FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computer Science & Mathematics*

BSc DEGREE
IN
Information Systems

PROJECT REPORT

Name: Simran Johal

ID Number: K1403042

Project Title: Student Attendance Monitoring System (SAMS)

Project Type: Build

Date: 23/04/2018

Supervisor: Graeme Jones

Kingston University London

Declaration

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks ("…"). Neither in part nor in its entirety have I made use of another student's work and pretended that it is my own. I have not asked anybody to contribute to this project in the form of code, text or drawings. I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

Date: 21/04/2018

Signature:

Acknowledgement

I would firstly like to thank Professor Graeme Jones who was an outstanding supervisor and client. Without the guidance and support received, I would not have been able to push beyond the limits of my capabilities. I express the utmost gratitude for the advice, motivation, encouragement and knowledge along the way. I am eternally grateful.

To my friends and family I've found the best support system I could have asked for. The journey has been challenging and far reaching, but I could not have done it without each and every one of you.

Most of all I would like I would like to say the most heartfelt of thank yous to my parents. Everything I do, I do for you.

TABLE OF CONTENTS

1 INTRODUCTION **6**
1.1 Background and Problem 6
1.2 Solution 7
1.3 Aims and Objectives 8
1.4 SWOT Analysis 8
1.5 PEST analysis 9
1.6 Project Management 10

2 REQUIREMENTS ANALYSIS **11**
2.1 Stakeholder Identification and Evaluation 11
2.2 Eliciting Requirements 13
2.3 Users & User Stories 13
2.4 System Requirements 17
2.5 MoSCoW 19
2.6 Conclusion 19

3 LITERATURE REVIEW **20**
3.1 Data Protection Act 1998 20
3.3 Computer Misuse Act 1990 23
3.3 Conclusion 24

4 DESIGN **26**
4.1 Data Modelling 26
4.2 Process Modelling 31
4.3 User Interface 34
4.4 Conclusion 43

5 IMPLEMENTATION **44**
5.1 Tools, Technologies and Architecture 44
5.2 Implementing the Database 48
5.3 Constructing the User Interface 51
5.4 Security Concerns and Precautions 61
5.5 Conclusion 62

6 TESTING AND VALIDATION **64**
6.1 Validation of Functional Requirements and System Needs 64
6.2. Validation of the Database 65
6.3 Requirement Testing 67
6.4 Usability testing 69
6.5 Conclusion 72

7 CRITICAL EVALUATION **73**
7.1 Achievements 73
7.2 Problems Encountered and Limitations 74
7.3 Critical Analysis and Future work 75
7.4 Conclusion 76

BIBLIOGRAPHY **77**

APPENDIX A: PROJECT MANAGEMENT AND SOFTWARE METHODOLOGY, RISK ANALYSIS AND CONTINGENCY PLAN AND DELIVERABLES WORK FLOW **79**

APPENDIX B: FULL USER STORIES **82**

APPENDIX C: FULL USE CASE DIAGRAM **84**

APPENDIX D: FULL USE CASE TEXTUAL DESCRIPTION **87**
APPENDIX E: FULL MOSCOW ANALYSIS ON FUNCTIONAL REQUIREMENTS **94**
APPENDIX F: EXEMPTIONS FROM THE DPA 1998 ..... **101**
APPENDIX G: NEW GDPR DATA PROTECTION PRINCIPLES **102**
APPENDIX H: INITIAL ENTITY-RELATIONSHIP DIAGRAM **103**
APPENDIX I: FULL BREAK DOWN OF ERD **104**
APPENDIX J: FUNCTIONAL REQUIREMENTS SQL QUERIES CREATED TO TEST DATABASE DESIGN **106**
APPENDIX K: FULL DATA DICTIONARY **108**
APPENDIX L: ACTIVITY DIAGRAMS **111**
APPENDIX M: SEQUENCE DIAGRAMS **113**
APPENDIX N: LOW FIDELITY WIRE FRAMES **114**
APPENDIX O: HIGH FIDELITY WIREFRAMES **118**
APPENDIX P: FULL NAVIGATION DIAGRAMS/ SITE MAP **127**
APPENDIX Q: METHODS IN POPULATING AND VALIDATING DATA **129**
APPENDIX R: $_POST VS $_GET **131**
APPENDIX S: ENTITY-RELATIONSHIP DESIGN AND DATABASE CONSTRUCT TESTING **132**
APPENDIX T: DATABASE ROBUSTNESS & QUERYING THE DATABASE **135**
APPENDIX U: DATABASE ROBUSTNESS & QUERYING THE DATABASE **136**
APPENDIX V: BLACK BOX TESTING FOR FUNCTIONAL REQUIREMENTS **138**
APPENDIX W: BLACK BOX TESTING FOR NON-FUNCTIONAL REQUIREMENTS **144**
APPENDIX X: USABILITY TEST TASKS AND POST QUESTIONNAIRE RESULTS **146**

1 Introduction

1.1 Background and Problem

According to University and Colleges Admissions Service (UCAS) 2017 end of cycle report, the UK holds the highest number of UCAS applications, with a staggering 82% (UCAS, 2017). However, (HESA, 2018) 2016-2017 results show that almost one in eight university students drop out after their first year. These could be for a number of unknown reasons. However, studies show that academic success can be vastly influenced by attendance. Founded in 1899, Kingston University supports 17,547 students in four campuses, offering hundreds of courses including foundation, undergraduate, postgraduate and short courses (Kingston University, 2018). As Kingston University accommodates for over thousands of students, it is easy for students who have not attended to go unnoticed. As a result, students subsequently fall into patterns of consistently missing events and disengaging from their modules. The spiral of missing events and underachieving can cause student to terminate their university career.

The current paper-based attendance system lacks efficiency, reliability and convenience. Students can easily forge signatures and impact the validity of attendance data. The sensitive information this data obtains runs the risks of breeching laws if lost, misplaced or fallen into the wrong hands. Presently, there is an inability to spot trends, patterns or relationships with attendance data. There is also no way to share or track attendance over modules, furthermore no platform for individuals of interest to act accordingly. Overall all the current system lacks engagement, acknowledgement, awareness of presence and integration. Figure 1.1 demonstrates a rich picture which involves a summary of the issues with the current system. A solution must involve identifying and intervening in early stages of disengagement to maximise student performance and academic experience.

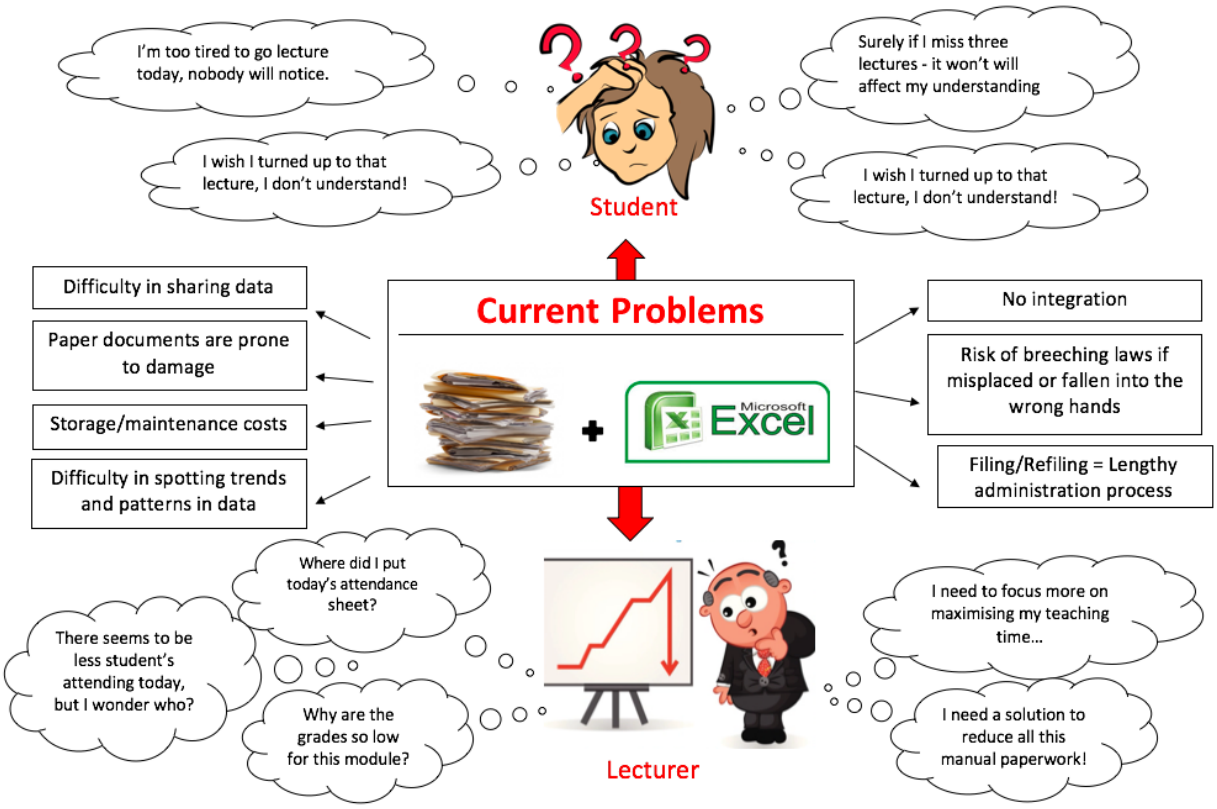


Figure 1.1: Rich Picture on Current Attendance Problems

1.2 Solution

The Student Attendance Monitoring System (SAMS) is an innovative and proactive solution to monitor the attendance of students studying in Higher Education (HE). This system aims to boost engagement, performance and maximising student potential. It will operate using an electronic attendance system. Lecturers will take attendance which will be stored in a database behind various security precautions to prevent violation, sabotage and misuse of data. This system will integrate the administration tasks to effectively prioritise time elsewhere. Shifting the current system to an online web-based application will increase the reliability and security of data. The system will aid users with a visual representation of the data, allowing a different perspective to the standard paper-based platform. The proposed system will solve the miscommunication between users. Most importantly, this system creates a solution to spotting trends or patterns in declining attendances. This in turn gives institutions, such as Kingston University, the opportunity to facilitate and accommodate the needs of students with the aim of increasing levels of engagement. It is a project seeking a resolution to the missed steps between disengaging students and a fruitful education.

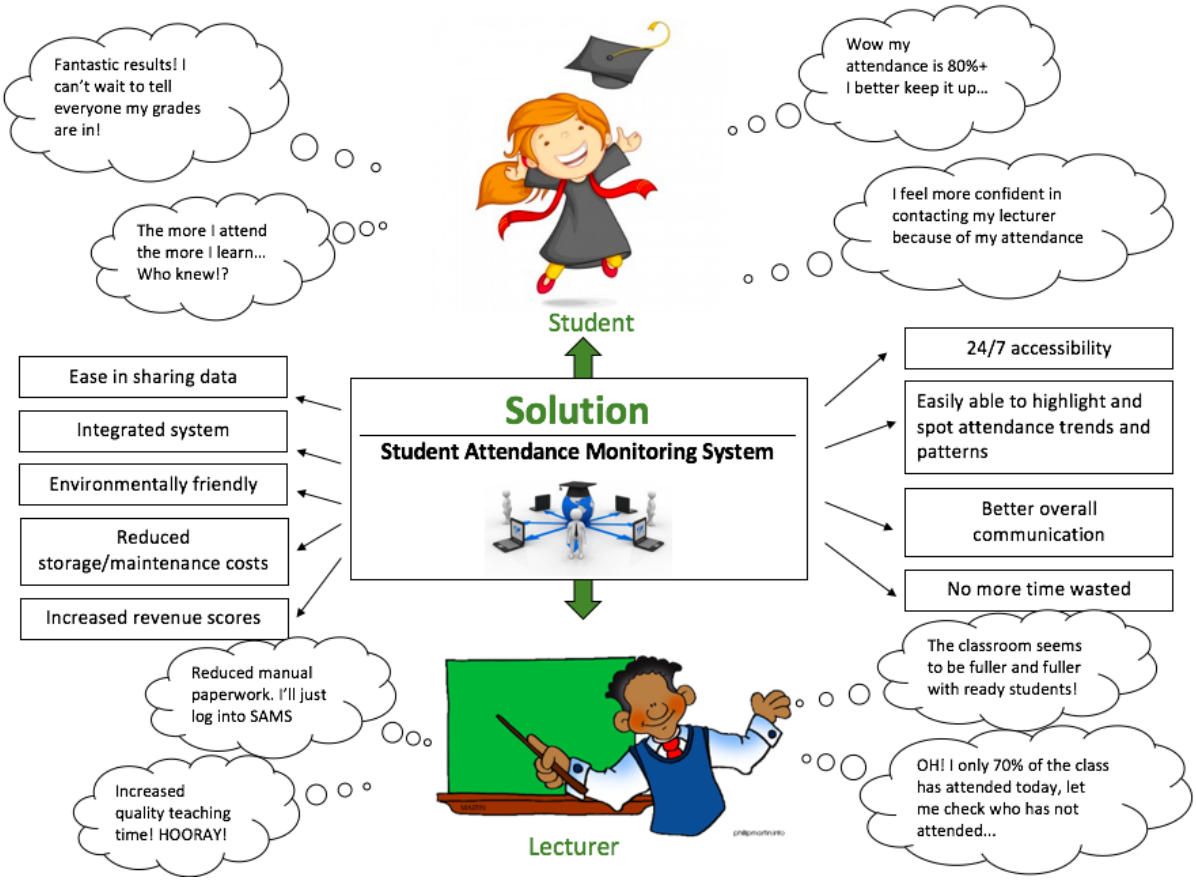


Figure 1.2: Rich Picture on Solution by Incorporating SAMS system

1.3 Aims and Objectives

In order for the SAMS web application to reach a successful end product, various goals will need to be accomplished. Therefore, the project will go forth with the overall aim and objectives of the following:

Aim:

"To increase attendance and engagement at Kingston University"

Objectives:

- To develop a user-friendly interface to be used daily.
- Each operation executed will provide feedback within 2 seconds.
- Reduce the time to complete the lengthy administration process of attendance by more than 50%.
- To implement a database design which can withhold over a thousand entries.
- To increase healthy communication between students and lecturers.
- Ensure data is accessible across faculty.
- Increase student retention through monitoring attendance data.

1.4 SWOT Analysis

SWOT is an analytical framework which identifies a projects strengths, weaknesses, opportunities and threats, both internally and externally. It can evaluate the position of the project and analyse the strength of survival and accomplishing aims and objectives set. Table 1.1 demonstrates the SWOT analysis conducted on the SAMS application. However, before doing so the criteria for the analysis is the following:

- **Strengths** - Favourable aspects of the project to assist with project success.
- **Weaknesses** - Unfavourable aspects of the project that could cause the project harm.
- **Threats** - Factors that could affect or threaten the success or completion of the project.
- **Opportunities** - Favourable factors of the project that will assistant with achieving the end result and giving it a competitive advantage.

Strengths	Weaknesses
✓ Previous experience with database designing, front end coding, and UML modelling. ✓ Accessibility to resources and assistance. ✓ High levels of knowledge of system users. ✓ Loyal users readily available to test system and give feedback. ✓ All users will be able to monitor attendance. ✓ A unique, positive and active perspective on engaging students. ✓ A centralised integrated system for sharing information amongst users. ✓ Online facility over paper based - reduced length administration processes. ✓ Increased overall productivity leading to high achievement rates.	✓ Limited/lack of experience in some programming languages. ✓ Pressures from meeting deadlines and other commitments, could affect project outcome and deliverables due. ✓ Weaknesses causing a delay in project deliverables and deadlines. ✓ Poorly designed interface and slow processing speeds could discourage users from making full use of the system.
Opportunities	**Threats**
✓ Product enhancement to add more/better functionality to the system. ✓ An interface that aimed to improve attendance can cause a rise in success rates	✓ Database will hold sensitive data about users, and therefore hold a threat of being hacked and/or manipulated. ✓ Database connection is mandatory for

therefore increase revenue scores for the university. ✓ Marketing platform to new undergraduates to attract them to Kingston University. ✓ Exploring the success amongst other universities to measure their engagement with their students – national venture. ✓ Exploring and adapting system to fit other needs for other institution sectors e.g. professional working environment attendance. ✓ Can be later sold on to potential interested customers/partners. ✓ Faculty can track and record attendance at their convenience.	system to work, failure of this will mean an insufficient system or incorrect data submitted and therefore in reports during this time. ✓ Updates to any interfaces used may cause the system to become obsolete in the future. ✓ Competitors can create a similar system with better technical expertise and minimise all opportunities for expansion. In additional, already maintain market share.

Table 1.1: SWOT Analysis on the Project

1.5 PEST analysis

The PEST framework analyses four external business factors, 'Political, Economic, Social and Technological', and how they will affect the projects performance and future actions. Fundamentally, the PEST framework "assumes that external and indirect circumstances are able to influence organisational operations, strategic planning, and management solutions" (Lin H. 2013). Table 1.2 demonstrates a PEST analysis conducted on the SAMS web application.

- **Political** - Government regulations and legal factors are assessed to address their impact on the project.
- **Economic** - Economic issues that could have a heavy impact on the project.
- **Social** - The socio-economic environment is analysed to assist in assuring how to meet and understand customer/user needs and what drives them for market purchase.
- **Technological** - Assesses how technology can either positively or negatively impact the introduction of this project.

Issues	Impact on Business
Political	
• The government introduce new and stronger policies in mandatory student attendance in higher education. • Government introduce stronger policies which require higher education institutions to monitor the progress of students' academic achievement over their time at University.	• Online computerised resource for storing and monitoring attendance saves costs, time and effort. • Reduced lengthy-administrative time used by staff resulting to more time spent efficiently teaching. • Monitoring attendance will highlight disengagement and improve overall interaction with students. Therefore, leading to improved academic results, increased revenue scores and standards of the institution.
Economical	
• Global economic pressures resulting in reduced disposal income per household to accommodate further education. • Increased costs in higher education fees.	• Unwillingness of parents/students to spend or loan such large amounts of money to continue with education. • Students may continue with other career paths.

Social	
• Decline in students attending higher education or finding an alternative path, reflecting national trends. • Closure of local firms seeking to employ students and therefore deterring away from higher education. • Inability to attract teaching staff or staff moving on in their careers.	• Reduced revenues and increase in costs – introduce extra features to assist education success and attract student to the University e.g. academic achievement prediction. • Impact in the reputation for University - the system will improve overall productivity and communication between all faculty and students and can later introduce a new parent aspect to the system.
Technological	
• Increase in modernised attendance technology focused on higher education. • Time required to maintain system.	• Increase in competition - improve and enhance system communication and features between the institution and its users. • Improve overall productivity and system resources.

Table 1.2: PEST Analysis on Project

1.6 Project Management

Adopting an efficient project management method can enable over all control and management throughout the project lifecycle. Many obstacles and contingencies may impact the project flow and therefore employing the most appropriate methodology is crucial for project success. It was decided that the most suitable approach to manage this project and develop the software is DSDM/Agile methodology. There are a number of reason for this decision, outlined in the following:

- This methodology prioritises requirements by value. Therefore, the system will adopt the most crucial requirements in the limited time frame. Additional qualities will bring the system extra value in project delivery.
- Client and stakeholder involvement is crucial in any project. This approach will enable the developer to work closely with the client and stakeholders to ensure the project success. In addition, the developer is able to specify in many system areas, features and components.
- By adopting a single methodology for both project management and software development will enable more time to be utilised effectively on the project.
- The agile methodology employs small time framed “sprints”. As the project has a time constraint, these sprints will create minimal threat when delivering the planned deliverables.
- This management style is flexible and dynamic, allowing each stage to be iterated and recursive for the most effective effect. In addition, this approach also enables to
- This approach enables testing throughout the process. Therefore, requirements can be modified at any stage without having a detrimental impact on the project flow.

Further detail on the methodology, including the Risk analysis and Contingency Plan and Project deliverables can be found in *appendix A*.

2 Requirements Analysis

The services that the system will provide and the limitations on its functions are mapped by the requirements. Requirement engineering is a procedure which involves discovering, capturing and analysing requirements and their constraints. If requirements are unclear or vague, this can cause disruption through the various phases of the project, thus resulting in delay or failure. This chapter will identify each stakeholder and the effect they have on the system, differentiating stakeholders to primary system users. Through a series of interviews with clients and user, the developer is able to derive the desired user goals. By successfully capturing all user goals, it will determine the strength of the project. As a result, these goals can be translated into requirements to design and implement the SAMS web application. This chapter will comprise of various tools and techniques in order to elicited and showcase the system and user requirements. In addition, the developer will work closely with the client to assure the foundations of the project are firmly composed. The requirements will be captured via user story's, use case diagrams, use case textual descriptions, they will be prioritized using MoSCoW prioritization. But firstly, the project must outline the stakeholders involved to understand the underlying system needs. Stakeholders will then be analysed through a stakeholder matrix.

2.1 Stakeholder Identification and Evaluation

A stakeholder is a person, group or organisation who has an interest or concern in an organisation and its activities. They can have heavy effect on an organisations actions, outcome and desires. Stakeholders can be both internal and external actors who contribute a positive and negative influence on an organisation's projects. For this project there are serval stakeholders, not all which are users, but all which wish to see the system succeed. By identifying and evaluation the strength of each stakeholder the developer is able to appropriately compose the system around the underlying requirements from the stakeholders. This can then be utilised to shape the project and application.

Internal stakeholders

- *Lecturer*
  Lecturers will be able to take attendance and closely monitor disengaging students. Administration time used to register and chase absent students will be saved due to the new system. In addition, patterns and trends in attendances will be highlighted for future judgement and decision making. A lecturer could also be one of the following:
  - *Module Leader*: Module leaders are able to view module attendances to get a better understanding of student attendance.
  - *Course Director*: Course directors can view the modules individually and the course as a whole, to figure out underlying issues with attendance.
  - *Head of Department*: The head of department will want to see department, with courses and modules attendance results included. In these stages, users are able to capture and overview of patterns and trends in attendance.
- *Students*
  Students attending university can view their attendance and the effect it has on their achievement. They can leave comments related to their absences.
- *Administrator*
  The administrator will not use the system as such but will focus on maintaining the system for the users. Administrator will have full control of the database and website, able to create, review, edit and delete.
- *Vice Chancellor*
  The vice chancellor will have high interest levels in the attendance to analyse reports and draw conclusions for future University needs.

External stakeholders

- *Parents/guardians*
  Parents/guardians have little control and knowledge of student attendance. However, in consideration with the expenses and energy, they care greatly about the outcome of their child's education.
- *Government*
  Key stake holder which provides finance, certain rules and regulations to follow for a successful institution.
- *Future Recruiters*
  Recruiters will be looking for newly educated graduates to start their career in their companies. They will care to see the punctuality of these students.

Stakeholder Power/Interest Matrix

Stakeholder Matrix is a useful tool used to analyse the position of stakeholders within a project. As illustrated in figure 1.1, each stakeholder has been categorised by the metrics of power and interest towards the project. The matrix is sub-categorised into four segments: keep satisfied, manage closely, monitor, and keep informed. This enables the developer to gain a deeper understanding on the stakeholders influence upon project and crucial decisions. In addition, such diagrams can assist in managing and communicating with stakeholders throughout the project. They can highlight and clarify the best channels and tools in which to effectively communication with all interested parties. However, the matrix will need adapt and modify throughout the project lifecycle to ensure accuracy due to stakeholders influence at each stage. This is key to ensure all criteria is met for the project success.

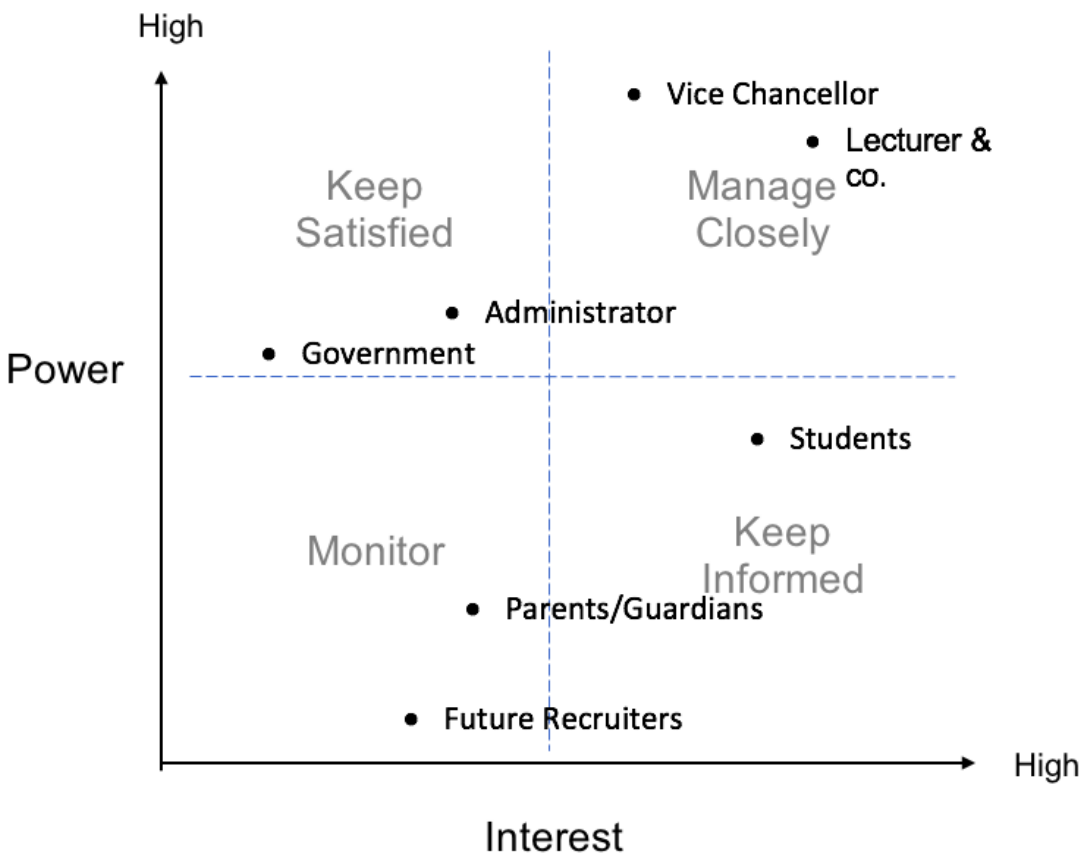


Figure 1.1: Stakeholder Power/Interest Matrix

2.2 Eliciting Requirements

The key measurement of success for an application is the degree in which the proposed requirements are met. Requirement gathering is a key phase in the development of an application, demonstrating the importance of the projects "real-world goals" which motivate the development of the software. Requirements gathering is crucial to seek what problem must be solved, and identify system boundaries. The client and users are the most critical medium through which requirements are captured and conveyed. Having established the key stakeholders involved, the next step- requires addressing the requirements of the system in further detail.

A series of interviews were conducted to gain a deeper understanding of the client and user needs. This beneficial aspect outlined the desired requirements for the system and how it is expected to operate. This process will aid the developer to create efficient user friendly system to be utilised on a daily basis as demonstrated in the objectives for the product. The client and potential users were questioned about the current system and what would drive a new more effective system to success.

Interview questions

- How your tasks are currently completed?
- What is the done with this data after?
- How does the current system lack inefficiency?
- Is there any flow or processes in the current system you would like to enhance or discarded?
- What are the key functionalities of the proposed system you would like for it to perform?
- Is there any extended functionality which could be incorporated?
- What are you expectations when it comes to accessibility, responses and usability?
- How do you intend to interact with the system?
- How technically advances are your users?
- What age are the typical users for this system?
- Is there user manual or additional training required to utilise the proposed system?

The questions above, enabled enriched knowledge to come forth with the project. This information can now act as the foundations of the project. However, the client and users will need to be informed of the process along the way as forecasted in the stakeholder analysis to deploy success.

2.3 Users & User Stories

After the interviewing process in order to understand how the system will unfold, it is crucial to consider the tasks and goals of each user type. Although there are many individuals who partake an interest in the system the requirements will be predominantly catered to the users of the system. The knowledge gathered from the interviews enriched the idea of user and their needs:

- Lecturer

This project will be heavily beneficial for lecturers as the current lengthy administrative resource will be replaced by a user-friendly efficient attendance system. Thus, lecturers will be the primary user of this system. The system will provide a means of monitoring student's attendance and engagement. A chance to intervene in early stages of disengagement to assure that student potential is maximised during their education is provided. Lecturers may also be one of the following: module leader or course director or head of department. Therefore, the system will present unique additional privileges according to the nature of their job roles.

- Student

Attendance will be taken by lecturers in every session, which can be reviewed and monitored through the SAMS application. Furthermore, students are given the opportunity to view and monitor their own attendance. Although the students do not have as many access rights, they are able to upload correspondences to provide rationale for any absence.

- Administrator

The administrator will have overall control of the different components of the system. The administrator will have the privileges to create, review, edit and delete system entries. Although having a great deal access to the system the administrator will not be using the system for its planned purpose and is therefore an indirect user.

Each user will partake a role upon the system executing certain tasks. A user story will grasp an enriched description of user tasks leading to their end goal. This agile software tool captures an insight on software features from a user's perspective. The tool provides a clear descriptive vision of the tasks in hand and the purpose for all too easily understand. User stories are captured from all primary actors (administrator, lecturer and students) to allow a better understanding of the system and system needs. By collecting this information, it creates a starting point for use cases to be derived which will in turn generate the system requirements. Table 1.1 illustrates the user story of the Lecturer. Full user stories can be found in *appendix B.*

User	Task in hand	For the purpose of
Lecturer	Take Attendance	I can record and later monitor student attendance.
	Confirm Attendance	I can assure that all attendance details are correct and submit to the database.
	View Attendance	I can review student's attendance for the current event.
	Add new session	I can create a new session to take attendance for.
	Edit Attendance	I can modify student attendance records.
	View Attendance Report	I can review student attendance records.
	Search Student Attendance Report	I can track down particular student attendance report to monitor.
	Save Attendance Report	I can keep student attendance reports for records.
	Print Attendance Report	I can keep student attendance reports for records.
	View Course Attendance (specific)	I can review overall course attendance.
	View Module Attendance (specific)	I can review overall module attendance.
	View Student Comments	I can view student queries and respond.
	View Student Correspondences	I can view student reasons for absence and respond.
	Contact Student	I can contact students in relation to absence and disengagement from a module.
	Edit Personal Details	I can modify personal records.
	Reset Password	I can reset my access to the system should there be a need to

Table 1.1: Lecturer user story

2.3.1 Use cases diagram

The user stories outline the goals of the users in context. These can be utilised in conjunction with use case diagram which clearly identify, organise and illustrate how to system will respond to various tasks. Such diagrams enable a detailed description of each user interacting with the system, highlighting and ultimately revealing issues in early stages of the project. Overall, it is declaration of how to perform the user story systematically and how the system behaves. Therefore, derived from the interviews with the client and potential users, the use case diagrams for SAMS have been split

into the different user's types for a deeper understanding. Each use case consists of narrative text which is extremely beneficial for all stakeholder to understand the clear route for the proposed system. However, it is key to note before implementing a use case diagram one must be familiar with the conventions as such:

- **System boundary**: System boundary is a border around the user cases encapsulating the information. Actors are generally shown around the border.
- **Actors**: Actors are illustrated through stick figures, as shown in Figure 1.2. They play a critical role in the system however they do not have to be human, just a being with behaviour. They can be generally known as primary actors. At times an external actor is required to portray the effects of the system actions. Such actors are known as supporting actors of secondary actors.
- **Use Case**: Use case is demonstrated as an oval. Forecasted in a verb-noun format, they generally depict the functional requirements of the system. Use cases are the goals to be completed by the actor.
- **Use Case Relationships**: The association between actors and use cases are demonstrated in the table 1.2. These relations can also be found from use cases to use cases in some cases.

Relation	Demonstration	Rationale
Association		These are the relationships between actors and a use case, to demonstrate association.
Include	«include»	A use case may require a functionality from another use case in order to perform the given task.
Extend	«extend»	Extending the functionality of a specific use case to another.
Indirect Association		These relations are the indirect association with a secondary actor. The goals achieved can forecasts an effect on these actors or they may assist to achieve the goal.

Table 1.2: Notations for use case relationships

Figure 1.2 illustrates demonstrates the lecturers activity with the system. The lecturer will have direct access to the system and therefore is a primary user. The lecturer is able to perform tasks which include extended optional functionality such as "view attendance" could extend to "edit attendance". On the other hand, goals such as "edit personal details" will include "save changes" in order to be successfully performed. Furthermore, there are some cases in which the user goals will involve a secondary actor, the database to retrieve data. Therefore it is crucial to depict this association to showcase the systems behaviour to user tasks. In addition, tasks that will be performed could have an indirect association with secondary users. Figure 1.2 showcases indirect association to the user when the task "contact student" is performed. As mention, module leader, course director and head of department are subsumed of lecturer. They have the same accessibility rights as a lecturer with extended functionality due to the nature of their role, such as a course director is able to "view course attendance". Use case diagrams for the remaining actors can be found in *appendix C.*

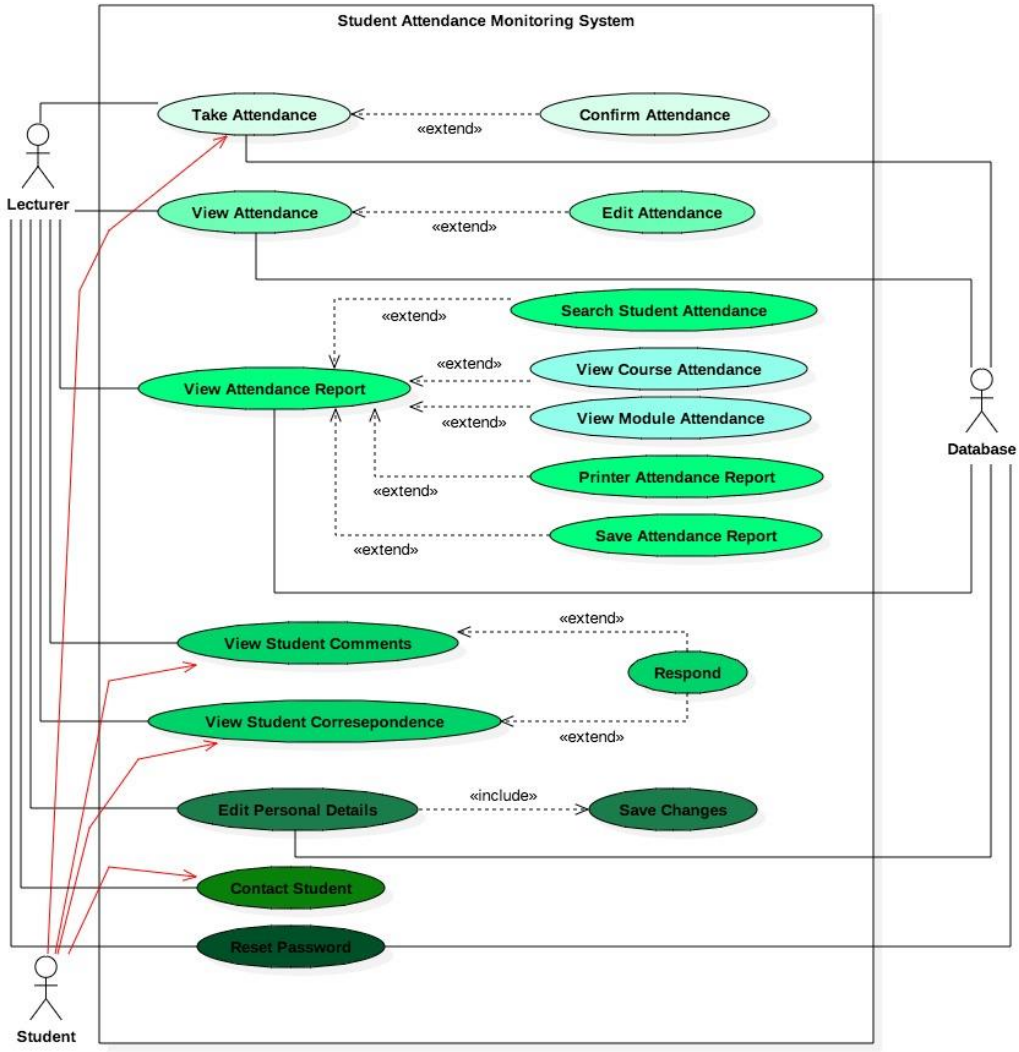


Figure 1.2: Lecturer use case diagram

2.3.2 Use case description

Use case descriptions are utilised in conjunction with the use case diagrams, in order explore each actors goal and the steps undertaken to achieve it. The use descriptions showcase the preconditions, successful outcome and failed end condition to prepare desired solution to each scenario. In addition, use case descriptions depict alternative flows which enable the developer to capture premature problem areas which could surface, enabling a smooth user journey. In order to declare important each use case has initially been prioritised through MoSCoW methodologies, with contributing factors to influence it such as frequency. MoSCoW prioritising will sought in further detail, later in the report. Table 1.3, demonstrates a typical use case description for the requirement "Create User". Further Use case textual descriptions are found in *appendix D*.

UC01	Create User (Student)	
Actor	**Administrator**	
Goal in context	To create a new student	
Preconditions	Administrator should be logged into the system.	
Success End condition	New user (student) is created.	
Failed End condition	New user (student) was unable to be created.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button "Manage Student".
	2	Administrator clicks on button "View Student".
	3	Administrator clicks on button "Add New Student".
	4	Administrator enters student's id.
	5	Administrator enters student's first name.
	6	Administrator enters student's surname.
	7	Administrator enters student's gender.
	8	Administrator enters student's date of birth.
	9	Administrator enters students contact number.
	10	Administrator enters email address.
	11	Administrator enters password.
	11	Administrator checks all student details are entered correctly.
	12	Administrator submits request to insert into database.
Alternative Flow	Step	Action
	N/A	N/A
Moscow		Must
Frequency		Every time a new student comes to the institution (start of term).

Table 1.3: Administrator use case descript for "Create User"

Although use case descriptions are extremely beneficial in line with use case diagrams, they withdraw several drawbacks. One of which is the time taken to successfully compose the different descriptions, this can impact the outcome of the project it is a large project with a small team. In addition, utilising a DSDM/Agile methodology for this project means that requirements and activities can be constantly changing. Therefore the use case descriptions can become useful and invalid. Nevertheless, as they form a basis for key tasks these are unlikely to drastically change to make an impact on the project.

2.4 System Requirements

The gathered knowledge from the tools and methodologies utilised has enabled the generation of a list of system requirements. The system requirements details of how the software or hardware operate. Deriving these requirements are critical to ensure the clients expectations and desires of the proposed system has been met. Undertaking this process has enabled ease in categorising these requirements into two parts:

- **Functional requirements**: Such requirements depict the functional behaviour that the system and its components should perform also. This is generally displayed in a Verb-Noun format, these requirements define the systems functionality and what it should do.
- **Non-Functional requirements**: These requirements describe the attributes of the system to judge the operation and the quality of the system. Therefore, generally there are given constraints on functionality such as performance, reliability, usability, etc.

The services that the system should offer and how it will behave will depend on the user inputs and system outputs, these in turn define the requirements. The functional requirements for the SAMS application are as followed**:**

Functional requirements

Administrator

- Create User
- Search User
- View User
- Edit User
- Delete User
- Create Module
- Search Module
- View Module
- Edit Module
- Delete Module
- Create Course
- Search Course
- View Course
- Edit Course
- Delete Course
- Create Department
- Search Department
- View Department
- Edit Department
- Reset Password
- Enrol Student
- Enrol Lecturer

Lecturer

- Take Attendance
- Confirm Attendance
- View Attendance
- Edit Attendance
- View Attendance Report
- Search Student Attendance Report
- Print Attendance Report
- Save Attendance Report
- View Course Attendance
- View Module Attendance
- Add Session
- Edit Session
- Delete Session
- View Student Comments
- View Student Correspondence
- Contact Student

Student

- Edit Personal Detail
- View own attendance
- Upload Correspondence
- View Correspondence
- Edit Correspondence
- Delete Correspondence
- Contact Lecturer
- Edit Personal Details.

Non-functional requirements

Security

- User passwords are hashed when stored in the database.
- Different user types will have different account privileges to restricted unauthorised access.
- The system should be able to authenticate the user when logging in.
- The system shall ensure the user is logged out when closing down the browser.

Performance

- The database should be able to accommodate over 1,000 records without system faults.
- The system should be able to run smoothly with over 500 users using it concurrently.

Compatibility

- The system must operate on all modern browser versions of the following: Google Chrome, FireFox, Opera and Safari.
- Responsive design adapting to different screen resolutions.

Design

- The GUI design must be clear and consistent through all pages.
- The colour scheme must be suitable for different lighting conditions.

Usability

- The system shall give guidance to users when completing system tasks.
- The system should be user friendly for expert and novice users.

2.5 MoSCoW

MoSCoW prioritisation is part of DSDM framework to assist in prioritising the most important requirements to enable efficiency in delivery. As the project is limited to 300 working hours, this tool will allow the developer to focus on implementing the most essential requirements. Prioritisation of these requirements allows an escalation in the value of work by decreasing typical activities that would take far development time. Table 1.4, illustrates a subsection of the functional requirements of the SAMS application prioritised due to their influence on the systems success. Full MoSCoW analysis is found in *appendix E*.

#	Functional Requirement	Priority	Justification
		Administrator	
1	Create User	Must Have	An administrator must be able to create users in order for the system to be fully functional. These users will test the system and use it on a daily basis.
2	Search User	Should Have	An administrator should be able to search users via their details e.g. ID, NAME, COURSE, etc. There will be over thousands of user records from lecturers to students, therefore without a search function the usability of the system becomes very difficult. However, there will be no detrimental effect to the system if this was not implemented.
3	View User	Must Have	An administrator must be able to view all the user records to clarify details of what courses/modules they are enrolled on, personal details and to check if there is a need to modify or update records.
4	Edit User	Must Have	An administrator must be able to edit user details in the case of incorrect records, they are able to efficiently modify details without having to create a new user.

Table 1.4: Fragment of MoSCow prioritisation of functional requirements

2.6 Conclusion

The requirements phase is one of the most crucial subsections of the project lifecycle. The processes and methodologies utilised in this chapter has clearly defined the requirements in order to develop the system to meet the client's needs. These will format as the projects foundations, in which the research, design and implementation processes can form on top of. However it is critical to note, due to the nature of the agile methodology utilised for his project, these requirements are subject to change. As the project begins to form, this can be problematic and stunt the progression of the project. Nevertheless, the core requirements have been agreed upon with the client and this is where the project will begin. If there is room for change, the developer is able to prioritise working functionality to meet the change. As the system requirements withhold a solid foundation for the project, further research can now be conducted around d these area to ensure an efficient end-product

3 Literature Review

Organisations globally, irrespective of size and years of establishment, require significant protection of their business assets. It is well considered that data is one of the most valuable assets in today's society. With the readiness of data processing now handled around the clock, sufficient protection can minimise business expenditure and failure. Legislations and regulations are instilled to assure the continuity and compliance of data protection, defining what information must be retained, for how long and under what conditions (Petrocelli, 2005).When protecting personal data in industries such as education, it is vital to discuss and evaluate what provisions to implement. This is particularly pertinent as the data subjects, individuals whom data is about, are young pupils. It may be argued that they have "relatively less power to resist or reject the surveillance they are subjected to on a routine basis" (Taylor, 2011). In such cases the Data Protection Act (DPA) 1998 provisions must be fully understood and made clear as to the extent to which society can adhere to them.

Devised from the basis of the EU Data Protection Directive 95/46/EC (DPD), the DPA 1998 explores the foundations of processing and managing personal data. Data must be processed fairly and lawfully, one of the eight DPA principles which must be complied with by the Data Controller. These are individuals whom determine the purpose for which personal data is or is to be processed. Most organisations will process personal data relating to employees, customers and suppliers, the act refers to these organisations as Data Controllers. In addition, a Data Processor is "any person (other than an employee of the data controllers) who processes data on behalf of the data controllers" such as storage of data on a third party sever (ico.org.uk, n.d.). The act clarifies the distinction between two in order to recognise that not all institutions processing personal data share the same degree of responsibly. Exemptions from the act which are applied on a case by case basis. From the 25th May 2018, the new General Data Protection Regulation (GDPR) will become law. Although the new legislation holds great similarity to the current DPA, this review will explore the new condition and the impact it will have on society.

The Computer Misuse Act 1990 (CPA) could be arguable an extension of the DPA 1998. The legislation was designed to protect users from the deliberate attack and theft of information on computers, after the events in R v Gold & Schifreen (1988). The act explores offences such as hacking, purposeful spreading of malicious software, etc. This literature review will explore the hidden depth of both laws and their governing principles in protecting society. It will reflect upon the introduction of such laws, the components involved and how it may affect the project and future for society. Furthermore, this knowledge will enforce foundations and principles inclined for this project. The Student Attendance Monitoring System (SAMS) will need to consider the vast amount of sensitive data which will be processed through implementation. The developer will be able to create a sustainable and secure system with the knowledge of protection regulations and security measures without hindering usability.

3.1 Data Protection Act 1998

The UK's first DPA became law on 12 July, 1984. This act incorporated the regulations for handling personal data and the rights to access and process it. However, these regulations were revised to bring the UK in line with other European Union States. This triggered the birth of the DPA 1998. The 1998 statute is significantly larger than its 1984 precursor, with seventy-five sections and sixteen schedules. The extended provisions include new rules enforcing restrictions when passing data to non-participating states (Antell, et al., 2007). As in the 1984 act, the DPA 1998 aims to prevent the misuse of the processing and holding of personal data relating to 'a living identifiable individual' (Leigh-Pollitt & Mullock, 2001). The new legislation grants data subjects more rights in the processing of their personal data. The 1998 act imposes new obligations upon the data controllers to comply with the enhanced eight data protection principles.

3.1.1 Eight data protection principles

In an effort to create a comprehensive and unified protection system, the eight data protection principles outlined in DPA 1998 were significantly influences by the principles within the Directive 95/46/EC. Data controllers and processors must comply by eight data protection principles. In detail, these are that (ico.org.uk, n.d.):

1. Personal data must be processed fairly and lawfully, and shall not be processed unless:
    a. at least one of the conditions in Schedule 2 is met, and
    b. In the circumstances of sensitive personal data, at least one of the conditions in schedule 3 is also met.

These will be discussed in further detail within the following segments of this review. The remaining conditions outline as such:

2. Personal data is obtained for specified and lawful purposes and shall not be further processed in any manner incompatible with those purposes.
3. Personal data shall be an adequate amount to fulfil the purpose and is not excessive.
4. Personal data is accurately stored, and where necessary kept up to date.
5. Personal data is not kept for longer than its necessary purpose.
6. Data will be process in accordance with the rights of the data subjects under this act.
7. Appropriate measures shall be taken against the unauthorised and unlawful processing of data and against accidental loss or destruction to persona data.
8. Personal data should not be transferred or processed, without an ensured high standard of protection for the data subjects, to a country or territory outside the European economic.

The revised requirements from the old 1984 act enforce new developed conditions in the first principle. Data controllers must meet at least one of the conditions outlined in Schedule 2 when processing personal data, and in the case of sensitive data a condition involved in Scheduled 3 also.

Schedule 2: Processing of any personal data

Schedule 2 outlines that processing must comply with legal obligations to which the data controller is subject to, other than an obligation imposed by contract. This processing is necessary to protect the interests of the data subject. Furthermore, data processing is necessary for certain public figures including the Administrator of Justice, the Crown, a Minister of the Crown or Government department (legislation.gov.uk, 2004).Legally, personal data is defined as 'any information relating to an identified or identifiable natural person (data subject)'. However, the DPA is much more complicated in comparison to these simple definitions. Personal data can be extended to include, 'any expression of opinion about the individual and any indication of the intentions of the data controller or any other person in respect of the individual'. These meanings originally coincided within the 1984 legislation, however widely criticised as the statement was seen as personal to the data controller rather than the data subject (Lloyd, 2011). The then Data Protection Registrar expressed his concerns that the provisions of the 1998 act should perhaps been seen to remove the unsatisfactory distinction. Nevertheless, although the legislation is vague when differentiating between data subjects and data controllers, it can still be successfully applied to businesses, organisations and the governments nationally.

Schedule 3: Processing of any sensitive personal data

Schedule 3 examines the conditions when processing sensitive data. Sensitive data includes ethnic background, religious beliefs, criminal records and physical or mental health of the data subject. This data requires a higher degree of protection.

In the 1984 legislation, the attempts to identify sensitive data was critiqued harshly by government ministers who compared the effort to "the quest to find a unicorn", both seen as 'mythical creatures' (Lloyd, 2011). What is now known as sensitive data, was only then considered sensitive in extreme conditions such as if in the hands of a terrorist organisation. The context in which personal data was held or used was considered more important than the data itself. The new 1998 legislation adopts additional categories and examples for data which requires a higher degree of security, classified as 'sensitive data'. As a result, Schedule 3 enforces the compliance of at least one of its conditions for the processing of sensitive data to be legal (Fell, 2007).

3.2.1 Exemptions from the act

The DPA is provisioned to apply generically to all individuals, businesses and organisations. However, there are some exemption from the act to accommodate for special circumstances. As advised by the Information Commissioners Office (ICO), each exemption must be considered on a case-by-case basis. Therefore, depending on the circumstances the act declares that an individual will be exempt from the following requirements:

- To register with the ICO (to "notify"); and/or
- To grant subject access to personal data; and or
- To give privacy notices; and/or
- Not to disclose persona data to third parties (ico.org.uk, n.d.)

Examples for exemptions from the DPA 1998, can be found in *Appendix F*.

3.2.2 Future development of DPA

The evolution of the DPA has enhanced significantly, an ongoing process to keep up with societal changes. In recent development, the European Union bring forth a new regulation to ensure a higher degree of protection for individuals and further conditions upon those who process individual's data. May 25th 2018 is the passing of the new regulation for data protection; the General Data Protection Regulation (GDPR). Similarly, the GDPR is underpinned by many data protection principles which drive compliance. Although there are many similarities between the GDPR and the DPA 1998, certain principles have been developed. The principles relating to the processing of personal data are enlisted in Article Five of the GDPR, can be found in *Appendix G*.

Data Protection Act 1998 vs General Data Protection Regulation

In addition to the enhanced principles, there are also many other areas the GDPR provides additionally benefits. The GDPR, unlike the DPA, will bring forth full alliance with the European Union, applying the outlined provisions to all those within the union. Furthermore, global organisations which holds data on EU citizens will also need to comply with the new legislation. In addition, ach European country will be appointed a Supervisor Authority to assurance compliance. Clearly established in its context, GDPR significantly applies to organisations which hold individual data. Organisations with employees more than 250 will need a delegated a Data Protection Officer to assurance compliance.

Data subjects are given a greater control over their processed data. Data access requests are now free of charge and available within 30 days. Furthermore, one of the most important aspect of the new legislations is the data subject's 'right to be forgotten'. Bringing forth the new legislation, individuals will have the right to erasures of their data including web records. The new GDPR regulation extends sensitive data provisions to include online identifiers, location data and genetic data to be protected. With the growth of technology appealing to younger ages, a child protection clause requires the parental consent of processing data for minors. Any breech of notification is now mandatory within 72 hours. Many of the GDPR's concepts and principles are similar to the current DPA. However, both are not enough to form an overall shield of protection on society. Although, there are new elements and significant enhancements, there are other legislations which play as an

extension paying close attention to data on a computer. An example of which, the Computer Misuse Act (CMA) 1990.

3.3 Computer Misuse Act 1990

The Computer Misuse Act (CMA) 1990 came into force on August 29, 1990. The act was influenced by the case of R v Gold & Schifreen (1988). The two 'hackers' gained unauthorized access to the mailing service of British Telecoms(BT) sending messages announcing the award of honors 'from' Prince Philip. They were initialing charged and convicted with forgery of BT's engineering password. However, both were freed on the grounds that unlike material goods, information cannot be stolen or forged (Anderson , 1994). This brought chaos to media, proclaiming to be "hackers charter" and began the uprising to the CMA 1990.

A hacker is an individual who illegally gains access to information in a computer system. There are different types of hackers with various intentions, from those who look to hack 'for fun' to those with the malicious intent to steal, exploit or sell data. When CMA was first passed, it was questioned why the unauthorised access to confidential information on a computer was considered an offence when the same information on a card index was not (Macewan, 2008). The ongoing debate on whether the law should directly apply to the theft of information continues to run today. Nevertheless, CMA shelters offenses to access and even attempt to access a computer system without the proper authorisation. In further detail, the act surrounds the aspects of the invasion of confidentiality. It aims to "make provision for securing computer material against unauthorized access or modification; and for connected purposes" (legislation.gov.uk, 2004). However, MacEwan (2008) argues the CMA is just "a compromise between radical thought and minimal action". The CMA inhabits several offenses in which compromise the act.

3.3.1 Computer Misuse Offences and Penalties

The CMA (1990) recognises and prosecutes by the following offences:

1. **Unauthorised access to computer material.**
   This offense was originally a summary offense but is now triable (Macewan, 2008). Those found guilty have the intent to secure unauthorised access to any program or data held in any computer. Guessing or using an individuals' password to log into their access area, even without intent to cause harm, is seen as illegal. This offense carries the risk of imprisonment not exceeding two years and/or a fine.
   a. **Unauthorised access with intent to commit or facilitate commission of further offences.**
      A perpetrator guilty if they have committed offence 1 with the intent to commit or facilitate further offences. The sole difference between the second and first offence is the intention and purpose of doing something illegal. This can lead to up to five years in prison and/or a fine.
2. **Unauthorised acts with intent to impair, or with recklessness as to impairing, operation of computer,etc.**
   This is the replacement offense of: "unauthorised modification of computer material", as notion was deemed too broad and triggered claims about loopholes (Macewan, 2008). The perpetrator is found guilty if they intentionally commit any unauthorised act on a computer by impairing or hindering computer operations or data. This includes the unleashing viruses. A computer virus is a type of malicious software designed to spread from host to host with the possibility of replicating itself, causing nuisance within the computer program and to its users. If found guilty, the defender can be convicted with a term of a maximum ten-year prison sentence and/or a large fine.
3. 3a. **Making, supplying or obtaining articles which can be used in computer misuse offence** (legislation.gov.uk, 2004).

This is an extension from offense 3, known as 3A. Offence 3A was introduced to meet to the UK's commitments under the Cyber Crime Convention; with the aim to criminalise 'hacking tools' (Murray, 2013). An individual is guilty of this offense if they create, supply or obtains any article with the intent to be used in committing a crime under this act, this includes the writing or creating of computer viruses. Should such viruses or methods of committing CMA offenses be supplied or obtained with the intent to commit computer system damage, the individual can endure a maximum ten-year prison sentence and/or a large fine.

3.3.2 Issus with CMA 1990

There are been much speculation about loopholes within the CMA 1990. Debates have arisen as to whether the CMA thoroughly protects victims and the cases of computer attacking crimes. Many consider Denial of Service (DOS) attacks to be a loop hole in the CMA 1990. DOS attacks occur when an attacker prevents legitimate users from accessing computer systems. This can be done through loss of network connectivity by overloading computer resources, consuming user bandwidth. An attack in which several compromised systems target another system such as a server, imposes an even bigger threat (Distributed Denial of Service). Such occurrences have become extremely regular to extort money or expose data. However, it is argued that the controversial opinions to these circumstances are due to each case being unique in the eyes of the law (Macewan, 2008) . Furthermore, recommended by the Law Commissioner the act from reframes from containing definitions such as "program" or "data", as it permits the courts to determine whether the particular facts fall within the offenses of the act (All Party Internet Group, 2004). Nonetheless, such pressure has resulted in further government legislation on the issues of DOS attacks.

CMA offences are sustainably underreported. It has been suggested that cooperate victims often wish to avoid the adverse publicity, fearing an excessive loss of capital overnight. However, such revelations could permit vulnerability for repeated attacks. On the other hand, individuals fail to report attacks because of the perception that the police are powerless in cybercrimes (All Party Internet Group, 2004). Indeed, it was highlighted that such crimes are not one of the target measures by which police performance is assessed. A study conducted by the University of California in 1995, found that 40% of police professionals receive no formal training on computers, an additional 20% of professionals receive no more than 2 hours of computer instruction (Goodman, 1997). However, seen that in society we are now in the fourth industrial revolution, technology is embedded in its everyday course. There has been major advancement from this study inclining that officers are capable of knowing enough to impact the likelihood of cybercrime.

Lastly is the issue with proving guilt for the act. In many cases individuals use ignorance as a defence and the sentencing of these crimes are much more lenient than the maximum available. In England and Wales, from 1990 to 2006, of the 214 defenders prosecuted under the CMA, 75% were found guilty (Macewan, 2008). Although this figure is considered reasonably high, the lack in enforcement is certainly not misplaced. By the time the offender is caught the damage to the system has already be committed and in many cases irreversible if not on the data, then on the reputation of many corporate companies.

3.3 Conclusion

The DPA 1998 has been instilled in effort to protect personal data. The act includes eight protection principles which ensure the processing of persona data is complete in the best interest of the data subject. If found guilty of the act, there are consequences of imprisonment and penalties. Exemptions from the act are tried on a case-by-case basis. The evolution of the DPA has introduced a new legislation which becomes law on the 25th May 2018 known as the General Data Protection Regulations (GDPR). GDPR enforces a higher degree of protection and a greater focus on data subject's rights over their personal. The CMA 1990 harmoniously work alongside the DPA. An

individual who illegally gains access to and in cases tampers with information in a computer system is committing an offense of the CMA 1990. The act includes four offences in which individuals are convicted by. Perpetrators can be stopped with different forms of security. Physical security can include locks and guards or software security such as implementing access rights, hashing passwords, system logs to see account activity, encryption and firewalls. Overall, this legislation was instilled to deter criminals from hindering or infiltrating data stored in a computer.

Through the knowledge acquired from this literature review, the successful implementation for the Student Attendance Monitoring System (SAMS) is crucial. Protection flaws and security precautions must be considered. Preventing perpetrators from accessing the sensitive data held by SAMS can be carried out in many ways, such as implementing encrypted password and taking additional steps to prevent attack such as SQL injection. However, with the constant reviewing and clarifying of new protection principles, the system will have to consider regular updates in order to keep up with the new protection laws and prevent authorised interception.

4 Design

Having conducted in-depth research of the surrounding aspects of SAMS web application, this chapter will showcase the design of the system. The chapter aims to expand upon the ideas and principles captured in previous chapters and introduce the design components of the finished application, catering to all user types. This phase will comprise of various tools and techniques as it begins to construct the user interface. The chapter will investigate four sub-sections to designing a user-friendly application: data modelling, process modelling, user Interface and lastly, the conclusion.

Data modelling consist of exploring different UML diagrams to produce an efficient database for this project. Through a set of processes, this segment will examine the creation of the database with regard to its elements, structure and format. A class diagram will illustrate the main classes required and their basic dependency. To expand, the collated information is enhanced in the creation of an entity-relationship diagram, where all conflicting associations are resolved. A data dictionary with further in-depth content on the individual table structures is composed before implementation. This information is used to understand the processing of the data, through dynamic behavioural diagrams called sequence diagrams. This segment explores the necessary components required to understand the flow of a user task. Topics in usability are explored to assist in comprising a user-friendly system firstly illustrated through wireframes. Usability is extremely important for product survival; therefore, wireframes are created to understand a workflow for each user type. Lastly, navigation diagrams/site maps are utilised to display coding content layout at a single glance for any stakeholder to examine. By using these design technologies and resources analysing and evaluating the system design will effectively ensure efficient development and implementation. A conclusion will draw together the components of the design chapter with recommendations and procedures that will assist the next course of events.

4.1 Data Modelling

Data modelling includes the documenting, defining and structuring of a database. This subsection of the report will explore various UML diagrams to construct and evaluate an appropriate database design for the SAMS web application. During this design process, the functional requirements of the system will be considered and how they can be properly executed through the database. This section will include a class diagram to understand the underlining blue prints of the database design. This UML diagram will be expanded upon to include resolving conflicts and attributes within an entity-relationship diagram (ERD) and lastly with the given constraints and data types in a data dictionary. The information gathered in this subsection will enable the developer to ease the process in implementation.

4.1.1 Conceptual Class diagram

A class diagram is a UML static structure which illustrates class dependencies and their basic attributes. Such diagrams assist in the conversion of the user requirements into a data model. Each individual class represents a database entity, initially derived from project scope meetings and interviews with the client. The major components gathered from the client's requirements have been captured in Figure 4.1. The diagram clearly illustrates the main users of the system, categorised into their user type; Administrator, Student or Lecturer. Module Leader, Course Director and Head of Department are all types of User also. These User types will have particular privileges in the system as well as inheriting the privileges of those below their hierarchy. For example, a Course Director will perform certain tasks, as well as the tasks of the Module Leader and Lecturer. However, a Course Director will not have the access rights to perform the tasks of a Head of Department. These subclasses all inherit attributes from the superclass, User, this is known as generalisation.

Generalisation is the process of minimising differences between entities by identifying their common characteristics. Generalisation classifier is an unfilled arrow head. Furthermore, Figure 4.1 illustrates how each class will associates with one another.

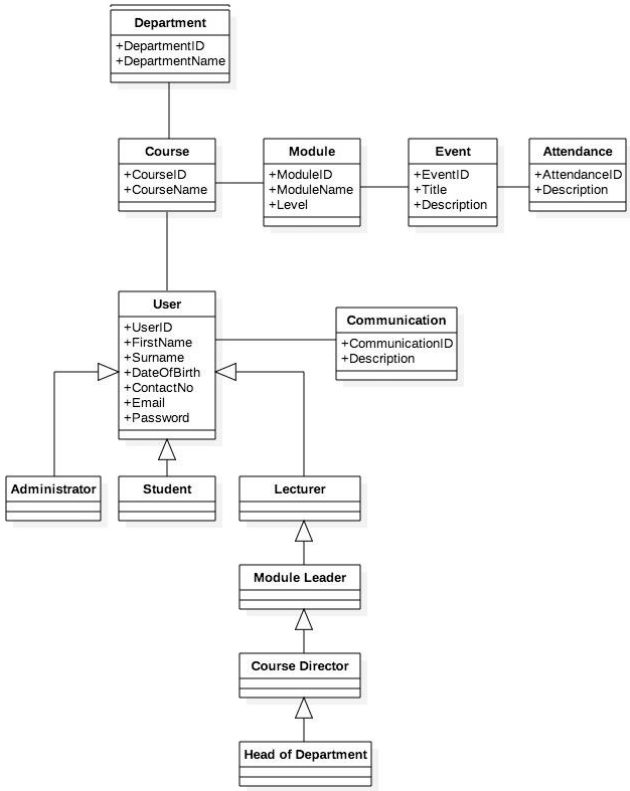


Figure 4.1: Class diagram of SAMS

In terms of relational data however, class diagrams cannot resolve any many-to-many relationships. The class diagram will act as a blueprint of the ERD, outlining the main entities of the database and capturing a deeper understanding of the system's needs. The collated information will be utilised to create an efficient database design.

4.1.2 Entity relationship diagram

An entity-relationship diagram (ERD) is a useful UML construct illustrating the data structure and architecture of the desired database. ERDs capture the relationship between entities and their relevant attributes, as seen briefly in the class diagram. Ultimately, thoroughly designing the ERD means resolving any conflicting associations. To diminish the multiplicities between two classes an intersecting class must be introduced. Intersecting classes contain the unique values of both original classes, and additional attributes if necessary. Each table must have a primary key (PK), a distinguished record which will uniquely identify the table. This attribute can also aid in establishing association with other tables by acting as a foreign key (FK). The associations (lines between the classes) demonstrates the relationships among classes and the cardinality relationship (e.g. One-to-One, One-to-Many etc.). By limiting each table to a single purpose, it reduces the occurrences of duplicated data and eliminates anomalies from database modification i.e. delete, update and insertion. This is known as normalisation. Thus, before the final ERD is created, three normalisation steps must be carried out to ensure the system database is in 3rd normal form. Irrelevant attributes and entities were withdrawn to minimise the impact on database performance. Through the early stages of the ERD development, certain relationships were reworked to optimise the performance,

functionality and processing of the system. *Appendix H* showcases the initial ERD, displaying the improvements and adaptations in the final design. Figure 4.2 demonstrates the ERD for SAMS web application. This has been partitioned into manageable segments to gain a deeper understanding of the database and the intended functionality.

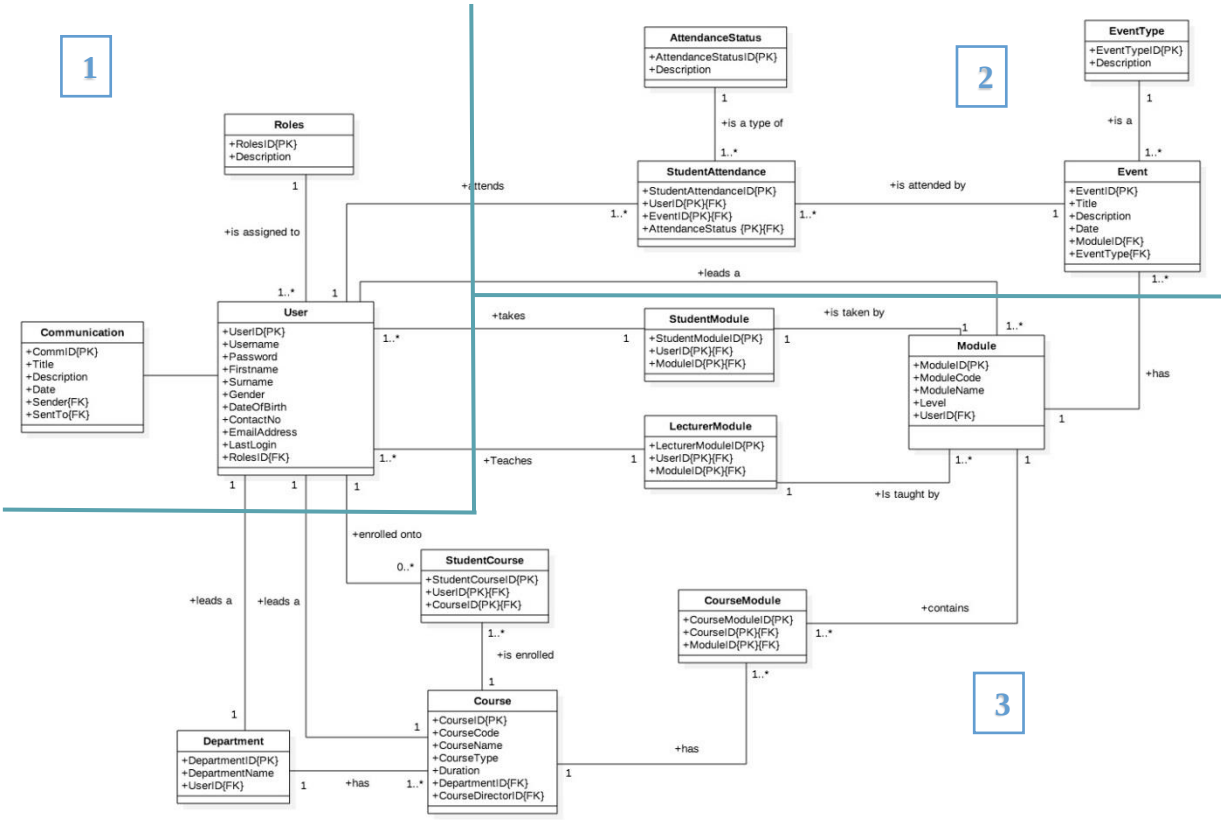


Figure 4.2: Entity Relationship Diagram

Figure 4.3 mirrors the real-world association within the University, of Students and Lecturers to their according courses and modules. These associations will store the modules taught by Lecturers and the attendance data collected. There are many cases of resolved multiplicities in this subsection of the database, an example being the relationship between Lecturers and their Modules. A Lecturer (User) will teach one-to-many Modules, and therefore this occurrence of multiplicity is resolved by an intersecting class, LecturerModule. A request by the client involved a centralised system which allows the University to share and view data across modules and faculty. As a result, direct relationship is created between a User and the associative entities. For example, UserID is enlisted as a foreign key in Course depicting a one- to-many association. Likewise, found in the occurrence of Department and Module. Full breakdown of the ERD for SAMS can be found in *Appendix I*. However, it is crucial to be aware due to the nature of agile methodology, there may be minor changes to this initial design.

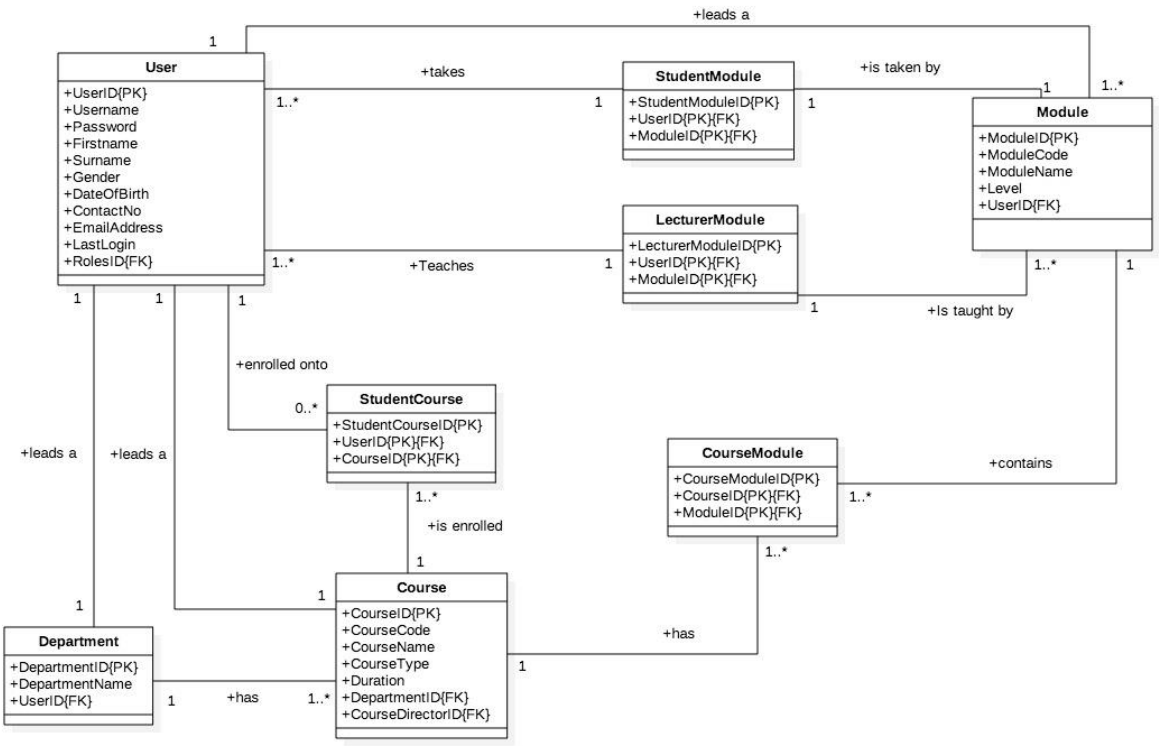


Figure 4.3: ERD - Users, Modules, Courses and Departments

4.1.3 Assumptions

In addition to those revealed above, the following assumptions were generated during the creation of the ERD:

- All members will be classed as active.
- Each Module is taught by one Lecturer; however, a Lecturer may teach one-to-many Modules.
- Each User has one Role, but a Role can belong to many Users. It was decided that Module Leader, Course Director and Head of Department will not get their own section for the system by are subsumed into Lecturer with additional privileges.
- User will send one-to-many Communication and Communication is sent by a User (at a given time).
- A Department/Course/Module can be led by one User.

These assumptions are outlined to ensure system functionality agreed with the client is successful. The client has enriched the developer with sufficient detail about the desired system, resulting in a well thought out database design for the SAMS web application. The client's requests and requirements have been carefully considered in the production of the desired database. To ensure validity of the database a list of SQL queries was generated based upon the requirements, found in *Appendix J.* However, before the implementing the database, the constraints and data types must be considered to ensure data integrity.

4.1.4 Data Dictionary

Data dictionary is an outline of the format and structure of each table of the desired database. A data dictionary ensures the correct data objects, attributes, referential relationships and potential constraints are in place. Each individual column of the data dictionary outlines the following:

- Relation: the table name.
- Attributes: the attributes employed by the table.

- Data Type (Size): The two common data types are INT, for numeric entries, and VARCHAR, for character entries. The brackets declare the maximum characters which can be inputted into the attribute field. This is to ensure inserted data is precise and valid. Data types such as DATE and DATETIME have also been utilised.
- PK/FK: The PK and, if applicable, FK will be displayed.
- Not Null: This column displays the requirements of a Not Null constraint (when data is inserted, the attribute cannot be left empty).
- Other constraints.
- Referential Relation: An additional column has been added to showcase the referential parent table of the foreign key if applicable.

As shown in table 4.1, the password attribute requires 255 characters. Each user password will be stored hashed and therefore requires a vast amount of storage space. Furthermore, Contactno is stored as a VARCHAR2 rather than INT as records entered with '0' will not be recognised by the database and consequently stored inaccurately. For example, when stored as an INT datatype, a number entered as "0765" would be stored '765', however VARCHAR2 resolves this matter. LastLogin, was an additional feature to monitor the systems usage. This utilises the datatype DATETIME with the constraint of CURRENTTIMESTAMP, therefore is updated every time a user is logged in. In addition, other constraints include AUTO_INCREMENT and DEFAULT, found in table 4.2. AUTO_INCREMENT has been applied to each PK in the database. It will automatically generate sequential numeric values each time a new record is inserted into the table. Furthermore, the DEFAULT constraint provides a default value for column, if not specified. Data entry is done via the front end of the system through a form. The full data dictionary can be found in *Appendix K.*

Relation	Attribute	Data Type(Size)	PK/FK	Not Null	Other Constraints	Referential Relation
User	UserID	INT(8)	PK	✓	AUTO_INCREMENT	
	Username	VARCHAR(10)		✓		
	Password	VARCHAR(100)		✓		
	Firstname	VARCHAR(20)		✓		
	Surname	VARCHAR(20)		✓		
	Gender	VARCHAR(8)		✓		
	DateOfBirth	DATE		✓		
	ContactNo	VARCHAR(11)		✓		
	EmailAddress	VARCHAR(50)		✓		
	LastLogin	DATETIME		✓	CURRENTTIMESTAMP	
	RolesID	INT(11)	FK	✓		Roles

Table 4.1: Data dictionary: User table

Relation	Attribute	Data Type(Size)	PK/FK	Not Null	Other Constraints	Referential Relation
StudentAttendance	StudentAttendanceID	INT(11)	PK	✓	AUTO_INCREMENT	
	UserID	INT(10)	FK	✓		User
	EventID	INT(10)	FK	✓		Event
	AttendanceStatusID	INT(11)	FK	✓	DEFAULT "2"	AttendanceStatus

Table 4.2: Data Dictionary: StudentAttendance table

A data dictionary is a useful template describing the format and structure of the database and the relationship between its entities. An established data dictionary can provide a significant advantage by improving data quality and reducing data redundancy. Overall, promotes better data management.

4.1.5 Conclusion

In conclusion, the methodologies and techniques undertaken in the data modelling subsection has enabled the developer to gain an enriched design of the database. The class diagram acted as the foundations of the ERD, outlining the main entities for the database, capturing a deeper understanding of the system's needs. This basis was enhanced through the ERD and data dictionary to understand the individual table's structure including attributes, constraints and primary/foreign keys. In addition, specialised input fields and validation constraints are applied on each input field within the application reduce the workload on the server. The next stages of the report forecast the interaction between the user and the system to in process modelling such as sequence diagrams.

4.2 Process Modelling

4.2.1 Activity diagrams

Relying on the use of use case diagrams alone will not suffice when understanding and defining the goals required of the system. Use case diagrams do not consider how the implemented system is proposed to function as such. Therefore, activity diagrams can be utilised to inhabit the intended behaviour of the system through a flow of activities for a single use case. Such diagrams are seen to "capture the underlying static structure of a software system at various levels" (Russell, et al., 2006). Therefore, activity diagrams have been utilised in this project to easily correspond the system behaviours to the client and stakeholders. The conventions outlined to interpret activity diagram are the following:

- ✓ The **activity state** is represented by a rounded rectangle which displays the activity taking place. For ease of understanding this is written in verb/noun format.
- ✓ The **control flow** is represented by an arrow transitioning from one activity state to another.
- ✓ **Nodes** are represented by a black circles to signify the beginning and end of a series of activities. The end node has an additional circle around it.
- ✓ **Decision and branching** are represented by a diamond shape which considers alternative paths. These are often conditional scenarios, where two diamonds are showcased to demonstrate the different options and how they branch back to a final option. They can also be utilised to represent for, while and do while loops.
- ✓ Synchronisation is utilised when a single flow becomes a multiple concurrent flow. Also known as **forks/joins**, they are represented by two thicker lines in parallel. This is demonstrating concurrent behaviour.
- ✓ **Swim lanes** will demonstrate the notion of applying various tasks to an object e.g. user.

To properly comprehend activity diagrams they are generally read from a starting node which will follow the sequence of events until the end node. Figure 4.4 illustrates the activity of taking attendance. The diagram runs through a series of activities between the user and GUI to land onto the page where student attendance list is shown. Once the attendance is submitted, a conditional scenario is presented to check if there are any database violations. If so the user will be redirected display an error message, on the other hand if the submission is successful a success message is displayed. This is the end of the series of activities demonstrated by the end node.

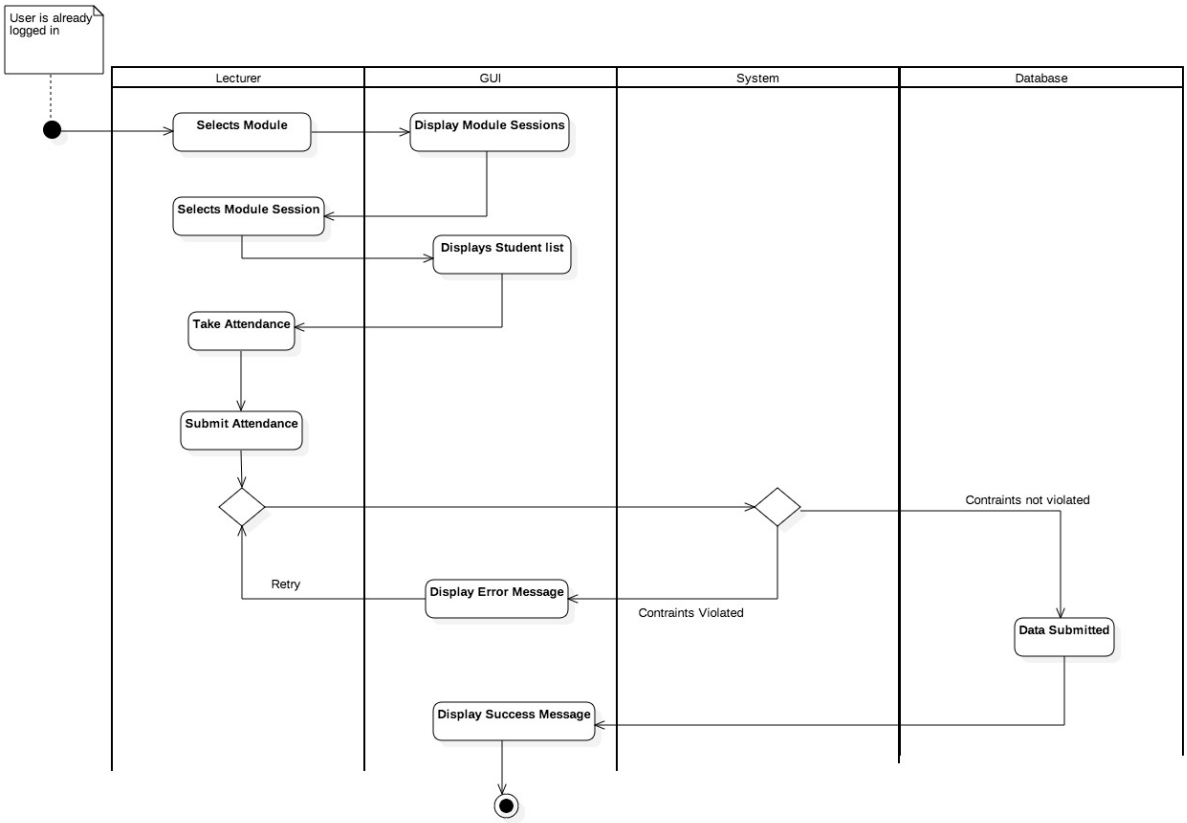


Figure 4.4: Activity Diagram of functional requirement "Take Attendance"

Although an excellent collaboration tool drawing the knowledge of the system and requirements to showcase to all stakeholders, allowing the discussion of design decision in concrete term. Many would argue that activity diagrams ignore the dynamic behaviour of the system. Therefore, sequence diagrams have also been utilised to explore beyond the static structure of the system. Nevertheless, activity diagrams are an excellent tool to embark upon the clients and stakeholder before further design decisions can be finalised. The remaining activity diagrams can be presented in *Appendix L*.

4.2.2 Sequence Diagram

Sequence diagrams, also known as event diagrams, convey the interactions among actors and objects in the exchange of messages over a time-order. Such behavioural diagram give great visualisation and validation of the different runtime scenario's a user will experience with the system. Developers are able to explore alternative flows, elicited from the user requirements. The following conventions are applied when utilising sequence diagrams, helping all stakeholders understand the flow of events to achieve a user goal:

- ✓ Rectangles on the top layer of the sequence diagram represents the objects involved in the process. The vertical dashed lines from the object indicate the objects presence over time (lifeline).
- ✓ The arrows indicate the messages across the different objects, depending on their type of message they are differentiated with distinctive arrowheads.
    - a) Synchronous (Call)/Return - A caller of a synchronous message must wait for a return in order to regain control. The return message is optional, however useful to establish a clear control flow. The timeline is shown activated when the rectangle between the two arrows appears.

b) Asynchronous (Send) – A caller of an asynchronous message does not have to wait for a response in order to perform other tasks. It normally corresponds to signal exchanged amongst systems, asynchronous message is normally paired with another asynchronous message.

- ✓ Frames/fragments represent behaviour conditions such as loops, alternative processing and if-else constructs.

In order to correctly interpret a sequence diagram, it is best to start from the first actor and arrow on the left-hand side. Sequence diagrams are generally initialised by an action on the system. Sequence diagrams elaborate on the different actors and objects involved and how different requirements may link to one another as alternative paths. It was decided to avoid creating sequence diagrams for functional requirements which took a linear approach such as adding a user, instead focusing on more complex tasks which involved alternative routes to assist the understanding with the client and all other stakeholders.

Figure 4.5, illustrates the functional user requirements as followed: view attendance, take attendance, confirm attendance and edit attendance within one dynamic sequence diagram. Actors are shown in the top layer of the sequence diagram. The assumption is made that the user is already logged into the system. The lecturer will then navigate to their chosen module selecting the session i.e. lecture, workshop, etc. These actions are shown through several pairs of asynchronous messages. Once a session has been selected, the request is sent via the GUI to the system to retrieve a list of students attending the session. Messages are sent as such to complete the entire process of recording attendance. As an alternative procedure, there is an additional option to view past sessions. This path is surrounded by an additional frame signifying an alternative route. Similarly, after selecting a session a list of students will appear, the Lecturer will be able to resubmit their request which will be an additional update query.

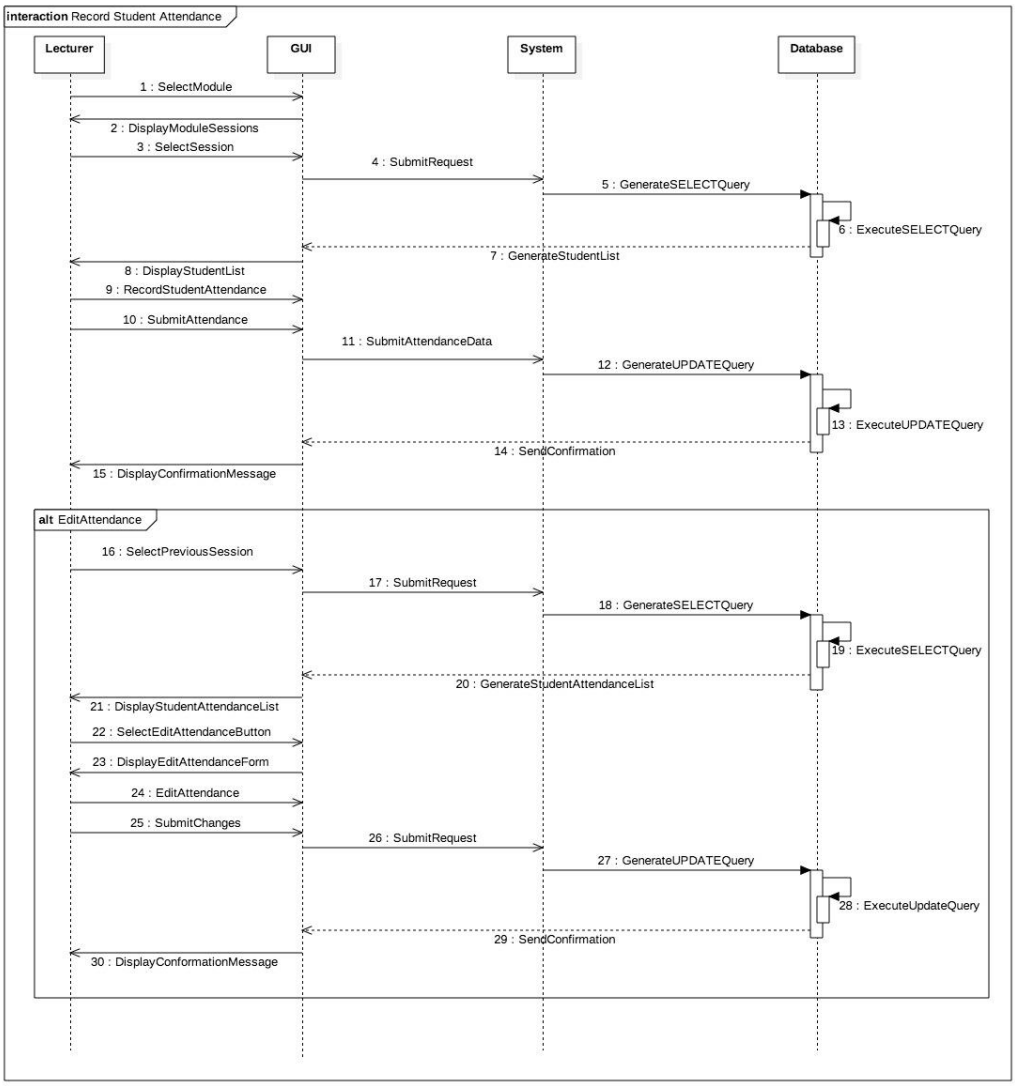


Figure 4.5: Sequence Diagram: Record Student Attendance (Lecturer)

Each tasks is numbered, clearly indicating a starting scenario and a successful outcome as well as possible alternative sequences. Sequence diagrams can be extremely beneficial for a project as they help the developer discover architectural, interface and logical problems prematurely. Further sequence diagrams can be found in *Appendix M*.

4.3 User Interface

Over the years, the technology industry is becoming increasingly competitive. Therefore, it is not enough to solely consider the functionality of a system but the pleasure of use for the intended users also. Human-computer interaction (HCI) is the study of how technology interacts with a user. HCI aims to "understand the designs that people need and the design processes" before development (Smith-Atakan, 2006). However, is argued that this approach pays less attention to the user experience and more so on optimising system performance.

Interactive design (ID) bridges this gap between technology and user experience by emphasising on designing an engaging and user-friendly interface. This approach enables deeper understanding to how a user will interact with the system, fixing premature issue and discovering better ways of doing things. ID focuses on understanding the user, their intentions and behaviour towards an interface in order to develop a sufficient design. In order to create an engaging experience, it is necessary to

understand the user's emotions and feelings towards the system (Rogers, 2011) . (Rogers, 2011) outlines four activities to achieve an ID which adopt an iterative and recursive cycle for success, this can be applied to the SAMS application:

1. Establishing requirements
2. Designing alternatives
3. Prototyping and,
4. Evaluating

Enhancing User Experience

Usability is best described as the interaction with a system and how easy it is. (Nielson, 2012) uncovers five quality components which measure a systems usability: learnability, efficiency, memorability, effectiveness and satisfaction. Incorporating these metrics when considering an interactive design will enhance the systems usability. Further development can occur after usability testing. However, before testing can be conducted several questions much be considered whilst designing the application:

- How does the user directly interact with the system?
- Does the appearance give the user insight on how to navigate? i.e. colours, shapes, etc.
- Can the user predict the outcome of their action with the information provided? i.e. meaningfully labelled buttons
- What feedback is given to a user once an action is performed?
- Are familiar structures utilised?
- Is the user experience simplified as much as possible?

Neilson emphasises the importance of testing to ensure product survival. If the proposed system is difficult to use, navigate or accomplish user tasks, individuals are more likely to divert from using the system (Nielson, 2012). This will be included in chapter six. In order to create a sufficient system, usability must be at the centre with an interactive design.

Navigation and Information architecture

It is vital to consider some user interface design principles which can enhance the overall user experience. By following industry practise, placing the navigation bar at the top of the page and key information in the middle, users are able to interpret the system with a familiar understanding. (Pernice, 2017) presents the theory of how users will scan a page, known as the "F-shaped pattern". The theory adopts the idea that users will generally fixate their attention "across the upper part" of the page, followed by their gaze motioning in a "horizontal movement". Lastly, users will move further down the page to cover "a second horizontal movement that typically covers a shorter area than the previous movement", forming an F shape pattern (Pernice, 2017). By following a design structure which supports this theory, the SAMS application can enhance the information intake and ease of completing a task.

Additionally, introducing white space to the information architecture can ensure content is fully legible for all user types. Overbearing users with too much information can divert them from using the SAMS application as a whole. A study conducted by (Lepper & Iyengar, 2001) showcases the effect of shoppers with too much choice. This study supports what is mentioned above, when too much information and choice is presented, users will deter from interacting with a system or completing their process (Weinschenk, 2009). Designers can blindly have the perception and displaying more information will increase user activity, however this is not the case. As the SAMS system will caters for a range of users it is vital to adopt a "one-fits all" design structure, ensuring relevant information is clearly visible with effective feedback following a clear cut structure. Overall in order to deploy an efficient user journey and experience it "involves simulating human behaviour,

which varies according to spatial layouts, context, environment and interaction with other users" (Sharma & Fisher, 2018). It is key to consider this through the page design process.

4.3.1 Page Design

Low fidelity wireframes

Initial sketches can conceptualise the application before major design decisions are finalised. Such approach is ideal for grasping design ideas, which can be easily erased and redrawn to suit the nature of the problem. This process can include informative feedback from client and potential users to depict how they would like to interact with the system during their user journey. Figure 4.6 is an example of a sketch drawn up to depict an interface structure idea, the remaining can be found in *Appendix N.* The sketches include annotations of desired deign decision whilst considering user experience.

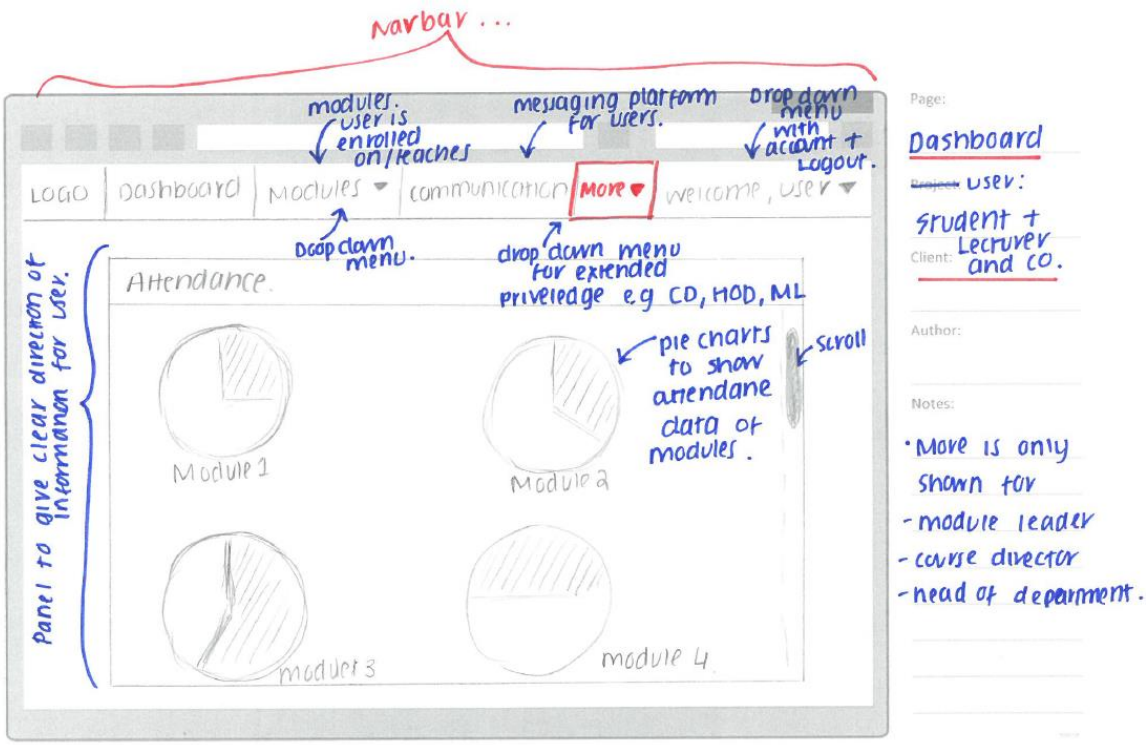


Figure 4.6 Student and Lecturer & co dashboard

High fidelity wireframes

After closely analysing the sketches with the client and the different user types, the informative feedback received has influence the wireframes demonstrated within this subsection. Essentially, wireframes are best described as "a simplified view of what content will appear on each screen of the final product, usually devoid of colour, topographical styles, and images" (Brown, 2011). Such diagrams enable visualisation of the end product, rising opportunities for further feedback prior implementation. The software utilised to compose the proposed wireframes is Sketch. Sketch is a free sources tool, which enabled design toolkits with the components of bootstrap to be downloaded. Each wireframe will be explained for the processes taken by the different user types. Further wireframes are shown in *Appendix O.*

Lecturer

After logging into the system each user will fall onto their own dashboard. The students and faculty dashboard look very alike, whilst the administrator differs due to the nature of his role. For each user a navigation bar will be located at the top of every page, this is to follow best practice for usability purposes. As seen in Figure 4.7, the Lecturer has four options;

- Dashboard
- Modules: drop down menu of modules he teaches to take attendance.
- Communication: a messaging platform between lecturers and students.
- On the right-hand side a welcoming message personalising the platform for each user, "Welcome, first name". Here, additional personal information is displayed such as account details and log out.

As seen in Figure 4.7, the Lecturer's dashboard will display the different modules he teaches through pie charts to show the overall attended vs non-attended. Pie charts are utilised to present data visualisation to the user, depicting a universal understanding for all. The attendance data within the pie chart is shown on a weekly basis, as requested by the client. By clicking each module this will lead him through to page displaying the individual session attendance within the module as shown in Figure 4.8. One the right-hand side, the upcoming events are displayed for the next seven days which include lectures and workshops.

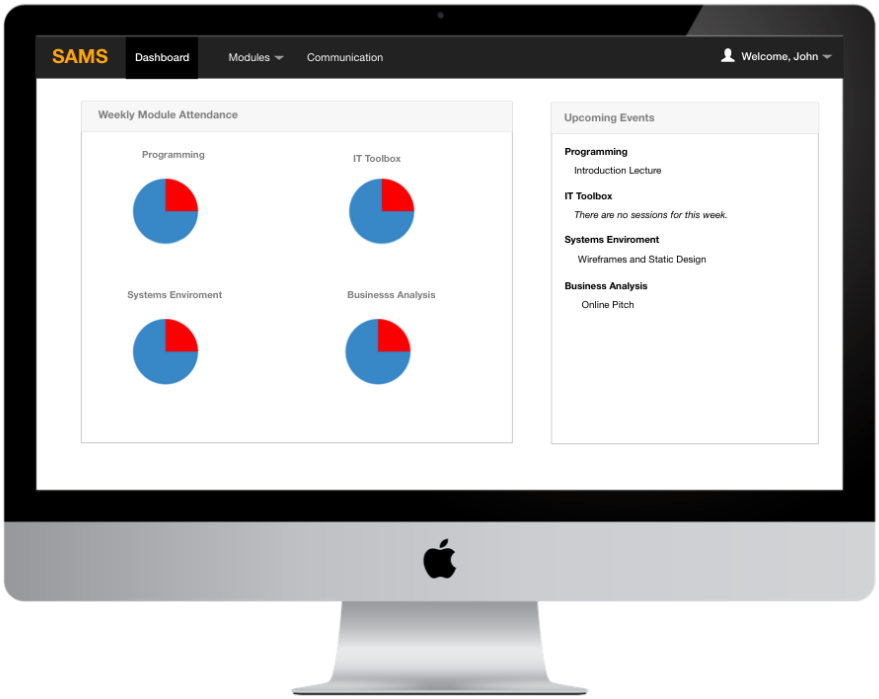


Figure 4.7: Lecturer Dashboard

Figure 4.8 showcases the information about each session attendance to date. This includes the percentage of attendance data captured from the different sessions. The percentage of attendance is displayed through primary colours red and green. For attendance data which is 70% and above, the data is shown as green, this is to promote success. Whereas attendance data depicted below 70% is displayed red for danger and to alert the user. Each page is given the same bootstrap panel view to promote consistency. The client requested a system which enabled the "drilling down" of data to the students it refers to. Therefore, an option to see each session attendance is provided and described in Figure 4.9.

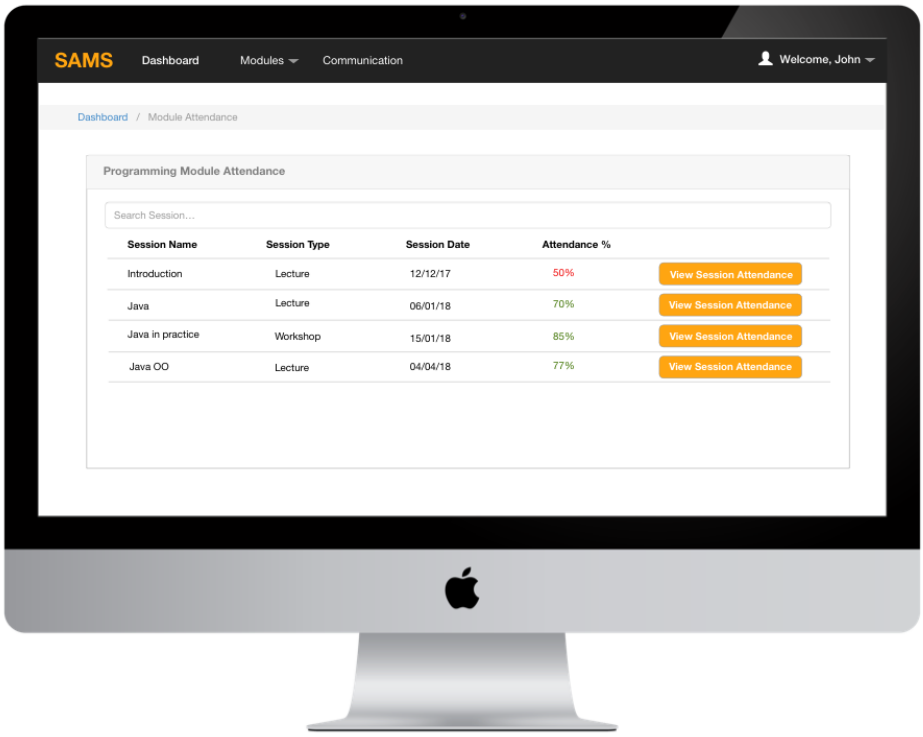


Figure 4.8: Break down of Module Attendance

Figure 4.9 demonstrates the attendance data submitted for each student for the session selected. Here the Lecturer has a clear and descriptive view of who is not attending the sessions, this can be taken a step deeper through to view student profiles. Individual student-based view is illustrated in Figure 4.10.

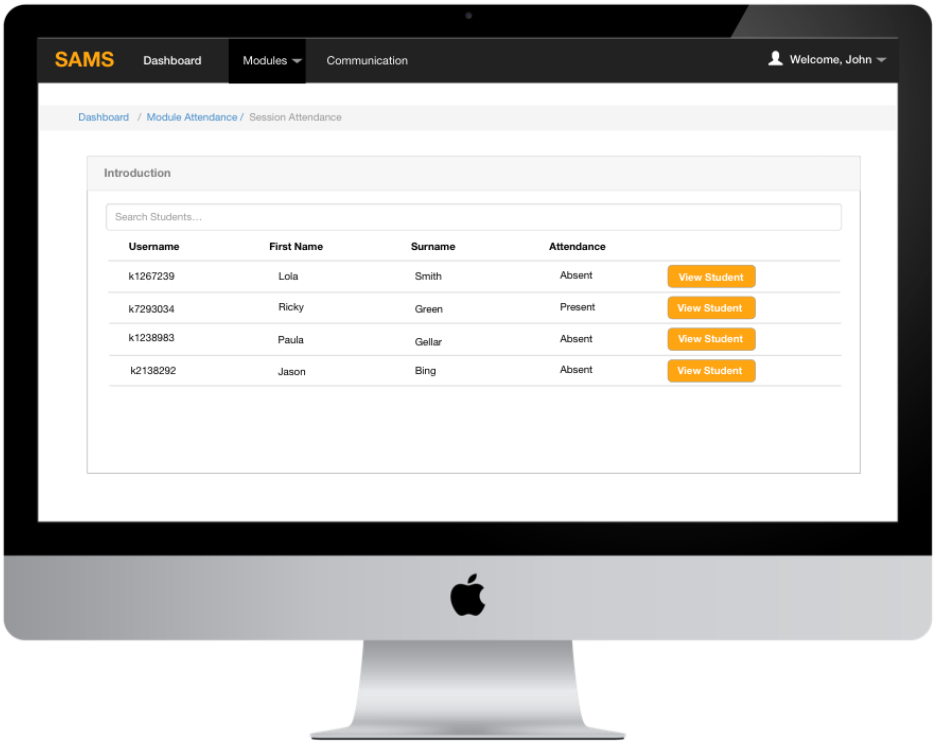


Figure 4.9: Break down of individual session attendance

Figure 4.10 illustrates a single student's profile. It showcases all the modules they are currently enrolled upon and their personal details as well. One of which is the attachment of their record of progress. This is inserted to enable correlations of attendance and grades to be clear. This will in turn, highlight areas in which students are disengaging and weather this as an effect upon their academic success.

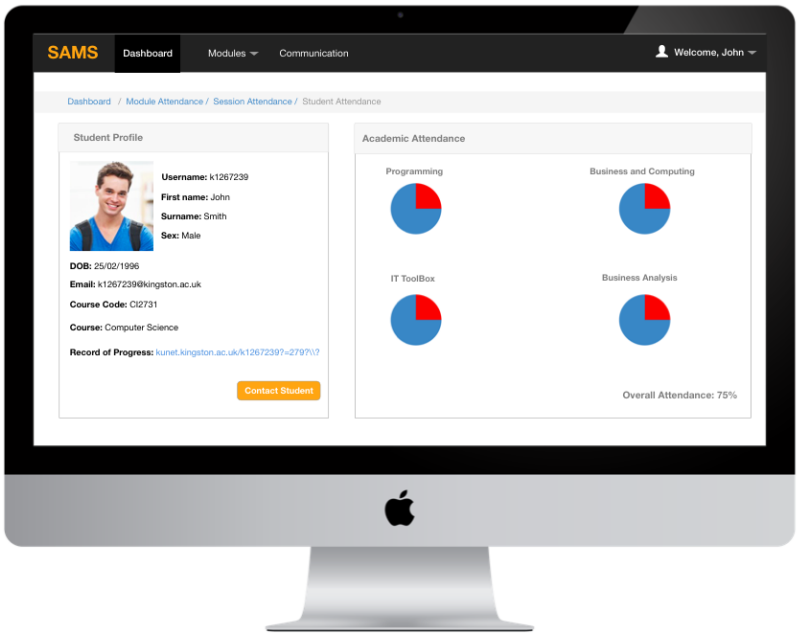


Figure 4.10: Individual Student view

Course Director

Module Leaders, Course Directors and Head of Department will be subsumed of the Lecturer. They will share the same accessibility rights on tasks, however include extended privileges depending on their role. As seen in Figure 4.11, the course director can view the module attendance within the Computer Science course. This can be drilled down into individual students as seen in the flow for the lecturer in Figures 4.7 - 4.10.

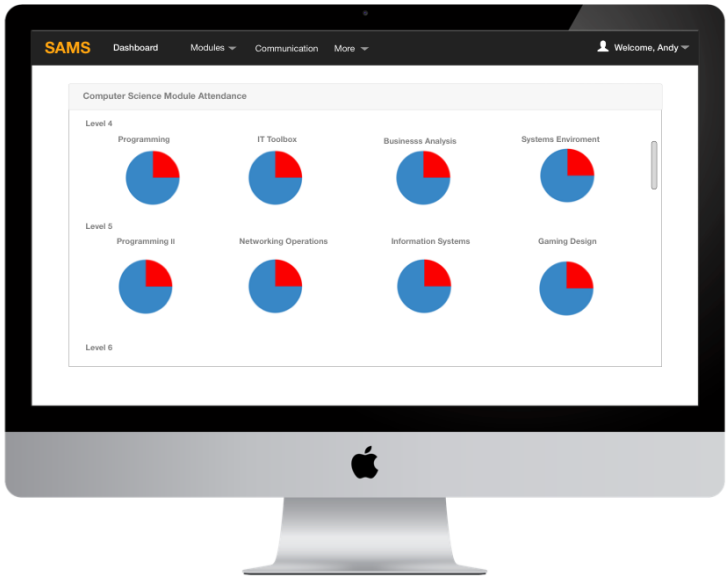


Figure 4.11: Course Director View of modules within Computer Science Course

Student

The Student will be welcomes with the same homepage as the Lecturer with their according modules. Figure 4.12, displays the attendance data for a selected module, shown as “absent”, “present” or “late”. Additionally, there is also a view of the overall attendance percentage. Here students will be able to monitor their own attendance.

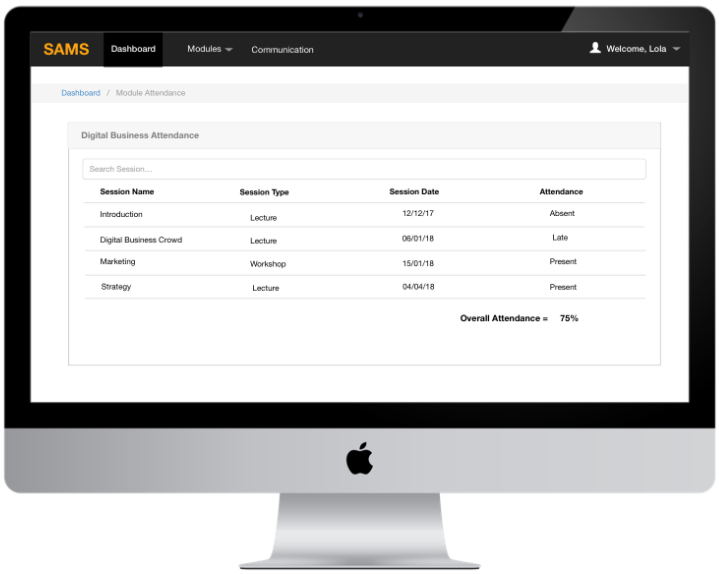


Figure 4.12: Student module attendance view

Administrator

The Administrator manage the different components of the system e.g. students, staff, modules, courses and departments. Figure 4.13 showcases the view of the users of the system. This includes additional options for adding a new user, editing, deleted and resetting user password.

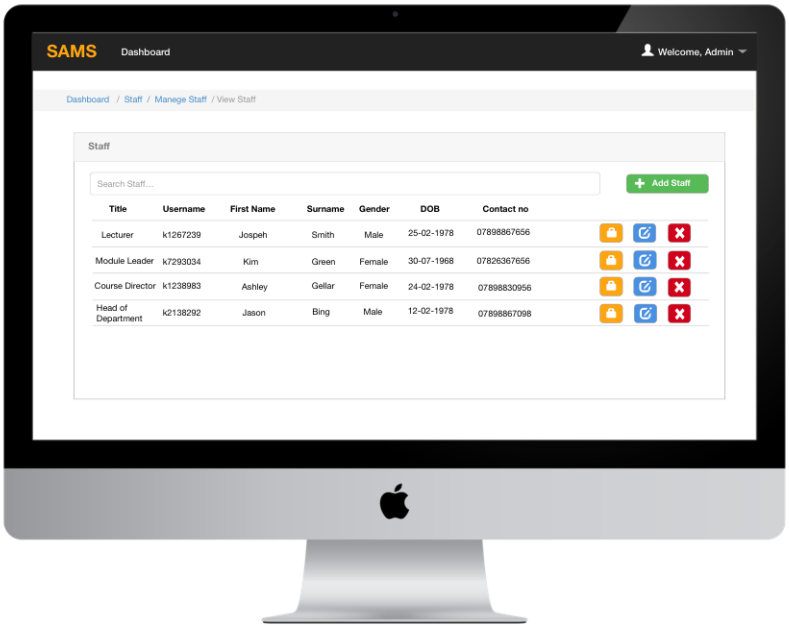


Figure 4.13: Admin manage staff

4.3.2 Navigation Diagram/Site Map

Navigation diagrams show a hierarchy structure of the application design. Also known as site maps, they are a useful tool widely used by web designers to provide an overview of a sites content and coding layout at a single glance. The diagram showcase the typical routes of the user, these include the critically important return journeys. Return journeys tend to be recursive relationships such as Log out which redirects back to the log in page, to complete the system circle. The menu structure for each user is unique, demonstrating their navigation abilities. However, there are features which will be the identical for everyone such as: log-In page, edit personal details and log out. Once the user has gained access to the system, they will be welcomed by the first layer of the navigation diagram. Each user will land directly onto their Homepage (dashboard). This layer also consists of the option to log out which is in the same location for all users to promote consistency. Figure 4.14, illustrates the navigation/sitemap for the Administrator. Further navigation diagrams can be found in Appendix P.

From all three user types' menu structures, the Administrator is visibly the most diverse. The user is given extended options to create, review, edit and delete in the following categories: staff, student, department, course, module and resetting user passwords. Deemed to be the most complicated navigation menu, a segment of the diagram has been partitioned for further detail in understanding, in Figure 4.15. A component in which the administrator will have control over is staff. One of the categories in which the Administrator has access over is Staff. After selecting this category, the administrator is given a further two options to "View Staff" and "Enrol Staff". The options depicted within "view staff" enable the administrator to create, edit and delete entries, all in which include a return journey to "view staff" once completed. These conventions apply similarly to the rest of Figure 4.14, and therefore an in-depth understanding of one fragment is adequate.

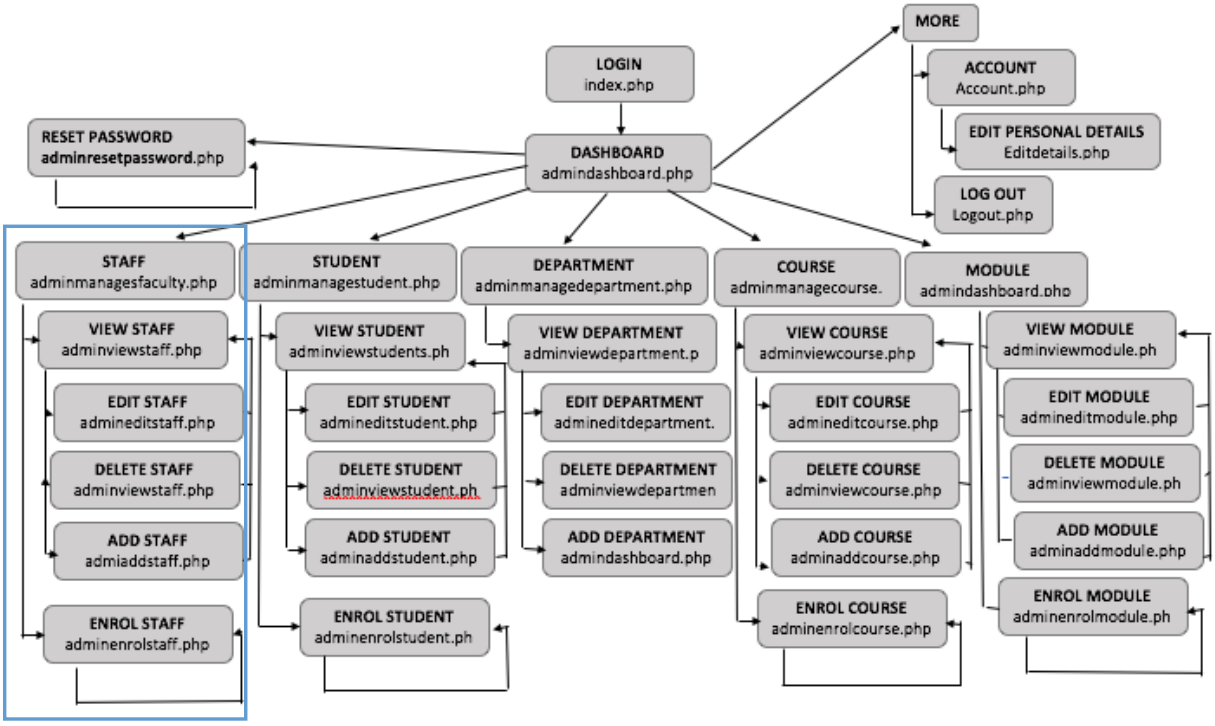


Figure 4.14: Administrator Navigation Diagram/Site Map

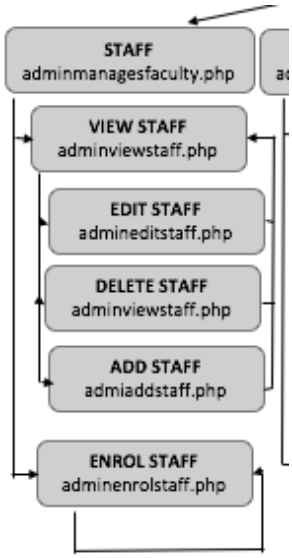


Figure 4.15: Administrator Navigation Diagram/Site Map (Staff)

Due to the agile nature for the project, the implementation phase may not directly map the same structure as seen with the navigation diagrams/site maps. However, these design phases are crucial to formulate a foundation for the implementation phases.

4.4 Conclusion

As well as considering recommendations from previous chapters, the design process saw to meet all user and client needs. As a result, an efficient and logical database design was generated, resolving conflicting relationships between classes. Such designs will enable the database to be implemented and populated with the necessary data. Furthermore, enough valuable information was gathered to understand how the system and the user should interact. Sequence diagrams for complex tasks such as record student attendance, assisted in establishing a foundation for the user interface design. Consequently, user interface wireframes was designed to showcase the users interaction in general tasks and how each page is formed together to create a seamless user journey. Nevertheless, due the time limitation of the project, there was little time to create further enriched users designs which would have been valuable to identify any issues at an early stage.

There were a few obstacles during the design phases which resulted in some delay, one in which was the functional requirement taking attendance for a Lecturer. Special requests were made by the client, to assure that the time taken to record of attendance was kept to a minimal. The time taken to generate the solution caused a great deal of pressure when producing the other sections of the report. Nonetheless, a solution was created and deliberated with the client, to implement a default value of for each entry when a new attendance list is generated, shown in Table 4.2 (StudentAttendance). The next chapter will consider and tie together the principles of the design chapter when implementing the SAMS web application.

5 Implementation

The tools and methodologies utilised in the previous chapter enabled the developer to demonstrate the blueprints for the proposed system to the client. Having outlined in detail, the design for the proposed system, this chapter will exhibit and implement the processes to create the end product. Research has been conducted to discover the most appropriate tools, technologies and resources to build the SAMS web application. Each tool has been tested and compared to understand the benefits and features to ease the implementation process. By drawing upon the strengths of these components it will enable the developer to create and execute an efficient and dynamic web application. This chapter will also reflect upon the database construct which was planned during the design phases of the project. Undertaking these crucial steps will build the backbone of the system. Moreover, a typical user workflow enduring different functional requirements of the system, will showcase the code, languages and libraries implemented. This segment will demonstrate and discuss the limitations and problems encountered during the implementation phase. Overall, this section will showcase the components and provisions behind the Graphic User Interface (GUI). Lastly, security exploitation can cause a detrimental effect to the system performance, therefore the implications to prevent this will also be visited. Therefore, this chapter will build upon the preliminary designs showcased in chapter four and convert them into a sufficient application to meet the client's requirements and in which users are able to successfully interact with.

5.1 Tools, Technologies and Architecture

This segment of the report will explore the vast range of tools and technologies that are used to execute the SAMS web application. It is crucial to research and test the different tools which are required, to understand how to maximise their features and ease the implementation process. By drawing upon the strengths of these resources, the developer is able to create a well-executed, effective and dynamic application. In addition, an overview of the system architecture has been composed and agreed upon with the client to showcase how each component work together harmoniously.

5.1.1 Tools

Atom

The SAMS web application will be fully developed using the text editor *ATOM*. ATOM is a free open-source editor which supports various programming languages such as C++, Python, PHP, JavaScript and HTML5. The text-editing software provides a user-friendly environment with syntax corrections, code completion suggestions and well-presented colour-coded text as visual aid to assist the user. This is especially beneficial for novice users getting the grips of new languages finding errors, testing and problem-solving code created. For this project a package which enabled FTP (File transfer Protocol) was installed into atom. It enabled generated code to be instantly uploaded to the server as soon as modifications were made and saved.

Alternatively, other code editors which could have been utilised such as Integrated Development Environments (IDE) for example NetBeans. Although NetBeans provides additional integrated development modules for the different programming languages, due to the over complicated format and the additional training required for advanced features and tools it was disregarded. Overall, ATOM enables the developer to minimise time expenditure through suggested code assistance and will be able to uphold the maintainability of code.

PhpMyAdmin

PhpMyAdmin is used to administer the database for the application. This tool will be utilised in implementing, generating and storing data. The open source administration tool primarily written in

PHP, is seen as the one of the most powerful MySQL tools for web hosting services. PhpMyAdmin enables data manipulation through PHP/SQL requests, by performing various tasks such as creating, modifying or deleting databases, tables or data entries; embedded constraints to prevent error in data entry; or managing users and their permissions. The web application will be hosted on Kingston University's StudentKUNET server which is pre-packaged with PhpMyAdmin. Therefore, the two software's are able to work harmoniously together to form the back-end for this project.

StudentKUNET server allows little flexibility in comparison to other web servers, it is a free host which enables processing request in communication with the server to be performed quickly and efficiently. Using a local server via applications such as *XAMP* can also substitute as an efficient alternative allowing code to be quickly uploaded through an internal host. Nevertheless, the StudentKUNET was deemed sufficient for this project as it allowed the developer to test the application on various devices.

5.1.2 Technologies

HTML & CSS

Hypertext Mark-up Language (HTML) is a well-known language used for creating and composing web pages and applications. Working alongside *Cascading Style Sheets (CSS)* and *JavaScript,* HTML provides the visual presentation and interaction between user and an application. Furthermore, the use of CSS enables manipulation of HTML code for better presentation by adding fonts, shapes, colours and influencing the layout. Together the two languages create and finalise the appearance of a webpage. The advantages of using HTML, are outlined as followed:

- HTML is widely supports by every web browser.
- HTML is a logical thought-out code and therefore is easily learnt and used globally.
- It is also widely used therefore there is endless help from the online community.

HTML allows for different languages to be embedded within one page of code in order to create a fully functional web application. However, there is much debated criticism of mixing different languages as it is portrayed as merging business logic with presentation. Employing frameworks such as Model View Control (MVC) can create a solution to this problem. MVC enables the development of code to be organised in a practical elegant matter by dividing the application into three interconnected parts. However, as the developer wanted to focus on the creation of the application and strengthening the knowledge on PDO, this approach was considered too time consuming to learn.

JavaScript

JavaScript is a scripting language which implements dynamic content visible on the client-side interface. The developer is able to create a highly responsive interface with dynamic functionality to will improve the user journey and experience. Therefore, the use of JavaScript is beneficial for the following reasons:

- Snippets of code can be written to extend the functionality of certain webpages.
- As the code is executed client side, the processing is almost instant as it does not require to be processed by the server.
- Similarly seen with HTML, JavaScript is very versatile ad can work closely along other languages such as PHP.

An extension of JavaScript, which is widely practised today is *JQuery.* JQuery is a library of JavaScript functions which enables certain tasks to complete such as respond to user events. The purpose of JQuery is to make it easier to use JavaScript across webpages. It built upon the functionality of the browser via the JavaScript Document Object Model (DOM) API, enabling the developer to accomplish the same task in fewer lines of code.

PHP PDO

PHP is a server-side programming language which is especially suited for web development within an application. It is a flexible, easy-to-learn language which is significantly supported by online resources and a large coding community to assist when solving implementation issues. As an extension of PHP, this application will be written in *PHP Data Objects (PDO)*. PDO enforces the connection between the application and the database by incorporating objects when coding. Thus, the SAMS's web application will incorporate the use of PDO when dealing with the server-side of the application due to the following reasons:

- PDO support many different database drivers. Therefore, as well as the vast selection of recourses and convenience, PDO is also able to enforce maintainability of code.
- PDO is great for code which requires object mapping. This enables closer alignment with the database.
- PDO library provides SQL injection security, provided the developer codes the intended way.

On the other hand, an alternative to PDO is the enhanced MySQLi. MySQLi also deploys an object-oriented interface with enhanced debugging capabilities. However, MySQLi does not support as many different database drivers. In addition, PDO is more secure and prevents countless SQL attacks by embedding provisions to cleanse of data.

5.1.3 Frameworks

Bootstrap

Bootstrap is a free open-source front end library used to make an application responsive to different devices. The framework incorporates HTML, CSS and JavaScript, thriving for a responsive design. By using bootstrap, more time can be applied to the main functionality of the application focusing on a user-centric approach. The Bootstrap framework already contains pre-styled components such as Navbar and Bootstraps Grid, which already have responsiveness built in so the developer is able to save a significant amount of time in the front-end presentation. Thus, the application will incorporate the use of Bootstrap due to the following reasons:

- Bootstraps offers a range of pre-styled components which can be modified through an internal CSS.
- Bootstrap is packaged with JavaScript within its components, which will include simple operating functionality.
- Bootstrap offers a number of components which can enhance the responsiveness of a system such as the grid layout. This feature amends the layout to the appropriate screen resolution.
- One of the biggest advantages of Bootstrap is speed of the development. The developer is able to focus more time in the back-end because of the many advantages within bootstrap.

5.1.4 System Architecture

As request by the client, the SAMS's web application will be compatible with an array of devices such as PCs, tablets and smartphones. Assuming that these devices have a preinstalled web browser, all interactions from user to application will be conducted this way. Figure 5.1 illustrates the communication from server to user, incorporating system interaction in between each stage. The browser will send a HTTP requests to the Web Server, which will store the SAMS website components onto the host device ready to be accessed by a specific domain. The Web Server will work with the front-end components (HTML & CSS, JavaScript & Bootstrap) and the back-end components (PHP PDO & SQL) to communicate with the Database Server. As soon as the requested file is found, the Web Server will respond with a HTTP request back through to the browser and displaying the retrieved data results to the user.

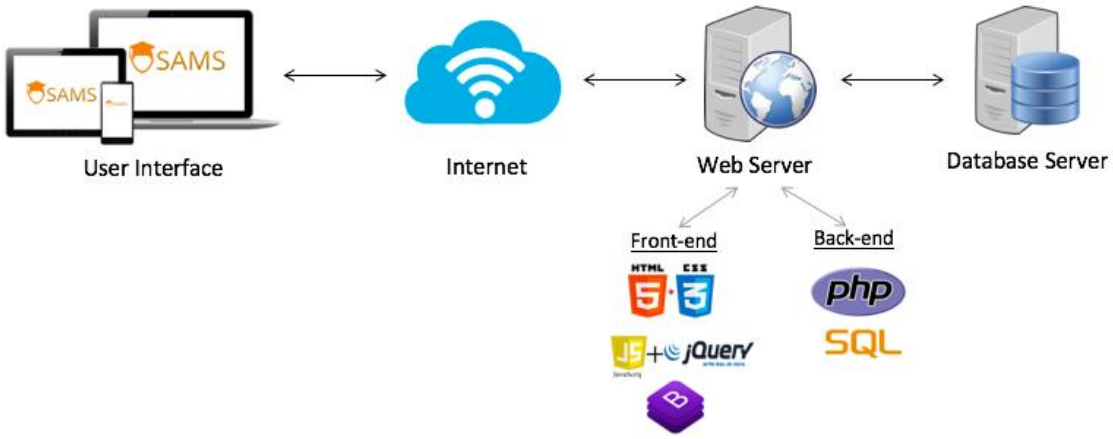


Figure 5.1: SAMS System Overview

In each implemented page of the SAMS web application will contain numerous libraries within <head> tags. When sending a request to a specific host it is fulfilled through a link provided by the *href* attribute illustrated in figure 5.2. The web server will respond with the associated page formatted and presented onto the user's browser. Furthermore, for requests which require database access, the database server will also be incorporated in the request/response procedure. The Database Server will read the embedded SQL queries and execute the request retrieving the appropriate data.

To create a more responsive design Meta viewpoint is implemented to assist the browser with rescaling the dimensions for the page. Each <link> tag fetches the necessary documents from numerous hosts. Lines five and seven are style sheets, supporting the presentation of the application including Bootstrap CSS and an internal .CSS files. The internal style sheet can overwrite the bootstrap styling to enable a more efficient front-end design catering to the client's requirements. Alternatively, the Bootstrap style sheet can be downloaded and saved onto the host server alongside the internal CSS file. However, due to frequent updates the current approach is more practical is it only requires the adaptation a single line of code. In addition, line six displays an icon on the user's browser, as shown in figure 5.3, purely to give the application a personal touch. The remaining script tags import the required JavaScript libraries to create a highly responsive interface with dynamic functionality.

```
<head>
  <title>Dashboard</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link rel="icon" type="image/png" href="Grad-hat.ico">
  <link rel="stylesheet" type="text/css" href="custom.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.1/Chart.bundle.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
```

Figure 5.2: Head tags for the SAMS application

Figure 5.3: SAMS browser icon

As the SAMS application is significantly database reliant, a connection to the database is established to allow manipulation of data through request/response cycles. A single file was created which stored the according configuration details for the application. This was to ensure high levels of efficiency and maintainability throughout the code. As a result, each page would only require a single line of code, as shown in Figure 5.4. The PHP function 'require_once' calls upon the connection.php file, shown in Figure 5.5, to obtain a database connection.

Figure 5.5 illustrates use of "try" and "catch" PHP functions to attain the database connection. These functions demonstrate the attempt to capture the database connection (try) and where the connection has failed to be attained (catch). The information required to connect with the host is stored within individual variables, shown in lines five to seven. These variables create a new object of the PDO class and can be accessed through the global variable $pdo. As shown in Figure 5.5, the new object must take three parameters; details of the host, the username for database entry and the subsequent password (concealed for security reasons). However, if connection to database does fail, the error message will be outputted, and the attempts to connect will "die". The pages, in which the code corresponds will be noted in the caption.

```php
<?php
require_once 'connection.php';
?>
```

Figure 5.4: Require_Once Database Connection

```php
try {
  $servername = "kunet.kingston.ac.uk";
  $username [redacted]
  $password [redacted]
  $dbname = [redacted]


  $pdo = new PDO(
  "mysql:host=$servername;dbname=$dbname", $username, $password);
  //echo "Connected";
} catch (PDOException $e) {
print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```

Figure 5.5: Connection to Database (connection.php)

5.1.5 Conclusion

It is vital to select the most appropriate tools and technologies in order to have a definitive understanding of the system architecture throughout the development process. Although they work independently, the front-end and back-end tools have been chosen to ensure maintainability and sustainability of the overall application. As UX is an extremely important factor to consider when creating the SAMS web application, additional tools such as JavaScript have been utilised to create a highly responsive interface with dynamic functionality. Once validated by the client, the tools will be firstly executed to implement the database following the design criteria in chapter four.

5.2 Implementing the Database

The database structure was planned during the design phases for this project. A range of techniques and methodologies were utilised to propose the most appropriate database structure. These procedures enabled the developer to establish the different relationships, attributes and

constraints. This segment of the report will reflect upon the implementation, evaluating and analysing the effectiveness of the design.

5.2.1 Creating tables and constraints

The tools and methodologies utilised during the design phase enabled the developer to gain a deeper understanding of the logical structure. Each tool has been uniquely applied to elicit information to build the database with minimal time expenditure. This included the entities required to achieve the project goals. In addition, the data dictionary methodology assisted the developer to clearly visualise the format and structure of the desired database including attributes, referential relationships and potential constraints. The database is managed through PhpMyAdmin where StudentKUNET provides hosting grounds. With the design and structure readily available, the developer was able to implement the appropriate tables agreed upon with the client. There are two approaches which can be taken when creating database tables. The first method is one in which all user expertise can easily navigate; implementing tables through a Graphic User Interface (GUI). As seen from Figure 5.6, the approach displays a simple user-friendly interface which requires no coding. The structured form allows the user to effortlessly recognise the different components necessary for table creation, including indicators for a Primary Key (PK). This method prompts the users for any missing information deterring from human-error. On the other hand, a more code-based alternative is composing SQL statements, as seen in Figure 5.7. This process promotes Data Definition Languages (DDL) to manipulate the database in table creation, insertion and deletion. This approach is for the more advanced SQL coders as the language requires a sequential structure in order to be executed correctly.

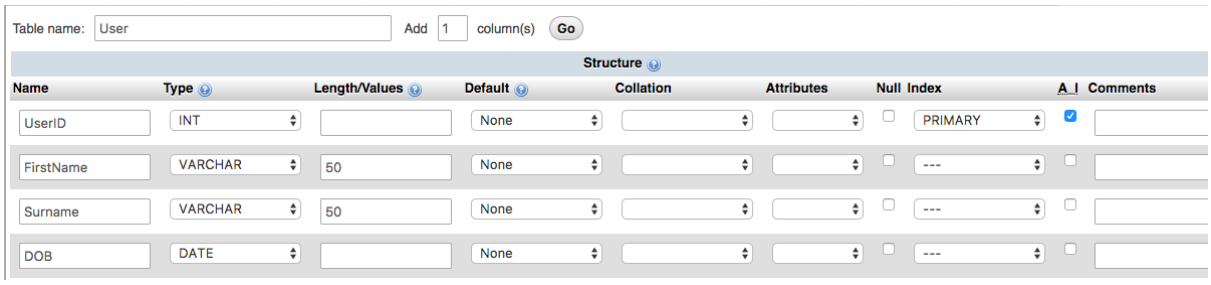


Figure 5.6: Table Creation via GUI

```
1 CREATE TABLE User
2     (UserID int(8) PRIMARY KEY AUTO_INCREMENT,
3      FirstName varchar(50) NOT NULL,
4      Surname varchar(50) NOT NULL,
5      DOB date)
```

Figure 5.7: Manual Table Creation Process

Both methodologies share implementation issues when applying a Foreign Key (FK) between tables. For this project the GUI approach was taken, as more time can be focused on implementing the core functionality of the application. Figure 5.8 demonstrates how a FK must be declared in such cases. Firstly, the desired attribute must be made into an index, this allows for FK constraints to be placed (Figure 5.8 image 1). Secondly, the table must be viewed from the 'Relation View' Tab where the relationships between the child and parent table can be established (Figure 5.8 image 2). Once the foreign key has been created the final step is to implement action constraints declared by "On Delete" and "On Update". These constraints prompt the system with direction when the FK is deleted or updated. For this project, all FK's constraints will be set to CASCADE for "On Delete" and

"On Update", as shown in Figure 5.8. A FK with CASCADE option selected declares that if a record in the parent table is deleted/updated then the corresponding records in the child tables will mirror this action. This will uphold a consistent and an up-to-date database, complying with the eight protection principles in the Data Protection Act 1998 explored in chapter three. However, to avoid accidentally deleting a necessary record, the developer will encode prompts to the user for confirmation before pursuing the request. Furthermore, Figure 5.9 illustrates the final tables for the database. Each referential relationship was validated through the insertion of dummy data to ensure all constraints were fully functional, prior populating each table with the necessary real data required. Methods of populating the data can be found in *Appendix Q.*

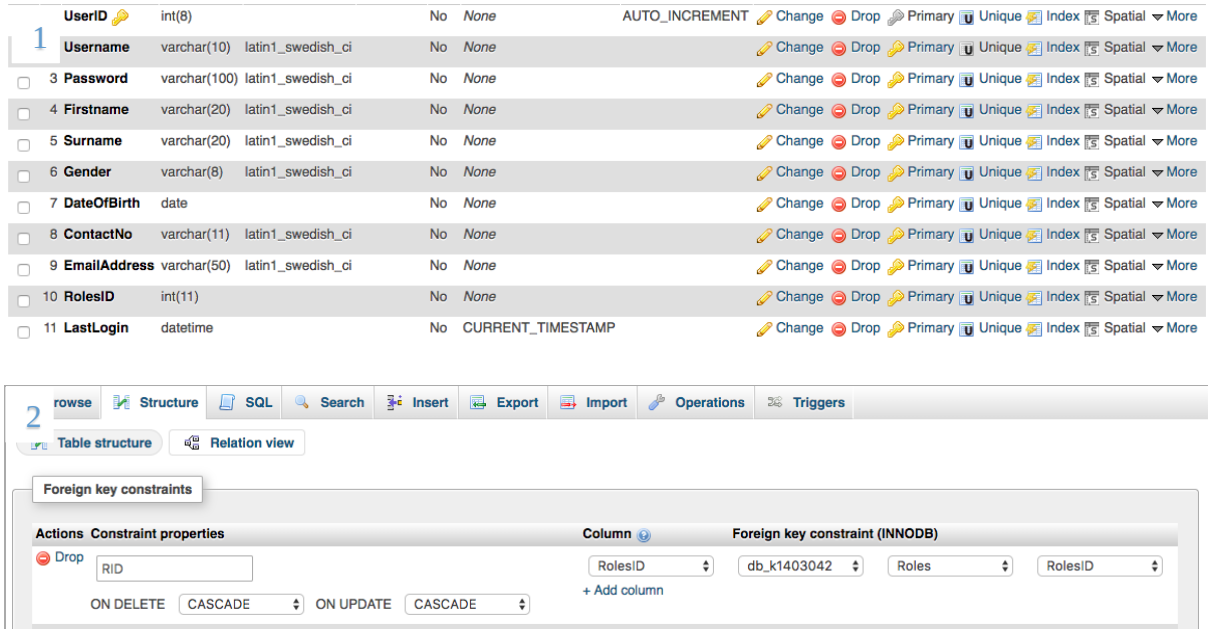


Figure 5.8: Establishing a Foreign Key via GUI

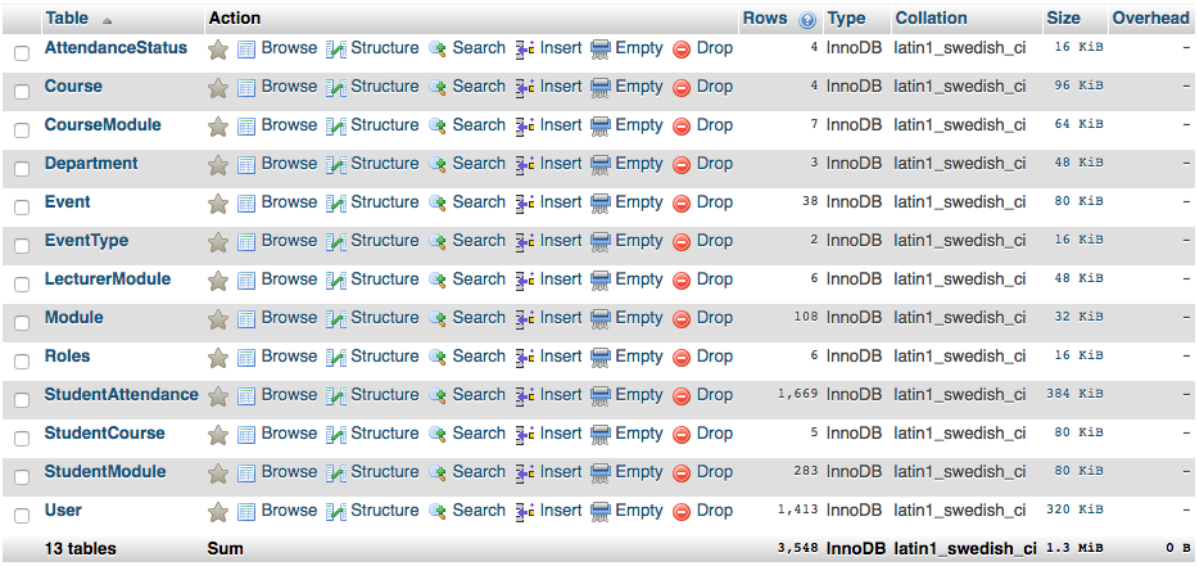


Figure 5.9: Implemented tables for SAMS application

5.2.2 Conclusion

The project adopts an agile methodology allowing for an iterative and recursive project process. Therefore, during the implementation there were some minor modifications applied to enforce a

better understanding of the database. Nevertheless, the main design principles outline in chapter four gave the developer a clear form of direction when implementing the database. Overall, a sufficient database for the propose system according to the requirements set by the client has proven successful. This will stand as the back bone of the system which the interface will be built upon. Therefore, next is to build the user interface which will communicate with the database through different programming languages.

5.3 Constructing the User Interface

Once the database is successfully implemented, the user interface will be constructed following the design wireframes outlined in chapter four. When constructing the user interfere, a range of programming languages have been used to compose the most appropriate interface. These languages include HTML/CSS, JavaScript/JQuery Bootstrap, PHP and SQL. These tools and technologies have been adapted and embedded within one another, harmoniously working towards a sufficient system. The segment of the report will go through a typical user workflow showcasing the different components utilised at each step. The following functional requirements will be executed and discussed in further detail:

- Login
- Create a new session which will concurrently generate an attendance list of students.
- Take and Submit attendance.

This segment will explore the code implemented to create the SAMS web application, and the obstacles encountered and conquered in the process. Overall, this section will showcase the components and provisions behind the Graphic User Interface (GUI).

5.3.1 Login

The functionality of the system will run through several functional requirements outlined in this segment, combined they will sequence to a typical user workflow. One of the most important stages of the system is the login which is briefly touched upon. The login is an important gateway to the system which automatically differentiates the different user's accessibility rights and start of their system journey.

Fetch() and FetchAll()

When a user submits their credentials to login, the system will query the database and retrieve the results. Once executed, the results of the SQL statement is utilised by two functions, Fetch() and FetchAll() as seen in Figure 5.10. This was an implementation decision carried out by the developer to increase the efficiency of the system. Fetch() is a definitive PHP function which fetches the first row of user details which match the conditions set in the query. This is then iterated through as session variables and the relevant data is pulled through the session onto the next page. On the other hand, FetchAll() prepares an array of results and compares these to a match in the database, allowing access to the system if successful. Additionally as seen in Figure 5.11, the function sizeof() is utilised to check the results from the FetchAll() execution to ensure that only one unique entry matches the criteria. As an alternative, the count() function could have been utilised. However, SizeOf() is only an alias of count() and therefore is not replicated the same in other languages when collaborating code. This would avoid ambiguity across languages. Figure 5.12, demonstrates the login GUI.

```php
<?php
  session_start();
  require_once "connection.php";

    $loginusername = $_POST['kuUsername'];
    $loginpassword = encrypt("Encryptdpassword", $_POST["kuPassword"]);

    $sql = "SELECT * FROM User WHERE Username='$loginusername' and Password='$loginpassword'";
    $stmt = $pdo->prepare($sql);
    $stmt->execute();
    $row = $stmt->fetch();

    $stmt = $pdo->prepare($sql);
    $stmt->execute();
    $result = $stmt->fetchAll();
```

Figure 5.10: Query he database with users log in credentials (login.php)

```php
if (sizeof($result) === 1)
  {
      $sql = "UPDATE User SET LastLogin = NOW() WHERE Username='$loginusername';";
      $stmt = $pdo->prepare($sql);
      $stmt->execute();
  if ($stmt->execute() === TRUE)
    {
      echo "<p>Record updated successfully</p>";
      echo "<p>Successful log in</p>";
    } else
        {
          echo "Error updating record: " . $stmt->error;
        }
```

Figure 5.11: Redirecting user to desired dashboard (login.php)

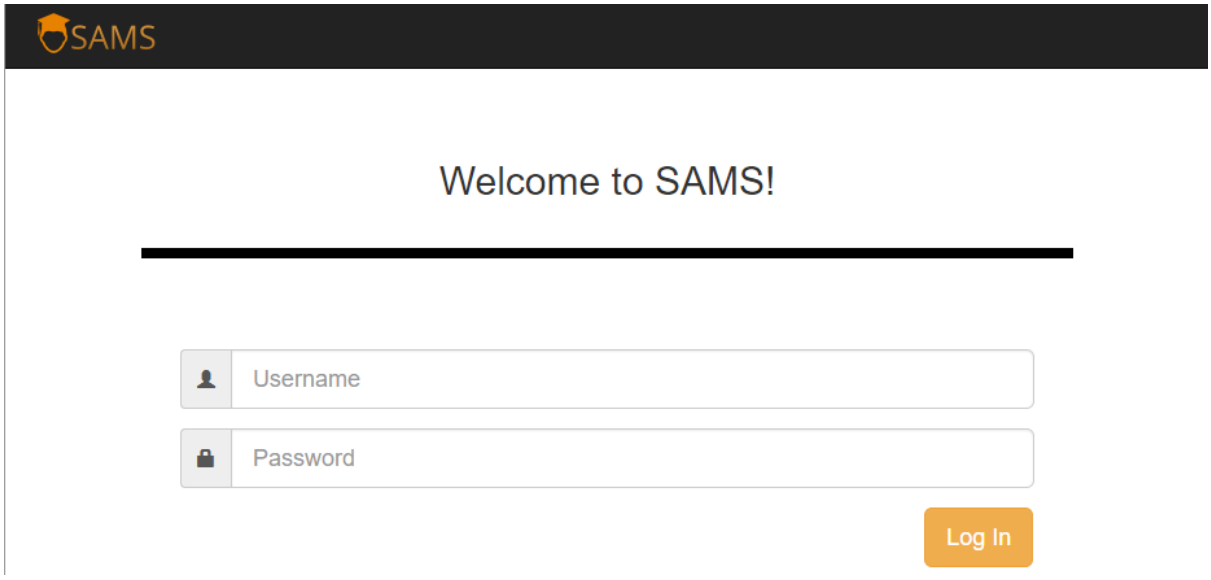


Figure 5.12: Login GUI.

Functions

PHP has numerous built in functions which create an efficient stream of code to be repeated in the program and is only executed when called upon. The function sendTo was created which took one parameter, $usertype as seen in figure 5.13. This function encloses the navigation to a particular page, in regard to user type. In figure 5.14, the function is called assigning a value to the parameter which is RoleID from the database.

```
function sendTo($usertype)
  {
      if($usertype == 1)
        {
          header("Location: studentdashboard.php");
        }else if($usertype == 2 || $usertype == 3 || $usertype == 4 || $usertype == 5){
          header("Location: lecturerdashboard.php");
        }else if($usertype == 6){
          header("Location: admindashboard.php");
        }
  }
```

Figure 5.13: Differentiating User access rights (login,php)

```
sendTo($row['RolesID']);
```

Figure 5.14: Calling function to redirect Users to their dashboard (login.php)

5.3.2 Dashboard

After user is logged in with the correct credentials, each user type will be directed onto their dashboard. The Administrator's dashboard differs from other users with icons displayed to manage the different components of the system. On the other hand, Lecturer and Student dashboards are similar to one another. These user dashboards will have the module attendance displayed by a pie chart. Pie charts are an easy-to-understand and effective communication tool universally understood across all audiences. Therefore, pie charts are chosen to pin point and emphasise pieces of data and enhance the user journey. However, composing these steps involved the combination of PHP PDO, HTML/CSS and JavaScript.

Complex SQL query

In order to retrieve the data required for the pie chart, an SQL query is generated querying three tables for data. There were two different queries in which had to be executed to retrieve the data required: students who attended and those who did not. Figure 5.15, demonstrates a for each loop iterating results for each module the lecturer teach. Nested within the loop is the two queries retrieving the unique attendance data for each module. For each loop is a useful construct to collaborate the information required for the Lecturer's Dashboard. The query to generate this data outlined a number of conditions through the WHERE clause. These were utilised to assure the most effective query request was executed. The conditions included the following:

- Specified module in question.
- Attendance data to be retrieved. For example, for the query which derived the attended data, the conditions will outline studentattendance.AttendancestatusID to be either 1 (attended) or 3 (late).
- Dates specified between the current data and a 7-day substrate interval. The SQL function DATE_SUB() will return the values between the two dates.

When executed, the query initially retrieved the data in two array forms. In order to resolve this issue, the developer employed an alias to define the single data value, "Attended". Once the query was executed, the results were stored in a variable. This could then be used to plot the data into the pie charts. Each variable created was associated with the variable $i to uniquely identify the data

retrieved. As shown in figure 5.15 section 1, $i is set to 0 before the loop begins and increased by one at the loop ends shown in figure 5.15 section 2.

```
<div class="row">
  <?php
    $i = 0;
    foreach ($results as $result):?>
      <a href="lecturermoduleattendance.php?moduleID=<?= $result['ModuleID']?>">
      <div class ="col-sm-4 col-sm-offset-1" style="height:40px; padding-bottom: 230px;">
      <canvas id="<?= $result['ModuleID']?>"></canvas>
      <?php
      $mid = $result['ModuleID'];

      $sqll = "SELECT COUNT(StudentAttendance.AttendanceStatusID) AS Attended
              FROM StudentAttendance
              INNER JOIN Event ON Event.EventID = StudentAttendance.EventID
              INNER JOIN Module ON Module.ModuleID = Event.ModuleID
              WHERE Module.ModuleID = $mid AND StudentAttendance.AttendanceStatusID = 1
              OR StudentAttendance.AttendanceStatusID = 3
              AND EDate BETWEEN DATE_SUB(CURDATE(), INTERVAL 7 DAY) AND CURDATE()";
        $stmt = $pdo->prepare($sqll);
        $stmt->execute();
        $attended[$i] = $stmt->fetch()['Attended'];
```

1

```
          $i++;
        ?>
```

2

Figure 5.15: Retrieve attendance data for dashboard (lecturerdashboard.php)

Chart JS

For this segment of the project, JavaScript was used in order to create the dynamic and responsive pie charts with attendance data. JavaScript is a scripting language which enables a highly responsive interface with dynamic functionality. ChartJS is a JavaScript library which enables the developer to generate data visualisation through charts on the client-side. Figure 5.19 illustrates the creation of the pie charts. The pie chart is adapted from labels and titles through to the colour, front, size and style. In order to be correctly references, each pie chart must be uniquely identified. The <canvas> tags are embedded within the code in Figure 5.15 section 1 and identified by each moduleID. This ID was used to reference each canvas tag when adapting the pie chart, shown in Figure 5.16. The two data results retrieved from the SQL query were imported into the JavaScript code. The GUI is shown in Figure 5.17.

```
<?php
$i = 0;
foreach ($results as $result):?>
<script>
  var myChart = document.getElementById('<?= $result['ModuleID'] ?>').getContext('2d');

  Chart.defaults.global.defaultFontFamily = "Helvetica";
  Chart.defaults.global.defaultFontSize = 18;
  Chart.defaults.global.defaultFontColor = '#777';
  var attChart = new Chart(myChart, {
    type:'pie',
    data:{
      labels:['Attended', 'Unattended'],
      datasets:[{
      label:'Attendance',
      data:[ <?= $attended[$i] ?> , <?= $unattended[$i] ?>],
      backgroundColor: [
      '#22533D',
      '#1C9E32',
    ],
    borderWidth:1,
    borderColor: '#777',
    hoverBorderWidth: 2,
    hoverBorderColor: '#000',
      }]
    },

    options:{
      title:{
        display:true,
        text:'<?= $result['ModuleName'] ?>',
```

Figure 5.16: JS for pie chart (lectuerdashboard.php)

Figure 5.17: GUI for pie chart

5.3.3 Add a Sessions (Workshop or Lecture)

A proposed functional requirement for the SAMS application was to create a new session e.g. lecture or workshop. This request is submitted by the lecturer who is not only able to create a new session, but able to edit and delete sessions also. Furthermore, the client emphasised upon an efficient system to the current paper-based system. Therefore, the solution proposed outlined that when a new session is created, an attendance list of students will be created in conjunction using the event ID as a reference.

Building a query string

Typically, the user will navigate to the appropriate module in order to create a new session. In this process a query string is built which appends the chosen ModuleID on the end of it, as shown in Figure 5.18. This is necessary to associate the event to the correct module, when redirecting the user.

```
<a class="btn btn-success" href="lectureraddevent.php?moduleID=<?= $moduleid ?>">
  <span class="glyphicon glyphicon-plus"></span>Add New Session
</a>
```

Figure 1.18: href Query String (lecturermoduleview.php)

Forms

Forms are generally used by the administrator to set up the various components of the SAMS application. This is crucial for the core functionality of the system. These include the following:

- Create, Review, Update and Delete all Users.
- Create, Review, Update and Delete Departments.
- Create, Review, Update and Delete Courses.
- Create, Review, Update and Delete Modules.
- Reset User passwords.

However, in the case of this workflow, the Lecturer completes a form to "Add a Session". This was one of the more difficult forms to implement as it required the details of the chosen Module in order to be submitted into the database correctly. In addition, all forms consisted of the same design lay out in order to create a consistent and assertive design throughout. Forms are surrounded by <form> tags which are crucial for information to be processed correctly. In order for the information within the form to submit to the correct processing page, the first form must include a method and action, as seen in figure 5.19. The method will specify how the data will be sent over, by either GET or POST. Both methods have their advantages and disadvantages, outlined in *appendix R*. In this case the POST method has been used to ensure sensitive data is not visible to users or attackers, this method is overall seen to be more secure than GET. Each form will contain input fields in order to obtain data from the user. In the SAMS web application, each input field follows a similar structure as demonstrated in Figure 5.20. The database follows constraints which ensures that when data is inserted there are no NULL values. To enforce this provision, the boolean attribute 'required' is present in the input field to ensure that the field is filled out before submitting the data, shown in figure 5.21.

```
<form method="POST" action="editstaff.php">
```

Figure 15.19: Form tags with method and action page (adminaddevent.php)

```
Event Title:<br>
<input type="text" class="form-control" name="ruename" id="rueventname" size="40" required>
<br>
```

Figure 15.20: Input field structure (adminaddevent.php)

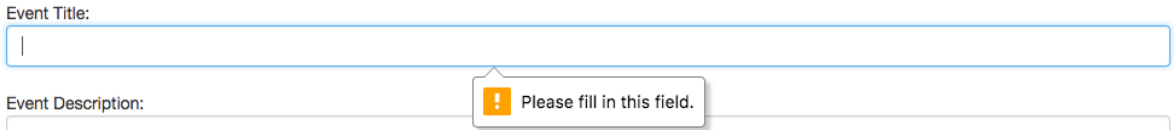


Figure 5.21: 'required' attribute present in input field (adminaddevent.php)

Furthermore, <input> elements can also be of the type "hidden". Hidden input fields allow the developer to include data which cannot be seen through the Graphic User Inter (GUI). This attribute is typically used to process and track database records which are created, updated or deleted when a form is submitted. However, hidden input fields can be tampered with creating a security hole in the system. Shown in figure 5.22, the associated ModuleID is passed through as a hidden input to the processing page. This data is used to not only create a new session but generate an attendance

list of students. This process will be discussed further in the report with additional security precautions.

```
<input type="hidden" name="ruemodule" value="<?= $moduleid ?>">
```

Figure 5.22: href Query String (adminaddevent.php)

Figure 5.23 demonstrates the drop-down menu populated with data from the database. A SELECT query is executed through a WHILE loop iterating through each row in the database until there are no remaining data entries. The event description is displayed to the user in the drop-down menu (figure 5.24), whereas the event TypeID is iterated as the VALUE of the row. The pre-set value enables clear navigation to the user, whilst the back-end prepares the processing and inserting of a new session into the database.

```
Event Type:<br>
<div class="form-group">
  <select name="ruetype" class="form-control">
      <?php
          $stmt = $pdo->query("SELECT *
                        FROM EventType");
                        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)):
      ?>
        <option value="<?= $row['EventTypeID']?>"><?= $row['EventDescription']?></option>
     <?php endwhile; ?>
    </select>
  </div>
```

Figure 5.23: While loop to display dropdown list in the form (adminaddevent.php)

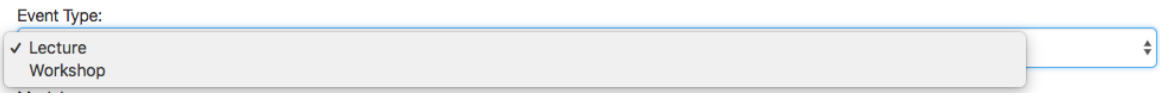


Figure 5.24: Drop down menu show in the user via GUI (adminaddevent.php)

Database Default Value

As the task of creating a new session also concurrently generates a list of students who will be attending the session it was important to devise an appropriate execution of both requirements. A solution was proposed ensuring minimal effort for the user. A default value is set each time a student attendance list is generated, which would display a pre-selected option on the GUI to the user. The AttendanceStatusID for the Student Attendance table was therefore defaulted to "2" (absent), as shown in figure 5.25. Overall, each time a session is created an according attendance list is generated, which will now be discussed in further detail.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	AttendanceID	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	UserID	int(10)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	EventID	int(10)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	AttendanceStatusID	int(11)			No	2		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Figure 5.25: Default Value set in StudentAttendance table within the database

Inserting data and placeholders

When the form has been submitted, the data is ran through a parametrised query, as seen in figure 5.26. However, before this can occur an IF statement is implemented to check that an input field has been set to avoid NULL data entries. Furthermore, IF statements are great when evaluating a condition and returning a Boolean value; true or false. If the test is true, then the body is executed,

else if the test equates to false the body is by passed. When generating the SQL query ready for execution, the table attributes are matched with placeholder values. This is a provision implemented by the developer to sanitise the data before insertion to avoid the interception of malicious code from attackers. This statement then executes an array of values retrieved from the form via the $_POST method and inserts the new data into the database. Figure 5.26 demonstrates the creation of a new session with the additional precautions taken to enforce a higher degree of protection from attack.

```php
<?php
 session_start();
 require_once "connection.php";

if (isset ($_POST['ruename']))
  {
$sql = "INSERT INTO Event (Title, Description, EDate, EventTypeID, ModuleID)
        VALUES(:ename, :edescription, :edate, :etype, :emodule)";
        $stmt = $pdo->prepare($sql);
        $insert = $stmt->execute(array(
          ':ename' => $_POST['ruename'],
          ':edescription' => $_POST['ruedescription'],
          ':edate' => $_POST['ruedate'],
          ':etype' => $_POST['ruetype'],
          ':emodule' => $_POST['ruemodule']));
           }
```

Figure 5.26: Insert Query using placeholders for form (addevent.php)

lastInsertID()

After a new session is created by the lecturer, a second segment of code is run to generate a student attendance list for the new session. In order to execute this correctly the EventID of the session must act as a foreign key. This was executed by the utilising a LastInsertId() function to retrieve the EventID from the newly created session, as shown in Figure 5.27. LastInsertId() returns a string value which represents the row ID of the last row inserted into the database. The assumption was made that one Lecturer will be creating a new event at one time. When the generated SQL query was prepared the last inserted ID was utilised as a condition for the query. The statement retrieves the UserID of the students whom are assigned to the associated module and the recently created event. The results were stored into the variable $results and ran through a foreach loop executing each result and inserting each student record with the default Attendance status as 2 (absent), as shown in figure 5.28. Figure 5.29 displays the newly submitted attendance list on the GUI, while Figure 5.30 illustrates the submitted data in the database. Although, this procedure was successfully implemented a great deal of time in the project scope was taken to create a solution which is best suited to the client's needs.

```php
$moduleid = $_POST['ruemodule'];
      if ($insert === true) {
      $last_id = $pdo->lastInsertId();
```

Figure 5.27: lastInsertID function executed (addevent.php)

```
$calllist = "SELECT User.UserID
             FROM `User`
             LEFT JOIN StudentModule ON StudentModule.UserID = User.UserID
             LEFT JOIN Module ON Module.ModuleID = StudentModule.ModuleID
             LEFT JOIN Event ON Event.ModuleID = Module.ModuleID
             WHERE Module.ModuleID = $moduleid AND Event.EventID = $last_id";
      $slist = $pdo->prepare($calllist);
      $slist->execute();
      $result = $slist->fetchAll();

    }

     foreach ($result as $studentlist)
      {
       $sql = "INSERT INTO StudentAttendance (UserID, EventID)
               VALUES (:userid, :eventid)";
       $stmt = $pdo->prepare($sql);
       $insert = $stmt->execute(array(
             ':userid' => $studentlist['UserID'],
             ':eventid' => $last_id));
      }
```

Figure 5.28: Inserting Student Attendance List into database (addevent.php)

Username	First Name	Surname	Attendance
k1111115	John	Albert	Present Absent Late
k7687285	Tisa	Cahill	Present Absent Late
k5021089	Abbey	Jeffrey	Present Absent Late
k7798231	Felica	Jennings	Present Absent Late

Figure 5.29: Newly created attendance list with default value shown via GUI

AttendanceID	UserID	EventID	AttendanceStatusID
1639	21	49	2
1640	125	49	2
1641	832	49	2
1642	306	49	2
1643	1332	49	2
1644	546	49	2
1645	845	49	2
1646	547	49	2
1647	1337	49	2
1648	312	49	2
1649	549	49	2
1650	1343	49	2
1651	312	49	2
1652	549	49	2

AttendanceID	UserID	EventID	AttendanceStatusID
1645	845	49	2
1646	547	49	2
1647	1337	49	2
1648	312	49	2
1649	549	49	2
1650	1343	49	2
1651	312	49	2
1652	549	49	2
1653	1343	49	2
1654	658	49	2
1655	295	49	2
1656	1319	49	2
1657	836	49	2
1658	298	49	2
1659	1320	49	2

Figure 5.30: Inserted for student attendance list (defaulted to = 2)

5.3.4 Submit Attendance

After creating an event and generating the corresponding student attendance list, the lecturer is able to take and submit attendance. Figure 5.30, demonstrates the typical graphic interface in which the lecturer will take attendance upon. Initially, the interface displays all students defaulted to absent ready for the lecturer to modify the attendance. This is to minimise the time lost in selecting individual attendance options for each student. In turn the lecturer is able to focus on lesson

planning and effectively chasing students who are disengaging from the module. However, before this can be accomplished the correct attendance data must be submitted into the database.

Radio Buttons and generating unique values

In order to submit attendance, the use of radio buttons have been utilised as shown in Figure 5.30. There were many different options in which could be selected such as drop-down menus or check boxes. User experience expert, Jakob Neilson, explored the benefits and issues in all three display methods. Nielsen declared that radio buttons are a clear and distinct presentation to users and are more comprehensive than the likes of check boxes. In addition, radio buttons require a “lower cognitive load” from users as all options are permanently visible and easily compared. Furthermore, Neilson confirmed that including a defaulted option “enhances users' ability to predict what a control will do and how they'll operate it” (Nielson, 2004).

Each radios button must carry a distinct “name” attribute in the <input> tags as seen in Figure 5.31. Figure 5.31, illustrates the creation of a new variable called $varname, which includes the string “attstatus” appended with the UserID of the student, creating a unique value for the name attribute for each row. This is then submitted to the associated processing page to submit the modified attendance data to the database.

```
<?php
 $user = $row['UserID'];
 $varname = "attstatus" . $row['UserID']
 ?>
   <?php
    if($row['AttendanceStatusID'] === "1"){ ?>
         <div class="text-center">
             <label class="btn btn-success">
                     <input type="radio" name=<?= $varname?> value="1" autocomplete="on" checked><span class="glyphicon glyphicon-ok"></spar
    </label>
```

Figure 5.31: Radio Buttons utilised in the form (lecturerregister.php)

Rebuilding an associative variable

The code shown in figure 5.32 is composed between <form> tags. When the attendance is submitted, to process the data the variable which was first generated for the “name” attribute must be re-created. Therefore, the developer reconstructed the query which was executed and shown to the lecture (list of student attendance). The condition outlined in this query was to order by surname ascending, this will generate the order to be the same throughout both pages. Then was to recreate the same variable with the string and appended UserID as shown in Figure 5.32. The attendance status was then posted from the previous page and inserted into the database on the conditions of Student ID and Event ID.

```
$stmt = $pdo->query("SELECT StudentAttendance.UserID, StudentAttendance.AttendanceStatusID
                     FROM User
                     INNER JOIN StudentAttendance On StudentAttendance.UserID = User.UserID
                     WHERE StudentAttendance.EventID = $eventid
                     ORDER BY User.Surname ASC");
while ($row = $stmt->fetch(PDO::FETCH_ASSOC))
      {
        $userid = $row['UserID'];
        $varname = "attstatus" . $userid;
        echo "<br>" . $varname;
        $attendancestatus = $_POST[$varname];
        echo " " . $attendancestatus;

            $sql = "UPDATE StudentAttendance
                     SET AttendanceStatusID = $attendancestatus
                     WHERE StudentAttendance.userID = $userid and StudentAttendance.EventID = $eventid";
            $stm=$pdo->prepare($sql);
            $insert=$stm->execute();
```

Figure 5.32: Rebuilding the variable (attendance.php)

Although this method works effectively for the task at hand, there are other techniques which would increase the overall efficiency, system performance and user experience. Using the likes of AJAX and JQuery combined to retrieve the data, there would be less strain on the servers with request/response requests. This could have been implemented by creating the radio buttons as demonstrated in figure 5.32, but surrounded by a with a referring id to target with JavaScript. These values will be sent and processed with an on click function in JavaScript embedded by the according Ajax. The input value from the user could have sent an Ajax request to another page to process the data accordingly. This processing page will need to have database connection through the global $pdo variable and the correct insert statement. These changes can be made in real-time with little strain on the servers to re-process the whole page. However, due to limited time this method could not be implemented in the project framework.

5.4 Security Concerns and Precautions

When implementing a system with a vast range of users and a sustainable amount of sensitive data, it is important to consider the security implications and precautions to undertake to prevent attack. There are numerous tools utilised in this project which can leave much room for attackers to cause havoc upon the system. Therefore, it is crucial to implicate additional security measures to minimise threat of attack. One of the most well-known forms of security exploitation is SQL Injection. This is an attack upon a system, typically through form input fields, inserting mischievous SQL statements to cause harm to the system or its data. This segment of the report will explore how the developer has attempted to secure the system from this attack through parameterised queries. Furthermore, as extra layer of security Secure Sockets Layer (SSL) protocol has been enforced to prevent attack by encrypting the code as it is sent between servers. Both these safety measures are implemented after the appropriate risk management was undertaken to highlight areas of vulnerability.

5.4.1 SQL Injections

Chapter 3 explores the various laws and legislations which protects data subjects and their personal data. The Computer Misuse Act (CMA) 1990 outlines access of a computer system without proper authorisation with the intent to cause nuisance or damage. Although these laws embark a security blanket upon society there are extra provisions that should be acknowledged and implemented in the case of attack. SQL injection is a code injection attack where the attacker executes malicious SQL statements to gain access to resources or manipulate data in the database. This type of security exploit can be used to drop vital database tables or even retrieve sensitive data. This vulnerability is one of the oldest security invasion techniques generally undertaken through an input field. However, there are many provisions which can be used to minimise this kind of attack.

Prepared SQL statements and placeholder values have been utilised to ensure the data is cleansed before being submitted to the database, as seen in Figure 5.33. These are known as parameterised queries. Parameterised queries contain placeholders which are used as parameters. The values are then supplied to the placeholders at the time of execution. Each placeholder value is iterated through an array and inserted into the database. The POST method is used to retrieve the information from the html form as it is a secure alternative to the GET method which sends data visibly through URL.

```
$sql = "INSERT INTO Event (Title, Description, EDate, EventTypeID, ModuleID)
        VALUES(:ename, :edescription, :edate, :etype, :emodule)";
        $stmt = $pdo->prepare($sql);
        $insert = $stmt->execute(array(
          ':ename' => $_POST['ruename'],
          ':edescription' => $_POST['ruedescription'],
          ':edate' => $_POST['ruedate'],
          ':etype' => $_POST['ruetype'],
          ':emodule' => $_POST['ruemodule']));
```

Figure 5.33: Execution of data by parametrised queries (attendance.php)

5.4.2 SSL protocol

Secure Sockets Layer (SSL) is a cryptographic protocol which provides a secure channel of communication between two devices operating over the internet. This can also include an internal network such as the communication between a web browser and its server. Subsequently the usual HTTP address is modified to HTTPS, with the 'S' signifying 'secure'. After vast research was conducted on the proper provisions for security outlined in the literature review, this project has established the HTTPS protocol. This is demonstrated through the browser to users as a padlock symbolising the enhanced security, as shown in Figure 5.34.

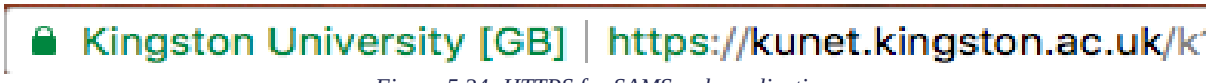


Figure 5.34: HTTPS for SAMS web application

With the new General Data Protection Regulation permitting higher degree of protection for personal data from the 25th May 2018, it is crucial to implement the SAMS web application with a secure communication platform. The SAMS database aims to hold a sustainable amount of sensitive data and therefore a high risk and target for attack. SSL complies with the following security principles:

- Protected data transmission through encryption e.g. browser to server. SSL inserts random characters embedded into the original data, and therefore is not comprehendible to an attacker.
- SSL certificate provides authentication ensuring that all data transmitted and received is connected to the correct server without being intercepted. This includes data integrity; all data requests are delivered successfully.

However, performance is a major disadvantage of the SSL protocol. As all data must be encrypted and submitted more time is take than the usual HTTP requests. Nevertheless, the advantages outweigh the disadvantages tremendously as it is critical not only for the system security but to abide by UK law to protect data.

5.4.3 Conclusion

An unprotected website or application is at a high risk of exposure and loss of capital. When dealing with a great deal of sensitive data of an individual it is crucial to ensure the necessary provisions are undertaken to minimise threat of attack. Through the application the developer has ensured when processing data parametrised queries and placeholders are utilised to cleanse user input before directly inserting into the database. In addition, by encoding the processing information when submitted by using SSL protocols the system is able to abide by regulations inflicted by the UK. However, these provisions still permit security loop hole throughout the system. By utilising additional security provisions such as using web application firewalls attack can be monitors and highlight areas of vulnerability. These sophisticated tools can filter potential dangerous web requests with additional SQL injection defences to catch and prevent most attempts of attack.

5.5 Conclusion

To conclude, the implementation chapter outlined the process in which was undertaken to build the SAMS application with the knowledge gather from the previous chapters. Due to conducted research, many of the desire component to build the system fit together smoothly, however there was many challenges along the way which cause for vital time to be lost in the project schedule. To reflect, the chosen tool and technologies were utilised effectively to create the SAMS web application. The extensive online help and support that is accommodated the programming languages selected meant that the implementation issues and problems were able to be quickly solved to create a dynamic system. In addition, the wireframes found in chapter four were mirrored

upholding a consistent design throughout. However, the closely coupled front-end and back-end coding proved difficult for the developer to refractor the code when modifications were required. This could have been avoided by decoupling the code through a framework such as Model View Controller (MVC). The framework provides an elegant manner in which to format and present the code, creating a more maintainable product. In addition, this would have enabled the developed with a deeper understanding of how the components interconnected. The knowledge gathered from the literature review, enabled the development to implement additional security precautions to not only increases the protection around the sensitive data involved but to comply with the legislations of the UK law. However, there is still much room for improvement, but due the project schedule there was not enough time to implement all which was researched. The next stage is to validate the requirements requested by the client for the users, through testing and validation. This is to ensure the desired product has been delivered, through debugging and testing function requirement

6 Testing and Validation

So far, the project lifecycle has iterated through serval phases from the requirement analysis through to initial implementation of the SAMS web application. Each stage has collated crucial information and knowledge in order to research, design and build the end project to a high standard. However, before the final product can be showcased to the client there is the remaining testing segment. Testing is an essential process during the project lifecycle as it highlights any defects or errors made during the development phases. Thus, to ensure the SAMS web application is fit for purpose several tests have been carried out to explore and evaluate the different aspects of the application. Furthermore, it is vital that the decisions carried out throughout the project scope are validated to ensure the correct product has been created. Therefore, as a result the following questions arose and were validated by the developer through testing:

- Do the Functional Requirements of the system meeting the user's needs?
- Is the design of database structured correctly to retrieve the correct data?
- Is the database durable enough to hold the data required to take the system to industry?
- Is the implementation correct against the user and system requirements?
- Are the Wireframes the best design for the system to reach the users goals and is the implementation efficient and user friendly enough?

Overall, the testing strategy that will be utilised will aim to answer the above questions surrounding function and non-functional requirements, database construct, implementation and lastly a usability test to ensure that the different aspects of the application is functioning correctly prior to further development.

6.1 Validation of Functional Requirements and System Needs

When constructing a system, the underlying foundations and the systems purpose stem from the requirements. Once all requirements have been met, the developer can create a successful product by the client's request. Therefore, requirement validation is a critical activity for product success. The process of requirement engineering was undertaken in chapter two. Requirement engineering involves discovering, capturing and analysing requirements and their constraints. The validation of these requirements can be complete through the client and users. Furthermore, if requirements are vague or unclear it can hinder the process of the project, thus resulting in delay or failure. Therefore, it is crucial to understand the standard which is required from the system by the client and other stakeholders, through requirement validation.

In order to validate the functional requirements for the system, the developer conducted regular meetings with the client to elicit the information on how the system will perform. The client was able to project the ideas in which they have for the proposed system. In turn, these meetings enabled the client to understand what they wanted and whether it was deliverable from the developer's end. In reflection, the on-going process included understanding the need of the system and what the users should be able to do, in turn the client and the developer were able to discuss whether it was deliverable from a technical perspective. In addition, further research was conducted with the potential users of the system. Interviews and questionnaires were conducted to gain a deeper insight on what the users expected from the system. In this process, the client will confirm or reject the requirements for the proposed system. As a result, a filtered list to the essential requirements for system was generates and agreed upon with the client. The final functional requirements, including a MoSCoW analysis was conducted within chapter two. However, due to the large project scope and time limitation, not all requirements are able to be implemented.

6.2. Validation of the Database

6.2.1 Entity-Relationship Design and Database Construct

The SAMS web application will be significantly database reliant. Therefore, the process of eliciting requirements for the system has influenced the database design. This enabled will the developer to create an appropriate Entity-Relationship Diagram (ERD). An ERD is a graphic representation of entities (object or concept) and their relationships with one another within the proposed system. Ultimately, an efficiently designed ERD will resolve any conflicting associations, by introducing intersecting classes. A list of SQL queries based on the functional requirements the client agreed upon have been generated to test the validity of the database. Table 6.1 demonstrates the test undertaken, including the execution of the functional requirements, retrieving the data desired and weather it was successful or unsuccessful. As many of the requirements were similar, following the CRUD (Create, Review, Update, Delete) technique, only a selection of essential requirements was tested. *Appendix S* demonstrates the full test results. The results were presented to the client, to ensure the result retrieved was correct against the functional requirements

#	Functional Requirement	SQL Query	Pass/Fail
1	Create User	INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID) VALUES ('K9999999', 'password', 'Paul', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')	Pass
	1 row inserted. Inserted row id: 1774 (Query took 0.0199 seconds.) INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID) VALUES('K9999999', 'password', 'Paul', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')		
2	View Student	SELECT Roles.Description, User.UserID, User.Username, User.Password, User.Firstname, User. Surname, User.Gender, User.DateOfBirth, User.ContactNo, User.EmailAddress, User.LastLogin, Course.CourseName, Course.CourseCode FROM (((`User` INNER JOIN Roles ON Roles.RolesID = User.RolesID) LEFT JOIN StudentCourse ON StudentCourse.UserID = User.UserID) LEFT JOIN Course ON Course.CourseID = StudentCourse.CourseID) WHERE Roles.RolesID=1 ORDER BY User. Surname ASC	Pass
	Showing rows 0 - 24 (293 total, Query took 0.0018 seconds.) [Surname: **ALBERT - CABRERA**] SELECT Roles.Description, User.UserID, User.Username, User.Password, User.Firstname, User.Surname, User.Gender, User.DateOfBirth, User.ContactNo, User.EmailAddress, User.LastLogin, Course.CourseName, Course.CourseCode FROM (((`User` INNER JOIN Roles ON Roles.RolesID = User.RolesID) LEFT JOIN StudentCourse ON StudentCourse.UserID = User.UserID) LEFT JOIN Course ON Course.CourseID = StudentCourse.CourseID) WHERE Roles.RolesID=1 ORDER BY User.Surname ASC		

Table 6.1 SQL queries for Functional Requirements

6.2.2 Database Constraints

Data integrity is a fundamental component of information security. Data integrity refers to the accuracy and consistency of data stored in a database. Data integrity is imposed within a database through error checking and validation routines, known as constraints. It is essential to implement a database with the appropriate constraints to ensure the long-term survival of the system. During the design phases in chapter four, the process of creating a data dictionary was conducted. A data dictionary utilises the information gathered in the ERD to complete the design with the correct data objects, attributes, referential relationships and potential constraints. Therefore, once implemented it was vital to test the constraints in place to validate the database design and ensure data integrity. There are three different types of integrity constraints which could be tested:

1. **Entity integrity**: Entity integrity ensures each table row has a unique and non-null PK.
2. **Referential integrity**: This is the association between tables i.e. FK relationship. Each FK must match a primary key value in the associated table.
3. **Domain integrity**: Domain integrity is a set of unique values permitted for an attribute. For example, if an attribute is set to INT then a string value cannot be accepted. Likewise, if an attribute is set to NOT NULL then these conditions cannot be violated through an empty data entry.

The database was interrogated with a number of SQL queries to test the validity of the design and the data integrity. A reliable database will ensure that constraints are upheld at all times. Table 6.2 illustrates a segment of the results from this permitted test. The results concluded that all necessary constraints were set up correctly. Only a segment of the data was tested, as similar process would be conducted on all tables and would consume valuable development time. The full testing can be found in *appendix T*. Additionally, further testing was conducted to test the database robustness and query optimisation, found in *appendix U*.

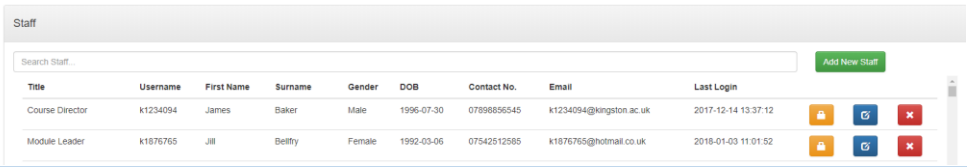
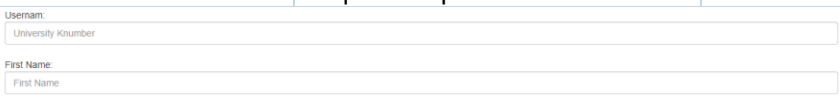
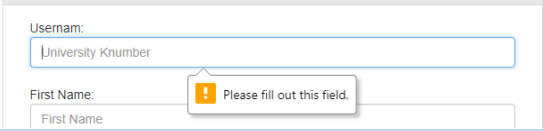
	Test Description	SQL Statement	Integrity	Result
1	Changing a PK to a value which already exists	UPDATE User SET User.UserID = 13 WHERE User.UserID = 14	Entity	Pass
	Error SQL query: UPDATE User SET User.UserID = 13 WHERE User.UserID = 14 MySQL said: #1062 - Duplicate entry '13' for key 'PRIMARY'			
2	Entering a FK value which doesn't match the values in the parent table.	UPDATE StudentAttendance SET StudentAttendance.AttendanceStatusID = 4 WHERE StudentAttendance.UserID = 21 AND StudentAttendance.EventID = 3;	Referential	Pass
	Error SQL query: UPDATE StudentAttendance SET StudentAttendance.AttendanceStatusID = 4 WHERE StudentAttendance.UserID = 21 AND StudentAttendance.EventID = 3 MySQL said: #1452 - Cannot add or update a child row: a foreign key constraint fails (`db_k1403062`.`StudentAttendance`, CONSTRAINT `dad` FOREIGN KEY (`AttendanceStatusID`) REFERENCES `AttendanceStatus` (`AttendanceStatusID`))			

Table 6.2 SQL queries testing database integrity

6.3 Requirement Testing

6.3.1 Functional Requirements Black Box Testing

Black box testing, also known as functional testing, is a software testing technique focusing on the functional requirements of the system. The test procedure emphases on the input and output of the system rather than the internal components and processes. Therefore, such tests do not focus on how the system produces the output, but instead weather the desire output is achieved. In order to test whether the product created is correct, a test to see if the functional requirements is conducted to validate the process so far. As the functional requirements of the system were not so clear cut, test cases were created around each requirement. These test cases evaluate whether the appropriate output is provided from the user input. Table 6.4 demonstrates the results from test cases surrounding the functional requirement "Create User" derived from the analysis phase. Alongside the results from the testing, an image to complete each test case demonstrating the different steps that may be undertaken to thoroughly evaluate the application. It has been assumed in some test cases that the user is already on the desired page to conduct their task. The functional requirement has been enlisted alongside each test case(s). Table 6.3 illustrates a single tested functional requirement test cases, the remaining test cases can be located in *appendix V.*

Test ID	FR	Test Description	Expected Result	Result (Pass/Fail)
1	Create User	1. User clicks 'Manage Staff/Student'	System displays further options to relate to users.	Pass
2		2. User clicks 'View Staff/Student'	System displays all records of users with user details.	Pass
				
3		3. User clicks 'Add new Staff/Student'	System redirects user to a page which displays a form with the required input fields.	Pass
				
4		4. User leaves a blank input field.	System displays alert notifying user that empty input field is required.	Pass
				
5		5. User enters string of characters into email address field.	System displays alert notifying user that '@' is required in the input field.	Pass

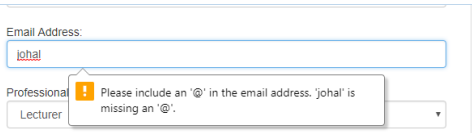
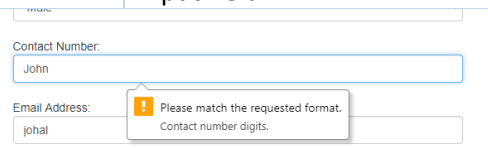
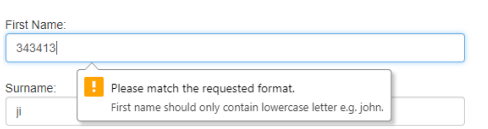
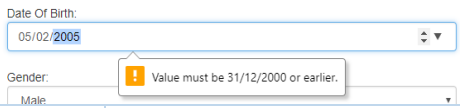
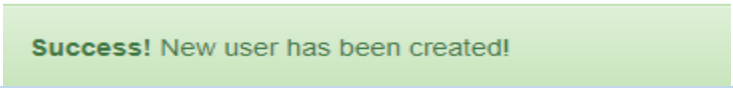
				
6		6. User enters text into contact number field.	System displays alert notifying user that text cannot be entered into a contact number input field.	Pass
				
7		7. User inputs numbers into First Name field.	System displays alert notifying user they must enter characters in the required input field.	Pass
				
		8. User inputs date of birthday greater than the year 2000		
				
8		9. User successfully submits form.	System displays success message.	Pass
				

Table 6.3 Functional testing example

It was essential to ensure that the input fields accepted the correct input, therefore prompts and alerts were implemented to prevent the user form entering incorrect characters and giving the user clear guidance when errors are made. Furthermore, in order to ensure a good user experience, it was vital to ensure the user knew that there was a clear path for their goals. This included preventing ambiguously labelled navigation or form fields. Success messages are extremely important to indicate the end of a user goal and to confirm system behave.

By testing the different scenarios and test cases, this enabled the developer to recognise any system bugs to rectify before further implementation. In addition, by following the same design principles and patterns throughout the implementation, the test cases were often similar and able to be copy and pasted throughout the test. A strong advantage of these having a unified set of design principles, met that code could be re-used and saved crucial time for the developer. Nevertheless, as the same implementation approach was used across different aspects of the system and functionality it meant that the developer was able to quickly resolve any issues found in the system.

6.3.2 Non-Functional Testing

Non-functional requirements specify how the system will operate with given constraints on its functionality. The functional requirements for the SAMS web application are measured in five different categories: Security, Performance, Compatibility, Design and Usability. Therefore, this testing technique will demonstrate how the system operates and the standard of service offered. Such testing can give the client and users indication on how dependable the system is, which in turn can affect the user's trust towards the application. Similarly, to the function testing, several test cases have been generated surrounding each non-functional requirement area to evaluate whether the system meets the necessary criterions. Table 6.4 demonstrates a segment of the test results from this test the remaining of which can be found in *appendix W*.

Test ID	NFR AREA	Testing Description	Expected Result	Result (Pass/Fail)
1	Security	Check password field in the database.	All passwords should be hashed using the appropriate algorithm	Pass
2		Attempt to create a new user with an existing username.	System should decline users request to create new user with existing username.	Pass
3		Attempt to create a new user with non-registered username.	System should accept users request to create new user with non-registered username.	Pass
4		Access system using incorrect log in detail.	System should redirect user to the log in page displaying error message.	Pass
5		Access system using correct log in detail.	System should grant access, redirecting user to the correct homepage and setting up sessions.	Pass
6		User closes down browser whilst logged in.	System should log out user and redirect them back to the log in page.	Pass

Table 6.4: Non-functional testing example

The different test cases exhibit weather the system is truly secure from attack and to the users. There are some areas in which were unable to be tested due to the availability of resources and data a therefore failed the testing scenarios. Nevertheless, this leaves much room for improvement and directs the development to the next phase of the project and for further implementation.

6.4 Usability testing

Usability measures the quality of a user's experience and therefore is a critical factor to consider when developing a system. An efficient user experience can be achieved by "by taking a user-centred approach to design, and thus ensuring that the product incorporates characteristics that support usability" (Bevan, 2009). This usability test will evaluate the considerations and design decision in the previous chapters to project UX success. The proposed usability testing has complied with Common Industry Format (CIF) Standard for usability testing. It is a framework which enables an effective approach to collecting quantitative data to measure and indicate the success in user performance. There are five critical elements to this approach measures the success of a sufficient system and usability testing. As mentioned in the design chapter these are the following: learnability, efficiency, memorability, effectiveness and satisfaction. These metrics will enable a thorough evaluation of the SAMS web application through task completion and user performance.

The testing was being executed under controller circumstances with set instructions to ensure valid and enriched data for effective feedback.

In order to carry out the usability testing, four participants have been recruited as show in table 6.5. The desired participants went through a pre-screening examination to scout individuals which fit the persona of the user types for the system. This included demographic, occupancy and computer literacy skills of each participant. In addition, each participant was asked to rate their technical experience on the scale shown below. These provisions were in placed to assure the participants closely mirrored the desired users of the system. Due to time constraint more participants were unable to tested, therefore only one participant per task was performed and according data was gathered.

#	Full Name	Age	Sex	Occupancy	Technical experience	Role Upon SAM application
1	Sonia Purewal	34	Female	Teacher	3	Lecturer
2	Amandeep Panesar	21	Male	Student	4	Student
3	Ajay Bhullar	35	Male	Data Analyst	3	Course Director
4	Sophie Kaur	28	Female	Assistant Phycologist.	2	Administrator

Table 6.5: Participant information for usability testing

*Technical experience was classified according to the following categories. The test will be undertake on a desktop:

1. **Inexperienced**: User has no experience with technical gadgets at all, and has limited knowledge on how to use the device chosen.
2. **Beginner**: User is able to navigate their way around basic device functionality and are familiar with the general concepts of the device.
3. **Somewhat experienced**: User is a regular user of this device and knows the functionality well. User is also has limited knowledge on its extended functionality.
4. **Experienced**: User if very experience, with knowledge of coding languages and how to navigate effectively on any device or application.

Tasks

Each participant partook the role of a particular user type upon the system. The participants were given a set of tasks to complete, with a given time frame of fifteen minutes. A further fifteen minutes were dedicated to a post-test questionnaire and interview about their results. If the task is not able to be completed within the time frame, this would indicate task failure. Each task which was chosen was strategically planned to assure that the user was able to navigate across the interface. This further tested the metrics: learnability and memorability. Tasks are outlined in *appendix X*.

Results

Participant 4: Administrator

Task #	No. of assists	No. of Errors	Test Result	Time Taken (MM:SS)
1	0	0	Pass	00:23
2	0	2	Pass	03:46
3	0	1	Pass	02:34
4	0	1	Pass	00:47

Participant 3: Course Director

Task #	No. of assists	No. of Errors	Test Result	Time Taken
1	0	0	Pass	00:32
2	1	3	Pass	03:02
3	0	0	Pass	00:57
4	0	0	Pass	00:46
5	0	0	Pass	01:03

Participant 2: Student

Task #	No. of assists	No. of Errors	Test Result	Time Taken
1	0	0	Pass	00:14
2	0	2	Pass	01:07
3	0	0	Pass	01:13
4	0	1	Pass	00:56

Participant 1: Lecturer

Task #	No. of assists	No. of Errors	Test Result	Time Taken
1	0	0	Pass	00:22
2	1	0	Pass	02:57
3	0	0	Pass	00.56
4	0	1	Pass	00.47

The post-test questionnaire sheet and interview answers can be found in *appendix x*. The usability test was 100% successful from all participants, and there were no assists required. The observer was able to note down some observations to do with the metrics involved in the testing approach:

- **Efficiency**: Measured the completion of task. Task completed was a 100% success rate from all participants.
- **Learnability**: Although participants started slowly with navigation, they were able to quickly navigate through the different webpages as they became familiar with the structure. This is also evident with post-test questionnaire.
- **Effectiveness**: Each task was completed and therefore effective testing.
- **Memorability**: The user were tested from a variation of one to two tasks to see if they became familiar with the website. This was evident with the answers also given for the post-test questionnaire.
- **Satisfaction**: Through the post-test questionnaire all participants were satisfied with the over aesthetics of the website and ease of use.

It was also discovered that the design was suitable for all age groups with the diverse range in computer literacy skills. However, there were further recommendations suggested in order to improve the usability of the application. Users requested that a search bar be implemented to filter the search results when locating a user. This is a very beneficial addition, as it is well practices within industry in order to facilitate search results. In addition, users also recommend to accommodate a filter facility in order to read the data in the more appropriate way. Generally, each participant was satisfied with their overall user journey as they were able to complete all tasks in the allocated time frame. Therefore, it can be concluded that the design of the application is suitable for all user types and the desired user experience is fully accomplished.

6.5 Conclusion

To conclude, the testing chapter is one of the most important segments of the project phases. This phase of the project life cycle indicates weather the design and implementation decisions have been successful in creating the SAMS web application. In addition, these segments strongly focused on testing the development of the SAMS web application to ensure that the end product meets all requested functionality from the client. To ensure that these requirements to test were the most appropriate, the numerous interviews conducted with the client and users enabled the developer to gain a deeper understating of the system. As the functional and non-function requirements of the user are the core for this product, test cases were generated and tested surrounding each one. These tests ranged from the user goals to testing the security, performance, compatibility, design and usability of the system. Although not all features were successfully implemented, the product still showcased a durable system meeting the majority of the functional requirements. As much of the system is implemented and successfully tested, it can be assumed that the prototype for the application is complete. Furthermore, the application sustainability proved that apply the agile methodology and appropriate programming languages enabled the developer to create a suitable end-product within the time scale allocated.

7 Critical Evaluation

Throughout the journey of the project many achievements have been accomplished and obstacles conquered. The chapter will discuss and critically review the achievements, issues and recommendations for future work. The analysis will assist in highlighting the positive and negative aspects of different methodologies, techniques and tools utilised in the duration of the project. Therefore, the different sub-sections includes achievements, issues, critical analysis and future work. Each segment will be analysed through a project and personal perspective. The critical analysis will showcase a summary of the overall journey, alternatives and justifications.

7.1 Achievements

The following sub-section will explore what worked well when building the SAMS web application. It will uncover the different methodologies and tools to accomplish the requirements and system goals. The achievements will be evaluated at a project and personal scale.

7.1.1 Project reflection

As a result of this project, a functional and dependable application was developed, conforming to the vast majority of client requirements. Working closely with the client proved successful, in delivering and validating aims and objectives outlined in the introduction chapter. By correctly channelling the client feedback throughout the project, the developer was able to gain a deeper understanding of the project scope and delivery. The main focus of the application surrounded two aspects which were closely monitored and explored to ensure an efficient system: Security and UX. The literature review significantly focused on the laws and regulations governed by the UK, whereas the principles explored in the design chapter helped shape the system towards an efficient user experience.

The requirements analysis generated a strong list of requirements to cement the foundations of the project. The non-functional requirements sparked a concern in the security aspects of the system which lead to a successful in-depth literature review exploring and discovering the laws and regulating governing the UK. Through the research and insight into the different regulations, the developer was able to understand the requirements to create and provision a secure system whilst complying with UK regulations. As a result, the SAMS web application conforms to the General Data Protection Relegation (GDPR) which will come into power 25th May 2018. During the design process this knowledge was utilised to generate a robust database design. By successfully researching and organising the components required, the implementation process was extremely smooth with little complications. Furthermore, the design process successfully mapped the implementation of the system. It was essential that UX drove these aspects of the system before implementation to ensure that the project took an Interactive design approach. (Rogers, 2011) explains that an interactive design considers the user experience and journey before implementation is undertaken. With this in mind, the developer was able to conjure questions and principles in which to consider throughout the design and implementation process. Usability testing results concluded that all areas tested were able to be successfully achieved. Although all requirements were not able to be implemented into the system, the results came back successful demonstrating a robust and efficient system.

Agile methodology enforced a great communication and rapport between the client and developer. This methodology enabled an iterative and recursive development process throughout the project lifecycle. As a result, the chosen agile methodology complimented the project progress in many ways. The developer was able to capture the requirements and feedback from the client at each stage in order to improve the application further. Weekly scheduled meetings with the client created a platform to analyse and evaluate the project based upon the project schedule outlined in

the beginning of the project. Overall, the methodology aided the successfully progression of the project with informative feedback for a successful end-product.

7.1.2 Personal Reflection

By introducing a real client into the project, the developer was able to enhance project management skills. Furthermore, the developer was also able to develop a range of soft skills when communicating with the client and different stakeholders. By confirming a list of deliverable dates with the client, the developer was able to fine tune her multitasking and time management skills to deliver the project on time. In addition, the developer was able to enhance her skillset by experimenting with new languages such as PHP PDO and JavaScript. In turn, this knowledge was enriched by vast problem-solving and research skills to find the most appropriate tools and methodologies to meet the user's needs. Lastly, this project has significantly improved the formal writing ability of the developer including grammar, spelling and punctuation.

7.2 Problems Encountered and Limitations

Although the desired product was successfully developed, there are many areas which were proven difficult to conquer. As a result, compromises had to be conjured in order for the project to move on in the time frame omitted. This segment will explore the problems encountered and the impact it made upon the project.

7.2.1 Project reflection

The flow of the project was generally seamless, however there were areas in which were proven difficult to conquer. As a result, significant time was lost to implement and execute more functionality. Creating and manipulating the pie charts displaying a visual overview for attendance, was deemed one of the most difficult and time-consuming features of the project. Incorporating the use of Chart JS and embedded data from the database involved several components. JavaScript was a new language for the developer and therefore background research was conducted in order to understand the mechanisms involved. In order to place the chart within the code, <canvas> tags were required. A for each was utilised to repeat the code for each module that was assigned to the user. However, to be manipulated by the DOM, each <canvas> tag required its own unique ID. This was efficiently resolved by using the module ID to identify each <canvas>. This loop had to be recreated to manipulate structure and attributes of each chart also.

After creating each chart, devising the appropriate SQL statement was the most challenging stage. The developer had to construct an SQL statement which would also be embedded within the for each loop to represent each module. The SQL consisted of several inner joins of different tables. However, It was proven difficult when sequencing the conditions involved in the WHERE declaration. In addition, a single statement was not enough to retrieve the desired results; therefore, two similar statements were composed. Embedding and manipulating the different languages to work harmoniously with one another was the most challenging of all stages of the project. As the different languages were coupled so closely together it was difficult for the developer to interpret. The overall process to implement this particular part of the dashboard took more time than intended. This in effect, impacted the rest of the project time scale including the time for testing strategies.

As discussed earlier, the usability testing scenario involved four participants. This was due to limitations on time. Thus, different testing methods such as repetitive and comparative, could not be executed. It was difficult to attain an understanding the true usability issues as different demographics, occupations and the range in computer literacy were not a true data sample. It could be concluded that four people were not enough to test an application of this standard. In addition, the results gathered were not enough to represent the true sample size for the application audience.

However, Neilson argues that the best outcome can come from testing three to five participants over three studies. Neilson states that after a single test the knowledge of the tester on the usability of the design is increased by a third, therefore the participant's size is efficient (Nielsen, 2000).

7.2.2 Personal Reflection

Although the developer was able to enhance her skillset, the research and time invested in learning and manipulating languages was challenging and time consuming. As no framework to structure the code was used throughout the project, it was difficult to completely understand how the different languages operated. Due to the limited knowledge on coding languages, the developer did not have enough knowledge to implement such approaches. As the code implemented within this project was vastly learnt within the last few months there was little time to follow an additional framework also. As a result, all code was closely coupled together portraying difficult to read. In addition, much time was awarded perfecting areas or the project which did not require the functionality. As a result, the developer had to compromise in other areas as time began to run short. This impacted the user experience in some cases. Nevertheless, the usability test results demonstrated that all task had a 100% success rate. Furthermore, eliciting and capturing the requirements from the client and users was interpreted differently by the developer at stages. This meant that many times the certain areas had to be clarified and chased up with each stakeholder to truly understand the journey for each user. This dynamically effected the time scale on the implementation process.

7.3 Critical Analysis and Future work

Due the limitations, the project was not able to be fully accomplished. However, along the journey the developer was able to gather knowledge to improve the standard of the application. As a result, the following recommendations have been composed in order to improve the exiting product:

1. Use MVC framework: As the developer was new to many of the languages and principles within this project, the code was difficult to maintain and sustain when implementing. Model View Controller (MVC) is a framework which enables the code to be interpreted in three interconnected components to increase the maintainability of code. Employing such frameworks can demonstrate the code in a more readable, organised and elegant manner.
2. Use PHP functions to hash passwords: MD5 is currently implemented into the application. However, through the knowledge gathered along the way, it has been noted that it is extremely dated and insecure. Therefore, by alternatively using PHP function such as password_hash() and php_verify() can create a strong one-way hashing algorithm to secure the system.
3. Responsive to multiple: As time became short towards the end the project, the developer was not able to create an application fully responsive for all devices. (Deloitte UK, 2018) concluded that students aged 18-24 use their mobile phones more than any age. In addition, they stated the best way to target student is through mobile applications, especially when dealing with higher education. Therefore, this can improve the overall UX.
4. Utilising client-side technology: Incorporating the use of AJAX and JSON webs services and enhance the usability and process speed. The limited JavaScript functionality within this project is not enough to create a dynamic and sustainable interface for such a large user base. Utilising AJAX and JSON will minimise the weight load on the server and create instant and real time data without refreshing the entire page.
5. Destroying Sessions: By destroying sessions after a certain period of inactivity, it will increase the security aspects of the application.
6. Implementing a dynamic in class attendance feature: By creating automatic in class attendance method, will mean that less administrative processes for the lecturer and the application is able to capture real time attendance. For example, generating random sequence of numeric characters and letters where students will enter into the application

once the session becomes “active” can make a more effective application for all stakeholders.

Recommendations one to five could improve the overall security, maintainability and sustainability of the application. However, as these issues were discovered late the process could not be reconstructed with the time remaining. Recommendation six will enforce a centralised system where all aspects of attendance monitoring will require minimal effort.

7.4 Conclusion

In conclusion, the project as transpired into an extremely valuable product. The effective communication between the developer and client reflected upon the project success. In addition from a personal reflection, a vast range of skills were gained throughout the project lifestyle. This included learning new languages to dynamic construct the application. This project has explored and highlighted different ways to tackle obstacles and comprise with limitations which have arisen throughout the development lifecycle. Overall, have achieved the majority of system goals the product and development process can be considered a success.

Bibliography

All Party Internet Group, 2004. *Revision of the Computer Misuse Act,* London: s.n.
Anderson , R. J., 1994 . *Liability and Computer Security: Nine Principles.* s.l., s.n., pp. 240-241.
Antell, J. et al., 2007. Data Protection Legislation. In: *IT Law.* Swindon: The British Computer Society, pp. 63-77.
Bevan, N., 2009. *Usability. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems.* s.l.:Springer, Boston, MA.
BhaumikPatel, 2018. *Nav Account Manager.* [Online]
Available at: https://bootsnipp.com/snippets/featured/nav-account-manager
Boostrap, 2018. *Components.* [Online]
Available at: http://getbootstrap.com/2.3.2/components.html
Brown, D. M., 2011. *Communicating design : developing Web site documentation for design and planning.* s.l.:Berkeley, Calif. : New Riders.
ChartJS, 2018. *Chart.js.* [Online]
Available at: https://www.chartjs.org/
Deloitte UK, 2018. How do today's students use mobiles?. *Game of phones.*
Fell, J., 2007. Data Protection Legislation. In: *IT Law.* s.l.:British Computer Society, pp. 63-70.
Goodman, M. D., 1997. Harvard Journal of Law & Technology Volume 10, Number 3. *Why The Police Dont Care About Computer Crime,* pp. 477-480.
HESA, 2018. *Non-continuation: UK Performance Indicators 2016/17,* s.l.: HESA.
ico.org.uk, n.d. *Are there any exemptions from the Data Protection Act?.* [Online]
Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/exemptions/
Kingston University, 2018. *Facts and figures: what you'll find at the University.* [Online]
Available at: http://www.kingston.ac.uk/aboutkingstonuniversity/factsandfigures/
legislation.gov.uk, 2004. [Online]
Available at: https://www.legislation.gov.uk/ukpga/1990/18/introduction
Leigh-Pollitt, P. & Mullock, J., 2001. The Data Protection Act Explained. In: *The Data Protection Act Explained.* s.l.:Osborne Clark, pp. 4-10.
Lepper, M. & Iyengar, S. S., 2001. When Choice is Demotivating: Can One Desire Too Much of a Good Thing?. *Personality and Social Psychology.*
Lloyd, I. J., 2011. Information Technology Law. In: *Information Technology Law.* s.l.:Oxford University Press, pp. 32-42.
Macewan, N., 2008. The Computer Misuse Act 1990: lessions from its past and predictions for its future. pp. 2-4.
Media, T., 22. *Getting Started With Chart.js.* s.l.:s.n.
Murray, A., 2013. Computer Misuse. In: *Information Technology Law.* s.l.:Oxford University Press, pp. 327-332.
Nielson, J., 2004. *Checkboxes vs. Radio Buttons.* [Online]
Available at: https://www.nngroup.com/articles/checkboxes-vs-radio-buttons/
Nielson, J., 2012. *Usability 101: Introduction to Usability.* [Online]
Available at: https://www.nngroup.com/articles/usability-101-introduction-to-usability/
Pernice, K., 2017. *F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant (Even on Mobile).* [Online]
Available at: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/
[Accessed 10 APRIL 2018].

Petrocelli, T., 2005. The Changing Face of Data Protection. In: *Data Protection and Information Lifecycle Management .* s.l.:Prentice Hall.
Rogers, Y., 2011. *Interaction Design.* 3rd Edition ed. s.l.:Wiley Ltd.
Russell, N., Aalst2, W. M. v. d., Hofstede, A. H. t. & Wohed, P., 2006. On the Suitability of UML 2.0 Activity Diagrams for Business.
Sharma, S. & Fisher, A., 2018. Simulating the User Experience: Design Optimisation for Visitor Comfort. pp. 62-65.
Smith-Atakan, S., 2006. *Human-Computer Interaction.* London: Middlesex University Press.
Sommerville, I., 2009. *Software Engineering.* s.l.:s.n.
Taylor, E., 2011. UK schools, CCTV and the Data Protection Act 1998. *Journal Journal of Education Policy .*
Tien , K., Pomaizlovaa, K. & Heywood , D., 2016. *The data protection principles under the General Data Protection Regulation.* [Online]
Available at: https://united-kingdom.taylorwessing.com/globaldatahub/article-the-data-protection-principles-under-the-gdpr.html
UCAS, 2017. *UCAS end of cycle report 2017,* London: s.n.
W3schools, 2018. *Introducing Javscript.* [Online]
Available at: https://www.w3schools.com/Js/
Weinschenk, S., 2009. *Neuro Web Design: What makes them click?.* s.l.:s.n.

Appendix A: Project Management and Software Methodology, Risk analysis and Contingency plan and Deliverables work flow

Project Management and software methodology

Adopting an efficient project management method can enable over all control and management throughout the project lifecycle. Many obstacles and contingencies may impact the project flow and therefore employing the most appropriate methodology is crucial for project success. It was decided that the most suitable approach to manage this project and develop the software is DSDM/Agile methodology. There are a number of reason for this decision, outlined in the following:

- This methodology prioritises requirements by value. Therefore, the system will adopt the most crucial requirements in the limited time frame. Additional qualities will bring the system extra value in project delivery.
- Client and stakeholder involvement is crucial in any project. This approach will enable the developer to work closely with the client and stakeholders to ensure the project success. In addition, the developer is able to specify in many system areas, features and components.
- By adopting a single methodology for both project management and software development will enable more time to be utilised effectively on the project.
- The agile methodology employs small time framed "sprints". As the project has a time constraint, these sprints will create minimal threat when delivering the planned deliverables.
- This management style is flexible and dynamic, allowing each stage to be iterated and recursive for the most effective effect. In addition, this approach also enables to
- This approach enables testing throughout the process. Therefore, requirements can be modified at any stage without having a detrimental impact on the project flow.

The DSDM/Agile methodology enables the project to be delivered in minor sprints but with much flexibility and room for improve from informative feedback from the client. Although, this methodology is the best suited for both the project management and software development, many other were considered by not chosen. The Agile methodology is very similar to the waterfall model. The waterfall methodology is a traditional approach which exercises a linear process. Each stage is iterated through in a seamless manner. Each stage must be completed and reviewed before another can begin. Due to this strict schedule the waterfall methodology was not chosen. The agile methodology enables the developer to improve and rectify areas at any stage of the project. Furthermore, the testing segment is towards the end of the project, making it difficult for developers to examine the products functionality throughout the project lifecycle.

Similarly, PRINCE2 was disregarded as it adopts a scheduled approach with little flexibility. The PRINCE2 methodology allows little room for improvement after feedback once different system components are finalised. Furthermore, SCRUM methodology requires daily meetings as it is a team-based approach. As the stakeholders involves are not able to meet regularly, this methodology would be difficult to abide by. Such methodologies, clearly indicate limitations which could significantly impact the project. Therefore, DSDM/Agile methodology is the most ideal approach moving forth with this project.

Risk Analysis and Contingency Plan

Risk in certain areas could inflict progression of the project. Therefore, a risk analysis and contingency plan has been conducted and derived from the threats outlined in the SWOT analysis.

This will enable the project manager to have a deeper understanding of the risks and how to tackle them in premature stages of the project. In turn, these mitigations can be considered when designing and implementing the system with additional provisions to enable a smooth-running project. Table 1, illustrates the risks, impact and likelihood of effect as well as mitigation and a contingency plan should these be an issue.

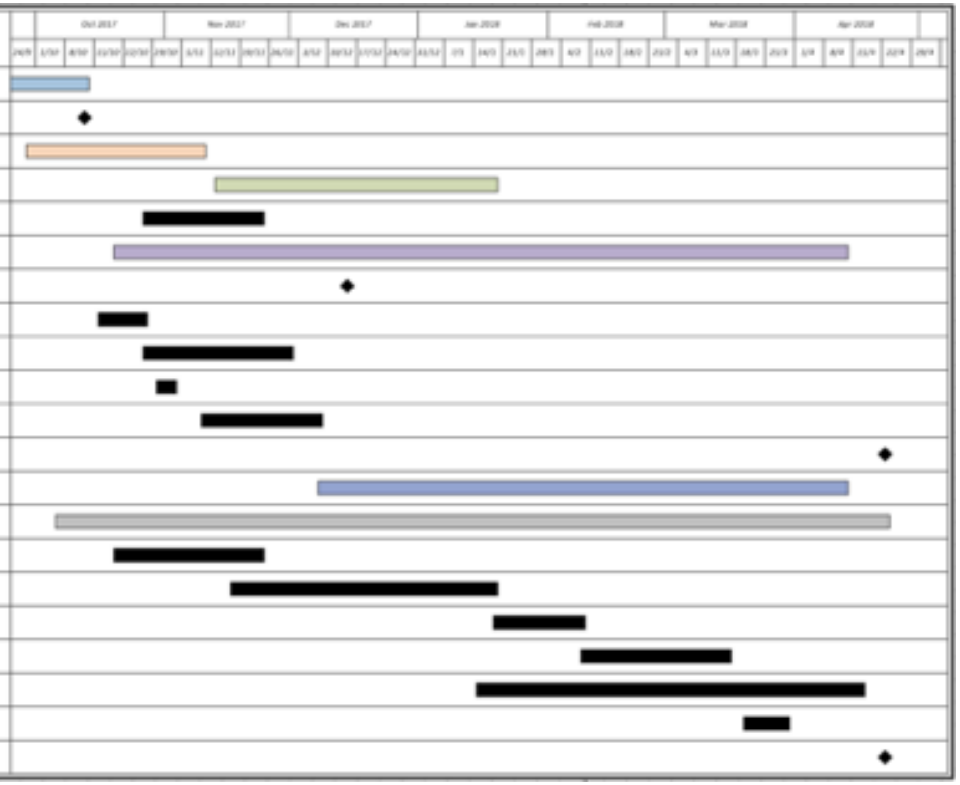
Risk	Likelihood	Severity	Contingency Plan
1. Not completing project in time	Low	High	Implement core functionality and what is finished.
2. Limited skillset for coding	Medium	Medium	Implement with knowledge gather, or task help from others. Look to re-organise tasks.
3. Client/user reject final prototype.	Low	Medium	Clarify major decisions with client, and sign of at end key stage. Revisit requirements if necessary.
4. Corruption of file	Low	High	Restore files from back-up.
5. Database failure	Low	High	Restore files from back-up.
6. Natural disaster occurs	Low	High	Restore files from back-up.
7. Poorly designed and implemented database.	Low	High	Implemented precautions to prevent SQL injection and hacking will minimise attack.
8. Not enduring to the correct methodology	Low	High	Re-organise task to meet timed deadline.
9. Data is not correctly stored in the database.	Low	High	Refactor data, implement constraints to prevent future mishap and follow regulations.

Table 1: Risk Analysis and Contingency Plan

Deliverables work flow

A proposed deliverables breakdown and key dates are outlined in a GAANT chart proposed at the start of the project. Shown in the figure below.

ID	Project Phases	Start	Finish	Duration
1	Research and Background	25/09/2017	13/10/2017	3w
2	Project Proposal	13/10/2017	13/10/2017	0w
3	Requirements Analysis	29/09/2017	10/11/2017	6w 1d
4	Design	13/11/2017	19/01/2018	10w
5	UML Diagrams	27/10/2017	24/11/2017	4w 1d
6	Implementation	20/10/2017	13/04/2018	25w 1d
7	Early Prototype Implementation	15/12/2017	15/12/2017	0w
8	Set up Database	16/10/2017	27/10/2017	2w
9	Webpage design implementation	27/10/2017	01/12/2017	5w 1d
10	PHP/HTML writing to database	30/10/2017	03/11/2017	1w
11	PHP/HTML reading from database	10/11/2017	08/12/2017	4w 1d
12	Viva	23/04/2018	23/04/2018	0w
13	Testing	08/12/2017	13/04/2018	18w 1d
14	Documentation	06/10/2017	23/04/2018	28w 2d
15	Requirements Chapter	20/10/2017	24/11/2017	5w 1d
16	Design Chapter	17/11/2017	19/01/2018	9w 1d
17	Review Chapter	19/01/2018	09/02/2018	3w 1d
18	Implementation Chapter	09/02/2018	16/03/2018	5w 1d
19	Testing Chapter	15/01/2018	17/04/2018	13w 2d
20	Draft Report For Greame	20/03/2018	30/03/2018	1w 4d
21	Report Submission	23/04/2018	23/04/2018	0w



Appendix B: Full User stories

User	Task in hand	For the purpose of
Administrator	Create User	I can keep student records up to date and accurate.
	Search User	I can track down particular students should there be a need to.
	View User	I can review user records.
	Edit User	I can modify student records.
	Delete User	I can keep student records up to date.
	Create Module	I can assign modules to users and courses.
	Search Module	I can track down particular modules should there be a need to.
	View Module	I can review module records
	Edit Module	I can modify module records.
	Delete Module	I can remove any unnecessary records.
	Create Course	I can assign modules and users to courses.
	Search Course	I can track down particular course should there be a need to.
	View Course	I can review course records
	Edit Course	I can modify course records.
	Delete Course	I can remove any unnecessary records.
	Create Department	I can assign courses to a department.
	Search Department	I can track down particular department should there be a need to.
	View Department	I can review department records.
	Edit Department	I can modify department records.
	Delete Department	I can remove any unnecessary records.
	Reset Password	I can reset to assist user access to the system.
	Enrol Student	I can assign students with a course and modules.
	Enrol Lecturer	I can assign lecturer teaching modules.

User	Task in hand	For the purpose of
Lecturer	Take Attendance	I can record and later monitor student attendance.
	Confirm Attendance	I can assure that all attendance details are correct and submit to the database.
	View Attendance	I can review student's attendance for the current event.
	Add new session	I can create a new session to take attendance for.
	Edit Attendance	I can modify student attendance records.
	View Attendance Report	I can review student attendance records.
	Search Student Attendance Report	I can track down particular student attendance report to monitor.
	Save Attendance Report	I can keep student attendance reports for records.

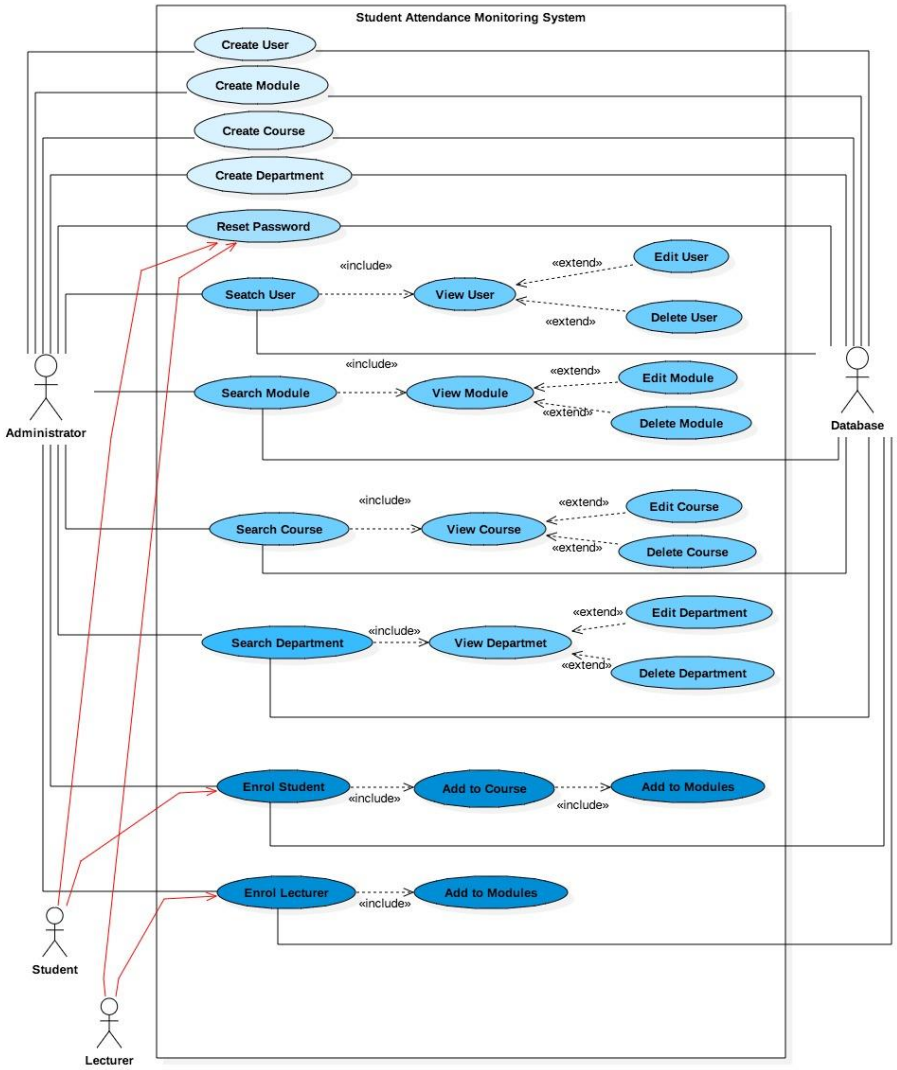
	Print Attendance Report	I can keep student attendance reports for records.
	View Course Attendance (specific)	I can review overall course attendance.
	View Module Attendance (specific)	I can review overall module attendance.
	View Student Comments	I can view student queries and respond.
	View Student Correspondences	I can view student reasons for absence and respond.
	Contact Student	I can contact students in relation to absence and disengagement from a module.
	Edit Personal Details	I can modify personal records.
	Reset Password	I can reset my access to the system should there be a need to

User	Task in hand	For the purpose of
Student	View Attendance	I can review my attendance.
	Upload Correspondence	I can give reason for my absence.
	View Correspondence	I can view my reasons for past absence.
	Edit Correspondence	I can modify my reasons for absence.
	Delete Correspondence	I can remove any unnecessary records.
	Contact Lecturer	I can contact lecturers with queries and my attendance.
	Edit Personal Details	I can modify personal records.
	Reset Password	I can reset my access to the system should there be a need to

Appendix C: Full Use Case Diagram

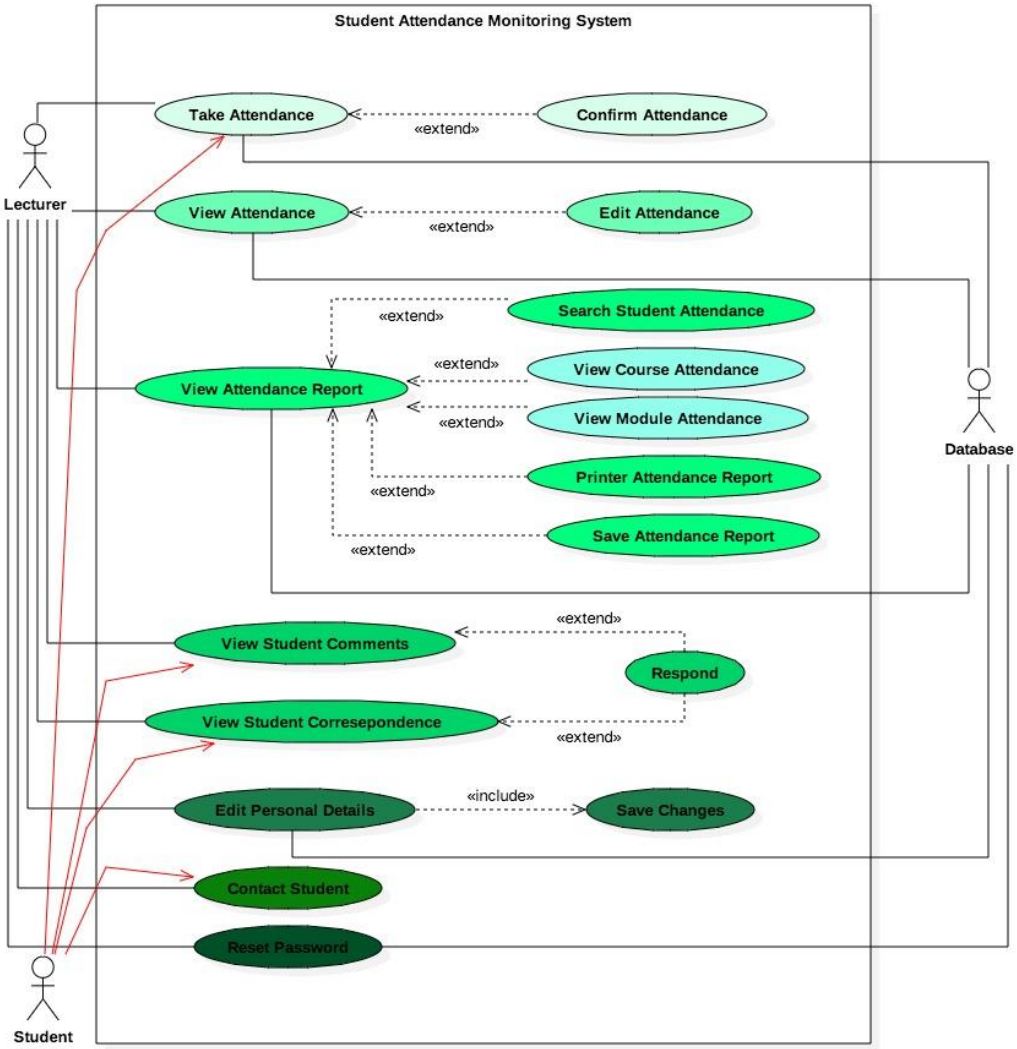
Administrator

The administrator will have full access to the system and will be able to create, review, edit and delete system entries. Lecturers and students will have direct associations with some of the Administrator's system tasks, this is displayed below.



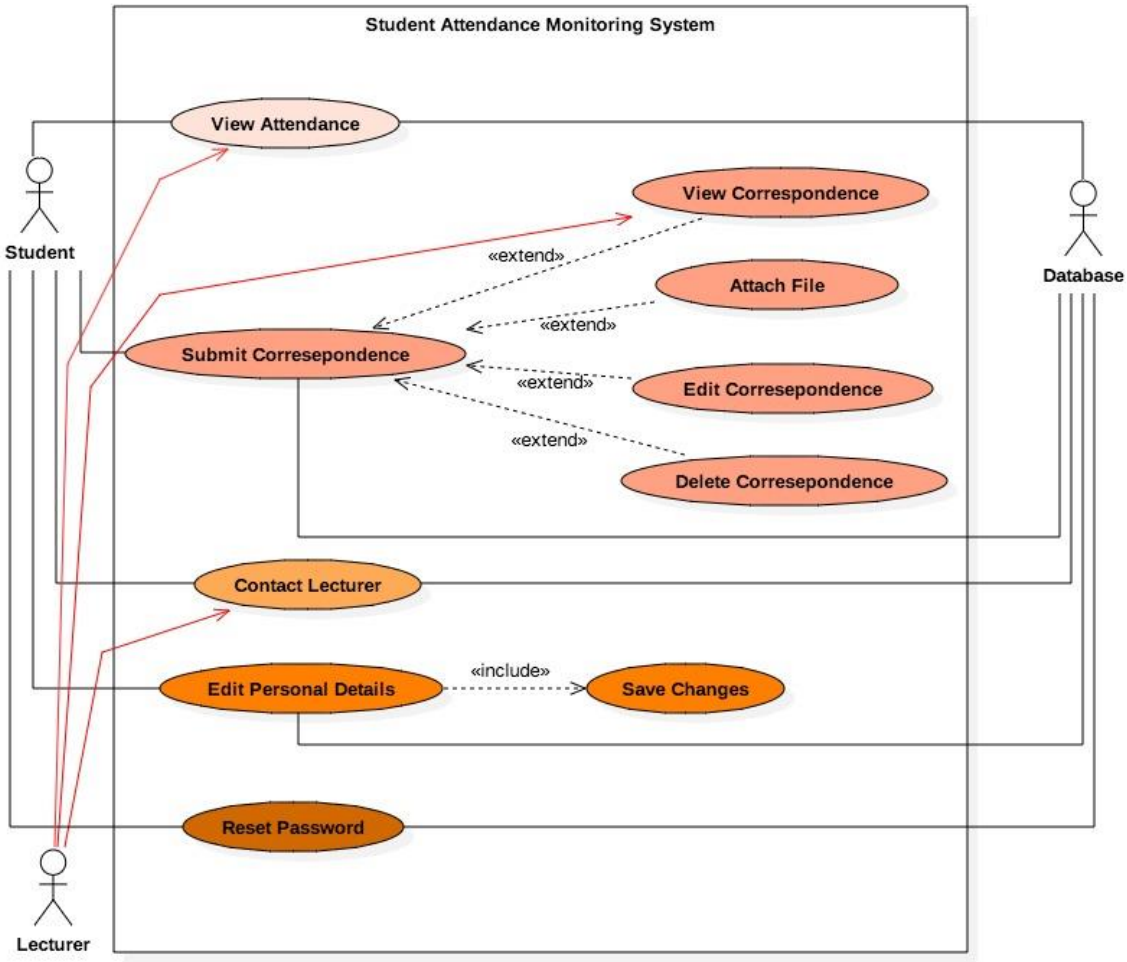
Lecturer

Lecturer will be able to have direct access to the system for its purpose and will be the systems key user. Specific lecturers will have different access rights depending's on their roles e.g. an employee can be both a lecturer and a module leader who will want access rights to view attendance from a particular module. Students will have direct associations with some of the lecturer's system tasks, this is displayed below.



Student

Although having access to the system, student will have the least rights of all primary users. However, this system will open up new avenues for students to monitor their own attendance. Lecturers will have direct activity with some of the student's system tasks, this is displayed below.



Appendix D: Full use case textual description

UC02	Enrol Student	
Actor	**Administrator**	
Goal in context	Enrol student onto course and modules.	
Preconditions	Administrator should be logged into the system.	
Success End condition	Student is enrolled onto a course with modules.	
Failed End condition	Student is not assigned to a course or modules.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage Student”.
	2	Administrator clicks on button “Enrol Student”.
	3	Administrator selects student from drop down list.
	4	Administrator selects course code.
	5	Administrator selects module codes.
	6	Administrator checks student details are correct.
	7	Administrator submits request to add to database.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		After new student is created.

UC03	Create Module	
Actor	**Administrator**	
Goal in context	Create a new module.	
Preconditions	Administrator should be logged into the system.	
Success End condition	A new module is created.	
Failed End condition	The new module is not created.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage Modules”.
	2	Administrator clicks on button “View Modules”.
	3	Administrator clicks on button “Add new Module”.
	4	Administrator enters module code.
	5	Administrator enters module name.
	6	Administrator selects module level.
	5	Administrator will select the courses the module will belong to.
	6	Administrator checks module details are correct.
	7	Administrator submits request to add to database.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When a new module is required.

UC04	Create Course
Actor	**Administrator**
Goal in context	Create a new course.

Preconditions	Administrator should be logged into the system.	
Success End condition	A new course is created.	
Failed End condition	The new course is not created.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage Course”.
	2	Administrator clicks on button “View Course”.
	3	Administrator clicks on button “Add New Course”.
	4	Administrator enters course code.
	5	Administrator enters course name.
	6	Administrator selects what department the course will belong to.
	7	Administrator checks course details are correct.
	8	Administrator submits request to add to database.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When a new course is required.

UC06	**Create Department**	
Actor	**Administrator**	
Goal in context	Create a new department.	
Preconditions	Administrator should be logged into the system.	
Success End condition	A new department is created.	
Failed End condition	The new department is not created.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage Department”.
	2	Administrator clicks on button “View Department”.
	3	Administrator clicks on button “Add New Department”.
	4	Administrator enters department code.
	5	Administrator enters department name.
	6	Administrator selects with faculty this department will belong to.
	7	Administrator checks department details are correct.
	8	Administrator submits request to add to database.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When there is a new department.

UC08	**Search User (Student)**	
Actor	**Administrator**	
Goal in context	To search a student.	
Preconditions	Administrator should be logged into the system.	
Success End condition	A student is searched.	
Failed End condition	Student was not able to be searched.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action

	1	Administrator clicks on button “Manage User”.
	2	Administrator clicks on button “View Students”.
	3	Administrator is shown a list of all the Students.
	4	Administrator clicks on search bar and types in student details to find a particular student e.g. student id, student name.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When a particular student needs to be found or to minimise search results.

UC10	**View User (Student)**	
Actor	**Administrator**	
Goal in context	To view a student’s details.	
Preconditions	Administrator should be logged into the system.	
Success End condition	Students details are shown.	
Failed End condition	Students details were not able to be shown	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage User”.
	2	Administrator clicks on button “View Students”.
	3	Administrator is shown a list of all the Students.
Alternative Flow	Step	Action
	1	Administrator clicks on button “Manage User”.
	2	Administrator clicks on button “View Students”.
	3	Administrator is shown a list of all the Students.
	4	Administrator clicks on search bar and types in student details to find a particular student e.g. student id, student name.
	5	Administrator clicks on student from the list of results.
	6	Administrator views student.
MoSCoW		Must
Frequency		When a particular student needs to be found.

UC11	**Edit User (Student)**	
Actor	**Administrator**	
Goal in context	To edit a student.	
Preconditions	Administrator should be logged into the system.	
Success End condition	Student is edited.	
Failed End condition	Student was unable to be edited.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage User”.
	2	Administrator clicks on button “View Students”.
	3	Administrator is shown a list of all the Students.
	4	Administrator clicks on search bar and types in student details to find a particular student e.g. student id, student name.

	5	Administrator selects student from list.
	6	Administrator clicks the edit button.
	7	Administrator modifies entry.
	8	Administrator saves changes.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When student records need to be modified.

UC12	**Delete User (Student)**	
Actor	**Administrator**	
Goal in context	To delete a student.	
Preconditions	Administrator should be logged into the system.	
Success End condition	Student is deleted.	
Failed End condition	Student was unable to be deleted.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Administrator clicks on button “Manage User”.
	2	Administrator clicks on button “View Students”.
	3	Administrator is shown a list of all the Students.
	4	Administrator clicks on search bar and types in student details to find a particular student e.g. student id, student name.
	5	Administrator selects student from list.
	6	Administrator clicks the delete button.
	7	System pops up box to confirm the request.
	8	Administrator confirms request.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When unnecessary records need to be deleted.

UC013	**Take Attendance**	
Actor	**Lecturer**	
Goal in context	To take student attendance in an event.	
Preconditions	Lecturer should be logged into the system.	
Post Condition	The	
Success End condition	Take student attendance for an event.	
Failed End condition	Student attendance is not taken.	
Primary, Secondary actors	Students.	
Main Flow	Step	Action
	1	Lecturer clicks on “Module” Drop down menu.
	2	Lecturer selects module.
	3	Lecturer clicks on “take attendance” on chosen session.
	4	Lecturer is shown an attendance list of students on the module
	5	Lecturer takes attendance.
	6	Lecturer submits attendance.

Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		At the start of every event.

UC014	**View Attendance Report**	
Actor	**Lecturer**	
Goal in context	To view overall attendance report.	
Preconditions	Lecturer should be logged into the system.	
Success End condition	Lecturer is able to view overall attendance report.	
Failed End condition	Lecturers are not able to monitor student attendance and act accordingly.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Lecturer click on “Module” drop down menu.
	2	Lecturer clicks on “View attendance report”.
	3	Lecturer selects date.
	4	Lecturer selects event on that date.
	5	Lecturer can search student and review attendance but module or event.
	6	Course Directors can view course attendances and module attendance.
	7	Module Leaders can view module attendance.
	8	Lecturer reviews attendance report.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When the lecturer would like to see progression of student attendance or when they are notified by the system of disengaging student.

UC015	**Contact Student**	
Actor	**Lecturer**	
Goal in context	Lecturer is able to contact student in accordance to their attendance.	
Preconditions	Lecturer should be logged into the system.	
Success End condition	Take student attendance for an event.	
Failed End condition	Student attendance is not taken.	
Primary, Secondary actors	Students	
Main Flow	Step	Action
	1	Lecturer clicks on “Communication” tab.
	2	Lecturer type in student name/id.
	3	Lecturer types in description.
	4	Lecturer adds attachments if necessary.
	5	Lecturer submits form.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When student is disengaging from events.

UC16	**View Attendance**

Actor	Lecturer	
Goal in context	To view attendance	
Preconditions	Lecturer should be logged into the system.	
Success End condition	Take student attendance for an event.	
Failed End condition	Student attendance is not available.	
Primary, Secondary actors	Students	
Main Flow	Step	Action
	1	Lecturer can see attendance on dashboard.
	2	Lecturer clicks pie charts and views session attendance.
	3	Lecturer clicks on session attendance and views students attendance.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		Whenever the lecturer wants to see attendance trends.

UC17	Edit Personal Details	
Actor	Lecturer	
Goal in context	To edit personal details.	
Preconditions	Lecturer should be logged into the system.	
Success End condition	Lecturer is able to modify personal details.	
Failed End condition	Student attendance is not taken.	
Primary, Secondary actors	Students	
Main Flow	Step	Action
	1	Lecturer clicks on “Account” tab
	2	Lecturer clicks on “personal details”.
	3	Lecturer clicks on the “Edit Button”.
	4	Lecturer edits areas for details to be modified.
	5	Lecturer clicks “save changes”.
	6	Lecturer is asked for confirmation for saving changes.
	7	Lecturer saves changes.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When the lecturer would like to update personal details.

UC19	View Attendance	
Actor	Student	
Goal in context	To view their own attendance.	
Preconditions	Student should be logged into the system.	
Success End condition	Student is able to view and monitor their own attendance,	
Failed End condition	Student is not able to view and monitor their own attendance,	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Student clicks on the “Modules” drop down menu.
	2	Student is able to select a module.
	3	Student views overall module attendance.
Alternative Flow	Step	Action
	1	Student is able to view attendance on dashboard.

MoSCoW		Must
Frequency		Whenever they log into the system.

UC20	**Upload correspondence**	
Actor	**Student**	
Goal in context	Student will be able to upload correspondence.	
Preconditions	Student should be logged into the system.	
Success End condition	Student is able to submit correspondence for reason for absence.	
Failed End condition	Student is unable to give reason for absence.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Student clicks on the “Communication” tab.
	2	Student clicks on “Upload correspondence”.
	3	Fill in form: Subject name, date, event, module and attachment.
	3	Submit form.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When student needs to give reason for absence.

UC21	**Contact Lecturer**	
Actor	**Student**	
Goal in context	Student will be able to contact lecturer.	
Preconditions	Student should be logged into the system.	
Success End condition	Student is able to contact lecturer.	
Failed End condition	Student is unable to contact lecturer.	
Primary, Secondary actors	N/A	
Main Flow	Step	Action
	1	Student clicks on “Communication” tab.
	2	Student type in Lecturer Name.
	3	Student types in description.
	4	Student adds attachments if necessary.
	5	Student submits.
Alternative Flow	Step	Action
	N/A	N/A
MoSCoW		Must
Frequency		When student needs to give reason for absence or contact lecturer.

Appendix E: Full MoSCoW analysis on Functional Requirements

#	Functional Requirement	Priority	Justification
	Administrator		
1	Create User	Must Have	An administrator must be able to create users in order for the system to be fully functional. These users will test the system and use it on a daily basis.
2	Search User	Should Have	An administrator should be able to search users via their details e.g. ID, NAME, COURSE, etc. There will be over thousands of user records from lecturers to students, therefore without a search function the usability of the system becomes very difficult. However, there will be no detrimental effect to the system if this was not implemented.
3	View User	Must Have	An administrator must be able to view all the user records to clarify details of what courses/modules they are enrolled on, personal details and to check if there is a need to modify or update records.
4	Edit User	Must Have	An administrator must be able to edit user details in the case of incorrect records, they are able to efficiently modify details without having to create a new users.
5	Delete User	Must Have	An administrator must be able to delete user records, this clears space for unnecessary records in the database.
6	Create Module	Must Have	An administrator must be able to create

7	Create Course	Must Have	modules and courses for the users to be enrolled onto. Without this feature there will be no attendance data to be taken for. The creation of module also allow deeper attendance insight for lecturers, module leaders and course directors.
8	Search Module	Should Have	An administrator should be able to search module/course via module/course details such as module/course code or name. Although there are not as many records as users it will create a simpler and quicker task for the administrator should they need to modify, edit, review any of these records. However, tasks can still be performed without this feature.
9	Search Course	Should Have	
10	View Module	Must Have	An administrator must be able to view module/course details to assure all data is correct and properly associated to one another and users.
11	View Course	Must Have	
12	Edit Module	Must Have	An administrator must be able to edit course/module details to assure that all records are correct and if they are not then they can efficiently modify and update these records without having to create a new entry.
13	Edit Course	Must Have	
14	Delete Module	Must Have	An administrator must be able to delete module/course records to clears space for unnecessary records in the database should modules of courses be replaced.
15	Delete Course	Must Have	
16	Create Department	Could Have	An administrator could create, search, view, edit and delete a department. The main focus for this project is to allow lecturers greater insight into student attendance. By opening up and
17	Search Department	Could Have	

18	View Department	Could Have	considering these requirements allows Head of department to have overview on the student engagement in this department, it is not a necessity to assure that the system runs correctly but never the less feature that will give the project value.
19	Edit Department	Could Have	
20	Delete	Could Have	
21	Reset Password	Must Have	An administrator must be able to reset user password otherwise when forgetting/getting the password incorrect too many times, the user can regain access through a password reset.
22	Enrol Student	Must Have	An administrator must be able to enrol students/lecturers onto courses/modules to create the correct associations when taking attendance.
23	Enrol Lecturer	Must Have	
24	Lecturer		
25	Take Attendance	Must Have	A Lecturer must be able to take attendance for the systems purpose to run its full functionality and to abide by laws.
26	Confirm Attendance	Must Have	A Lecturer must be able to confirm attendance to assure that the correct records of student attendance is submitted to the database to run further functionalities.
27	View Attendance	Must Have	A Lecturer must be able to view attendance so they can clarify that the record are correct, and if needed to modify/update those that are incorrect.
28	Edit Attendance	Must Have	A Lecturer must be able to edit attendance to assure that all attendance records are correct and if they are not

			then they can efficiently modified or updated for example when a student comes to an event late.
29	View Attendance Report	Must Have	A Lecturer must be able to view attendance report to be able to monitoring students and identify trends and patterns of disengaging students.
30	Search Student Attendance Report	Should Have	A Lecturer must be able to search student attendance report through their details such as ID, Name, etc. There will be over hundreds of individual student attendance records, therefore a search function will assit the usability of the system. However, there will be no detrimental effect to the system if this was not implemented.
31	Print Attendance Report	Won't Have	A Lecturer won't be able to print/save an attendance report in this current system, there are many other requirements that will need to be implemented for success before extra features are considered and added to the system.
32	Save Attendance Report	Won't Have	
33	View Course Attendance	Should Have	A Lecturer should be able to view attendance report through filters such as overall module/course attendance. This allows for further trends and patterns to be found through the student attendance report. However, there will be no detrimental effect to the system if this was not implemented.
34	View Module Attendance	Should Have	
35	View Department Attendance	Could Have	A Lecturer could be able to view attendance report through a department filter if the head of department is considered as a primary user. This allows for a deeper insight for further trends and

			patterns to be found through the student attendance report. However, there will be no detrimental effect to the system if this was not implemented.
36	View Student Comments	Should Have	A Lecturer should be able to view student comments/correspondence for their absence to make any changes to attendance records. This allows a communication platform between the lecturer and student on the specific case of attendance. Furthermore lecturers should be able to respond to students and t contact them back about event attendance.
37	View Student Correspondence	Should Have	
38	Contact Student	Should Have	
39	Edit Personal Detail	Should Have	A Lecturer should be able to edit their own personal details for their account e.g. title, name, role, to assure that all records visible to users on the system are correct. However, as the administrator already have "Must Have" rights to do this, however this will create a long winded process to change details this way.
40	Reset Password	Could Have	A Lecturer could be able to reset their own password. However for security reason it may be best to allow the resetting password access to limited users and leave it just to the administrator. However, this will create a long winded process to change a password this way.
41	Student		
42	View own attendance	Should Have	A Student should be able to view their own attendance to allow them to monitor and track their own attendance records. This requirement allows users to become

			more aware of their attendance.
43	Upload Correspondence	Should Have	A Student should be able to upload correspondence in reference to their absence, to allow greater communication between students, lecturers and their education on one platform.
44	View Correspondence	Should Have	A Student should be able to view correspondence to assure that all the details included are sufficient and correct for their absence.
45	Edit Correspondence	Should Have	A Student should be able to edit correspondence to modify or update any details that were left out in the original correspondence submission.
46	Delete Correspondence	Should Have	A Student should be able to delete correspondence if there is no longer any need for it to be submitted.
47	Contact Lecturer	Should Have	A Student should be able to contact lecturer about their attendance if they would like to provide further insight on attendance of any other matters. This allows a communication platform between the lecturer and student on the specific case of attendance
48	Edit Personal Details	Could Have	A Student should be able to edit their own personal details for their account e.g. name, role, to assure that all records visible to users on the system are correct. However, as the administrator already have “Must Have” rights to do this, however this will create a long winded process to change details this way.
49	Reset Password	Could Have	A Student could be able to reset their

			own password. However for security reason it may be best to allow the resetting password access to limited users and leave it just to the administrator. However, this will create a long winded process to change a password this way.

Appendix F: Exemptions from the DPA 1998

The DPA is provisioned to apply generically to all individuals, businesses and organisations. However, there are some exemption from the act to accommodate for special circumstances. As advised by the Information Commissioners Office (ICO), each exemption must be considered on a case-by-case basis. Therefore, depending on the circumstances the act declares that an individual will be exempt from the following requirements:

- To register with the ICO (to "notify"); and/or
- To grant subject access to personal data; and or
- To give privacy notices; and/or
- Not to disclose persona data to third parties (ico.org.uk, n.d.)

These requirements are detailed under Part IV, Schedule 7 of the DPA 1998. Complete exemption includes, data obtained and processed for the purposes of protecting national security and domestic purposes. However, in some circumstances it is appropriate to disclose personal information; these recognised as partial exemptions from the act. According to the ICO, they include some of the following:

- **Crime and Taxation**: In these circumstances personal data could be processed for the prevention/detection of crime, prosecuting offenders and the collection of tax.
- **Disclosures required by law**: Non-disclosure is exempt under any UK enactment, by any rule of common or by order of a court in any jurisdiction.
- **Education purposes**: In the cases of educational or social work records. These include examination marks before publication.
- **Physical or Mental Health**: The cases in which a professional see that granting the individual access to their data would trigger serious harm to themselves or someone else, are also exempt.
- **Secretary of state**: The security of the state can further exempt data by order (legislation.gov.uk, 2004).

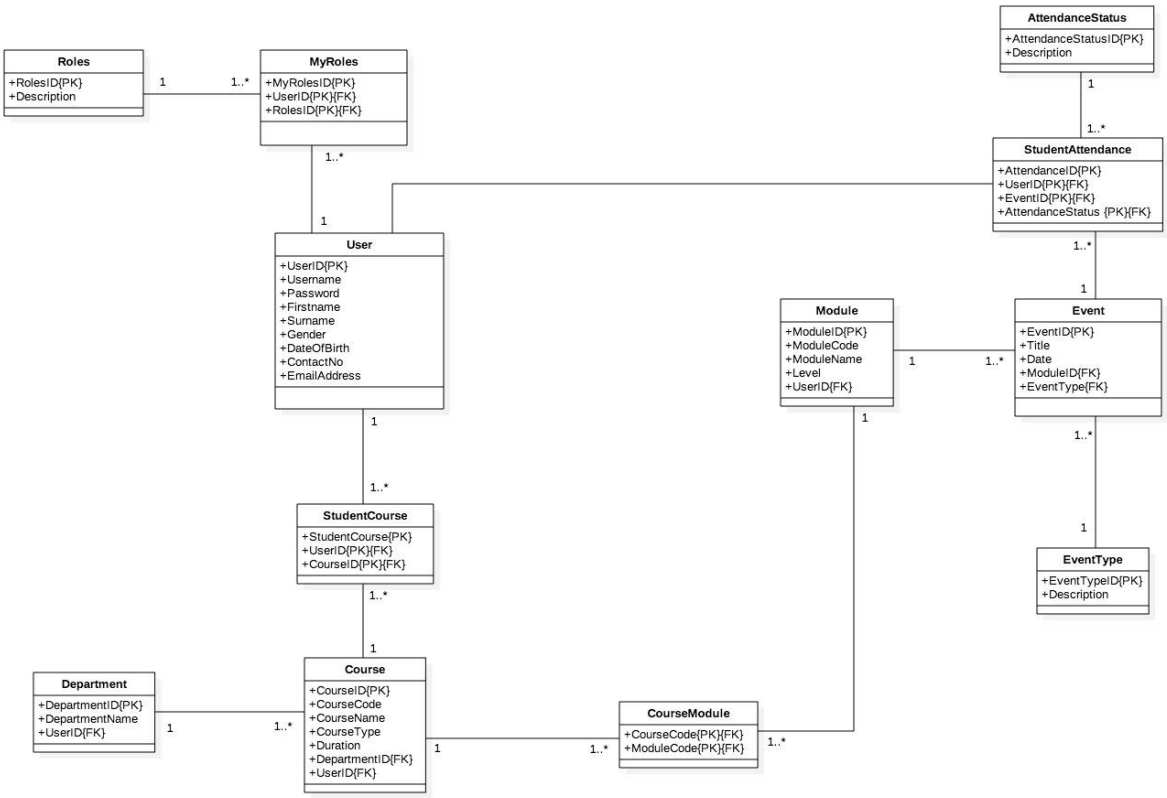
Although the exemptions are touched over lightly, this is vast improvement from the 1984 legislation whom legacy regretfully missed out the key principles which safeguard our data today.

Appendix G: New GDPR Data Protection principles

The evolution of the DPA has enhanced significantly, an ongoing process to keep up with societal changes. In recent development, the European Union bring forth a new regulation to ensure a higher degree of protection for individuals and further conditions upon those who process individual's data. May 25th 2018 is the passing of the new regulation for data protection; the General Data Protection Regulation (GDPR). Similarly, the GDPR is underpinned by many data protection principles which drive compliance. Although there are many similarities between the GDPR and the DPA 1998, certain principles have been developed. The principles relating to the processing of personal data are enlisted in Article Five of the GDPR, which require:

- **Processed lawfully, fairly and in a transparent manner in relation to the data subject;**
  This is similar to the first current data protection principle; however the processing of personal data must be complete in a "transparent" and concise manner, assuring ease in accessibility. In order to achieve transparency, the data subjects are kept informed before the data is collated and where any subsequent changes are made.
- **Collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes.**
  Similarly found in DPA principle two, the processing of data is only acceptable if it is compliant to the original purpose for which the data was collected for. However, exemptions include processing for the purpose of public interest, scientific or historical research purposes or statistical purposes.
- **Adequate, relevant and limited to what is necessary in relation to the purpose for which they are processed**
  Data controllers must ensure that the necessary personal data is attained only for its specific purpose, and enough data is attained to reach its purpose.
- **Personal data shall be accurate and, where necessary kept up to date**
  This condition is also found in the DPA 1998.
- **Kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed**
  Once the personal data has exceeded its purpose in which it was collected, the data should be deleted. It is advised that regular reviews are conducted to cleanse the database. Exceptions of this condition include purpose of public interest, scientific or historical research or statistical purposes, with account to implementation safeguards.
- **Processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures**
  This principle requires data controllers and processors to assess the risk, assuring appropriate security is implemented.
- **The controller shall be responsible for, and be able to demonstrate compliance with the GDPR**
  The principle of accountability is one in which major implications apply. The burden of responsibly has extended from the data controller to include the data processor also. In addition, it is not enough to comply, it much be seen their individuals are complying.

Appendix H: Initial Entity-relationship Diagram



Appendix I: Full break down of ERD

To differentiate the Users of the system, each User will have a “Role”. For example, a User can have the role of a Student. It is assumed that a User will only have one role in which they partake on the system. The surrounding aspects of the system all interact with User either directly or indirectly, therefore the User table is the focal point for this database design. The Users are Administrator, Student and Lecturer which is subsumed by Module Leader, Course Director and Head of department.

As requested from the client, a platform to allow the transmission of messages between Users has been considered. The details of these correspondences will be stored in the Communication table. As illustrated in Figure 4.3, Communication has two foreign key attributes, both directing to the User (UserID) table, as a source and a recipient of the messages. This is displayed as the attributes Sender and SentTo in the Communication table.

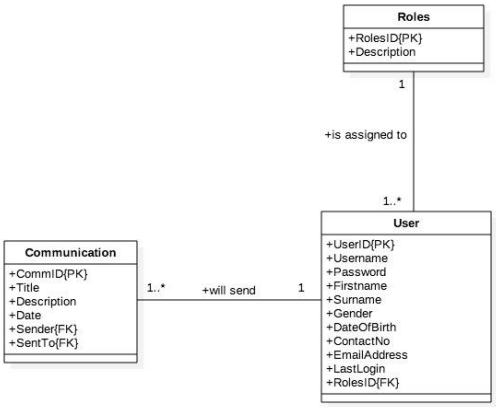


Figure 4.3: ERD - User Types and Messages

Figure 4.3 mirrors the real-world association within the University, of Students and Lecturers to their according courses and modules. These associations will store the modules taught by Lecturers and the attendance data collected. There are many cases of resolved multiplicities in this subsection of the database, an example being the relationship between Lecturers and their Modules. A Lecturer (User) will teach one-to-many Modules, and therefore this occurrence of multiplicity is resolved by an intersecting class, LecturerModule. A request by the client involved a centralised system which allows the University to share and view data across modules and faculty. As a result, direct relationship is created between a User and the associative entities. For example, UserID is enlisted as a foreign key in Course depicting a one- to-many association. Likewise, found in the occurrence of Department and Module.

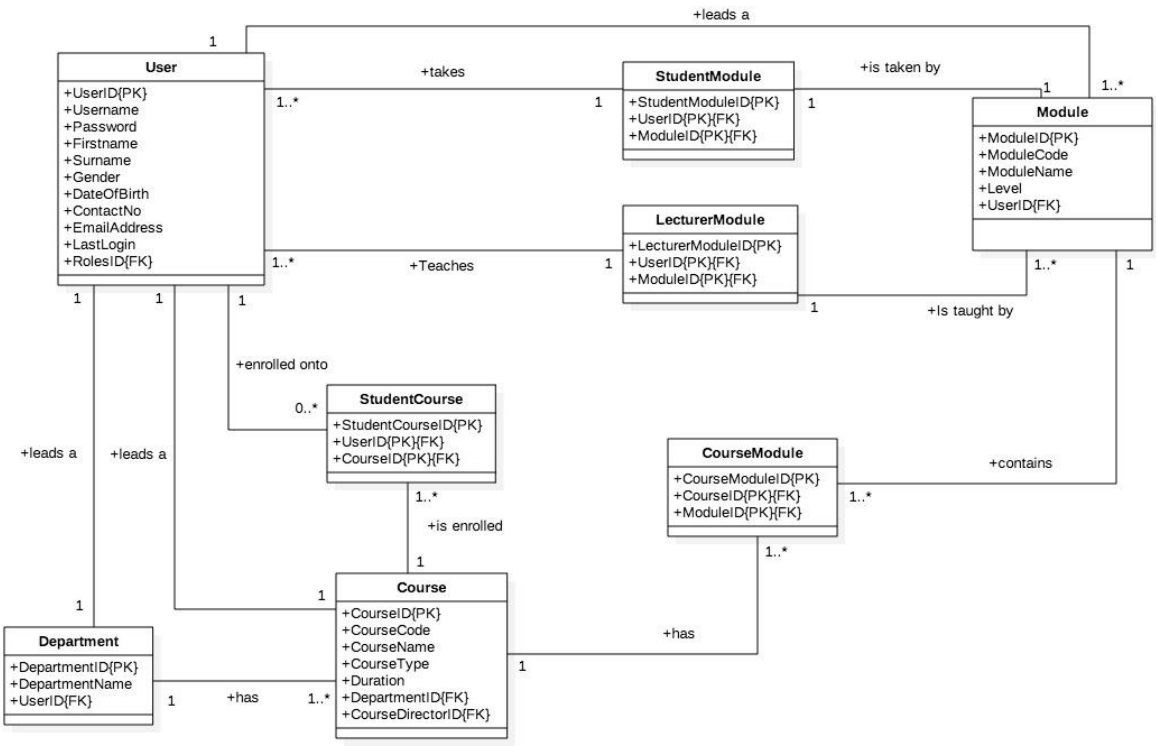


Figure 4.3: ERD - Users, Modules, Courses and Departments

Figure 4.4 concentrates on recorded the attendance of Students (User). This segment of the database illustrates the relationship between Students and their Module attendance. Each Module contains many Event's such as lectures and workshops. Students are enrolled onto several Modules, and therefore StudentAttendance store the attendance data at each Event. StudentAttendance is an intersecting table associating Users with Events attended, resolving the conflicting many-to-many relationships.

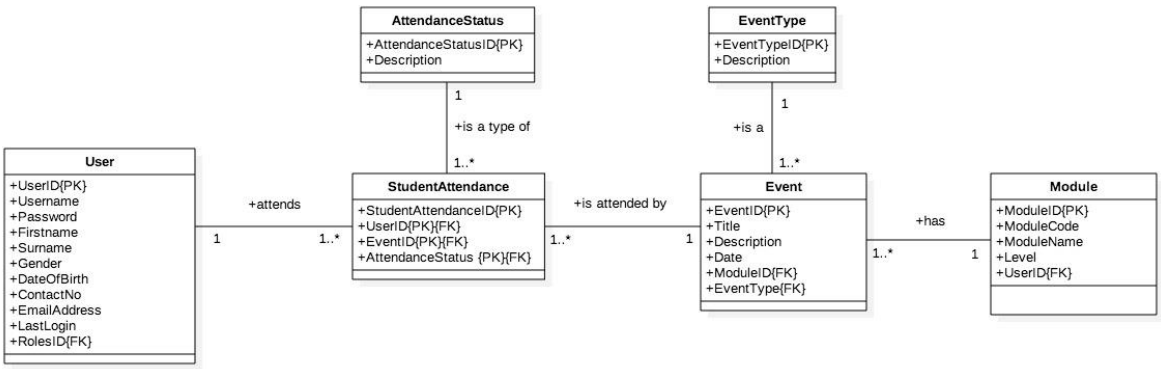


Figure 4.4: Entity Relationship Diagram: Users and their Attendance

Appendix J: Functional Requirements SQL Queries created to test Database Design

#	Functional Requirement	SQL Query
1	Create User	INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress)
2	View Student	SELECT Roles.Description, User.Username, User.Password, User.Firstname, User.Surname, User.Gender, User.DateOfBirth, User.ContactNo, User.EmailAddress, User.LastLogin, Roles.Description, Course.CourseName, Course.CourseCode FROM ((((`User` INNER JOIN MyRoles ON MyRoles.UserID = User.UserID) INNER JOIN Roles ON Roles.RolesID = MyRoles.RolesID) LEFT JOIN StudentCourse ON StudentCourse.UserID = User.UserID) LEFT JOIN Course ON Course.CourseID = StudentCourse.CourseID) WHERE Roles.RolesID=1
3	Edit User	UPDATE User SET Firstname = 'simon' WHERE Username = 'k1403042'
4	Reset Password	UPDATE User SET Password = '' Where User.Username = ''
5	Confirm Attendance	UPDATE StudentAttendance SET AttendanceStatus ='2' Where UserID=2 AND EventID=0
6	Edit Attendance	UPDATE StudentAttendance SET AttendanceStatus ='3' Where UserID=2 AND EventID=0
7	View Module Attendance	SELECT User.Username, User.Firstname, User.Surname, Module.ModuleName, EventType.Decription, Event.Title, AttendanceStatus.Description FROM (((((((`Module` INNER JOIN StudentModule ON StudentModule.ModuleID = Module.ModuleID) INNER JOIN User ON User.UserID = StudentModule.UserID) LEFT JOIN Event ON Event.ModuleID = Module.ModuleID) LEFT JOIN EventType ON EventType.EventTypeID = Event.EventTypeID) LEFT JOIN StudentAttendance ON StudentAttendance.EventID = Event.EventID) LEFT JOIN StudentAttendance Userid ON StudentAttendance.UserID = User.UserID) LEFT JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceID) WHERE Module.ModuleID = 1
8	View Department Attendance	SELECT Department.DepartmentName, Course.CourseCode, Course.CourseName, User.Username, User.Firstname, User.Surname, Module.ModuleName, EventType.Decription, Event.Title,

		AttendanceStatus.Description FROM (((((((((`Department` INNER JOIN Course ON Department.DepartmentID = Course.DepartmentID) LEFT JOIN StudentCourse ON StudentCourse.CourseID = Course.CourseID) LEFT JOIN User ON User.UserID = StudentCourse.UserID) LEFT JOIN StudentModule ON StudentModule.UserID = User.UserID) LEFT JOIN Module ON Module.ModuleID = StudentModule.ModuleID) LEFT JOIN Event ON Event.ModuleID = Module.ModuleID) LEFT JOIN EventType ON EventType.EventTypeID = Event.EventTypeID) LEFT JOIN StudentAttendance ON StudentAttendance.EventID = Event.EventID) LEFT JOIN StudentAttendance Userid ON StudentAttendance.UserID = User.UserID) LEFT JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceID WHERE Department.DepartmentID = 1
9	Edit Personal Detail	UPDATE User SET User.Firstname ='Josh' WHERE User.UserID=11
10	View Course Attendance	SELECT Course.CourseCode, Course.CourseName, User.Username, User.Firstname, User.Surname, Module.ModuleName, EventType.Decription, Event.Title, AttendanceStatus.Description FROM (((((((((`Course` INNER JOIN StudentCourse ON StudentCourse.CourseID = Course.CourseID) INNER JOIN User ON User.UserID = StudentCourse.UserID) INNER JOIN StudentModule ON StudentModule.UserID = User.UserID) INNER JOIN Module ON Module.ModuleID = StudentModule.ModuleID) LEFT JOIN Event ON Event.ModuleID = Module.ModuleID) LEFT JOIN EventType ON EventType.EventTypeID = Event.EventTypeID) LEFT JOIN StudentAttendance ON StudentAttendance.EventID = Event.EventID) LEFT JOIN StudentAttendance Userid ON StudentAttendance.UserID = User.UserID) LEFT JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceID) WHERE Course.CourseID = 2

SQL queries have been generated from the function requirement list of the user. This list will allow the testing of integrity on the database design, and if necessary make the required changes before the user interface implementation. This is also beneficial as it will save time, during the implementation phase to create SQL queries if they run successfully.

Appendix K: Full Data Dictionary

Relation	Attribute	Data Type(Size)	PK/FK	Not Null	Other Constraints	Referential Relation
User	UserID	INT(8)	PK	✓	AUTO_INCREMENT	
	Username	VARCHAR(10)		✓		
	Password	VARCHAR(100)		✓		
	Firstname	VARCHAR(20)		✓		
	Surname	VARCHAR(20)		✓		
	Gender	VARCHAR(8)		✓		
	DateOfBirth	DATE		✓		
	ContactNo	VARCHAR(11)		✓		
	EmailAddress	VARCHAR(50)		✓		
	LastLogin	DATETIME		✓	CURRENTTIMESTAMP	
	RolesID	INT(11)	FK	✓		Roles

Relation	Attribute	Data Type(Size)	PK/FK	Not Null	Other Constraints	Referential Relation
StudentAttendance	StudentAttendanceID	INT(11)	PK	✓	AUTO_INCREMENT	
	UserID	INT(10)	FK	✓		User
	EventID	INT(10)	FK	✓		Event
	AttendanceStatusID	INT(11)	FK	✓	DEFAULT “2”	AttendanceStatus

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
Roles	RoleID	Int(11)	PK	✓	AUTO_INCREMENT	
	Description	Varchar(30)		✓		

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
Course	CourseID	Int(10)	PK	✓	AUTO_INCREMENT	
	CourseCode	Varchar(50)		✓		
	CourseName	Varchar(30)		✓		
	CourseType	Varchar(20)		✓		
	Duration	Varchar(8)		✓		
	DepartmentID	Int(10)	FK	✓		Department
	CourseDirectorID	Int(8)	FK	✓		User

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
StudentCourse	StudentCourseID	Int(8)	PK	✓	AUTO_INCREMENT	
	UserID	Int(8)	FK	✓		User
	CourseID	Int(10)	FK	✓		Course

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation

Department	DepartmentID	Int(10)	PK	✓	AUTO_INCREMENT	
	DepartmentName	Varchar(100)		✓		
	UserID	Int(8)	FK	✓		User

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
Module	ModuleID	Int(11)	PK	✓	AUTO_INCREMENT	
	ModuleCode	Int(10)		✓		
	ModuleName	Varchar(200)		✓		
	Level	Int(2)		✓		
	UserID	Int(8)	FK	✓		User

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
CourseModule	CourseModuleID	Int(11)	PK	✓	AUTO_INCREMENT	
	CourseID	Int(8)	FK	✓		Course
	ModuleID	Int(10)	FK	✓		Module

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
StudentModule	StudentModule	Int(8)	PK	✓	AUTO_INCREMENT	
	UserID	Int(8)	FK	✓		User
	ModuleID	Int(11)	FK	✓		Module

Relation	Attribute	Data Type(Size)	PK/ FK	NOT NULL	Other Constraints	Referenti al Relation
LecturerModule	LecturerModuleID	Int(8)	PK	✓	AUTO_INCREMENT	
	UserID	Int(8)	FK	✓		User
	ModuleID	Int(10)	FK	✓		Module

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
Event	EventID	Int(10)	PK	✓	AUTO_INCREMENT	
	Title	Varchar(30)		✓		
	Description	Varchar(1000)		✓		
	Date	Date		✓		
	ModuleID	Int(11)		✓		
	EventType	Int(11)	FK	✓		EventType

Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation
EventType	EventTypeID	Int(11)	PK	✓	AUTO_INCREMENT	
	Description	Varchar(50)		✓		

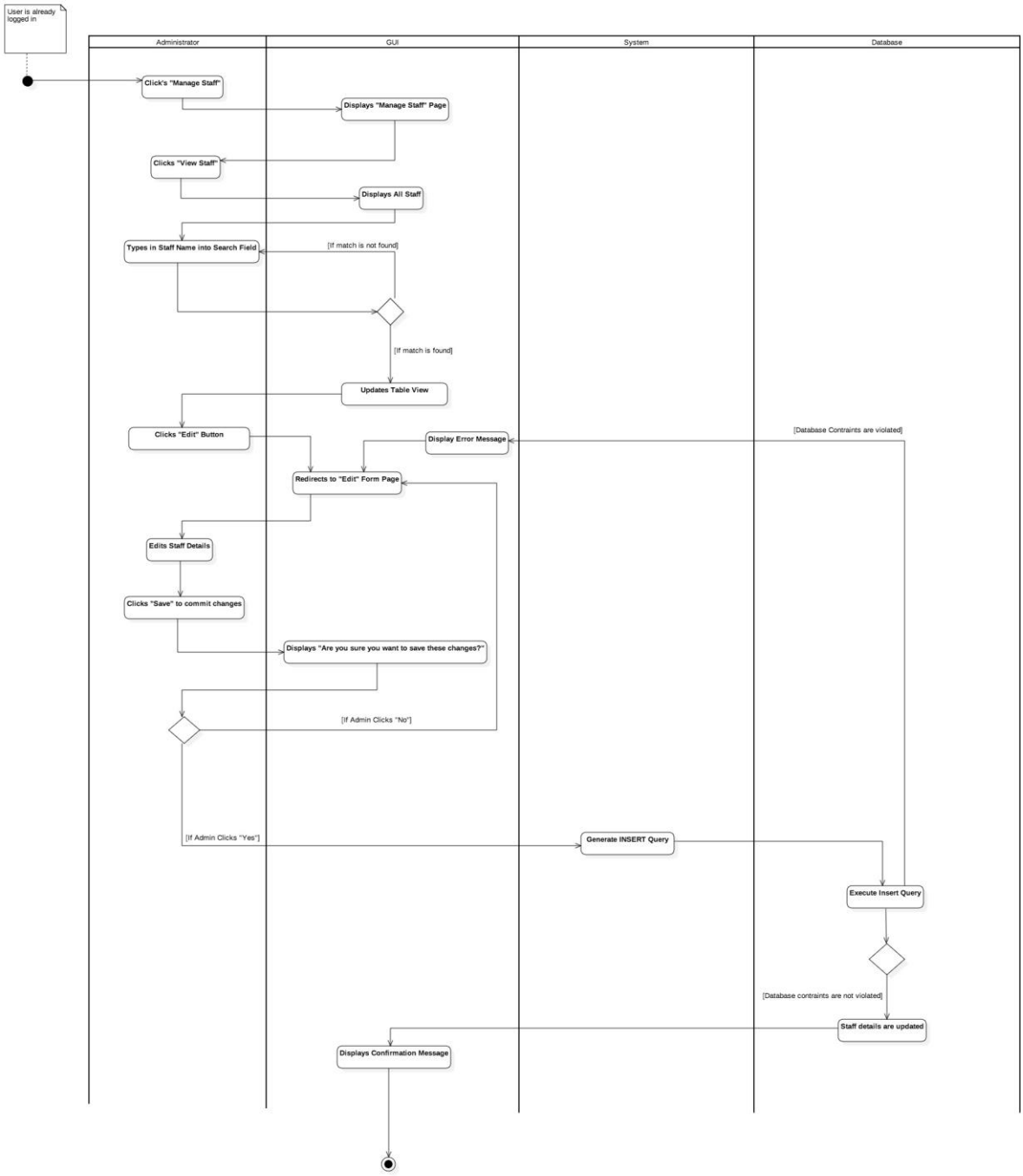
Relation	Attribute	Data Type(Size)	PK/FK	NOT NULL	Other Constraints	Referential Relation

AttendanceStatus	AttendanceStatusID	Int(11)	PK	✓	AUTO_INCREMENT	
	Description	Varchar(50)		✓		

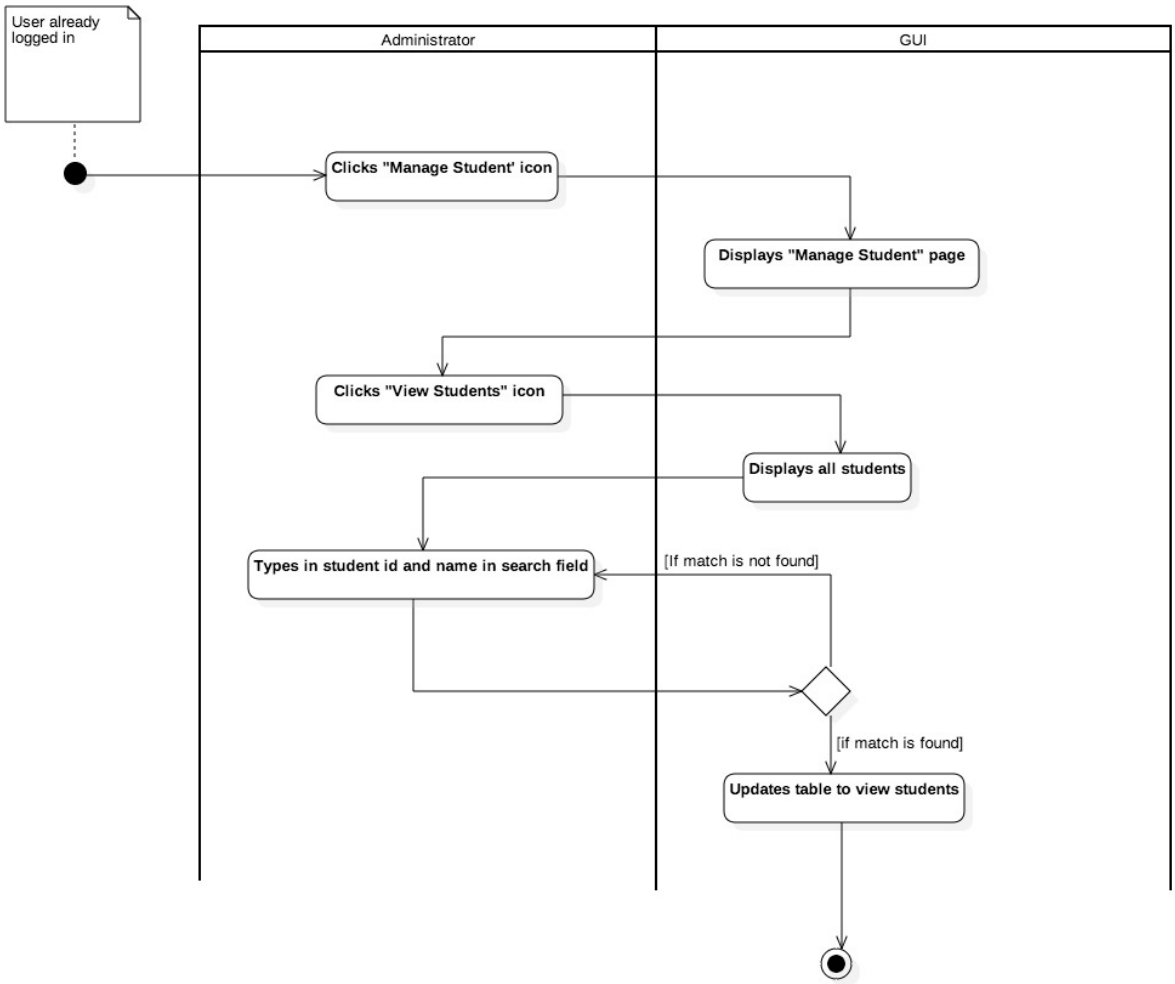
Relation	Attribute	Data Type(Size)	PK/ FK	NOT NULL	Other Constraints	Referential Relation
Communication	CommunicationID	Int(11)	PK	✓	AUTO_INCREMENT	
	Title	Varchar(50)		✓		
	Description	Varchar(1000)		✓		
	Date	Date		✓		
	Sender	Int(8)	FK	✓		User
	SentTo	Int(8)	FK	✓		User

Appendix L: Activity diagrams

Admin edit user:

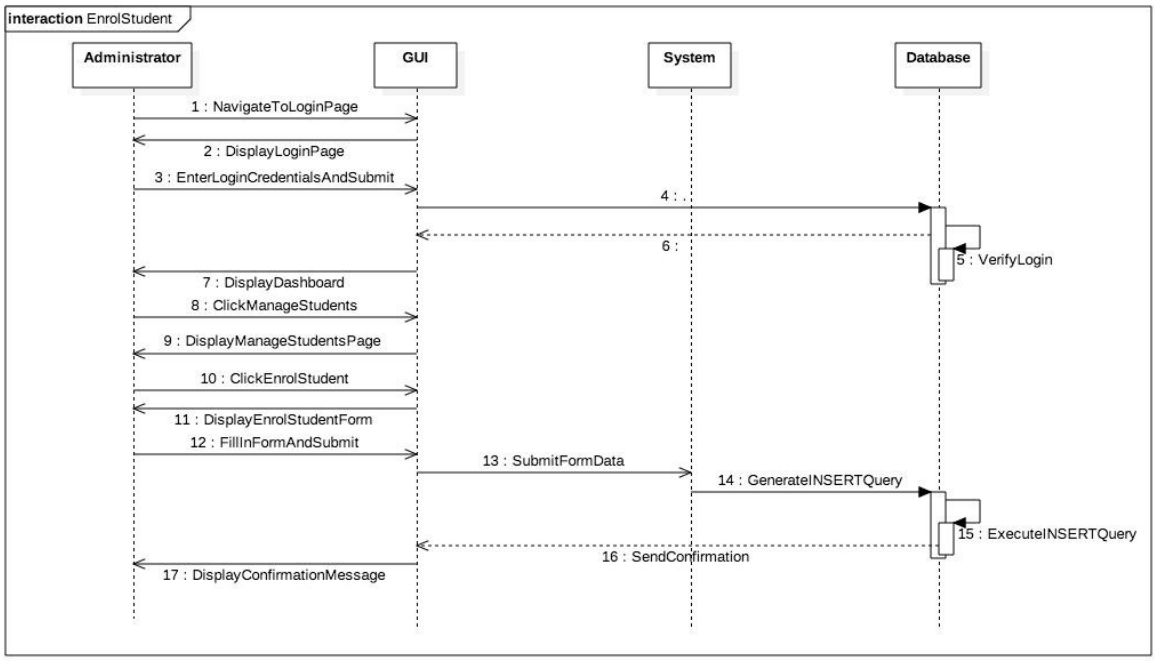


Search Student:

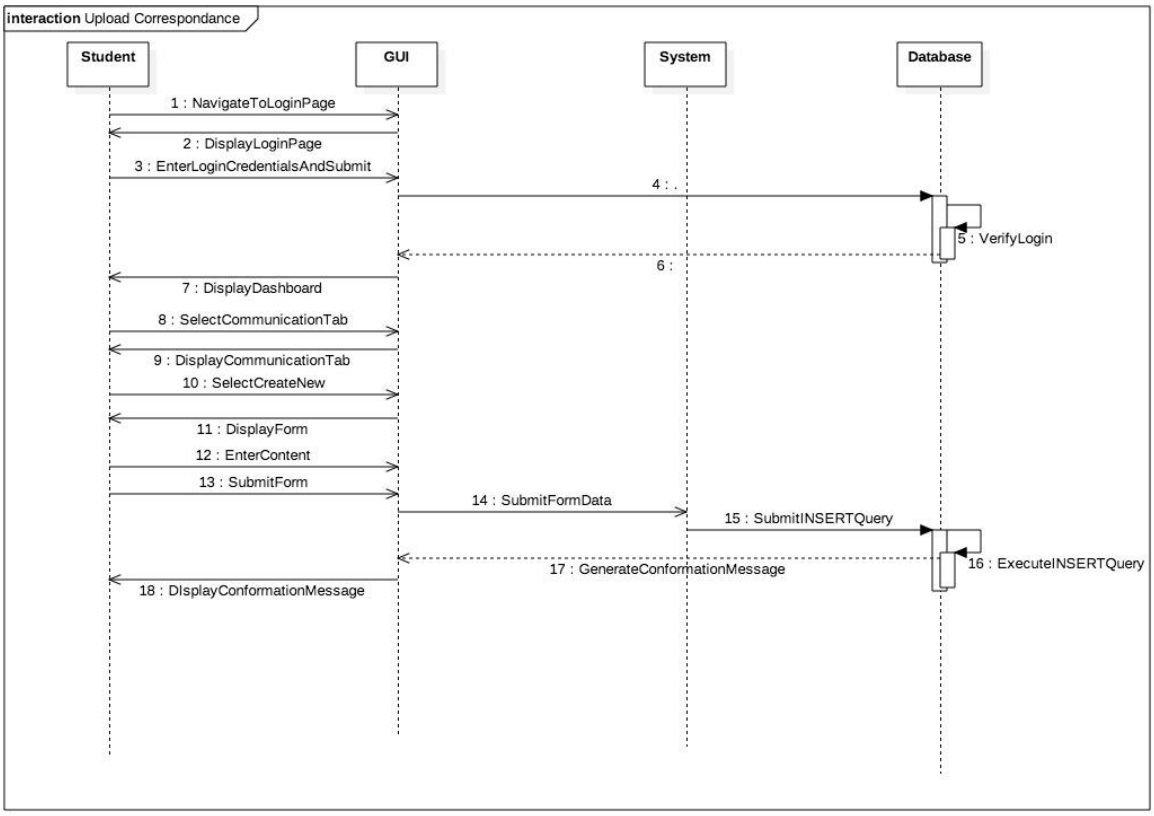


Appendix M: Sequence diagrams

Enrol Student:

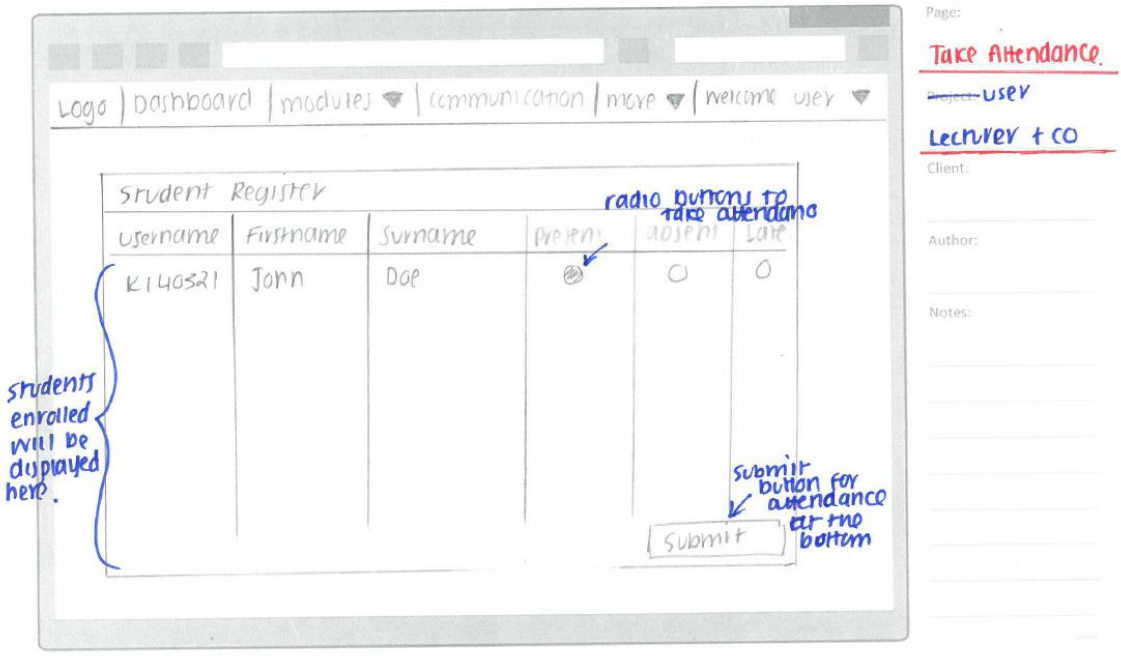


Upload Correspondence:

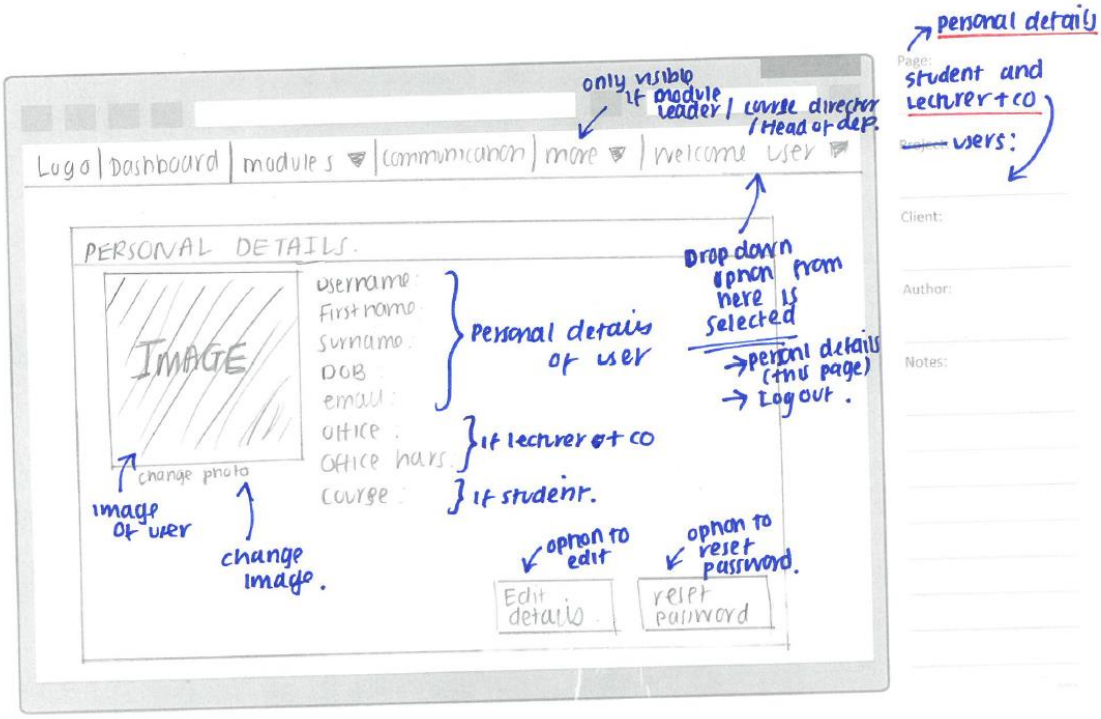


Appendix N: Low Fidelity wire frames

Lecturer & Co:

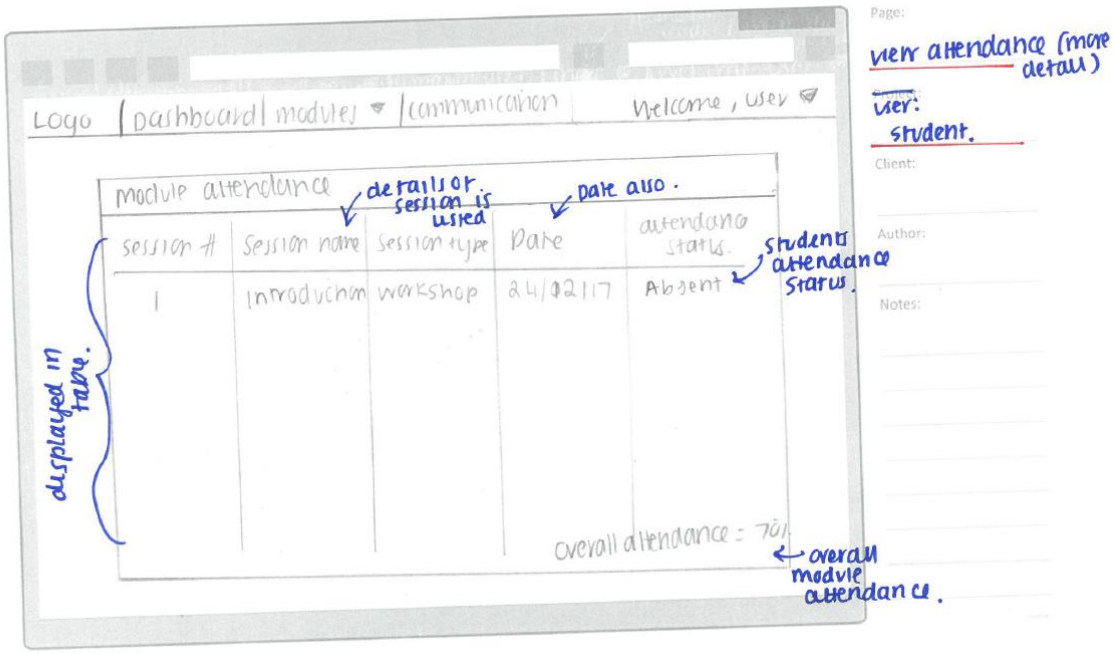


Take Attendance



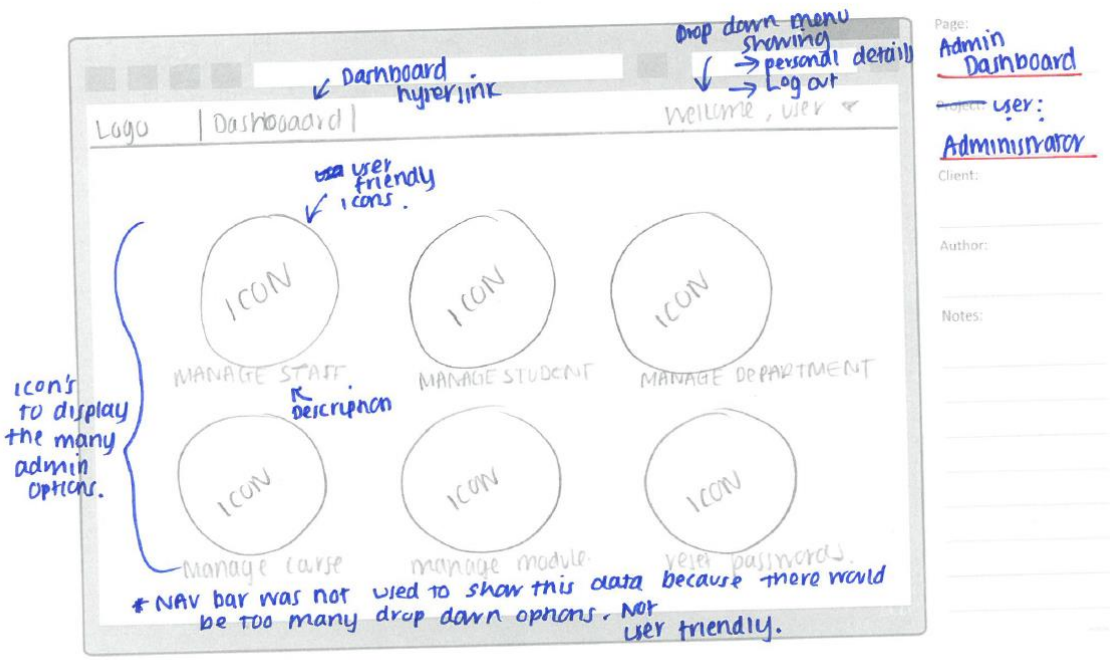
Personal Details

Student

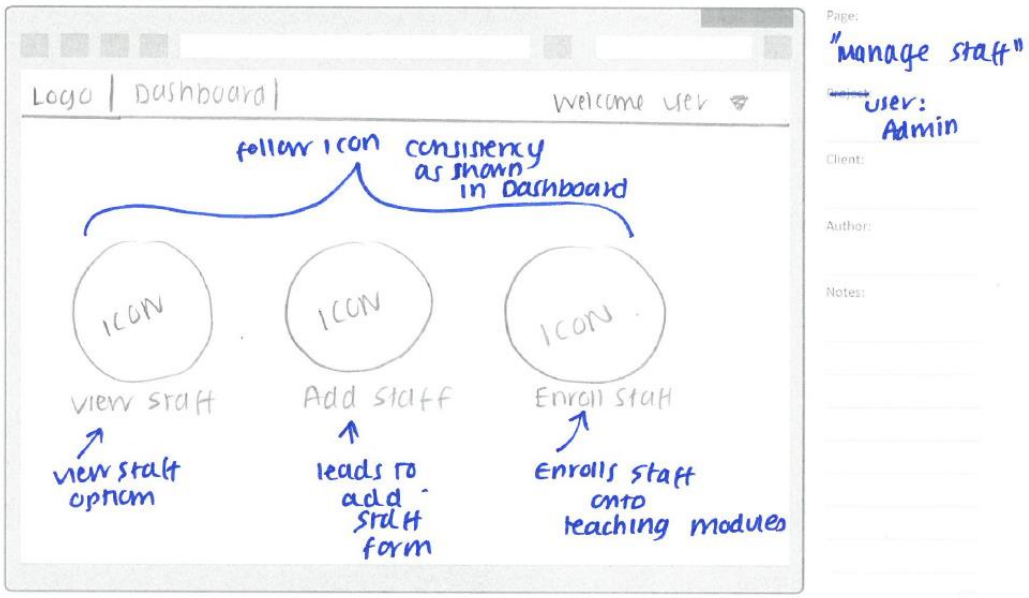


View Module Attendance

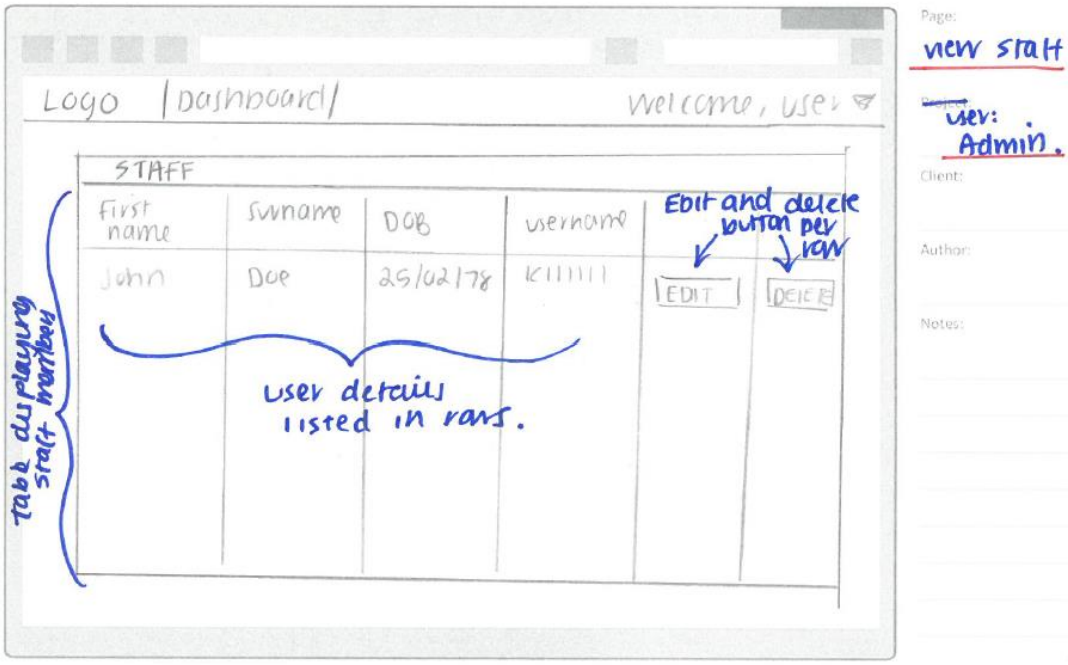
Admin:



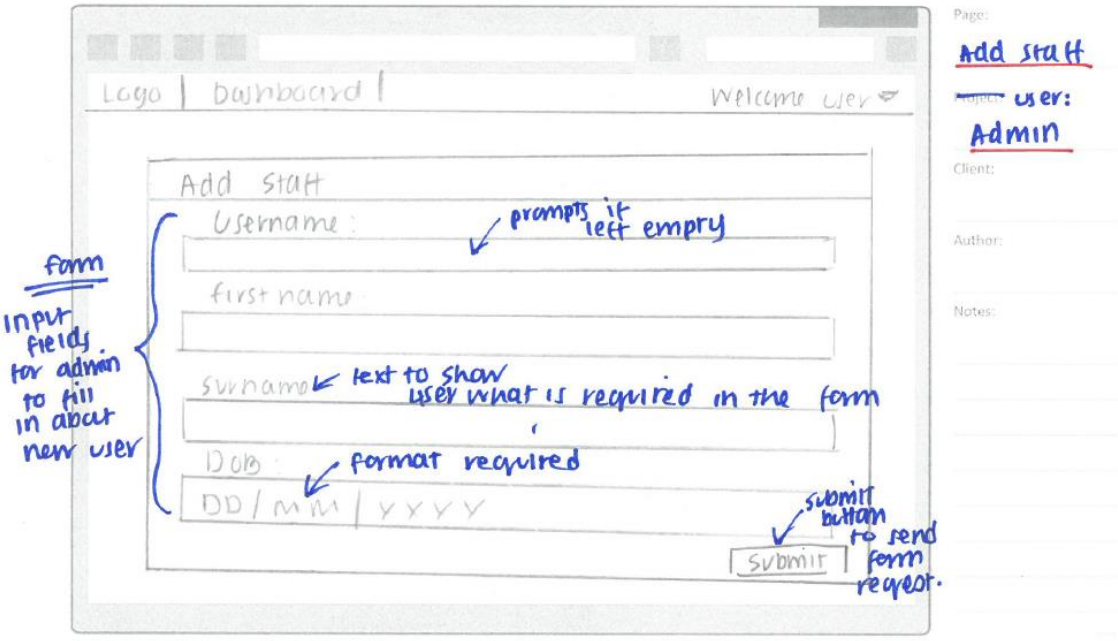
Admin Dashboard



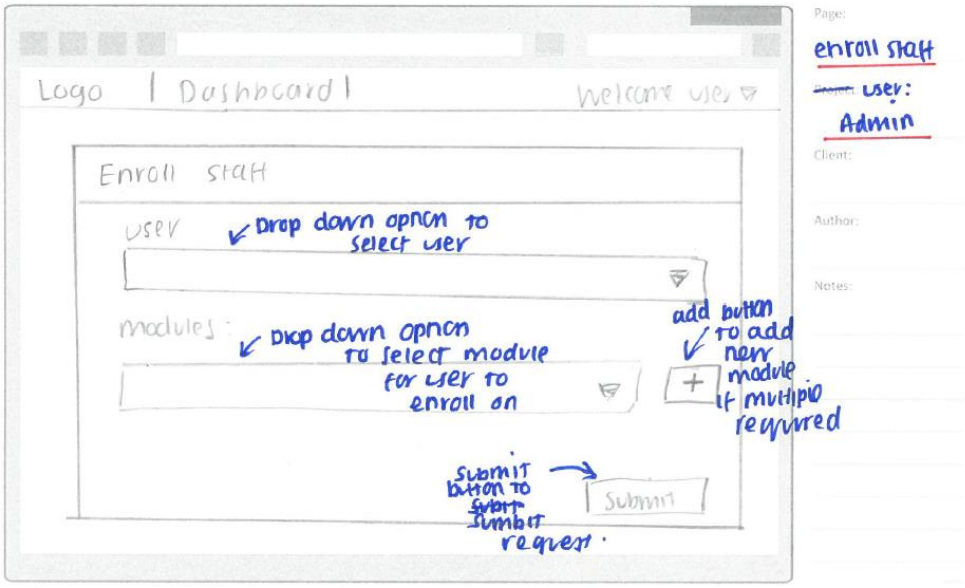
Manage Staff



View Staff

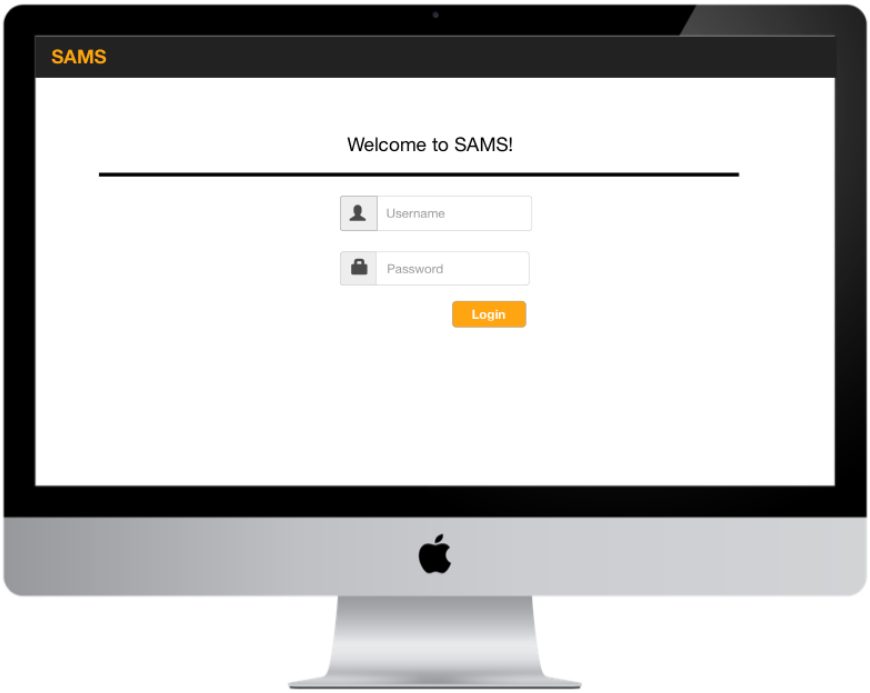


Add Staff



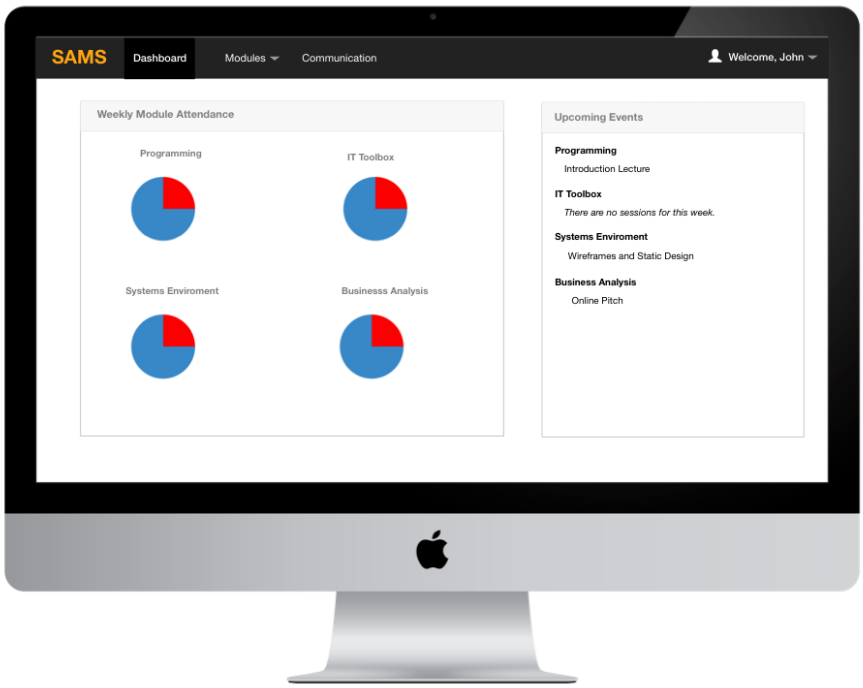
Enroll Staff

Appendix O: High Fidelity Wireframes

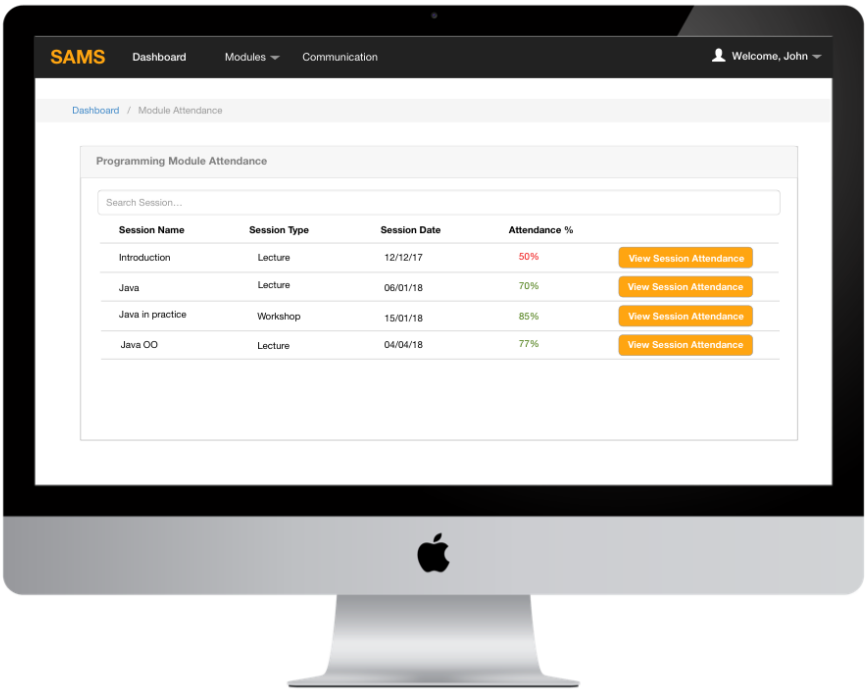


Login

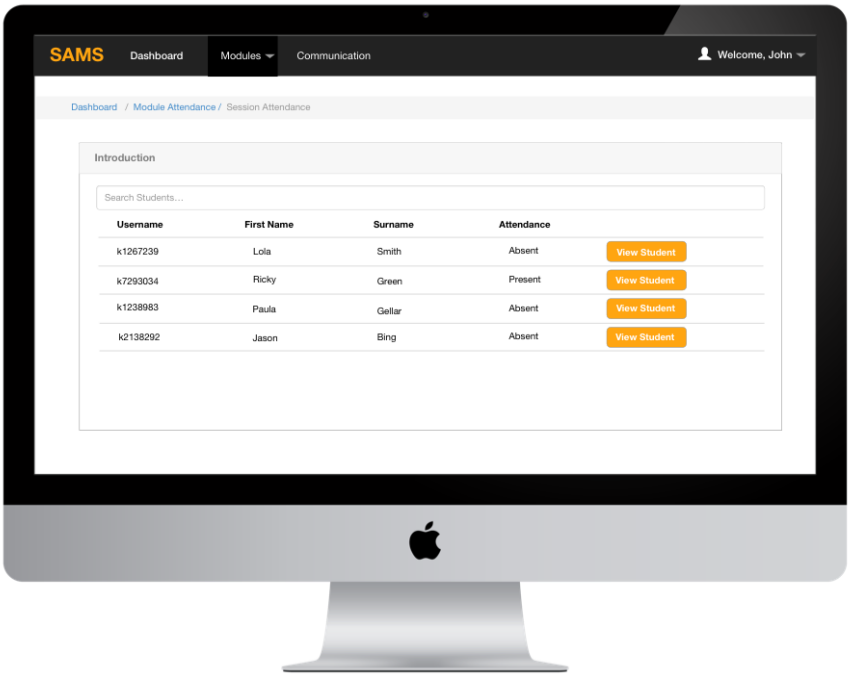
Lecturer



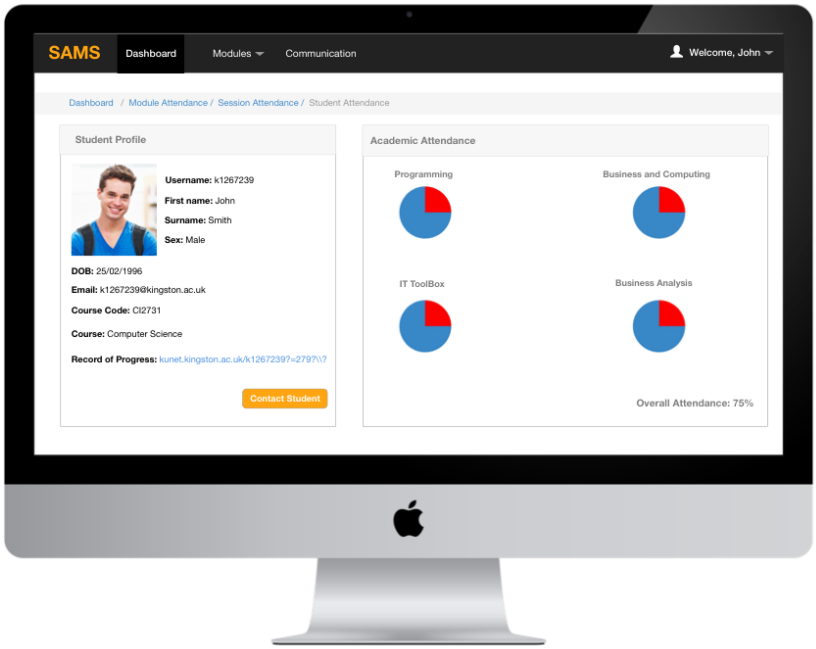
Lecturer Dashboard



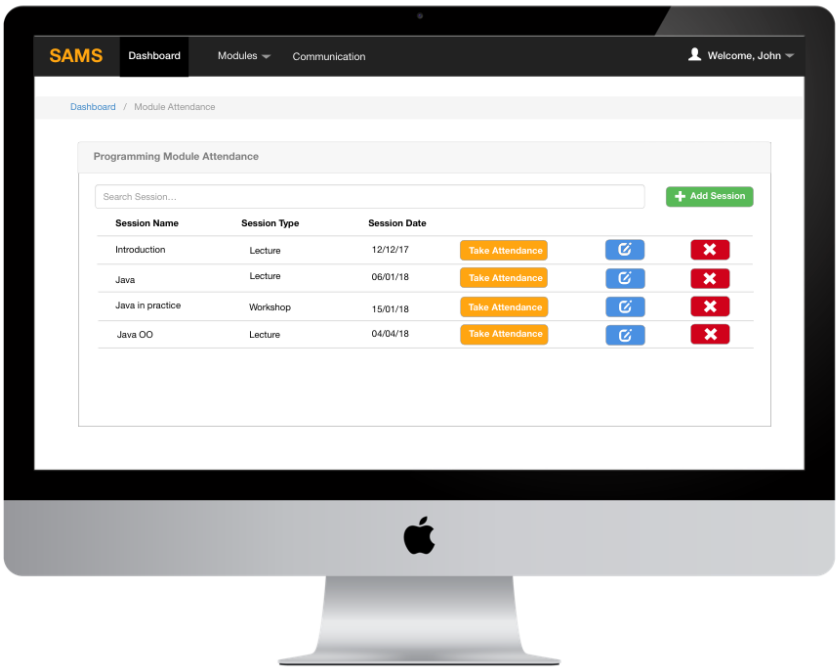
Breakdown of Module Attendance



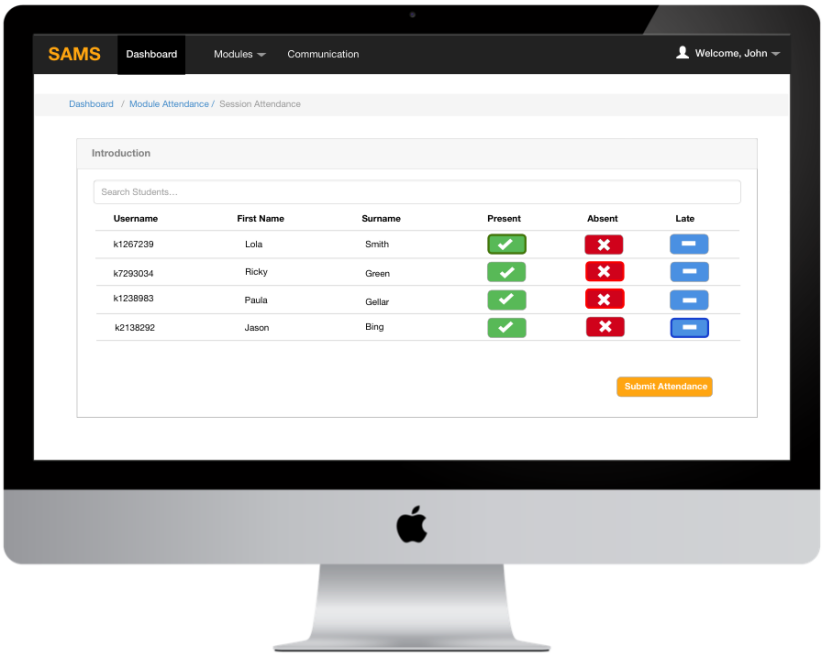
Breakdown of Session Attendance



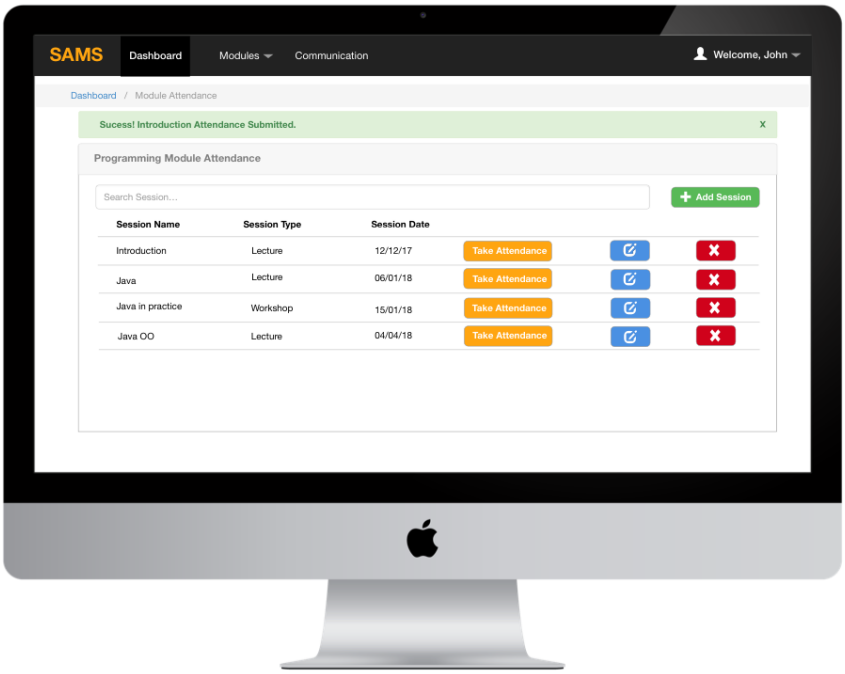
Attendance of individual Student



Selected Module through Dropdown to View Sessions

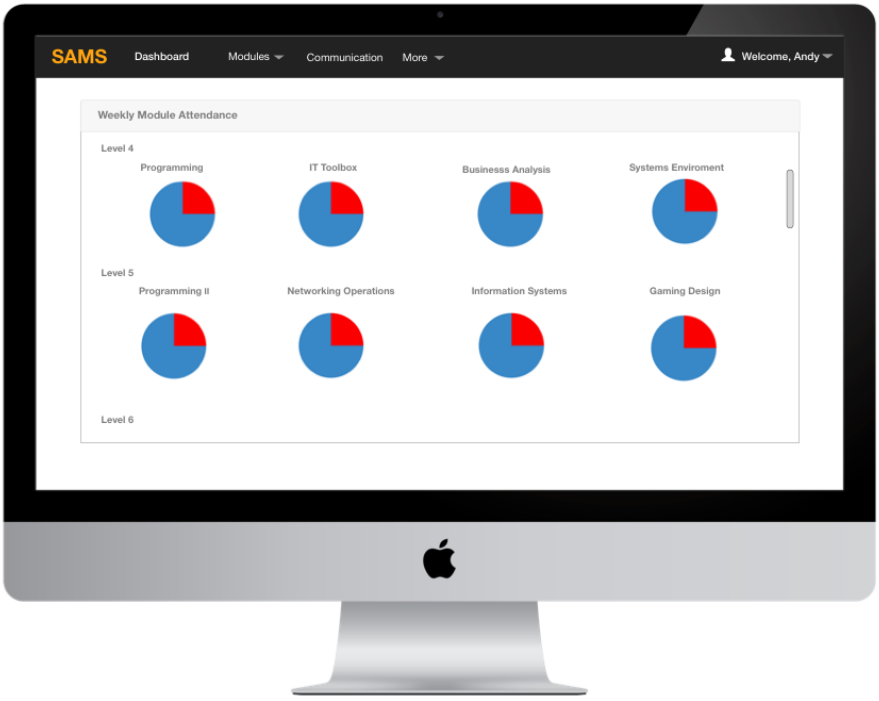


Lecturer take Attendance



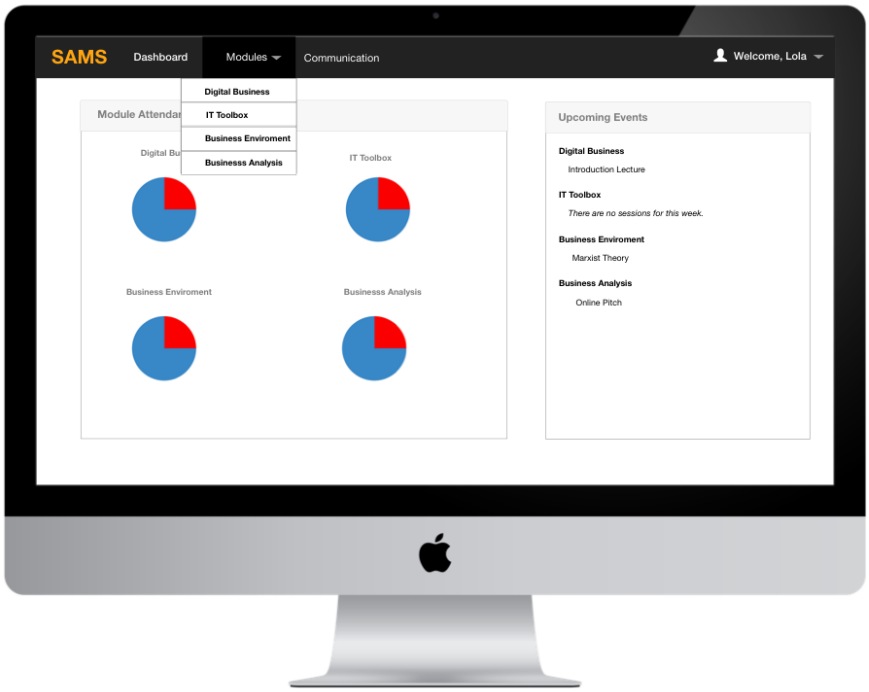
Attendance successfully submitted

Course Director: Will have a similar flow as the lecturer with extended features.

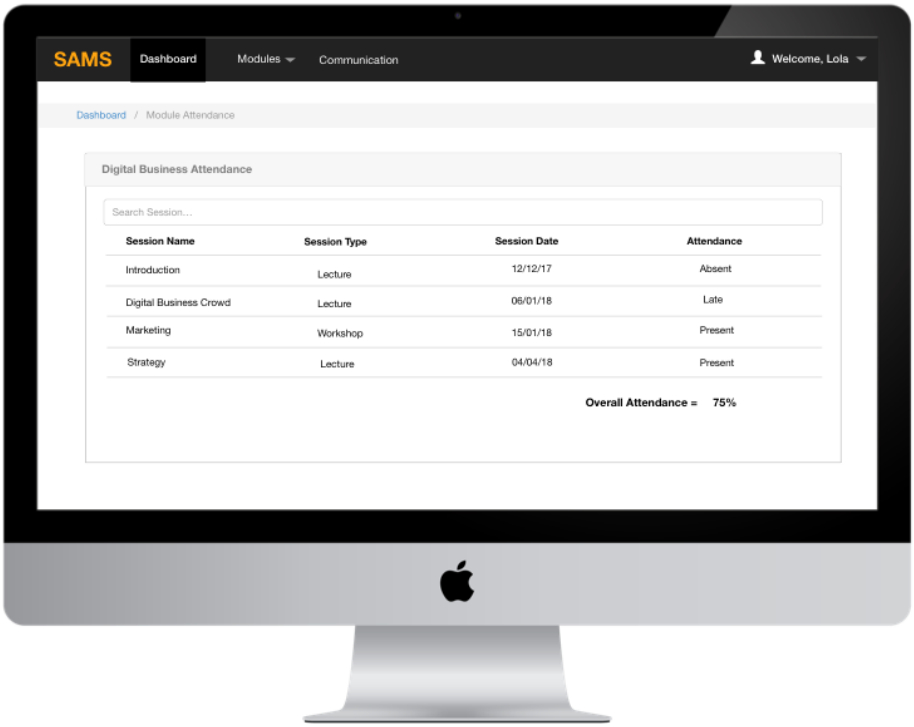


Course director View of Modules for the course

Student:

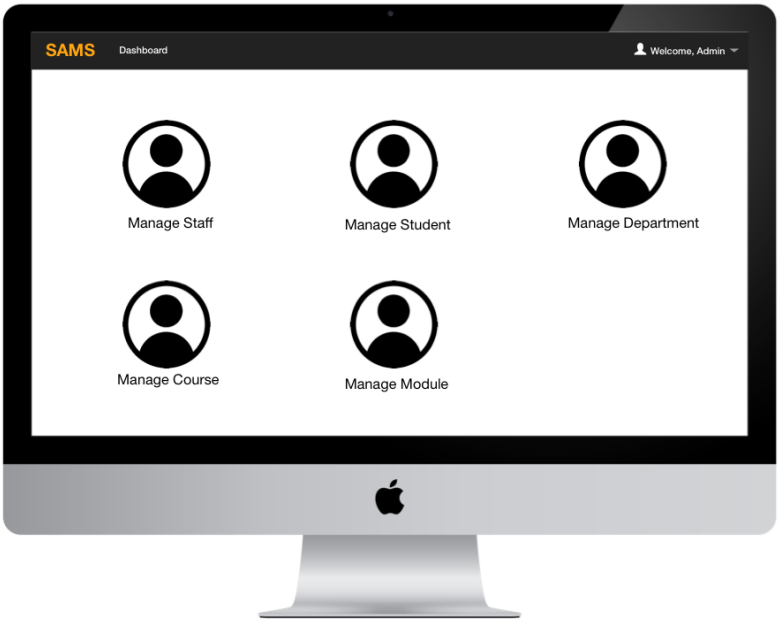


Student Dashboard & Module Options

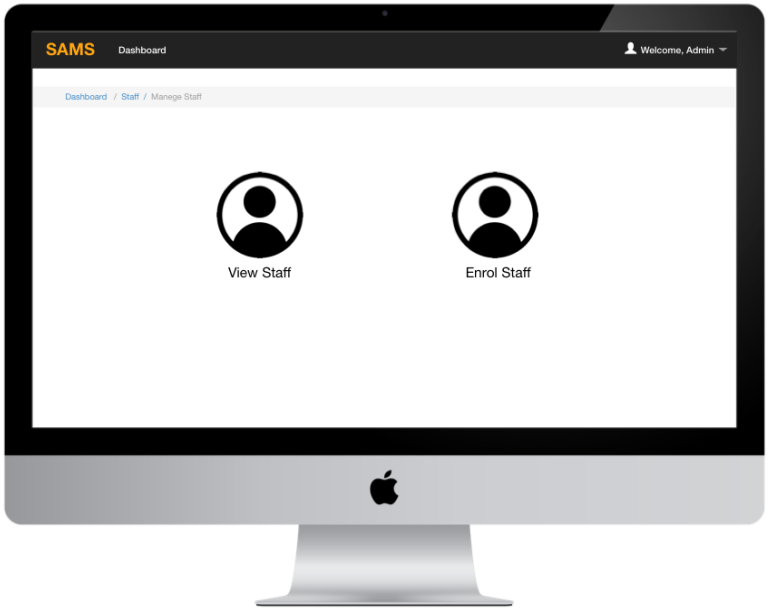


Student Dashboard & Module Options

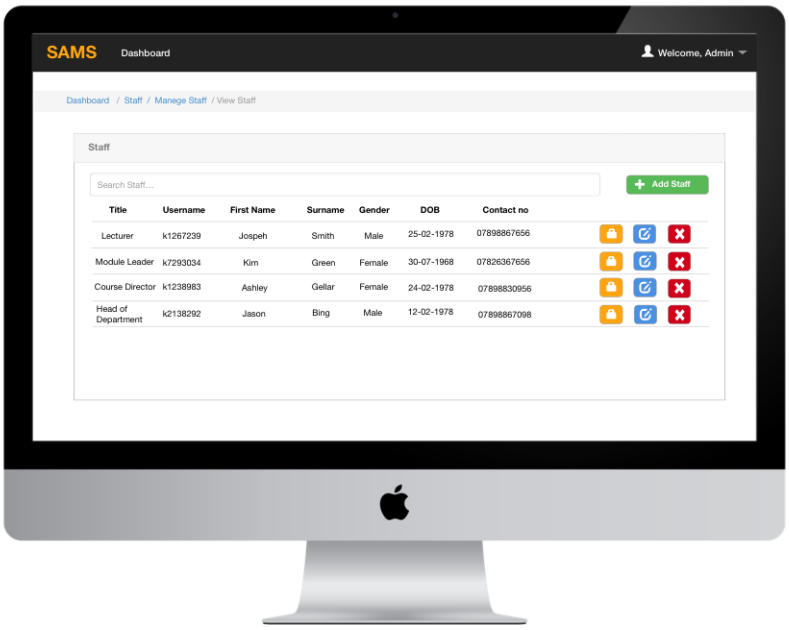
Administrator: Will have a similar work flow for each component of the system.



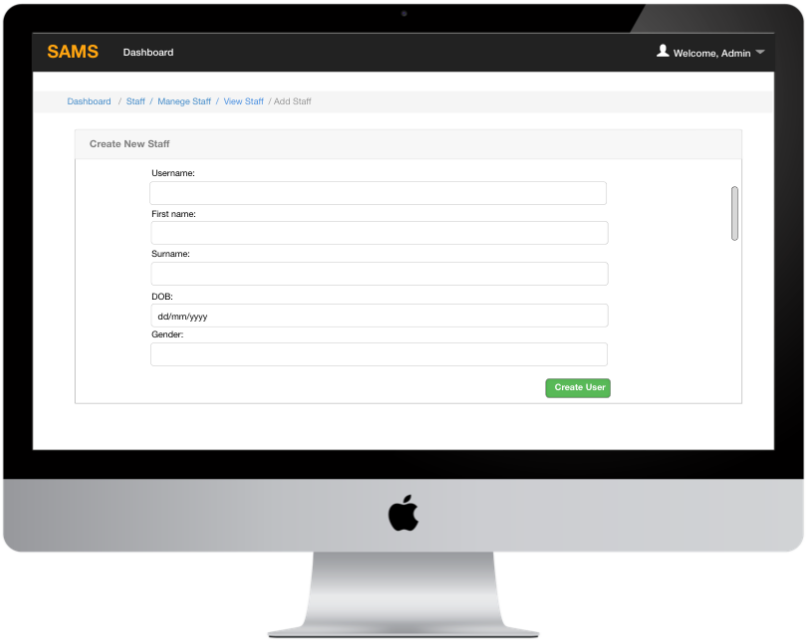
Admin Dashboard



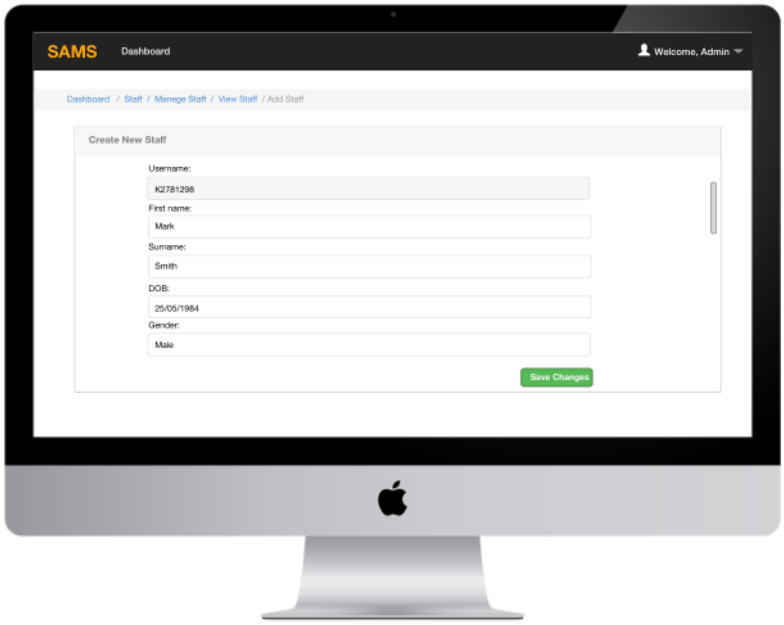
Manage Staff Options



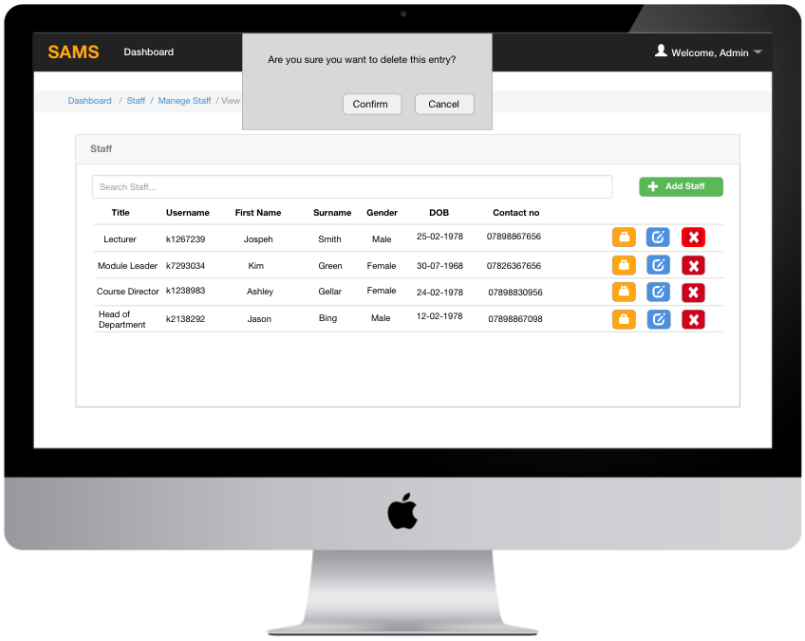
View Staff and Additional Options



Add New Staff



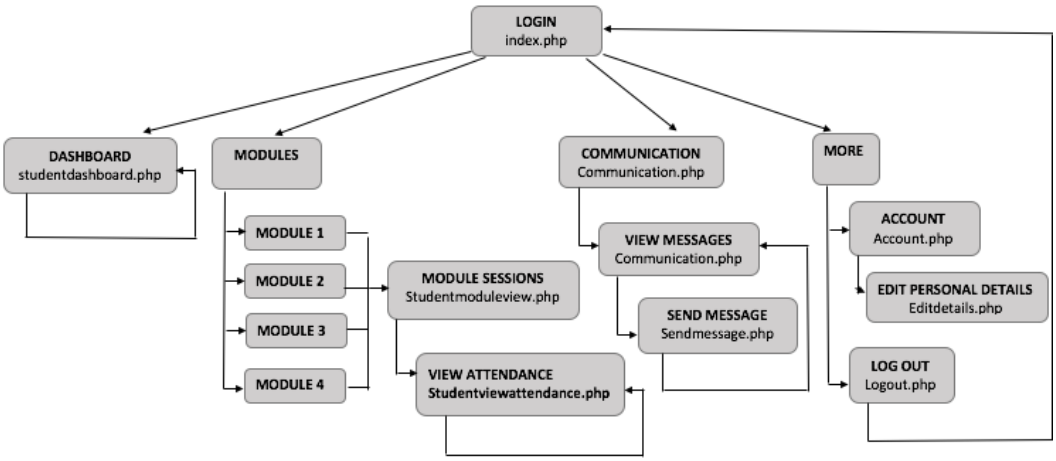
Edit Staff



Delete Staff

Appendix P: Full Navigation Diagrams/ Site Map

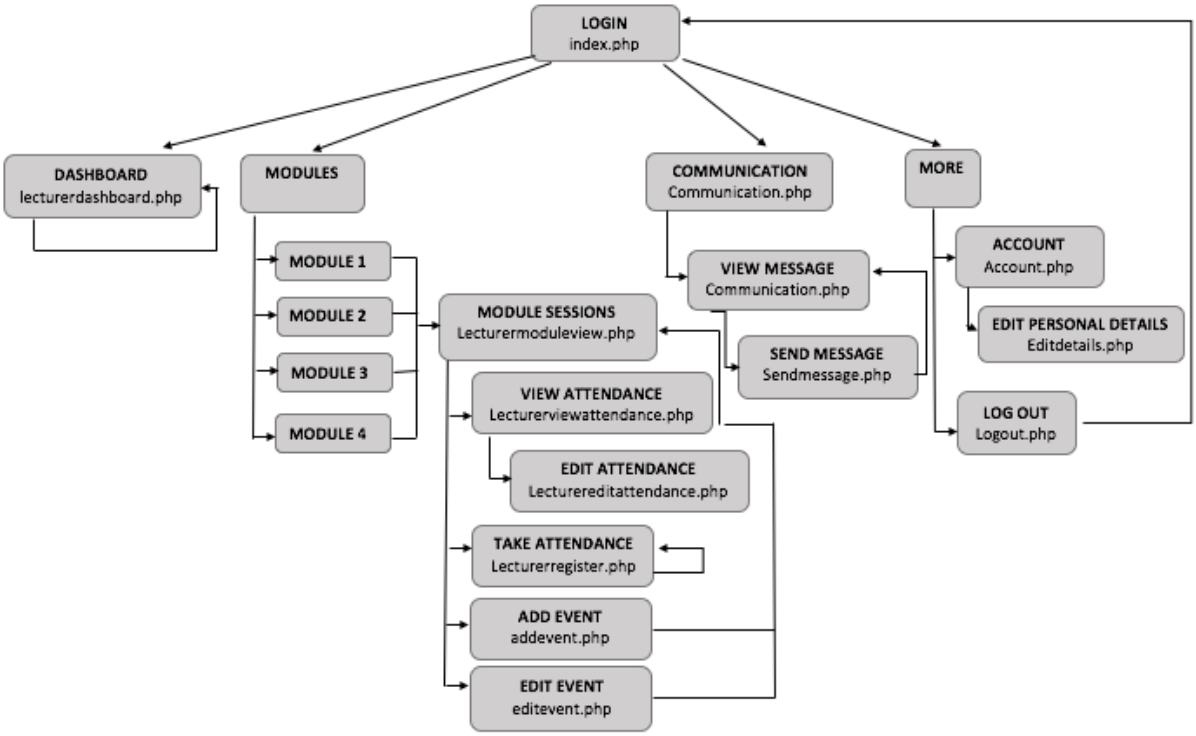
Student:



Student Navigation Diagram/Site Map

The Student navigation follows the procedure mentioned above, after logging in they are directed onto their dashboard. The top layer of the diagrams, indicates the options designed on the navigation bar. There are two thought out two drop down menus in this design, “Module” and “ More” leading to further a selection of tasks such as viewing attendance. There are some options which have return paths to the same page or the ending of the users path, for example when view attendance is selected there is no more options beyond which when clicked will lead to its own page, deeming it the end of this path. “More” shows the route for log out directing back to the log in screen as shown, ending the user cycle for the system journey.

Lecturer:



Lecturer Navigation Diagram/Site Map

Lecturer has an extremely similar lay out to student, top layer being the navigation bar with the same options. However they have further access rights when coming to publishing and editing student attendance. Once a Lecturer has submitted attendance, the page will refresh itself with a confirmation message, alerting the Lecturer of the new submission. This explains the recursive arrow shown in the figure above. To expand, Lecturers also have minimal creation rights in regards to events. Lecturers are able to create and edit events from the module sessions page.

Appendix Q: Methods in populating and validating data

As requested by the client, the system had to be robust enough to sustain vast amounts of data if it was to be perceived as industry standard. Fictional data was gathered from various websites. Once the data was gathered, Microsoft Excel was utilised to enable bulk population. Functions such as RANDBETWEEN(*top*, *bottom*) were used to easily generate data within a matter of minutes by returning a random number between a bottom and top choice. Each table was generated and saved as a MS Comma-Separated Value(CSV) file. CSV files stores tabular data in plain readable text, each line read as a data record to be inserted. Figure 5.10 demonstrates this approach with a segment of MS Excel file ready to be imported. Figure 5.11 illustrates the corresponding imported data through the PhpMyAdmin menu. This inflicted a positive impact on the project schedule, as large amounts of data were able to be implemented in a matter of days. An alternative to this approach is through the PhpMyAdmin GUI. This approach is perfect for tables which require little data or for those where the CSV file cannot be successfully inserted. Nevertheless, this approach is not practical for large amounts of data which could have had a detrimental effect on the project schedule.

In order validate the data has been imported correctly, MS Excel provides the ROW() function which calculates the accumulative rows within the file. Similarly, PhpMyAdmin provides a numeric representation of each table's data entries, demonstrated in Figure 5.12. It showcases the number of rows (data entries) for each table with an accumulated sum of all table entries. By comparing the two figures and ensuring they are equal, it can be assumed that the data has been imported correctly

k2715306	mc9jZx6Sb6Sb0ZCT3FLj	Aaron	Lamb	Male	09/12/1983	7416120083	k2715306@kingston.ac.uk	2
k4623684	6psmhklSLphPl4IxXUEa	Abbey	Lawson	Female	22/08/1971	7144004932	k4623684@kingston.ac.uk	3
k6433472	CsV8qmEtMVDvNF9PH1gl	Abbie	Jean	Female	22/01/1998	7975350786	k6433472@kingston.ac.uk	1
k8282273	1lqmLM0HLVOTfzokJVoJ	Abby	Jefferies	Female	17/12/1975	7552271699	k8282273@kingston.ac.uk	4
k4863485	g8fDJmVk9nblEOdmoztr	Parthenia	Jeffers	Female	25/07/1985	7602100064	k4863485@kingston.ac.uk	1
k6052265	UXDaPi8E51WNcKRCmdoZ	Moises	Jefferson	Female	02/05/1988	7723967936	k6052265@kingston.ac.uk	2
k3025031	8XQd9Ol4UFgYMGTwvl6j	Lottie	Jeffery	Female	04/07/1986	7661682206	k3025031@kingston.ac.uk	2
k5789105	Zkp3g6VgF5vRt3mD4tSy	Yon	Jeffrey	Female	23/01/1974	7653384751	k5789105@kingston.ac.uk	4
k9441238	UfWB9bCj7oQE1bbTQEI7	Syreeta	Jeffries	Female	10/02/1990	7657755014	k9441238@kingston.ac.uk	3
k6369253	t7lhl9HBaSn6cHX81Qls	Kina	Jenkins	Female	15/02/1990	7435432566	k6369253@kingston.ac.uk	3

Figure 2.10: CSV File with Data Ready to Import

UserID	Username	Password	Firstname	Surname	Gender	DateOfBirth	ContactNo	EmailAddress	RolesID
280	k2715306	mc9jZx6Sb6Sb0ZCT3FLj	Aaron	Lamb	Male	0000-00-00	7416120083	k2715306@kingston.ac.uk	2
281	k4623684	6psmhklSLphPl4IxXUEa	Abbey	Lawson	Female	0000-00-00	7144004932	k4623684@kingston.ac.uk	3
282	k6433472	CsV8qmEtMVDvNF9PH1gl	Abbie	Jean	Female	0000-00-00	7975350786	k6433472@kingston.ac.uk	1
283	k8282273	1lqmLM0HLVOTfzokJVoJ	Abby	Jefferies	Female	0000-00-00	7552271699	k8282273@kingston.ac.uk	4
284	k4863485	g8fDJmVk9nblEOdmoztr	Parthenia	Jeffers	Female	0000-00-00	7602100064	k4863485@kingston.ac.uk	1
285	k6052265	UXDaPi8E51WNcKRCmdoZ	Moises	Jefferson	Female	0000-00-00	7723967936	k6052265@kingston.ac.uk	2
286	k3025031	8XQd9Ol4UFgYMGTwvl6j	Lottie	Jeffery	Female	0000-00-00	7661682206	k3025031@kingston.ac.uk	2
287	k5789105	Zkp3g6VgF5vRt3mD4tSy	Yon	Jeffrey	Female	0000-00-00	7653384751	k5789105@kingston.ac.uk	4
288	k9441238	UfWB9bCj7oQE1bbTQEI7	Syreeta	Jeffries	Female	0000-00-00	7657755014	k9441238@kingston.ac.uk	3
289	k6369253	t7lhl9HBaSn6cHX81Qls	Kina	Jenkins	Female	0000-00-00	7435432566	k6369253@kingston.ac.uk	3

Figure 5.11: Data important via CSV file

Table	Action	Rows	Type	Collation	Size	Overhead
AttendanceStatus	Browse Structure Search Insert Empty Dro	4	InnoDB	latin1_swedish_ci	16 KiB	-
Course	Browse Structure Search Insert Empty Dro	4	InnoDB	latin1_swedish_ci	96 KiB	-
CourseModule	Browse Structure Search Insert Empty Dro	7	InnoDB	latin1_swedish_ci	64 KiB	-
Department	Browse Structure Search Insert Empty Dro	3	InnoDB	latin1_swedish_ci	48 KiB	-
Event	Browse Structure Search Insert Empty Dro	38	InnoDB	latin1_swedish_ci	80 KiB	-
EventType	Browse Structure Search Insert Empty Dro	2	InnoDB	latin1_swedish_ci	16 KiB	-
LecturerModule	Browse Structure Search Insert Empty Dro	6	InnoDB	latin1_swedish_ci	48 KiB	-
Module	Browse Structure Search Insert Empty Dro	108	InnoDB	latin1_swedish_ci	32 KiB	-
Roles	Browse Structure Search Insert Empty Dro	6	InnoDB	latin1_swedish_ci	16 KiB	-
StudentAttendance	Browse Structure Search Insert Empty Dro	1,669	InnoDB	latin1_swedish_ci	384 KiB	-
StudentCourse	Browse Structure Search Insert Empty Dro	5	InnoDB	latin1_swedish_ci	80 KiB	-
StudentModule	Browse Structure Search Insert Empty Dro	283	InnoDB	latin1_swedish_ci	80 KiB	-
User	Browse Structure Search Insert Empty Dro	1,413	InnoDB	latin1_swedish_ci	320 KiB	-
13 tables	Sum	3,548	InnoDB	latin1_swedish_ci	1.3 MiB	0 B

Figure 5.12: Number of Database Record Rows in Each Table

Appendix R: $_POST vs $_GET

GET Method:

- Appends form data onto the URL, making it visible to all. Therefore, this should not be used in the cases of sensitive data, however this method is very advantageous in cases of non-sensitive data such as query strings.
- The length of the URL is limited to around 3000 characters, therefore GET methods may not be able to hold the data sent through a form.

POST method:

- This method is more secure as the user-entered information is not visible in the URL; therefore, this method would be better suited for the cases where sensitive data is involved.
- For the POST method, there is a larger limit to the total data which can be passed through a form.

Both methods are super global variable provided by PHP which enables the developer to access all the information passed through $_POST/$_GET or submitted through a HTML form via the method = "get|post" attribute.

Appendix S: Entity-Relationship Design and Database Construct Testing

#	FR	SQL Query	Pass/Fail
1	Create User	INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID) VALUES ('K9999999', 'password', 'Paul', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')	Pass
	1 row inserted. Inserted row id: 1774 (Query took 0.0199 seconds.) `INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID) VALUES('K9999999', 'password', 'Paul', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')`		
2	View Student	SELECT Roles.Description, User.UserID, User.Username, User.Password, User.Firstname, User.Surname, User.Gender, User.DateOfBirth, User.ContactNo, User.EmailAddress, User.LastLogin, Course.CourseName, Course.CourseCode FROM (((`User` INNER JOIN Roles ON Roles.RolesID = User.RolesID) LEFT JOIN StudentCourse ON StudentCourse.UserID = User.UserID) LEFT JOIN Course ON Course.CourseID = StudentCourse.CourseID) WHERE Roles.RolesID=1 ORDER BY User.Surname ASC	Pass
	Showing rows 0 - 24 (293 total, Query took 0.0018 seconds.) [Surname: **ALBERT - CABRERA**] `SELECT Roles.Description, User.UserID, User.Username, User.Password, User.Firstname, User.Surname, User.Gender, User.DateOfBirth, User.ContactNo, User.EmailAddress, User.LastLogin, Course.CourseName, Course.CourseCode FROM (((`User` INNER JOIN Roles ON Roles.RolesID = User.RolesID) LEFT JOIN StudentCourse ON StudentCourse.UserID = User.UserID) LEFT JOIN Course ON Course.CourseID = StudentCourse.CourseID) WHERE Roles.RolesID=1 ORDER BY User.Surname ASC`		
3	Edit User	UPDATE User SET Firstname = 'simon' WHERE Username = 'k1111113'	Pass
	1 row affected. (Query took 0.0032 seconds.) `UPDATE User SET Firstname = 'simon' WHERE Username = 'k1111113'`		
5	Delete User		Pass
	1 row deleted. (Query took 0.0052 seconds.) `DELETE FROM User WHERE User.UserID = 654`		
4	Reset Password	UPDATE User SET Password = "learning" Where User.UserID = 1796	Pass

	1 row affected. (Query took 0.0027 seconds.) `UPDATE User SET Password = "learning" Where User.UserID = 1796`		
5	Submit Attendance	UPDATE StudentAttendance SET AttendanceStatusID = 1 WHERE StudentAttendance.userID = 21 and StudentAttendance.EventID = 3	Pass
	1 row affected. (Query took 0.0052 seconds.) `UPDATE StudentAttendance SET AttendanceStatusID = 1 WHERE StudentAttendance.userID = 21 and StudentAttendance.EventID = 3`		
6	View Module Attendance	SELECT Event.Title, Event.EDate, AttendanceStatus.Description, EventType.EventDescription FROM StudentAttendance INNER JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceStatusID INNER JOIN Event ON Event.EventID = StudentAttendance.EventID INNER JOIN EventType ON EventType.EventTypeID = Event.EventTypeID INNER JOIN Module ON Module.ModuleID = Event.ModuleID WHERE Module.ModuleID = 12 ORDER BY Event.EDate ASC	Pass
	Showing rows 0 - 24 (352 total, Query took 0.0014 seconds.) [EDate: **2018-02-03 - 2018-02-23**] `SELECT Event.Title, Event.EDate, AttendanceStatus.Description, EventType.EventDescription FROM StudentAttendance INNER JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceStatusID INNER JOIN Event ON Event.EventID = StudentAttendance.EventID INNER JOIN EventType ON EventType.EventTypeID = Event.EventTypeID INNER JOIN Module ON Module.ModuleID = Event.ModuleID WHERE Module.ModuleID = 12 ORDER BY Event.EDate ASC`		
7	View Module Attendance	SELECT Event.Title, Event.EDate, AttendanceStatus.Description, EventType.EventDescription FROM StudentAttendance INNER JOIN AttendanceStatus ON AttendanceStatus.AttendanceStatusID = StudentAttendance.AttendanceStatusID INNER JOIN Event ON Event.EventID = StudentAttendance.EventID INNER JOIN EventType ON EventType.EventTypeID = Event.EventTypeID INNER JOIN Module ON Module.ModuleID = Event.ModuleID WHERE Module.ModuleID = 12 ORDER BY Event.EDate ASC	Pass

Showing rows 0 - 24 (39 total, Query took 0.0005 seconds.)

```
SELECT StudentAttendance.AttendanceStatusID FROM StudentAttendance INNER JOIN Event ON Event.EventID = StudentAttendance.EventID INNER JOIN Module ON Module.ModuleID = Event.ModuleID WHERE Module.ModuleID = 7
```

Appendix T: Database Robustness & Querying the Database

	Test Description	SQL Statement	Integrity	Result
1	Changing a PK to a value which already exists	UPDATE User SET User.UserID = 13 WHERE User.UserID = 14	Entity	Pass

	Error
	SQL query:
	`UPDATE User SET User.UserID = 13 WHERE User.UserID = 14`
	MySQL said:
	`#1062 - Duplicate entry '13' for key 'PRIMARY'`			

2	Entering a FK value which doesn't match the values in the parent table.	UPDATE StudentAttendance SET StudentAttendance.AttendanceStatusID = 4 WHERE StudentAttendance.UserID = 21 AND StudentAttendance.EventID = 3;	Referential	Pass

	Error
	SQL query:
	`UPDATE StudentAttendance`
	`SET StudentAttendance.AttendanceStatusID = 4`
	`WHERE StudentAttendance.UserID = 21 AND StudentAttendance.EventID = 3`
	MySQL said:
	`#1452 - Cannot add or update a child row: a foreign key constraint fails (`db_k1403042`.`StudentAttendance`, CONSTRAINT `dsd` FOREIGN KEY (`AttendanceStatusID`) REFERENCES `AttendanceStatus` (`AttendanceStatusID`))`			

3	Inserting a record with NULL values.	INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID) VALUES ('', 'password', 'Joe', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')	Domain	Pass

	Error
	SQL query:
	`INSERT INTO User (Username, Password, Firstname, Surname, Gender, DateOfBirth, ContactNo, EmailAddress, RolesID)`
	`VALUES ('', 'password', 'Joe', 'Smith', 'Male', '25-02-1988', '07867765654', 'k1234231', '1')`
	MySQL said:
	`#1062 - Duplicate entry '' for key 'Username'`			

4	Inserting a string value into an INT attribute.	INSERT INTO Department (DepartmentID, DepartmentName, HeadOfDepartmentID) VALUES ('School of Arts', 'Bill Jones')	Domain	Pass

	Error
	SQL query:
	`INSERT INTO Department (DepartmentID, DepartmentName, HeadOfDepartmentID)`
	`VALUES ('School of Arts', 'Bill Jones')`
	MySQL said:
	`#1136 - Column count doesn't match value count at row 1`			

Appendix U: Database Robustness & Querying the Database

Database Robustness

It was a request permitted by the client, to implement enough data to mimic the typical capacity when the product is taken to industry. Therefore, the developer implemented over a thousand records to ensure the database was durable enough to hold the capacity of the SQL requests. Therefore, the SQL requests seen in *appendix J* were re-tested to assure the results were still achieved. These results were proven 100% successful. PhpMyAdmin provides a figure representing's the number of rows stored in the database, demonstrated in Figure 6.1. In addition, each table's data entries are accumulated to depict a subtotal of all entries. Through these figures, it can be assumed that enough data has been imported. These processes were shown to the client for final clarification of requirements and durability of the system and database.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	AttendanceStatus	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	16 KiB	-
☐	Course	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	96 KiB	-
☐	CourseModule	Browse Structure Search Insert Empty Drop	7	InnoDB	latin1_swedish_ci	64 KiB	-
☐	Department	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	48 KiB	-
☐	Event	Browse Structure Search Insert Empty Drop	38	InnoDB	latin1_swedish_ci	80 KiB	-
☐	EventType	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB	-
☐	LecturerModule	Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	48 KiB	-
☐	Module	Browse Structure Search Insert Empty Drop	108	InnoDB	latin1_swedish_ci	32 KiB	-
☐	Roles	Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	16 KiB	-
☐	StudentAttendance	Browse Structure Search Insert Empty Drop	1,669	InnoDB	latin1_swedish_ci	384 KiB	-
☐	StudentCourse	Browse Structure Search Insert Empty Drop	5	InnoDB	latin1_swedish_ci	80 KiB	-
☐	StudentModule	Browse Structure Search Insert Empty Drop	283	InnoDB	latin1_swedish_ci	80 KiB	-
☐	User	Browse Structure Search Insert Empty Drop	1,413	InnoDB	latin1_swedish_ci	320 KiB	-
	13 tables	Sum	3,548	InnoDB	latin1_swedish_ci	1.3 MiB	0 B

SQL queries testing database integrity

Querying the Database

The SAMS web application is designed to work closely in line with the database in order to meet the requirements for the system. The application will be responsive on many devices, used across different connection speeds and will operate with many users concurrently accessing the system. As a result, it is crucial to utilise the most efficient SQL queries to execute all system and user goals. This will in turn improve the UX, as the request/response time can be quickly and concurrently executed.
Data Manipulation Language (DML) is used to insert, create, delete and retrieve data. Before executed, each statement will be stored within a variable to prevent the risk of SQL injection. This will also uphold the maintainability of the code throughout the project.

```
$sql= "SELECT Module.ModuleName, Module.ModuleID
                    FROM ((Module
                    INNER JOIN LecturerModule ON LecturerModule.ModuleID = Module.ModuleID)
                    INNER JOIN User on User.UserID = LecturerModule.UserID)
                    WHERE User.UserID= $userid";
```

SQL Statement Stored within a Variable to Assist with Preventing SQL Injection

Throughout the implementation process the queries begin to vary as the functional requirements unfold. The general structure follows a linear approach whereby it is essential to SELECT the attributes FROM the desired table. This can then be enhanced through conditions to meet a

criterion i.e. WHERE UserID = 3. However, as the complexity of the system increases the requirement to gather data from numerous tables do also. In order to perform these requests and optimise query processing, a range of JOIN queries will need to be executed, which vary from the following:

- Inner Join: Will return all records which have matching records in both tables.
- Left Join: Will return all the records from the left table, and the matching records in the right table.
- Right Join: Will return all the records from the left table, and the matching records in the right table.
- Outer Join: Will return all records and matching records in either left or right table.

Whilst testing the different query structures, in was concluded the combination of inner joins and left joins were the most suitable for the SAMS web application. This conclusion was derived from a test conducted to measure the speed and efficiency of different queries. As the application is be significantly reliant on the database the results from different join techniques were compared. The two which seemed to have a detrimental effect on the processing was the OUTER and INNER joins. The results depicted that inner joins ran the quickest. In reflection to this testing inner join queries were utilised to ensure query processing was executed efficiently and quickly.

Appendix V: Black Box Testing for Functional Requirements

Test ID	FR	Test Description	Expected Result	Result (Pass/Fail)
9	2 Search User	1. User enters input into search bar.	System displays results matching criteria.	Pass
10	3 View User	1. User clicks 'View Staff/Student'	System displays all records of users with user details.	Pass
11	4 Edit User	1. User clicks 'View Staff/Student'	System displays all records of users with user details.	Pass
12		2. User clicks edit icon.	System generates form with selected user's details.	Pass
13		3. User tries to edit the selected users Username.	System displays disabled field. User is unable to edit.	Pass
14		4. User edits enters string of characters into email address field.	System displays alert notifying user that '@' is required in the input field.	Pass
15		5. User attempt to leave field empty	System alerts user that field is left empty.	Pass
16		6.User submits request to save edited detail	System displays confirmation message to successful editing.	Pass
17	5 Delete User	1. User clicks 'View Staff/Student'	System displays all records of users with user details.	Pass
18		2. User clicks delete icon.	System will display confirmation message of deleting user entry.	Pass
19		3. User clicks 'cancel' for delete confirmation message.	System removes confirmation message and displays original set of users with their user details.	Pass
20		4. User clicks 'confirm' for delete confirmation message.	System displays confirmation message of deleted user entry.	Pass
21	6 Create Module	1. User clicks 'Manage Module'	System displays further options about modules.	Pass
22		2. User clicks 'View Modules'	System displays all records of modules with module details	Pass
23		3. User clicks 'Add Module'	System displays form to add new module.	Pass
24		4. User leaves input	System alerts user that	Pass

		field empty.	field is left empty.	
25		5. User submits request.	System alerts user that field is left empty.	Pass
26	7 Create Course	1. User clicks 'Manage Course'	System displays further options about courses.	Pass
27		2. User clicks 'View Course'	System displays all records of courses with course details.	Pass
28		3. User clicks 'Add Course'	System displays form to add new course.	Pass
30		4. User leaves input field empty.	System alerts user that field is left empty.	Pass
31		5. User submits request.	System alerts user that field is left empty.	Pass
32	8 Search Module	1. User enters input into search bar.	System displays results matching criteria.	Pass
33	9 Search Course	1. User enters input into search bar.	System displays results matching criteria.	Pass
34	10 View Module	1. User clicks 'View Modules'	System displays all records of modules with module details.	Pass
35	11 View Course	1. User clicks 'View Course'	System displays all records of Courses with course details.	Pass
36	12	Edit Module (Out of Project Scope)		
37	13	Edit Course (Out of Project Scope)		
38	14	Delete Module (Out of Project Scope)		
39	15	Delete Course (Out of Project Scope)		
40	16 Create Department	1. User clicks 'Manage Department'	System displays further options about departments.	Pass
41		2. User clicks 'View Department'	System displays all records of courses with course details.	Pass
42		3. User clicks 'Add department'	System displays form to add new department.	Pass
43		4. User leaves input field empty.	System alerts user that field is left empty.	Pass
44		5. User submits request.	System alerts user that field is left empty	Pass
41	17 Search Department	1. User enters input into search bar.	System displays results matching criteria.	Pass
42	18 View Department	1. User clicks 'View Department'	System displays all records of departments with department details.	Pass
43	19	Edit Department (Out of Project Scope)		
44	20	Delete Department (Out of Project Scope)		
45	21	Reset User Password (Out of Project Scope)		
46	22 Take Attendance	1. Lecturer click 'modules' from	System displays drop down menu with	Pass

		navigation bar	modules.	
47		2. Lecturer clicks on module.	System displays lists of sessions within module	Pass
48		3. Lecturer selects session to take attendance on.	System displays all students within session selected.	Pass
49		4. Lecturer clicks button for choice of attendance.	System highlights button of choice	Pass
50	23 Confirm Attendance	5. Lecturer submits attendance details.	System displays success message for submission	Pass
51				
52	24 View Attendance	1. Lecturer click 'modules' from navigation bar	System displays drop down menu with modules.	Pass
53	25 Edit Attendance	2. Lecturer clicks on module.	System displays lists of sessions within module	Pass
54	26 View Attendance Report	3. Lecturer selects session to view attendance on.	System displays all students within session selected.	Pass
55		1. Lecture logs in	System displays attendance data on dashboard through pie charts	Pass
56		2. Lecturer Clicks on selected module pie chart	System displays list of session and attendance percentage	Pass
57		3. Lecturer clicks on selected session	System displays list of session and student attendance statuses.	Pass
58	27 Search Student	1. Lecture logs in	System displays attendance data on dashboard through pie charts	Pass
59	28 Attendance Report	2. Lecturer Clicks on selected module pie chart	System displays list of session and attendance percentage	Pass
60		3. Lecturer clicks on selected session	System displays list of session and student attendance statuses.	Pass
61		4. Lecturer Searches student	System finds results to search criteria	Pass
62		5. Lecturer clicks 'view student'	System displays students overall attendance for different modules.	Pass
63	44 View events (workshop or lecture)	1. Lecturer click 'modules' from navigation bar	System displays drop down menu with modules.	Pass
64		2. Lecturer clicks on module.	System displays lists of sessions within module	Pass
65		3. Lecturer selects session to view	System displays all students within session	Pass

		attendance on.	selected.	
66	45 Create event	1. Lecturer click 'modules' from navigation bar	System displays drop down menu with modules.	Pass
67		2. Lecturer clicks on module.	System displays lists of sessions within module	Pass
68		3. Lecturer clicks ' add an event'	System displays form for adding event.	Pass
69		4. Lecturer leaves input field empty	System prompts user of empty field	Pass
70		5. Lecturer submits field out form	System displays confirmation message	Pass
71	46 Edit event	1. lecturer clicks edit icon	System displays edit form.	Pass
72		4. Lecturer leaves input field empty	System prompts user of empty field	Pass
73		5. Lecturer submits field out form	System displays confirmation message	Pass
74	47 Delete Event	1. User clicks 'Module'	System displays drop down menu with modules.	Pass
75		2. User selects module	System displays lists of sessions within module	Pass
76		3. User clicks delete icon next to session name.	System displays confirmation prompt.	Pass
77		4. User clicks 'confirm' for delete confirmation message.	System displays success message of deleted entry	Pass
78		5. User clicks 'cancel' for delete confirmation message.	System hides the delete message.	Pass
79	28	Print Attendance Report (Out of Project Scope)		
80	29	Save Attendance Report (Out of Project Scope)		
81	30 View Course Attendance	1. Course Director clicks on option 'more' in navigation bar	System displays drop down menu.	Pass
82		2. Course Director clicks on course	System Displays module attendance for the courses.	Pass
83	31 View Module Attendance	1. Lecture logs in	System displays attendance data on dashboard through pie charts	Pass
84		2. Lecturer Clicks on selected module pie chart	System displays list of session and attendance percentage	Pass
85		3. Lecturer clicks on	System displays list of	Pass

		selected session	session and student attendance statuses.	
86	32	View Department Attendance (Out of Project Scope)		
87	33 View Student Comments	1. Lecturer clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
88	34 View Student Correspondence	1. Lecturer clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
89		2. Lecturer clicks "open" next to desired message.	System displays message from user.	Pass
90	35 Contact Student	1. Lecturer clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
91		2. Lecturer clicks "create new".	System displays new message modal.	Pass
92		3. Lecturer leaves input field empty	System prompts user for input.	Pass
93		4. Lecturer cancels message creation.	System displays inbox page again.	Pass
94		5. Lecturer clicks submit.	System displays success message.	Pass
95	36	Edit Personal Detail (Out of Project Scope)		
96	37	Reset Password (Out of Project Scope)		
97	38 View own attendance(Student)	1. Student clicks 'Modules' on navigation bar.	System displays dropdown menu of enrolled modules.	Pass
98		2. Student clicks on selected module.	System displays sessions within module and attendance	Pass
999		1. Student logs in	System displays pie chart showing student attendance	Pass
100		2. Student clicks drop down menu of account on navigation bar	System displays drop down options.	Pass
101		3. Student clicks 'my account'	System displays account details and attendance.	Pass
102	39	Upload Correspondence (Out of Project Scope)		
103	40 View Correspondence	1. Student clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
104		2. Student clicks "open" next to desired message.	System displays message from user.	Pass
105	42 Delete Correspondence	1. Student clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
106		2. Student clicks	System prompts user	Pass

		delete icon next to message.	with confirmation message.	
107		3. Student clicks confirm to the confirmation message	System displays success message that message is deleted.	Pass
108		4. Student clicks cancel to confirmation message.	System displays list of messages in inbox again	Pass
109	43 Contact Lecturer	1. Student clicks on the 'communication' in navigation bar	System displays list of messages from other users.	Pass
110		2. Student clicks "create new".	System displays new message modal.	Pass
111		3. Student leaves input field empty	System prompts user for input.	Pass
112		4. Student cancels message creation.	System displays inbox page again.	Pass
		5. Student clicks submit.	System displays success message.	Pass

Appendix W: Black Box Testing for Non-functional requirements

Test ID	NFR AREA	Testing Description	Expected Result	Result (Pass/Fail)
1	Security	Check password field in the database.	All passwords should be hashed using the appropriate algorithm	Pass
2		Attempt to create a new user with an existing username.	System should decline users request to create new user with existing username.	Pass
3		Attempt to create a new user with non-registered username.	System should accept users request to create new user with non-registered username.	Pass
4		Access system using incorrect log in detail.	System should redirect user to the log in page displaying error message.	Pass
5		Access system using correct log in detail.	System should grant access, redirecting user to the correct homepage and setting up sessions.	Pass
6		User closes down browser whilst logged in.	System should log out user and redirect them back to the log in page.	Pass
7		Check that when creating a new user page is secure.	System should not display any details in the URL	Pass
8		Attempt access admins page with student log in.	System should redirect use to the log in page.	Pass
9		Attempt to access Lecturers page with student log in details	System should redirect use to the log in page.	Pass
10		Attempt to access Student page with Lecturer log in details	System should redirect use to the log in page.	Pass
11		Attempt to access admin page with Lecturer log in details.	System should redirect use to the log in page.	Pass
12		Attempt to access lecturer page with lecturer log in details.	System should grant access, with welcome message.	Pass
13		Attempt to access lecturer page with Student log in details.	System should grant access, with welcome message.	Pass
14		Attempt to access lecturer page with admin log in details.	System should grant access with welcome message.	Pass
15		Attempt to access Lecturers page with admin log in details	System should redirect use to the log in page.	Pass
16		Attempt to access Student page with admin log in details	System should redirect use to the log in page.	Pass

17	Performance	Check time taken to compete action.	Time taken must be under 3seconds.	Pass
18		Database is able to run several queries.	Each query should return results in under 2 seconds.	Pass
19		Interact with system with over 1000 records in the database.	System should respond to request within 3 seconds.	Pass
20	Compatibility	Login to SAM through PC running windows.	System should automatically adapt to screen size.	Pass
21		Login to SAMS through laptop windows.	System should automatically adapt to screen size.	Pass
22		Login to SAMS through MAC running macOS.	System should automatically adapt to screen size.	Pass
23		Login to SAMS through IPhone 6s.	User should be able to access the system	Fail
24		Login to SAMS through Samsung S7	User should be able to access the system	Fail
27		Login to SAMS from latest Firefox	User should be able to access the system	Pass
28		Login to SAMS from latest Google Chrome Browser	User should be able to access the system	Pass
29		Login to SAMS from latest Opera	User should be able to access the system	Pass
30		Login to SAMS from latest Safari	User should be able to access the system	Pass
31	Design	Navigation bar is the same on all pages for student.	All pages must show four buttons.	Pass
32		Navigation bar is the same on all pages admin	All pages must show one buttons.	Pass
33		Navigation bar is the same on all pages lecturer	All pages must show four buttons.	Pass
34		Navigation bar is the same on all pages course director.	All pages must show five buttons.	Pass
35		Check interface is readable in sunny weather.	System is still readable.	Pass.

Appendix X: Usability test tasks and post questionnaire results

Administrator

Task	Time Allocated
Login using the following credentials: U: admin P: admin	3mins
Create a new member of staff with your choice of details.	5mins
Edit the Department "Department of Social Arts" to be called "Department of Science".	5mins
Log out of the system and let the observer know when you are done.	2mins

Course director

Task	Time Allocated
Login using the following credentials: U: k7777777 P: coursedirector	3mins
Find the lowest attending module for the computer science course	4mins
Look into this module and fine the lowest attended session	2mins
Within this session find "John Alba" and note down the four module they take	4mins
Log out of the system and let the observer know when you are done.	2mins

Student

Task	Time Allocated
Login using the following credentials: U: k6352635 P: student	3mins
Find your lowest attending module	4mins
Within this module how many events were you absent for?	3mins
Log out of the system and let the observer know when you are done.	3mins

Lecturer

Task	Time Allocated
Login using the following credentials: U: k12345611 P: lecturer	3mins
Create a new session for the module "Digital Business" with your choice of details.	6mins
View your newly created session and mark the following as absent: John Alba, Sim Sahota, Becky Climb, Submit your results	4mins
Log out of the system and let the observer know when you are done.	2mins

After the test, each participant took a post-test questionnaire which assessed their experience, initially after an interview was conducted. The questions generated are shown below:

1. How satisfied were you with your overall experience?
2. Was the design consistent and easy to remember?
3. How difficult was it to find the information required to complete your task?
4. Do you think the system gave effective feedback?
5. How easy was it to navigate around the application?
6. Would you use this application again for its intended purpose?

Participant #	Post-Test Questionnaire Answers
Participant 1	1. I was extremely satisfied and found that the website was easy to learn to make my way around quickly 2. Yes, I was able to get around the website very quickly and easily for my task because of consistency. 3. At first it was a little hard, but after completing one task it was easy to know where everything was. 4. Yes I do, for each task completed I knew when I was done. 5. Very easy, I required little to no help. 6. Yes I think this is a great idea!
Participant 2	1. A fair amount, the website was simple and clear cut. 2. Yes, everything was clear and the buttons were very descriptive to assist with the task. 3. Not very difficult, it was easy to understand the website because the key information was clear. 4. Yes, for the session created there was effective feedback. 5. Very easy, the navigation bar and breadcrumbs helped. 6. Yes I would I think it's a very good idea to keep on top of attendance.
Participant 3	1. Great experience, easy to use. 2. Generally pretty easy, I like the pie chart on the dashboard they were very engaging. 3. Not difficult at all, everything was clearly labelled and it was easy to navigate. 4. I didn't get a chance to see much feedback but it was quick in response to my task. 5. Very easy, as I mentioned before. 6. If required to do so, yes.
Participant 4	1. Very Satisfied, I was able to complete my task very quickly 2. Yes the big icons was very engaging and I knew where to go straight away, the rest of the pages has a similar layout also so it was easy once I explored one route. 3. Not very, everything was very clear with worded buttons and labelled icons. 4. (User was unsure what this meant so observer explained the context). Yes when I completed a task I knew I was done because of the error messages. 5. Very easy, the navigation bar helped a lot. 6. N/A