javamail/ssLINK/142892#nourie),(2006),Java Technologies for Web Applications, <http://www.oracle.com/technetwork/java/javamail/webapps-1-138794.html>, Accessed November 2014.

[23] Web Wiz,(-), What are active server pages, <https://www.webwiz.co.uk/kb/asp-tutorials/what-is-asp.htm>, Accessed December 2014

[24] Microsoft, Introduction to ASP.NET and Visual Studio, <http://msdn.microsoft.com/en-us/library/aa711425%28v=vs.71%29.aspx>, Accessed December 2014

[25] Oracle, (-), Database concepts –Intro, <http://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm>, Accessed December 2014

[26] Oracle, (-), Database concepts – Physical Structures, <http://docs.oracle.com/cd/E11882_01/server.112/e40540/physical.htm#CNCPT003>, Accessed December 2014

[27] Oracle,(20014), Version comparison, <http://www.oracle.com/us/products/database/enterprise-edition/comparisons/index.html>, Accessed December 2014

[28] Microsoft, SQL Server Overview, <http://technet.microsoft.com/en-us/library/ms166352%28v=sql.90%29.aspx>, accessed November 2014

[29]Wikipedia, (-), Microsoft SQL Server, <http://en.wikipedia.org/wiki/Microsoft_SQL_Server>, Accessed December 2014

[30] TSQL Tutorial,(-), <http://www.tsql.info/>, Accessed December 2014

[31] OTS Solutions,(-), Advantages and Disadvantages of MySQL, <http://gautambikash.hubpages.com/hub/Advantages-and-Disadvantages-of-MySQL>, Accessed Oct 2014

[32] John Mack, 2014, 5 Advantages and Disadvantages of MySQL, <https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/>, Accessed October 2014

[33] <http://javascript.about.com/od/reference/p/javascript.htm>, Accessed November 2014

[34] W3 Schools, http://www.w3schools.com/js/js\_intro.asp, Accessed November 2014

[35] W3 schools, http://www.w3schools.com/jquery/default.asp, Accessed November 2014

[36] Dojo, http://www.dojo.org

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  |  | | --- | --- | |  |  | |

|  |
| --- |
|  |

# Analysis: Describe clearly what your solution will do

This project will provide an easy to use all in one application to input, record and utilize data in small education centres and businesses. This application will be designed to be flexible, adjustable and extendable so it can be used for as wide a variety of organisations as possible. Outlined below is some of the functionality that will be developed for adult and community education centres.

The following is what the developed application aims to provide:

* **Secure login**  
  **Entering of user data**
* **Update, delete or archive student or staff data**
* **Add, update or delete courses and categories**
* **Student and tutor profiles presented in a user friendly fashion**
* **Create timetable**
* **Generate class schedule for class attendance and performance recording**
* **Statistic generation**
* **Perform basic network administration functions**

# Approach and Methodology

## AUP

Agile Unified Process (AUP) is a simplified version of the Rational Unified Process (RUP) and is seen as lying somewhere between it and extreme programming. It comprises of many of the features of RUP as well as adhering to the principles of Agile development including Test Driven Development, Agile Model Driven Development, Agile Change Management and Database Refactoring. AUP offers flexibility in what tools should be used during design development and testing.

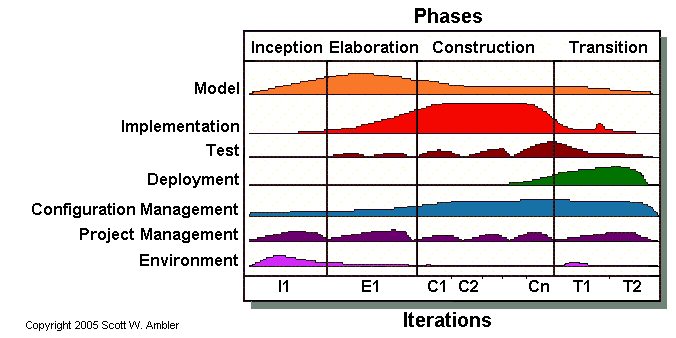


Fig 7 Graphical Representation of AUP (http://www.ambysoft.com/unifiedprocess/agileUP.html)

There are four serial phases of AUP

1. **Inception**

Define initial scope of project and investigate its feasibility and acquire approval. Potential architectures are also investigated

1. **Elaboration**

Further define the architecture and prove its achievable

1. **Construction**

Development of application in stages releasing at regular intervals

1. **Transition**

Integrate application into real world environment

AUP contains seven disciplines which are performed iteratively

1. **Model**

Important part of the process it covers RUP’s Business Modelling, Requirements gathering and analysis and design disciplines.

1. **Implementation**

Transform models into executable code and test

1. **Test**

Perform testing to ensure applications meets users requirements, non functional requirements and contains no defects

1. **Deployment**

Make system or part of available to the user

1. **Configuration management**

Manage access and changes to the different parts of the project

1. **Project management**

Direct different activities of project including specifying tasks and monitoring progress

1. **Environment**

Ensure a proper environment is set up with tools and platforms needed for development and testing

When following AUP software should be released in portions instead of all in one go. Having developed and tested a particular piece of functionality it should be releases in a staging area for QA and User acceptance testing before being integrated with other components for phased deployment into a real world environment [37].

**\*\*Please note\*\* Full research the area has been carried in this area and is to be included in the manual**

# Conclusion

AUP will be the model Software Development Methodology used as a guide for this project. Although it has been overtaken by other methodologies in recent years in terms of usage it is felt as this project is being undertaken by a single developer it is a good methodology to follow due to the fact it comprises of the best parts of RUP without its complexities and also adheres to Agile design principles. Other methodologies which were researched and discounted for various reasons include Waterfall which was discounted due to its inflexible nature, Spiral due to its complex nature and the fact its more suited to larger projects, Scrum due to the fact it is designed more for team projects, XP again mostly because it is designed for team projects and in particular that it specifies that programming should be carried out in pairs and RUP mainly due to its complexity.

This project has been broken up into phases which are specified in the user requirement template. Each phase contains sub phases which have been assigned in order of importance. As each sub phase is developed it will be release into a staging area for quality assurance (QA) and user acceptance testing (UAT). Once the entire sub phases of a particular phase have been tested sufficiently they will be integrated into an overall testing area for further QA and UAT test as well as integration and performance testing.

Development will be carried out using a bottom up approach. During development each initial use case will be taken and analysed to ascertain if it needs to be altered or expanded and to determine what classes and functions will be needed to achieve what is required. Starting at the data persistent layer the initial ERD and existing database will be analysed to determine if tables, procedures, triggers, views and functions exist which are sufficient for the task required. If not existing elements will be altered or new ones added as required. If any procedural programming and security measures such as views have to be implemented these will be tested before continuing. Next the business logic layer will be analysed to determine if any classes and functions exist to facilitate communication with the data persistent layer and also external systems such as the file server and the user interface. Again existing elements will be updated or new ones added as required. Any new functionality implemented will be tested locally to ensure it performs as required. Finally how the user interface will appear and interact with the business logic layer will be analysed. Tools such as screen shots of mock-ups and diagrams will be used to provide a clear picture of the desired workflow. Once coding is complete the new functionality will be tested as a complete module. If previously written functions or resources have been used to implement the desired functionality the requirement they were developed for originally will be tested to ensure they still perform correctly.

|  |
| --- |
| **Fig 8 Flowchart representing overall development flow** |

|  |
| --- |
| fig 9 Flowchart of individual requirement development |

# Design

## Technical architecture diagram:



Fig 10 architectural overview



Fig 11 architectural overview with technologies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | User interface |   *Application server*   |  | | --- | | *Business logic layer*  Call script for adding a new user providing any  Response handler  Credentials or ack  Request handler  Add user to database |  |  | | --- | | *Data persistent layer*  *Return credential flags for file server or an ack that data has been inserted* |  |  | | --- | | *Create a/c + folders return ack* |   *Linux file server*  *Fig 12. Example of a proposed workflow* |

## Other design documents

## 

|  |
| --- |
|  |

Fig 14 Use Case Diagram

# Initial Use Cases

Below are a set of initial uses cases which have been created for each of the requirements identified through gathering and analysis. These use cases are not finalised and have been created merely as an initial guide of what will be required. During development each use case will be taken to ascertain if it needs to be altered or expanded and to obtain a clear picture of what exactly will be required to implement the requirement.

|  |
| --- |
| Use case number: 1 |
| Use case name: Login User |
| Intent: To allow users to login with unique username and password |
| Precondition: User opens login screen |
| Use case initiation: Username and password entered |
| Dialog (Description) :   1. User: Hits submit button 2. System: Hashes password and checks against the database 3. System: Checks what group user belongs to 4. System: Logins user and presents appropriate screen   Alternate Flows:  3a. System rejects login request and provides error message |
| Use case termination: user logged in |
| Post Condition: Record of login in logs database |

Use case number one in the system design plan describes the requirement of allowing users to login to the application using a username and password which has been provided to them after their details had been entered into the system for the first time. As this requires to have user details entered and a hashed password generated and entered into the logins database it was decided to swap this with requirement number two which is the providing the ability to enter user information through a form.

Use Case Number 1

The first step in the implementation of this requirement was to create a user interface which contains a table displaying current users in the system and buttons which when selected will use a Bootstrap Modal which will contain a form for user detail insertion. The HTML template used for this requirement and indeed most of the phase one requirements is userindex.jsp with commonmodals.jsp and head.jsp also included. Any JavaScript, JQuery and Ajax functions being called can be found in AddUser.js.

Once the new user form has been completed and the submit button has been selected the function insertUser() is called. This takes all the fields from the form and submits them along with an instruction id as Json data string though an Ajax call to the InsertData servlet which is which can be found in the controls class UserInsert.java. Having established a database connection through the DBManager.java class and checked the instruction id in the request the servlet then calls the relevant function from the UserInsertManager class which in turn calls the relevant function(s) in the dao class InsertUserData to perform the insert to the database.

A bit has also been sent with the request indicating if the data needs to be checked for duplicates. If the bit indicates the database is to be checked for duplicate user details the function CheckForDups() is called. If it is found similar details already exist in the database the function ReturnPosDups() is called to relay these details back to the user to enable them to abort the operation if it is the case that duplicate data exists. If the user confirms they wish to proceed the function InsertSingleUser() is called which extracts the fields from the request and creates a random number which will be used to create the users unique id in the database. It then calls the stored procedure InsertUser in the database passing the relevant fields into it. This stored procedure enters all details into the TB\_User table. Once the insertion has been complete this returns a newly generated id from the table. Depending on the users role within the organisation two letters are appended to the id which is to serve as the users login name. This is then returned to the InsertUser.java where it is used to create an entry to the tb\_userlogin table. Before inserting into tb\_userlogin the createHash() function from the security manager is called to hash the users date of birth which will serve as the users login password. If the user has been inserted successfully a confirmation message is returned to the user and the table displays the new users details.

Use case number 2

Use case number two specifies the requirement of allowing a user to log on with their own username and password which is to be compared to the hash values in the database. if authentication is successful their user id and role id should be returned to be used as a session cookie. The role id will determine which initial screen is to be presented to the user initially and to which areas the user should have access to throughout the application.

Upon loading the application the user is presented with a logon screen which provides a field for a user name and password. Oncethe relevant data has been entered and submitted this are passed to the LoginUser servlet in the controls class LoginUser.java. The LoginUser servlet then calls relevant function from RetrieveManager which in turn calls the relevant function from dao.RetrieveUserData to extract the relevant data from the database. Once this data has been returned the password supplied is hashed and compared to the hashed value returned from the database using the validatePassword() function in the SecurityManager.java class[ ]. If successful the user is directed to the appropriate screen based on their role otherwise they are returned to the login screen.

Use Case Number 3

This use case was not developed during phase one development due to a combination of a lack of time and ineffective research. It was feared by proceeding with this requirement others may have been missed in phase two which were deemed more important for the overall functionality of the application. Once time permitted this requirement was researched further and was approached towards the end of phase two.

In order to facilitate the bulk upload of user data from .csv files a new button was added to the user interface. Upon selection of this button a modal appears with a file input. Once a file has been selected and submitted the file name is passed to the FileUploadServlet which handles the uploading and opening of the file as well as reading its contents. Each field except for the header fields are read into an array which is in turn passed to the InsertSingleUser function in the dao class InsertUserData for insertion to the database. This requirement at present is not considered completed an explanation for why is provided in the difficulties section for this phase.

Use Case Number 4

This use case outlined the requirement of allowing users to be reactivated in the system having been deleted previously. Again an oversight during the design phase saw the need for the reordering of requirements. In order to restore a user they must first be marked as deleted. Due to this fact it was decided to develop the delete user functionality at this point

Data protection laws specify that data must be kept for a specific period of time and be accessible upon request. These laws also state that data should only be available for the purpose it was intended. To comply with these rules and to allow users who have previously attended education centres to progress to further course user details are marked as archived when deleted and so are kept in the system hidden rather than deleted altogether. In order to facilitate this functionality a delete button was added to each row in the user table. Once a button has been the relevant user id is selected from the table as is the row it belongs to and both are passed to the function complteUpdateDel in AddUser.js which presents are bootbox containing a request for confirmation from the user. Once confirmation has been received the function upDel is called which passes the user id to the InsertData servlet in UserInsert.java which uses the RestoreArchiveUser function in the UserInsertManager to call the dao function RestoreArchiveUser. This function starts by selecting the user’s role from tb\_user to ascertain if the user is a student or tutor. If the user is a student enties in tb\_user and tb\_student\_courses are marked as archived. If the user is a tutor only the relevant entry in tb\_user is marked as archived. Once all updates are complete on the database a success message is returned to upDel which displays it to the user and deletes the relevant entry from the table and redraws it.

Use Case Number 4

To allow for users to be re-instated on the system a second button was added to the user management button. When selected the current table is replaced by a one that includes details of users which have been marked as deleted in the past. Each entry in the table contains a restore which when selected calls

Once a user has been restored successfully a confirmation will be presented to the user. The user should no longer be visible in the table of archived users and should be visible in table containing all user details.

An oversight in the design for this requirement was highlighted during user acceptance testing. It was pointed out that there was the lack of an option to restore multiple users at the same time. Time has not permitted this functionality to be added and so is marked for work in the future

Phase 2

Phase two mainly consisted of user requirements which concerned category and course management. The main user interface template for this phase is catcoursindex.jsp and its main js file is CatCourse.js. Each course which is run by the centre must be a member of at least one category. Courses can be assigned to different categories and each course name can be used in multiple instances providing they have a unique identifier.

Use Case 5: Insert Category

Requirement five specifies that users should have the ability to add a new course category to the system. This entails the user specifying a name and description for the new category. Once the relevant button on the user interface is pressed a modal is presented which contains a small form for the new category details. Once the relevant details have been submitted and the user confirms their intensions inserCatCourse() is called from CatCourse.js. This passes the data through an Ajax call to the CategoryCourses servlet which can be found in CategoryCourseInsert.java. This calls the dao function InsertCategory() through the CategoryCourse manager. InsertCategory() checks for any duplicate entries in the database by calling the stored procedure checkfordups() in the database. If a duplicate category name exists the operation is cancelled and a message stating so is returned to the user. Otherwise the data is entered into tb\_Category and a success message is returned to the user. The new category is now visible in the category table.

Use Case 6: Insert Course

This requirement specifies a user should be able to add a course to a category. For this requirement the user is to specify details such as what category the course should belong to. As stated previously course details can be used for different instances of that particular course. Instances may differ for different reasons such as what tutor is assigned to it or what capacity it will cater for. A button was added to the category and course home page which allows the user to view all course on offer at the centre. Category information is hidden at this point and a button to allow for the addition of a new course is shown. Once selected a modal presents a user form. This form includes selects which provide category name for which the new course can be assigned to. It also provides a list of tutors which can be assigned to the course. If an existing course is not to be used the user can specify a new course name. An existing accreditation body can be selected or if preferred a new one can be added. Other details such as the course level if applicable needs to be selected as does its capacity which must be under the organisations largest room size. Whether or not the course will require computing equipment must also be specified at this time Once the submit button has been selected and the user confirms their actions inserCatCourse() is called from CatCourse.js. This passes the data through an Ajax call to the CategoryCourses servlet which in turn calls the CategoryCourseManager which calls the insertCourse() function from InsertCourseData.java. This function calls the checkForDups stored procedure in the database which determines if there is a course with the same name and level in the database. If this is the case a message is relayed back to the user. If no duplicate exist a separate function InsertToCourses is called which calls the stored procedure addNewCourse in the database. This procedure determines if the course already exists in the database. If it this is not the case the procedure creates a new entry in TB\_Courses which will include the id of the category it belongs to, the course name, its level and accreditation body if applicable and whether or not the course will require equipment. The course id as well as the tutor id and capacity are inserted into tb\_tutorcourses. The course id is either the one sent from the servlet with the request or in the case of a new course being added the one that was generated during the insert. Once ted successfully a message is presented to the user and the new course is visible in the course table.

Use Case 6a: Insert Course from Category Table

This piece of functionality only became a requirement during development. It was realised that it would be make it easier for a user if they had the option to select a category from the table and add a course directly to that category rather than having to navigate to the course management screen. In order to facilitate this requirement a button is added to each table row as they being generated. Once selected a modal is presented to the user containing a form for the new course details. The category name is pre-populated in the form. If the category currently contains course those names are available for selection by the user. Once the submit button is selected the application carries out the same process as use case six.

Use Case 7: Update

|  |
| --- |
| Use case number: 8 |
| Use case name: Update /delete courses and categories |
| Intent: To allow course names and categories to be changed or details deleted completely |
| Precondition: Use logged in as administration staff |
| Use case initiation: Selection of category or course within courses page |
| Dialog (Description) :   1. System: Presents selected course or category with editable fields plus list of courses or categories for reassignment, plus a delete option 2. User selects action 3. System: Presents user with warning of the knock on affect of performing selected tasks 4. User: Proceeds with selected tasks 5. System: Runs stored procedure to perform selected actions and updates database. runs script to update file server 6. System: Confirms to the user that operation was successful   Alternate Flows:  4a. User declines to carry out updates and exits |
| Use case termination: User exits from manage course screen |
| Post Condition: Database contains changes both to course table and those associated to it and folder s updated accordingly on file server. |

|  |
| --- |
| Use case number: 9 |
| Use case name: Assign a tutor to a course |
| Intent: To allow an instance of a course to be instantiated with a tutor and date range |
| Precondition: Use logged in as administration staff |
| Use case initiation: Selection of manage courses tab within course administration |
| Dialog (Description) :   1. System: Presents list course/ categories and a search menu 2. User: Selects relevant course from list returned by category filter or search results 3. User: Selects qualified tutor and appropriate semester from drop down lists 4. System: Runs stored procedure to ensure the same entry doesn’t already exist and that tutor is qualified to deliver the selected course and creates entry in the database 5. System: Runs a script to create a folder within the file server name using course name, tutor name, semester and year 6. System: Confirms to the user that operation was successful   Alternate Flows:  4a. User informed entry already exists presented with the option to select another  4b. User informed tutor not qualified |
| Use case termination: User exits from course administration |
| Post Condition: Database contains data for new course instance and a folder exists on file server |

|  |
| --- |
| Use case number: 10 |
| Use case name: Add students to an instance of a course |
| Intent: To allow students to be added to an instance of a course |
| Precondition: Use logged in as administration staff |
| Use case initiation: Selection of manage assigned courses within course administration |
| Dialog (Description) :   1. System: Presents list of categories, associated courses and search menu 2. User: Selects relevant course from list or search result 3. System: Presents list of students 4. User: Selects student to be added to the course 5. System: Runs stored procedure to ensure the same entry doesn’t already exist and creates entry in database 6. System: Runs a script to creates a folder on the file server if required 7. System: Confirms to the user that operation was successful   Alternate Flows:  5a. User informed entry already exists presented with the option to select another |
| Use case termination: User exits from course administration |
| Post Condition: Database contains data for new course instance and a folder exists on file server |

|  |
| --- |
| Use case number: 11 |
| Use case name: View user profile and update/ delete details |
| Intent: To present user details in a visual appealing manner allowing fields to be altered or user deleted |
| Precondition: User logged in as administrator |
| Use case initiation: Selection of “users” tab |
| Dialog (Description) :   1. System: Presents list of users with filter and search options 2. User: Selects relevant user from list or searches 3. System: Presents user profile with all relevant details in editable fields 4. User: Selects update, archive or delete operation 5. System: Presents warning of consequences of action selected 6. User: Confirms action 7. System: Calls stored procedure to carry out relevant action. Calls script to update file server if necessary   Alternate Flows:  3a . System cannot locate any details informs user through error message |
| Use case termination: User exits from user |
| Post Condition: Database contains up to date data |

|  |
| --- |
| Use case number: 12 |
| Use case name: Create timetable |
| Intent: To allow admin staff to define term periods and create a timetable for them |
| Precondition: User logged in as administration staff |
| Use case initiation: Selection of timetables tab |
| Dialog (Description) :   1. User selects “Add new term period” 2. System: Presents already defined teaching periods with form to add a new one 3. User: Fills out required fields and selects relevant dates from a calendar 4. System: Calls stored procedure to Insert new time period and excluded periods into the database and create a temp table to hold work in progress table info 5. User selects: Add/ update timetable 6. System: Presents form with drop down options depending on whether user selected work in progress or new timetable 7. User: Selects relevant fields for timetable in sequence 8. System: Takes selected field and selects available attributes for next field using temp table as reference 9. User: Continues selecting until complete 10. System: Presents newly created timetable 11. User: Views and confirms as what is required and saves   Alternate Flows:  1a. User skips to 5  4a. date ranges do not match error message presented to user  7a. User doesn’t select in sequence warning message displayed  9a. No resources available, user must review timetable |
| Use case termination: User exits from manage classes |
| Post Condition: Database contains up to date performance data |

|  |
| --- |
| Use case number: 13 |
| Use case name: Select class list from schedule add data |
| Intent: To allow teaching staff select class lists from their personal daily schedules and enter student attendance and performance data |
| Precondition: System has generated a daily schedule from data |
| Use case initiation: User logs in as tutor |
| Dialog (Description) :   1. System: Presents calendar with generated class schedules 2. User: Selects preferred day 3. User: Selects relevant class from list 4. System: Presents list of students 5. User: Selects tick box for attendance and/or adds any notes/ marks 6. System: Checks to ensure the same entry doesn’t already exist 7. System: Creates/ updates entry in database   Alternate Flows:  6a. Record exists User presented with warning message |
| Use case termination: User exits from schedule |
| Post Condition: Database contains up to date performance data |

|  |
| --- |
| Use case number: 14 |
| Use case name: Generate Statistics for reports |
| Intent: To allow administration staff to generate statistics regarding individual and overall course participation |
| Precondition: User logged in as administration staff |
| Use case initiation: Selection of records tab |
| Dialog (Description) :   1. System: Presents options data and date ranges 2. User: Selects relevant criteria and output format 3. System: Presents data in selected format 4. User: Saves or prints   Alternate Flows:  3a. Date range out of bounds error message presented to user  3b. No data relating to selected criteria exists error message presented to user |
| Use case termination: User exits from records page |
| Post Condition: Records either saved, printed or discarded |

|  |
| --- |
| Use case number: 15 |
| Use case name: Manage groups |
| Intent: To allow administration staff to manage groups and their membership |
| Precondition: User logged in as administration staff |
| Use case initiation: Selection of “Manage groups” tab within users |
| Dialog (Description) :   1. System: List of groups 2. User: Selects relevant group 3. System: Presents group with a list of its current members plus alter and delete options 4. User: Performs desired actions 5. System: Updates the file server   Alternate Flows:  2a. If assignment doesn’t exist user adds before proceeding |
| Use case termination: User exits from manage classes |
| Post Condition: Database contains up to date performance data |

|  |
| --- |
| Use case number: 16 |
| Use case name: Perform partial backups |
| Intent: Allow users perform backups of the database or file server |
| Precondition: User logged in as administration staff |
| Use case initiation: Selection of “Manage groups” tab within “Users” |
| Dialog (Description) :   1. System: List of groups 2. User: Selects relevant group 3. System: Presents group with a list of its current members plus alter and delete options 4. User: Performs desired actions 5. System: Updates the file server   Alternate Flows:  2a. If assignment doesn’t exist user adds before proceeding |
| Use case termination: User exits from manage classes |
| Post Condition: Database contains up to date performance data |

|  |
| --- |
|  |

Fig 14. Initial ERD of logins databse

|  |
| --- |
| Fig 15 Partial Initial ERD for core database |
|  |

Fig 16 rest of initial ERD for core database

# Prototyping and Development

|  |
| --- |
|  |

Fig 17. Screen shot of prototype home screen

As a prototype a home screen has been created as well as a user administration screen where a limited number of user details can be entered and saved to the database. A message with a user id is presented to the user upon completion. A small amount of user data can also be retrieved and displayed in a simple table.

# Testing

Whitebox

(Testing carried out by a tester with indebt knowledge of code how it is supposed to perform and expected results.)[38]

### Unit testing

As stated previously the AUP model for software development will be used as a guide for this development process. As per the project design plan development will be carried out in phases which each phase containing sub phases which contain individual components for development. Once developed the new component will be tested by the developer to ensure it performs as expected. Before any development commences on a particular requirement clear guidelines on what it should and shouldn’t deliver will be developed to aid testing. Testing will not only be carried out to ensure the particular feature works but also to ensure it cannot be broken.

Performed Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Req ID | Test Case # | Test | Round | Result |
| 1 a | 1 | 1.0 | Insert User | 1 | Fail |
| 1 a | 1 | 1.0.1 | Insert User | 2 | Fail |
| 1 a | 1 | 1.0.2 | Insert User | 3 | Pass |
| 1 a | 1 | 1.1.0 | Insert Duplicate User (proceed) | 1 | Pass |
| 1 a | 1 | 1.1.1 | Insert Duplicate User (abort) | 1 | Pass |
| 1 a | 1 | 1.2.0 | Constraints Check | 1 | Fail |
| 1 a | 1 | 1.2.1 | Constraints Check | 2 | Pass |
| 1 a | 2 | 2.0.1 | Secure User Login | 1 | Pass |
| 1 a | 2 | 2.0.2 | Secure User Login | 2 | Fail |
| 1 a | 2 | 2.0.3 | Secure User Login | 3 | Pass |
|  |  |  |  |  |  |
| 1 b | 4 | 3.0.0 | Restore User | 1 | Fail |
| 1 b | 4 | 3 b | Restore User | 2 | Pass |
| 2 a | 5 | 4 a | Insert category | 1 | Pass |
| 2 a | 6 | 5 a | Insert course | 1 | Fail |
| 2 a | 6 | 5 b | Insert course | 2 | Pass |
| 2 a | 6 a | 6 | Insert course from on category | 1 | Fail |
| 2 a | 6 a | 6 b | Insert course from on category | 2 | Fail |

Testing was carried out continuously during development. Many issues where encountered during development mostly due to a lack of knowledge regarding JQuery, Ajax and JavaScript. Initial research had been carried out into these technologies but as with most new skills the best way to test new skills is to use them.

The table above includes testing results from unit test cases carried out by the developer after each sub phase had been completed. These tests were carried out on the newly developed piece of functionality in isolation. Details of all integration testing are outlined in the next section.

Phase 1 Unit Test Analysis

The first piece of development carried out was on the requirement which specifies a user should have the ability to add a user through a digital form. As the table shows the first round of testing resulted in failure. This was due to the fact that no date of birth was visible in the database. After some quick research it was discovered that the database expects the date to be added in the year month day format due to its default configuration. Upon further testing it was realised that the temporary external ids which were generated using a random numbers where not being updated with the newly generated database id’s which resulted in users having the same external id which caused an error during selection as more than one result was being returned. The first test also failed due to the fact the user table did not reflect the changes carried out even though they were present in the database. A substantial amount of research was required to find the most effective and efficient way to carry this out.

The next set of unit testing was checking the systems response when a duplicate first name and surname were entered in the form. Testing was carried out to ensure the user could proceed and insert the data or abort and close the form. Both tests passed. Tests were also carried out to ensure data integrity constraints on the user form were performing as expected. the initial test carried out failed as the constraints were too sever in some cases and didn’t work at all on the email address question.

The nest requirement to be tested was the ability to login with a secure password and username. The first round of testing carried out did not included the Security Manager functionality which hashes passwords when entering them into the database and un-hashes them when retrieving. This first round of tests passed. When the hashing functionality was added the next test failed. This was due to the password comparison function which contained a mistake in how the password and username was retrieved from the database for comparison. Once this issue was rectified the third test passed.

The next requirement to be developed was allowing a user which had previously been deleted from the system to be restored

### Integration testing

Once unit testing has been completed successfully on a particular component it will be integrated into the existing system and tested to evaluate its interaction with existing hardware and software. As well as testing Interoperability this testing will focus on the affect on performance and security after integration.

## Blackbox

(Testing carried out by a tester with no knowledge of the code and how it is supposed to perform)[39]

Once a sub-phase has been completed it will be released to users for testing. These users will have no knowledge of the code. Test-cases will be developed to cover different scenarios.

Round 1.

Integration Testing

|  |
| --- |
| **Test Case Number:** 1 |
| **Test Case Name:** Insert user data |
| **Purpose:** To ensure data for a new user can be added to the database through the application |
| **Procedure Steps:**   1. Select add user in the users screen 2. Fill out typical user information 3. Submit 4. View message |
| **Expected Results:**   1. **Success Message received** 2. **All user details in TB\_User** 3. **Same user in tb\_Logins** 4. **New user visible on screen** 5. **Ability to logon with user name and password** |
| **Results**  **1: Pass**  **2: Fail – no dob**  **3: Fail**  **4: Fail**  **5: Fail** |

Round 2

|  |
| --- |
| **Test Case Number:** 1 |
| **Test Case Name:** Insert user data |
| **Purpose:** To ensure data for a new user can be added to the database through the application |
| **Procedure Steps:**   1. Select add user in the users screen 2. Fill out typical user information 3. Submit 4. View message |
| **Expected Results:**   1. **Success Message received** 2. **All user details in TB\_User** 3. **Same user in tb\_Logins** 4. **New user visible on screen** 5. **Ability to logon with user name and password** |
| **Results**  **1: Pass**  **2: Pass**  **3: Pass**  **4: Pass**  **5: Fail** |

|  |
| --- |
| **Test Case Number:** 1 |
| **Test Case Name:** Insert user data |
| **Purpose:** To ensure data for a new user can be added to the database through the application |
| **Procedure Steps:**   1. Select add user in the users screen 2. Fill out typical user information 3. Submit 4. View message |
| **Expected Results:**   1. **Success Message received** 2. **All user details in TB\_User** 3. **Same user in tb\_Logins** 4. **New user visible on screen** 5. **Ability to logon with user name and password** |
| **Results**  **1: Pass**  **2: Pass**  **3: Pass**  **4: Pass**  **5: Pass** |

|  |
| --- |
| **Test Case Number:** 1a |
| **Test Case Name:** Insert duplicate user data |
| **Purpose:** To ensure data for a new user can be added to the database through the application |
| **Procedure Steps:**   1. Select add user in the users screen 2. Fill out form with typical user information except using same name as previously 3. Submit 4. View message |
| **Expected Results:**   1. Message provided to the user that there is a possibility of a duplicate entry in the system 2. User able to submit data |
| **Actual Result:**   1. **Pass** 2. **Pass** |

|  |
| --- |
| **Test Case Number:** 1b |
| **Test Case Name:** Ensure data integrity checks |
| **Purpose:** To ensure user cannot enter incorrect data type into fields |
| **Procedure Steps:**   1. Select add user in the users screen 2. Enter number into name fields 3. Skip address 1 filed 4. Enter single word into email field |
| **Expected Results:** user shouldn’t be able to proceed without correcting mistakes. Clear error message provided. |
| **Actual Result:** |

|  |
| --- |
| **Test Case Number:** |
| **Test Case Name:** Add new Category |
| **Purpose:** To Ensure New Categories can be entered |
| **Procedure Steps:**   1. Select add category from category home screen 2. Fill in name and description of category 3. Submit 4. View message |
| **Expected Results:**   1. **Success Message received** 2. **New category in tb\_category** 3. **New category visible on screen** |
| **Results**  **1. Pass**  **2. Pass**  **3. Fail** |
| **Notes:**  **Table not refreshed** |

|  |
| --- |
| **Test Case Number:** |
| **Test Case Name:** Add new Category |
| **Purpose:** To Ensure New Categories can be entered |
| **Procedure Steps:**   1. Select add category from category home screen 2. Fill in name and description of category 3. Submit 4. View message |
| **Expected Results:**   1. **Success Message received** 2. **New category in tb\_category** 3. **New category visible on screen** |
| **Results**  **1. Pass**  **2. Pass**  **3. Pass** |
| **Notes:** |

|  |
| --- |
| **Test Case Number:** |
| **Test Case Name:** Add new Course from Category Row |
| **Purpose:** To Ensure New Categories to a category when selected from a table |
| **Procedure Steps:**   1. Click on plus sign in desired row 2. Fill in/select details 3. Submit form 4. View message |
| **Expected Results:**   1. **Success message** 2. **Course in TB\_courses if new** 3. **New entry in tb\_tutor\_courses** |
| **Results**   1. **Fail** 2. **Fail** 3. **Fail** |
| **Notes:**  Select for tutor not populated |

This test failed during the first round of testing due to the fact that the tutor and course name selects did not get populated. This issue was rectified and the test was carried out again.

Above are examples of test cases which will be provided for testers. Testers will be made up of members of staff from an education centre, plus another who works in administrative role for an assurance company and has vast experience using data gathering applications, a software engineer as well as one tester who has little experience of software applications or IT in general. The main aim of this type of testing is to ensure the application meets the user requirements and is adhering to the usability criteria guidelines set out in Nielson’s Heuristics. Tests that result in unwanted results will are included to ensure firstly error are caught and also that users are informed with clear easy to understand error messages. Extremely short surveys will be used to analyze the testers overall thoughts of the application.

## Quality Assurance and User Acceptance Testing

Once the application nears completion it will be released to a set of users for the equivalent of user acceptance testing. Testers will be asked to use the application as if they were performing everyday work duties. Once testing has concluded surveys were sent out the tester will be asked to complete a survey which again will be used to analyze the applications overall performance.

# Issues and risks

An overall analysis of the developer’s academic performance over the past few years would highlight a weakness when it comes to development. For one reason or another developer has struggled in certain aspects of this discipline mostly in user interface design and development. A large attraction in developing a web application rather than another type of application was the opportunity to improve these development skills. Adding to the challenge is the choice of using JavaScript and JQuery to develop the user interface. The developer has no previous experience in these technologies but is keen to utilise them as they are widely used within industry today. In order to overcome these issues the developer has been actively practising implementing user interfaces with these technologies and plans on further enhancing theses skills over the coming weeks.

Although no one part of this project should be overly complex there are many pieces of functionality which need to be integrated which may all contain complexities and present issues when integrating which could present unforeseen time issues. As shown below a comprehensive plan has been developed which has split the work into phases and sub-phases each containing units of development. These phases have been planned so that the most important are developed first working down to least important. In phase “” future work has been included in the off chance there is some spare time available. it will be extremely important to manage time and stick to this plan as rigorously as possible to ensure the application contains some substantial functionality.

# Plan and future work

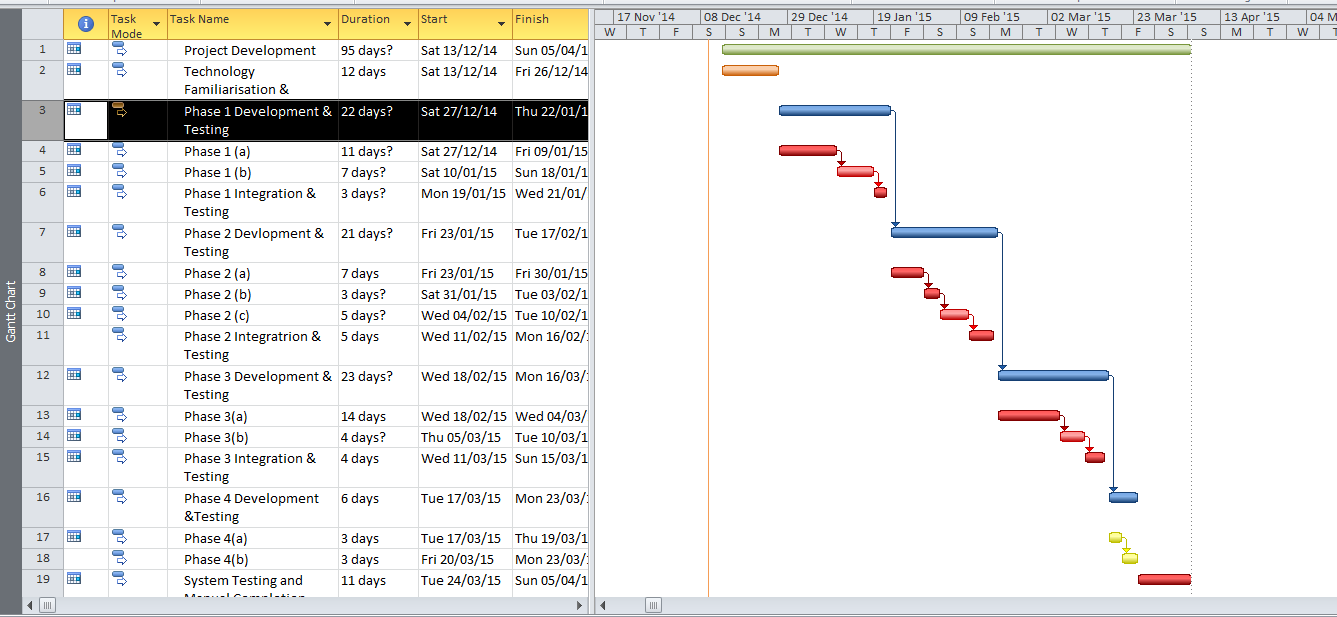
****

Fig 18. Gant chart displaying schedule of future work

# Conclusions

At the beginning of this project the main aim was to try and design and develop an application which would be useful to others and make their lives a little easier. A conscious effort has been made through requirements gathering and analysis to only include features that will be utilized and add worth to the application. Through personal observations and thoughts an idea of requirements had been formulated. In order to achieve my initial aim brainstorming sessions and interviews were set up and surveys were sent out to perspective users. It is felt that through these methods and research into existing systems that a relevant and achievable set of features have been outlined.

A personal goal in completing this questionnaire was to be left with a very clear idea of what was required and what would be needed to achieve these requirements. A good lesson has been learnt from filling out this form in that time should not be taken for granted and even the simplest of task can take a lot longer than expected. Even though a lot of work has been put into completing this form there are certain aspects that could be improved greatly. Going forward it is realised when researching notes should be recorded properly as should references as it will make things a lot easier in the future.

It is felt through the design of a comprehensive design and test plan that all crucial requirements are achievable within the project time frame. It is clear that discipline as well as organisational and time management skills will be will be crucial in order to get the project over the line but if these skills are used and a plan is stuck to success should be attainable.

Implementation

Phase 1(a)

Secure Login

Phase 2(b)

Manual Insertion of User Data

Further referneces

[37] <http://www.ambysoft.com/unifiedprocess/agileUP.html>.

[38] <http://www.webopedia.com/TERM/W/White_Box_Testing.html>

[39] <http://www.webopedia.com/TERM/W/Black_Box_Testing.html>