BSc DEGREE

IN

Computer Science

PROJECT REPORT

Name: Dirosan Sivarajah
ID Number: K1914569
Project Title: Student Attendance Monitoring System
Project Supervisor: Graeme Jones

Please complete your chosen Mark Allocation Model below. The "fixed" sections may not be changed. In the other sections, you may choose only ONE of them to assign only one (1) single point. Half points may not be allocated. The total must add up to 10 and the two fixed sections may not be changed. If this is not completed or not completed correctly, the default model will be used. See Canvas and lecture notes for more information about Mark Allocation Models.

Section	**Points Allocated**
Introduction and Literature Review	2
Methodology and Analysis	2
Design	2
Prototype/Implementation	2
Validation and Testing	1 (fixed)
Critical Review and Conclusion	1 (fixed)
Total	10

Declaration

I have read and understood the University regulations on academic misconduct and I understand the meaning of the word *plagiarism* and the university's definition of the word. 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 individual's work and presented it as my own. No other individuals have contributed to this work in the form of written prose, code, drawings or other figures. I did not allow and will not allow anyone to copy my work with the intention of presenting part or all of it as their own.

Date: 08/04/2022

Signature: Dirosan Sivarajah

Table of Contents

1 Introduction

This project will be developed by conducting various research to demonstrate how the application will be implemented. Ensuring that it is implemented using the right tools and technologies will help build a functional application that users will use to achieve their end goals. This project monitors student attendance to support the course director and Lecturer in identifying engaged and non-engaged students in the workshops and lectures.

1.1 Aim

This project aims to build an application that will allow the lecturer and course director to monitor students with lower attendance. This will enable the course director and Lecturer to help the student, and this could help improve the student's education and allow them to engage in the lecture and workshop.

1.2 Objectives

- Identify stakeholders to gather requirements
- Functional and non-functional
- UML models to see the flow of the system
- Wireframes to design the application to see what the user can view
- Identify the tools and technologies that will be used to develop the application
- Perform multiple tests using dummy data when implementing

2 Literature Review

2.1 Introduction

This report segment will review the Data Protection Act 2018 rules and regulations concerning people's sensitive information and how companies and online are safeguarded and will research the eight core principles of the act and why the act needs to be followed by everyone (Castagna, 2021). Also, discuss individual rights in more depth and when the individual rights apply. In addition to this, it will explore the database security and how the data should be protected from threats such as SQL Injection, (DDoS - Distributed Denial of Service), malware, Man-in-the-middle attack etc. Lastly, what are the penalties for companies or people for breaching the act and the impact and consequences of a data breach. The findings will be used to ensure that the user's data is stored securely by following the rules and regulations. Furthermore, what security measures could be taken to protect user data on this project.

2.2 Data Protection Act 2018

The Data Protection Act 2018 came into effect in May 2018. The act's purpose was to comply with the right to privacy of personal information from being threatened and maintain confidentiality between the user and the company that acquired the information. The Data Protection Act 2018 has eight core principles that everyone must follow when processing or handling personal data. There are consequences for breaching data and not following the Data Protection Act to protect data from threats. For instance, Sony company was fined for breaching the data protection act "Sony fined £250,00 after breaching Data Protection Act" (Smith, 2013). The data breach allowed the hacker to compromise the whole Sony PlayStation platform and give the hacker the personal information of millions of customers, such as names, addresses, account passwords, payment card details, etc. The data breach incident was because the Sony PlayStation platform was out of date, which put the Sony PlayStation platform vulnerable to threats such as SQL Injection, (DDoS - Distributed Denial of Service), malware, Man-in-the-middle attack etc. It could have been avoided if Sony had updated its software, which would have protected against data breaches, and customer data would have been secure.

2.2.1 Data Protection Principles

The data protection principles were enforced to ensure that the processing of induvial data is handled safely and protected from threats. This section will discuss in more depth each principle and when it comes into effect when processing data.

The first principle is that the data should be processed "lawfully, fairly, and transparent" (legislation.gov.uk). This principle ensures that the data is used according to the law and not anything against the law. For instance, when a company is collecting or processing customer information, the company must provide a reason for collecting the data. Before they collect, they must get consent from the customer before handling the data.

The second principle is that the data should be processed for "specified, explicit purposes" (legislation.gov.uk, no date b). This principle ensures that the data must be only used for legitimate reasons. For instance, data that the company collected should be only used for the purpose they collected and not use the data for any other purposes.

The third principle is that the data should be processed "adequate, relevant and limited to only what is necessary" (legislation.gov.uk, no date b). This principle ensures that only the company obtains and stores the necessary data. This prevents a company or online to hold unnecessary data about the users.

The fourth principle is that the data should be processed "accurate and kept up to date where necessary" (legislation.gov.uk). This principle ensures that the data the company collects must serve a purpose, and if the data doesn't serve a purpose, it should not be kept at all.

The fifth principle is that the data should be processed "in accordance with the rights of the data subjects under this act" (legislation.gov.uk). This principle ensures that the data the user has the right to how their data is processed. For instance, the user can deny the company from processing the user data for market and other unnecessary reasons.

The sixth principle is that the data should be processed "kept for no longer than is necessary" (legislation.gov.uk). This principle ensures that the company should not hold data about the user that becomes redundant. The company should securely delete the user data. For instance, when the user data serves its purpose, the company should automatically delete it from the system about the user. This reduces the risk of sensitive data leaking or being misused.

The seventh principle is that the data should be processed "handled to ensure appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction or damage" (legislation.gov.uk). This principle ensures that the company is responsible for the security of the user data. For instance, if the company is hacked and puts the system vulnerable, the company should take the necessary measures to respond to threats quickly and efficiently.

The eighth principle is that the data should be processed "shall not be transferred to a country or territory outside the European Economic Area unless the country has a high level of protection for the right and freedoms of user data" (legislation.gov.uk). This principle ensures that the company that holds user data should not transfer data without ensuring that the data will be kept secure from threats.

2.2.2 The Individual rights

The individual right was enforced for users to have ownership over their data and how it is processed. This was to ensure that the customer has control over the personal information that the company holds.

- The individual must be informed about what data the company will be collecting and how it will be processed.
- The individual has the right to access their data, and the company must be available to provide all the information they have collected at the individual request.
- The individual has the right to modify their personal information to ensure that the company holds updated information.
- The individual has the right to delete data that the company doesn't need and has the right to withdraw any consent they might have given when the company collected their data.
- The individual has the right to stop or restrict their data from processing.
- The individual has the right to data portability when the user requests the data the company holds to give the same data to another company.
- The individual has the right to object to how their data is processed for certain cases.

This individual right will allow the user to control how their data can be processed and used by a company.

2.2.3 Penalties

There are penalties when the Data protection act is a breach by a person or a company. The penalty depends on the severity of the breach, and there are various punishments, including fines, prosecution etc. However, companies that breach the rules will lead to a maximum penalty of £17.5 million or 4% of the company's global turnover (Information Commissioner's Office, 2021). The law states that when a company experiences a data breach, the company needs to information the (ico - Information Commissioner's Office) within 72 hours of knowing the breach. Furthermore, the company must notify all the individuals if their data was compromised in the data breach attack (Information Commissioner's Office, 2022).

2.3 Computer Misuse Act 2018

The Computer Misuse Act 2018 was introduced in 1990. The purpose of this act is to protect users from threats. It has four officed and penalties for breaking those offences. The Computer Misuse Act 2018 was introduced when two hackers hacked into BT and stole the login credentials of BT engineers. The two hacker names are "Regina V Gold and Schifreen" and they tried to access Duke Edinburgh's email information (McCallion, 2022). For instance, malicious hackers exploit or sell data to cause problems or for personal purposes. The Computer Misuse Act was enforced to protect those vulnerable people from threats, and some penalties can put people in jail for breaching Computer Misuse Act.

2.3.1 Computer Act Offences and Penalties

The Computer Act Officine and Penalties were enforced to keep users' information from being misused.

When "Unauthorised access to computer material. Commonly referred to as hacking" (legislation.gov.uk). This offence purpose is that it is a crime to access other user information without their permission or misuse any individual data for any purposes. The penalty for breaking the offence can put the individual face up to six months in prison or a fine.

When "Unauthorised access to computer materials with intent to commit a further crime etc. (legislation.gov.uk)". This offence purpose is that it is a crime to misuse an individual data to commit a crime, for example, the hacker could use the individual information to buy a gun. This could get the individual into a crime that they might not have committed. The penalty for breaking the offence can put the person face up to five years in prison or a fine.

When "Unauthorised modification of data" (legislation.gov.uk). This offence purpose is that it is a crime to modify individual information without their permission, and this could put the individual in a problem as they might lose some of their personal information, for instance, it can be their medical record etc. The penalty for breaking the offence can put the individual face up to ten years in prison or an unlimited fine to reset their life's

When "Making, supplying or obtaining anything which can be used in Computer Misuse offences" (legislation.gov.uk). This offence purpose is that it is a crime to steal or sell any individual information by creating, supplying, or obtaining computer viruses to cause damage to systems that holds user personal information. The penalty for breaking the offence can put the individual face up to ten years or in prison unlimited fine for reset of their life's.

2.4 Database Security

The database is essential as it stores valuable information, so the database must be well protected from any malicious threats. There are various threats such as SQL/NoSQL injection attackers, DoS/DDoS attacks, malware etc. (IBM, 2021). The hackers can use any threat methods to gain unauthorised access to the database. If the hacker compromises the database, that could cause a high impact on the company as valuable data will be exposed. For the company to run, they will have to solve the problem and what data has been lost etc. Furthermore, this could make the company lose moany and company partners as they will lose trust, the customer will leave, etc. To prevent a data breach, the company must have measures to protect valuable information before they lose all the data.

2.5 Conclusion

One of the key reasons companies take data protection seriously is that they can lose data, putting the company in a big problem as they will have to face the law. Therefore, this project will use dummy data rather than actual individual data. When this project is in the implementation phase, the security of the project will be taken into consideration. For instance, implementing a login system will only allow the user to access the application and ensure that the password is hidden. This research has given insight into how data is crucial and the consequences of breaching those acts.

3 Requirements Analysis

3.1 Introduction

The Requirement analysis segment will use various analysis methods to determine the requirements. The techniques that will be used include Stakeholders, SWOT Analysis, use case etc. the user of the techniques will help to prioritise and understand the requirements. For instance, User stories will be used to identify each user's requirements for this project. This will allow identifying the internal and external stakeholders that will have an effect on the project and ensure that all the discussion is made with the client every step of the way to ensure that the requirements and foundation of this project are strongly composed.

3.2 Stakeholders

The stakeholders are a group of people involved with the project, and there are two types of stakeholders which are internal and external. The internal stakeholders are involved with the organisation and have direct relationships such as ownership, investment, etc. The external stakeholders are not directly involved with the organisation, such as suppliers, public groups, etc. For this project, different types of stakeholders are involved, such as (Admin, Course Director, Lecturer and Student). This will help the developer ensure that all the stakeholders' requirements are met and necessary changes depending on the implementation of each requirement.

3.2.1 Key Stakeholders

To identify potential stakeholders that will benefit from the project, and they play an important role in this project. For this project, it is vital to evaluate each stakeholder to identify their strengths and what they can do in this project. This will allow the developer to design the system to the stakeholder's requirements. Furthermore, meetings were set up with the client to discuss the project's requirements further and allow the client to understand the internal and external stakeholders.

3.2.1.1 Internal

- **Administrator**
  The admin will only use the system to maintain the system for the users, and the admin has the permission to add, edit, update, delete, etc., and full control over the system.
- **Course Director**
  The course director can monitor all the student attendance in the modules by workshop and lecture. They have the ability to filter the student with low/high attendance.
- **Lecturer**
  The lecture can monitor all the student attendance in the modules by workshop and lecture. They have the ability to filter the student with low/high attendance
- **Student**
  The student can view the modules they are studying.

3.2.1.2 External

- **Government:**
  The government provides the finances for the students and institutions to run successfully
- **Parents/guardians:**
  The parents and guardians will care about their children getting a degree or being successful in for future career

3.2.2 Stakeholder Power/Interest grid

The Power/Interest grid is beneficial as it helps position current and future stakeholders based on their power and interest in the system. Figure 1 illustrates each stakeholder's power and interest in the system, which will help manage and communicate with the stakeholders during the project's development.

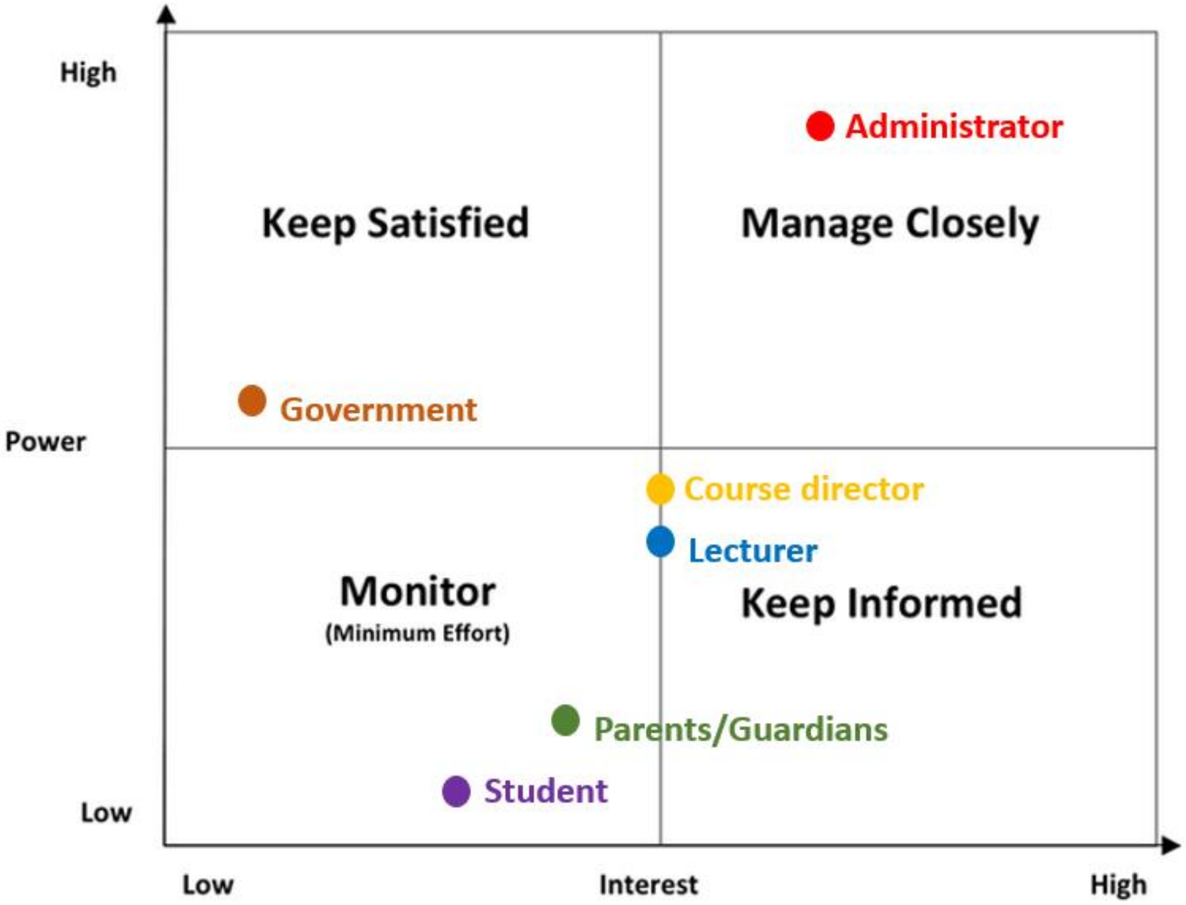


Figure 1 Stakeholder Power/Interest grid

3.3 SWOT Analysis

SWOT analysis is used to analyse the Strengths, Weaknesses, Opportunities and Threats of the project. It helps to determine the strengths and discover the project's opportunities and what can be improved on the project. Furthermore, it helps to identify the weaknesses and threats of the project beforehand, which will allow taking the necessary measure to solve the problem.

Strengths	**Weaknesses**
The Lecturer doesn't need to take attendance on paper and could help eliminate problems such as human error, misdirecting, etc. It helps to save time. This will help the Lecturer as they can monitor each module session and see the student's attendance. The Lecturer can find students with lower attendance percentages and try to help them. The students can monitor their attendance percentage on the system, encouraging them to attend each module session as they will miss out on all the teaching materials.	The data that might be stored can be corrupted, which stops the Lecturer from seeing student attendance or the data might not be accurate. The student might not be able to view their attendance as the data might be incorrect or misleading. Other academic associations might not use the system as it does not hold many functions that they might be interested in.

Opportunities	**Treats**
The students will be able to monitor their module session attendance. The system will help other academic associations that want to monitor their student attendance and help them to identify students that do not attend some module sessions or do not even attend any sessions. The Lecturer will be able to see each student's attendance, and they will be able to keep track of any students with lower attendance.	The threat can be when an unauthorised user tries to gain access to the system to steal all the sensitive data or cause system failure by inserting a malicious virus that could corrupt the data. Other systems function and have better requirements than this system. Many users might like the ease of use of the system etc.

3.4 Use Case

The user case outlines the user's goals, and it will illustrate how the system will respond to the various tasks that the user will be doing on the system. Furthermore, each user case consists of a narrative text that allows the stakeholders to understand how the system works to achieve each task. Use case diagrams for the remaining users can be found in the APPENDIX Use case.

Figure 2 illustrates the admin actions with the system, and they have full access to the system as they can add, edit, and delete data on the system. The Lecturer, Course director, and the student will directly associate with the admin as shown below, where Lecturer, Course director and student come under the admin action user.

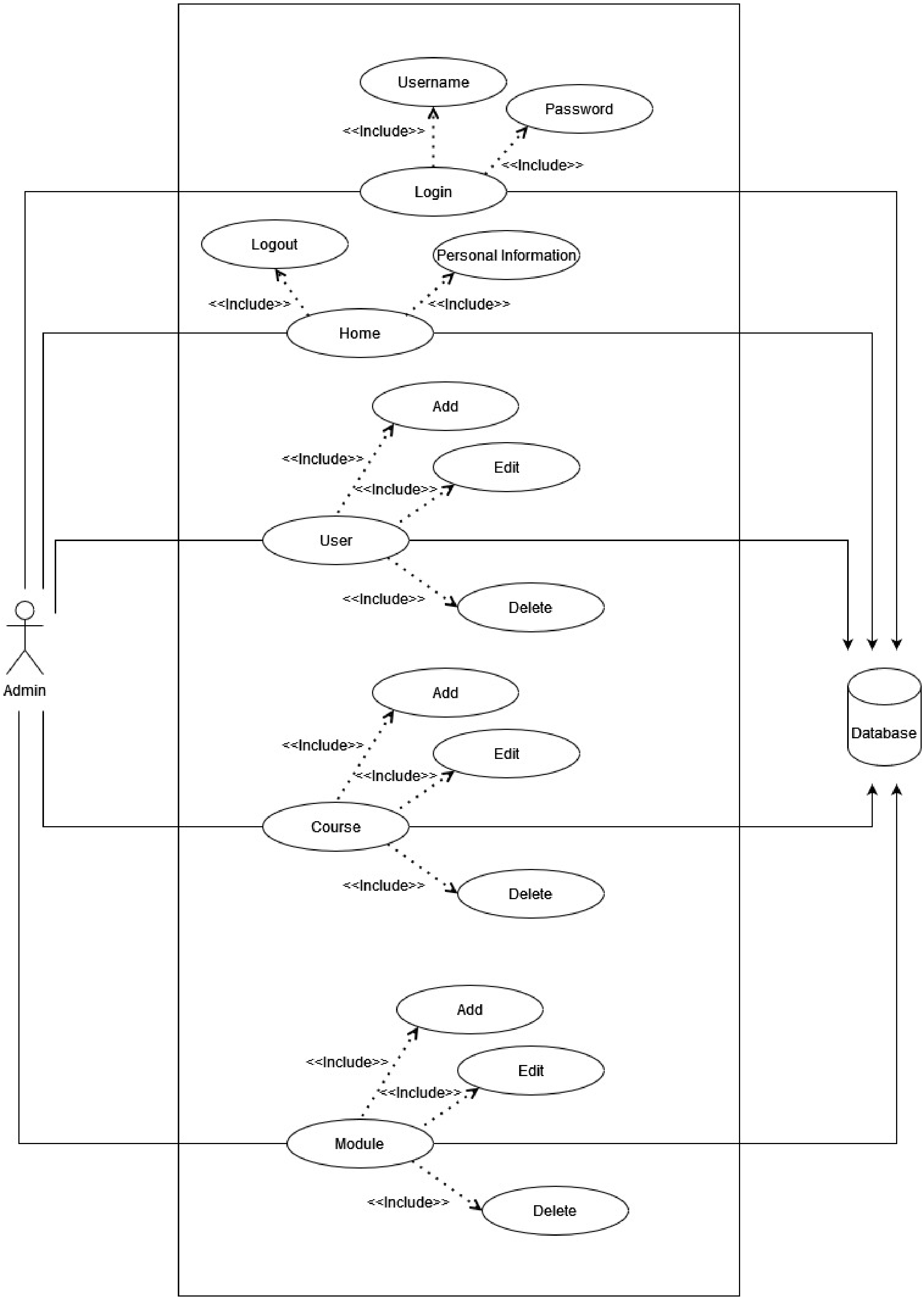


Figure 2 Admin Use Case diagram

Figure 3 is for the Lecturer as they are key users of the system, and the course director will directly associate with the Lecturer as both have the same access right permission on the system. Furthermore, the student will also directly associate with both the lecturer and course director as they are both available to view student attendances in the modules they study.

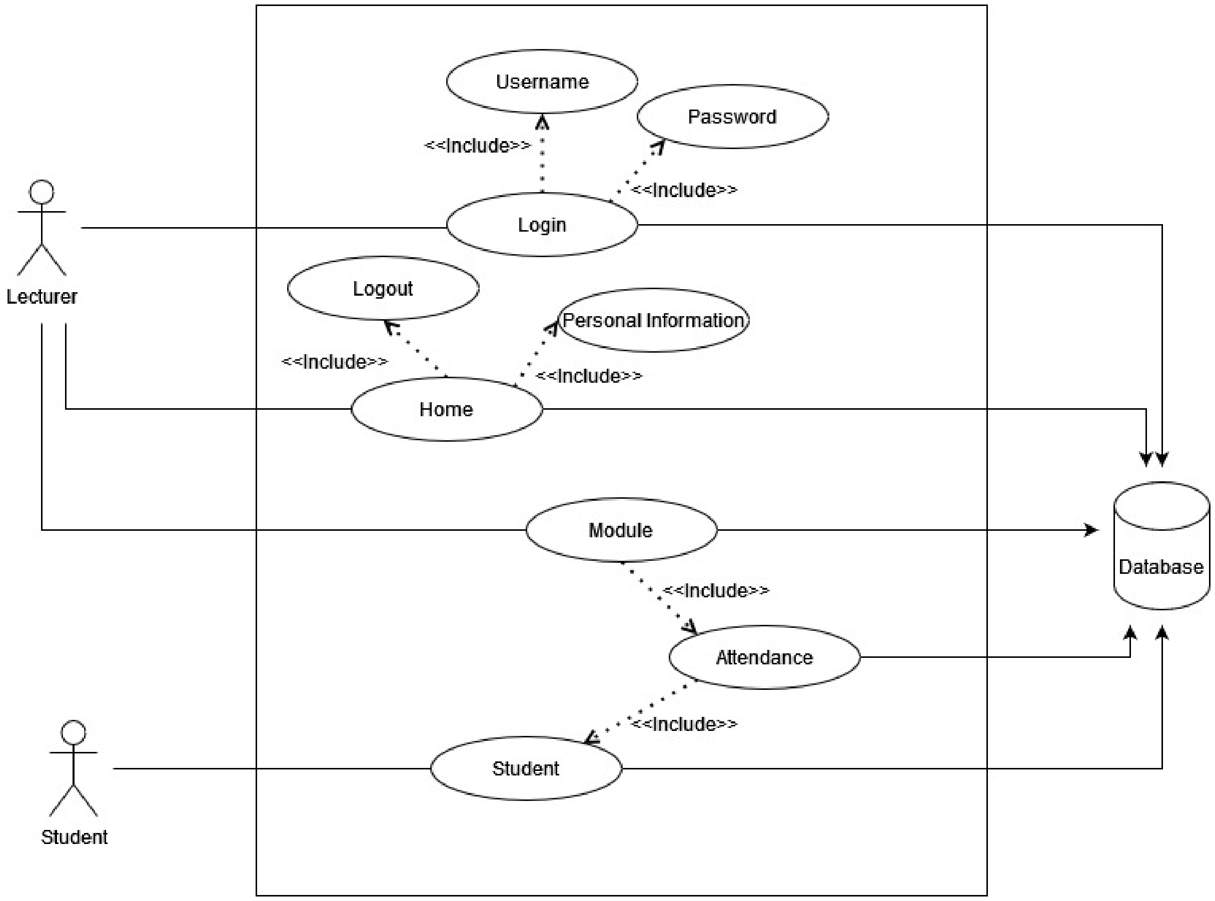


Figure 3 Lecturer Use Case diagram

3.5 Use case text

The user text directly correlates with the Use case diagram, which explores each user's actions to achieve their end goal. It shows the successful outcomes and the problems that might occur, which helps to identify solutions to the issues. This will benefit the developer more as it shows where problems could occur, and each table has been prioritised using MoSCoW to show which is essential for the system and outlines which requirements are crucial to this project. Table 4 is the use case description for reequipment to "Login to the system".

3.5.1 Admin

The user case text was created using the use case diagram. The purpose of creating the use case text was to demonstrate what the user can do on the application in more steps.

Table 1 is for the admin and shows the action the admin will do to log in to the application. The use case text shows the actor who will be doing the steps and the goal they are trying to achieve. The condition of the goal and the main flow will show how the user will achieve their goal by following the steps. All the tables have been prioritised using MoSCoW.

Use case 1	Login	
Actor	Admin	
Goal	Able to login to the system	
Condition	Login	
Main Flow	**Step**	**Action**
	1	Admin must be able to Login
	2	Admin must enter the username and password
	3	Admin press login button
	4	Admin must be able to view the content once logged in
Alternative Flow	**Action**	
MoSCoW	Must	

Table 1 Admin use case text Login

Table 9 is for the student showing how the student is available to view the home page with their personal information.

Use case 2	Personal Information	
Actor	Student	
Goal	Able to view Personal information	
Condition	View Personal information	
Main Flow	**Step**	**Action**
	1	Once logged in
	2	Admin must be able to view the content
	3	Personal information displayed
	4	Student must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 9 Student use case text personal information

Table 15 is for the Lecturer where it is showing how the Lecturer is available to view the student attendance.

Use case 5	Attendance	
Actor	Lecturer	
Goal	Able to view Attendance	
Condition	View Attendance	
Main Flow	**Step**	**Action**
	1	The Lecturer must be able to view Attendance detail
	2	The Lecturer must be able to view the content
	3	The Lecturer should be able to see Attendance code
	4	The Lecturer must be able to see induvial module attendance
	5	The Lecturer must be able to view induvial student attendance
	6	The Lecturer should be able to see the overall Attendance
	7	The Lecturer must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 15 Lecturer use case text attendance

3.6 User Type and Stories

The user type and stories determine what the user is available to do and the end goals they want to achieve using the system.

3.6.1 User Type

Administrator
The admin is available to (add, edit, assign, delete) users, courses, and Modules.

Course Director
The course director is available to view personal information and module. Furthermore, available to view attendance.

Lecturer
The Lecturer is available to view personal information and module. Furthermore, available to view attendance.

Student
The student is available to view personal information and module.

3.6.2 User Stories

As a/an	**I want to…**	**so that…**
Admin	log in to the system	I can manage the system
Student	view the course	I can view the modules that I am studying
Student	log in to the system	I can view my personal information
Course director	See all the Module	I can view the module attendance
Lecturer	want to see student attendance	I can identify which student has lower attendance
Admin	add users	they are available to access the system
Lecturer	log in to the system	I can view my personal information
Admin	delete users	I can delete users that don't need access to the system
Admin	assign module to student	the student is available to view the modules that they are studying
Course director	log in to the system	I can view my personal information
Admin	add course	I can add a new course to the system
Student	see individual module attendance	I can see which module my attendance is low
Admin	add Module	I can add new modules related to the new course

Table 21 User Stories

3.7 Requirements

The requirements section will determine what the user requires when they use the system to achieve their end goal. The requirement is separated into functional and non-functional and the user stories from the previous chapter to determine the user requirements.

3.7.1 Functional Requirements

The functional requirements are a verb-noun format that defines the system and shows how the system will function.

ID	**Functional requirements**
1	The users must be able to view their personal information
2	The users must be able to log in to the system using login credentials
3	The admin must be able to add users to the system
4	The admin must be able to delete users from the system

5	The admin should be able to edit users on the system
6	The admin must be able to assign courses and related modules by level
7	The student must be able to view the course and related modules
8	The course director must be able to view the modules attendance
9	The Lecturer must be able to view the students on the Module
10	The course director and Lecturer must be able to filter student attendance by low/high
11	The admin must be able to add a new course to the system
12	The admin must be able to add modules related to the new course
13	The admin must be able to delete course on the system
14	The admin should be able to edit module on the system
15	The admin must be able to delete module on the system
16	The admin should be able to edit course on the system
17	The student should be able to view overall attendance in a graph showing different module

Table 21 Functional requirement

3.7.2 Non-Function Requirements

ID	Non-Functional requirements	Description
1	Platform	The system must be able to run on devices such as Windows and iOS that has browser such as (Google chrome, Firefox etc.)
2	Maintainability	Anyone can easily maintain the system as it doesn't require any training
3	Design	The GUI should be straightforward for all the pages
4	Performance	The database should be able to retrieve data from the database and display it on the front end. The system should run smoothly with all the data coming from the database.
5	Security	Each user has different login privileges to restrict unauthorised users. The system should be able to accept any users with the login.

Table 22 Non-Functional requirement

3.8 MoSCoW

The MoSCoW is used to determine the importance of the system requirements, and it shows what functions the system hold and what the functions will do. As the developer, I will implement the more essential requirements needed on the system for the users. Table 1 shows the essential functions required on the system for the admin to manage all the users and provide the necessary data for them to view.

ID	**Functional requirements**	**MoSCoW**
1	The users must be able to view their personal information	MUST HAVE
2	The users must be able to log in to the system using login credentials	MUST HAVE
3	The admin must be able to add users to the system	MUST HAVE
4	The admin must be able to delete users from the system	MUST HAVE
5	The admin should be able to edit users on the system	MUST HAVE
6	The admin must be able to assign course and related modules by level to student	MUST HAVE
7	The student must be able to view the course and related modules	MUST HAVE
8	The course director must be able to view the modules attendance	MUST HAVE

9	The Lecturer must be able to view the students on the module	MUST HAVE
10	The course director and Lecturer must be able to filter student attendance by low/high	MUST HAVE
11	The admin must be able to add new course to the system	MUST HAVE
12	The admin must be able to add modules related to the new course	MUST HAVE
13	The admin must be able to delete course on the system	MUST HAVE
14	The admin should be able to edit module on the system	SHOULD HAVE
15	The admin must be able to delete module on the system	MUST HAVE
16	The admin should be able to edit course on the system	SHOULD HAVE
17	The student should be able to view overall attendance in graph showing for different module	COULD HAVE

Table 23 MoSCoW

3.9 Conclusion

Overall, the requirement analysis is the most crucial part of the project. It determines the project's lifecycle and clearly defines the user requirements that will be used in the project's development phase. The core requirements have been discussed and agreed upon with clients, and once the project is in the development phase, the requirements will be met. Furthermore, if any changes occur in the development process, I need to ensure that the end goal is the same for the system and further research to ensure the end goal is achievable.

4 Design

4.1 Introduction

This report segment will determine the design of the system and the aspects that will be part of the system. The research conducted in the previous chapters will be used to create the design components that will be used to develop the system. This report will cover different phases required for the system's design part, such as (ERD) Entity Relationship, Navigation, Activity, and other required diagrams.

Many aspects will be used to create a functional system that the users will use. Entity-Relationship (ERD) will help to solve the association between the tables. A data dictionary will show the in-depth content of individual tables, which will help create the database. A sequence diagram will be used to identify how the data will be processed and its behaviour. The segment will help explore the required components to understand the user tasks, and the wireframes will be used to understand the workflow of each user type. The navigation diagram will help identify the page layout on the system for each user type. The use of all the aspects will help evaluate the system design efficiently to implement the system and give the user the ease of use when they are using the system to do their tasks.

4.2 Data Modelling

The data modelling will capture the system database structure and explore various UML models made for the system design. The functional requirements will be considered to create the database. The information that is gathered will enable the developer to implement the system.

4.2.1 ERD Diagram

The Entity-Relationship diagram is important as it captures the relationships between the entities and the attribute required by the system. Each table holds a primary key (PK), which uniquely identifies the table entries and potentially, a foreign key (FK) that shows the association with other tables. In creating the ERD diagram, many changes, such as the tables' relation, have to be refined to optimise the system's functionality, processing, and performance.

Furthermore, it will help to create the database and put dummy data under those tables. The use of the methodology in this project will affect the database, where minor changes will be made to the initial design. Figure 6 shows the database structure, and the line shows the association between the table with (One-to-Many, One-to-One etc.)

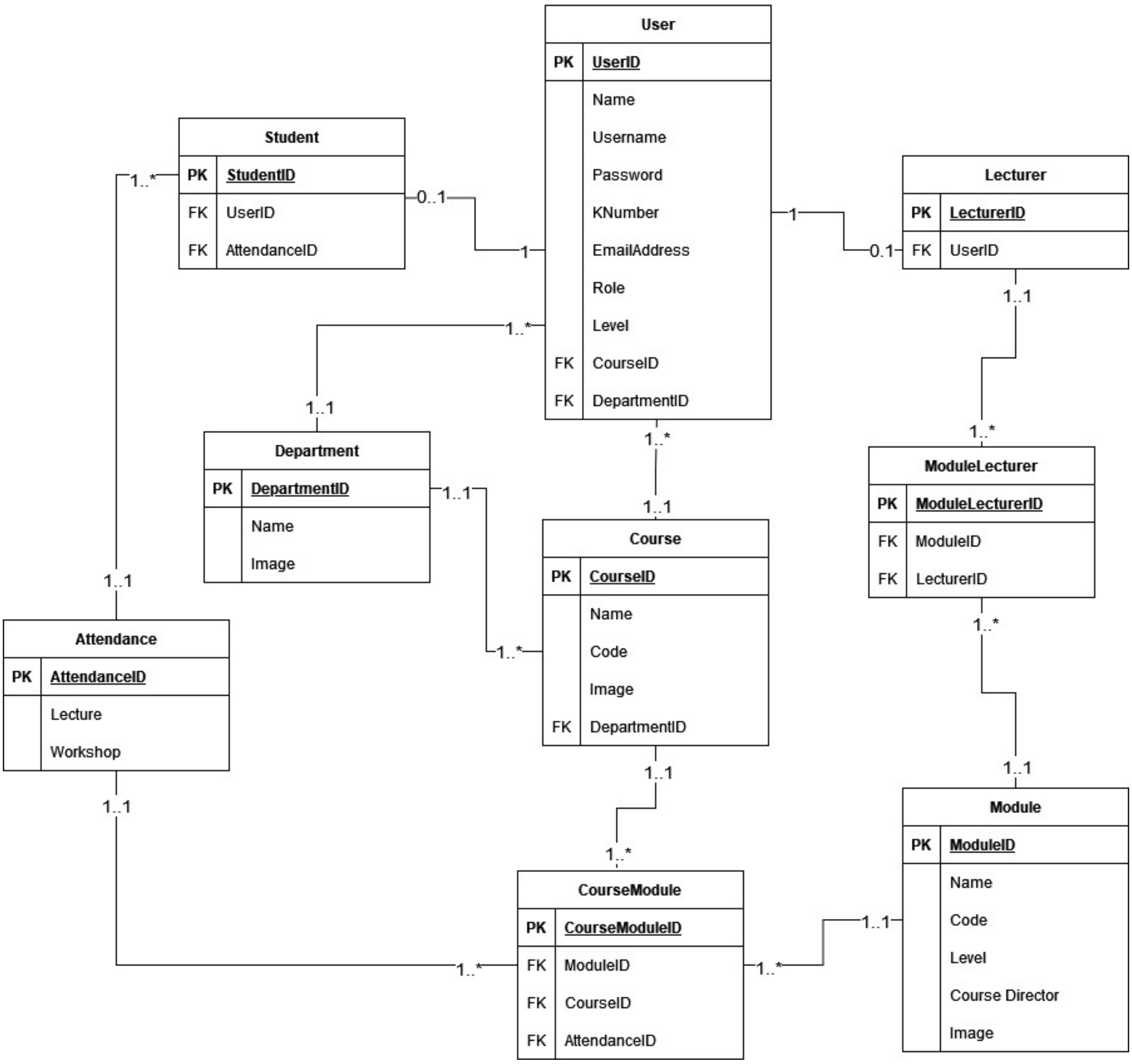


Figure 6 ERD (Entity-Relationship diagram)

4.2.1.1 Assumptions

The assumptions were identified during the creation of the Entity-Relationship diagram.

- A lecturer may teach one to many modules, and each module taught by a one-to-many lecturer
- Course may have one too many modules, and modules may have one too many courses

4.2.2 Data Dictionary

The data dictionary shows the format and structure of each table in the database. It ensures that the attributes have the right data type. Each table has a primary key, and if the table has a relationship, a foreign key is defined appropriately in the suitable tables.

- Relation: Specifies the name of the table
- Attribute: Specifies the attribute within the table
- Data type: Specifies the data such as INT - numerical value, VARCHAR - characters
- Key: Indicated whether the attribute is a primary key or a foreign key
- Not Null: Indicated whether the attribute value should be empty or not
- Constraints: It restricts the data type in the table
- Referential relation: Specifies the table with which it has an association

Table 24 was created for Users. It will contain all the user's details such as (name, username, password, etc.). It will have a foreign key of CourseID and DepartmentID as the user table is associated with the course and department table.

Relation	Attribute	Data Type	Size	Key PK/FK	Not Null	Constraints	Referential Relation
User	ID	INT		PK		AUTO-INCREMENT	
	Name	VARCHAR	50				
	Username	VARCHAR	50				
	Password	VARCHAR	50				
	Role	VARCHAR	50				
	KNumber	VARCHAR	50				
	Emailaddress	VARCHAR	50				
	Level	INT					
	CourseID	INT		FK			Course
	DepartmentID	INT		FK			Department

Table 24 User Data Dictionary

Table 30 shows the attendance table and the attributes that attendance table will have in the database.

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Attendance	ID	INT		PK		AUTO-INCREMENT	
	Lecture	INT	10				
	Workshop	INT	10				

Table 30 Attendance Data Dictionary

Table 31 is a joined table to solve the many relations between the course and module table. This table only holds ID from the tables to ensure that the tables can be communicated.

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
CouseModule	ID	INT		PK		AUTO-INCREMENT	

	ModuleID	INT		FK			Module
	CourseID	INT		FK			Course
	AttendanceID	INT		FK			Attendance

Table 31 CourseModule Data Dictionary

Table 32 is a joined table to solve the many relations between the module and lecturer table. This table only holds ID from the tables to ensure that the tables can be communicated.

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
ModuleLecturer	ID	INT		PK		AUTO-INCREMENT	
	ModuleID	INT		FK			Module
	LecturerID	INT		FK			Lecturer

Table 32 ModuleLeader Data Dictionary

4.2.3 Sequence diagram

The sequence diagram shows the system's behaviour and demonstrates how the objects interact with each other to achieve a task. This allows the developer to explore the complexity of the user requirements, and it will help to understand the functionality of each action step at a time.

The rectangle on top of the diagram represents the object of the system process, dash lines define the object lifeline, and the thin rectangle in red represents the operating elements. The two arrows are calling and returning messages to operate the system. For instance, the calling message sends a request to indicate that it wishes to perform a task and the return message shows the communication between the lifeline of an interaction. However, the caller message must wait for the return message to reclaim control over the system.

Figure 7 user login sequence diagram shows the process of login into the system for the user to log in, the login form will be displayed, and then there will be two input fields asking the user to enter the username and password. For instance, if the user enters an invalid username or password, the system will deny access to the system and display an error message to notify the user to recheck the username and password to proceed. If the username and password are correct, the system will fetch all the user information using the username and password from the database and display the information on the home page for users to view.

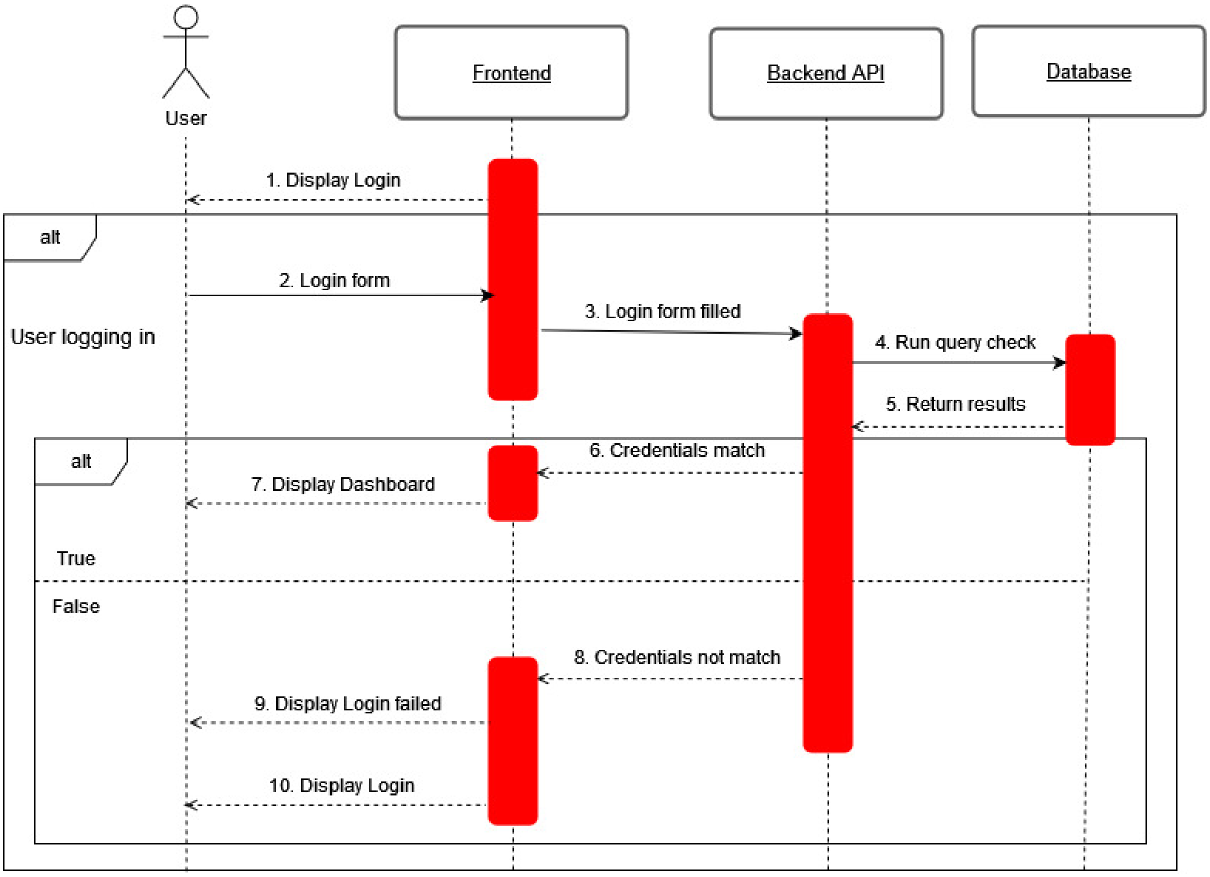


Figure 7 User sequence diagram

In figure 8, the form will be displayed when the admin registers a new user from the user page. Then fill all required input fields with the user information, and once the add button is clicked, it will validate the form. POST request runs a SQL query to store the data into the correct table by filling the attributes with the right values. However, if the user data entered is invalid, it will display an error message to the admin to double-check that the input fields are entered with the correct values.

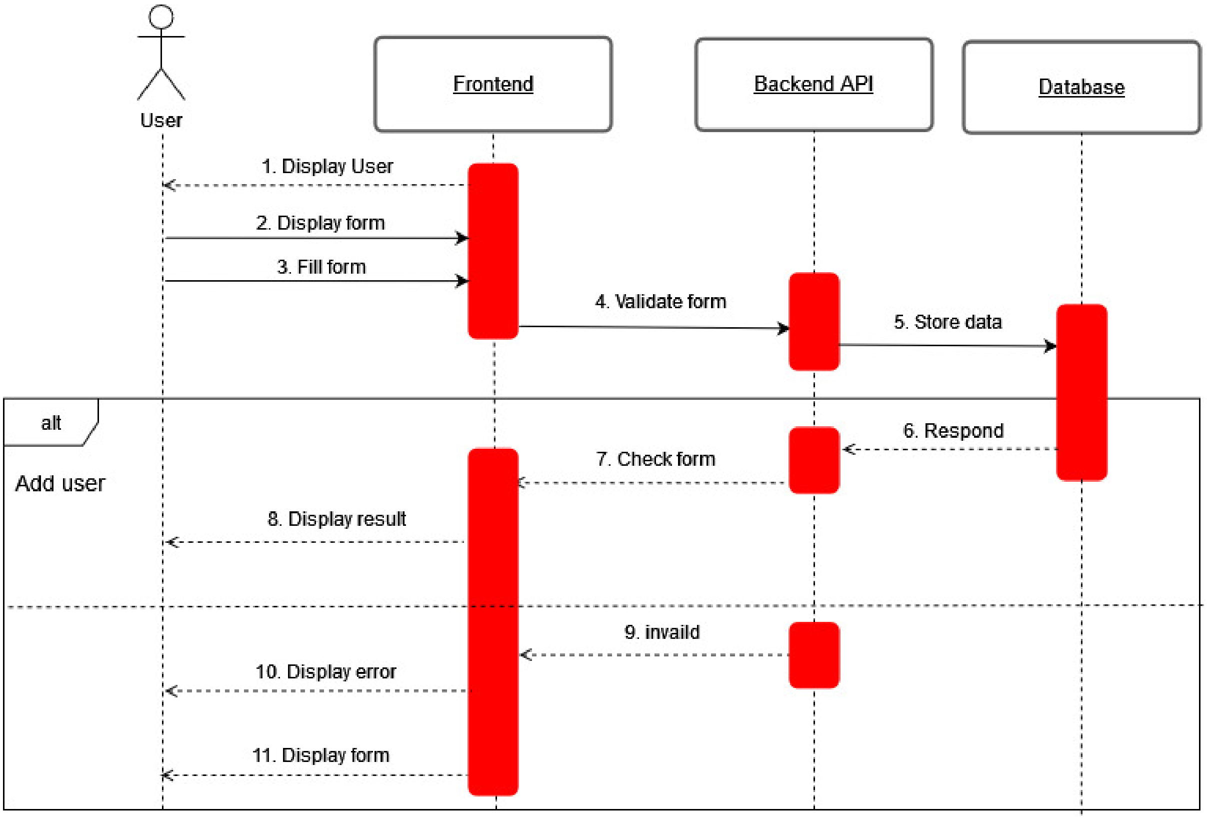


Figure 8 User sequence diagram

4.2.4 Activity diagram

The activity diagram demonstrates the dynamic aspects of the system, which helps the developer as it shows them how each function on the system will interact with each other to achieve the user tasks. The circle at the top is where it begins, and it ends a series of activities on the circle bottom. The rounded rectangle shows the activity taking place and the arrows connecting the activities. Finally, the diamond shape is where all the decisions are being made.

The Figure 9 demonstrates what the lecturer and course director is available to do on the system. The diagram was created to assume that the course director and Lecturer are available to view student attendance to identify the student with lower attendance in the module. The course director or Lecturer can click on the module page, then a list of modules will be displayed with a view button, which will take the course director and Lecturer to the attendance page of that module with the list of students fetched from the database with the individual attendance of the student in the modules. With the filter function, the course director and Lecturer are available to filter the student list by lower or higher attendance.

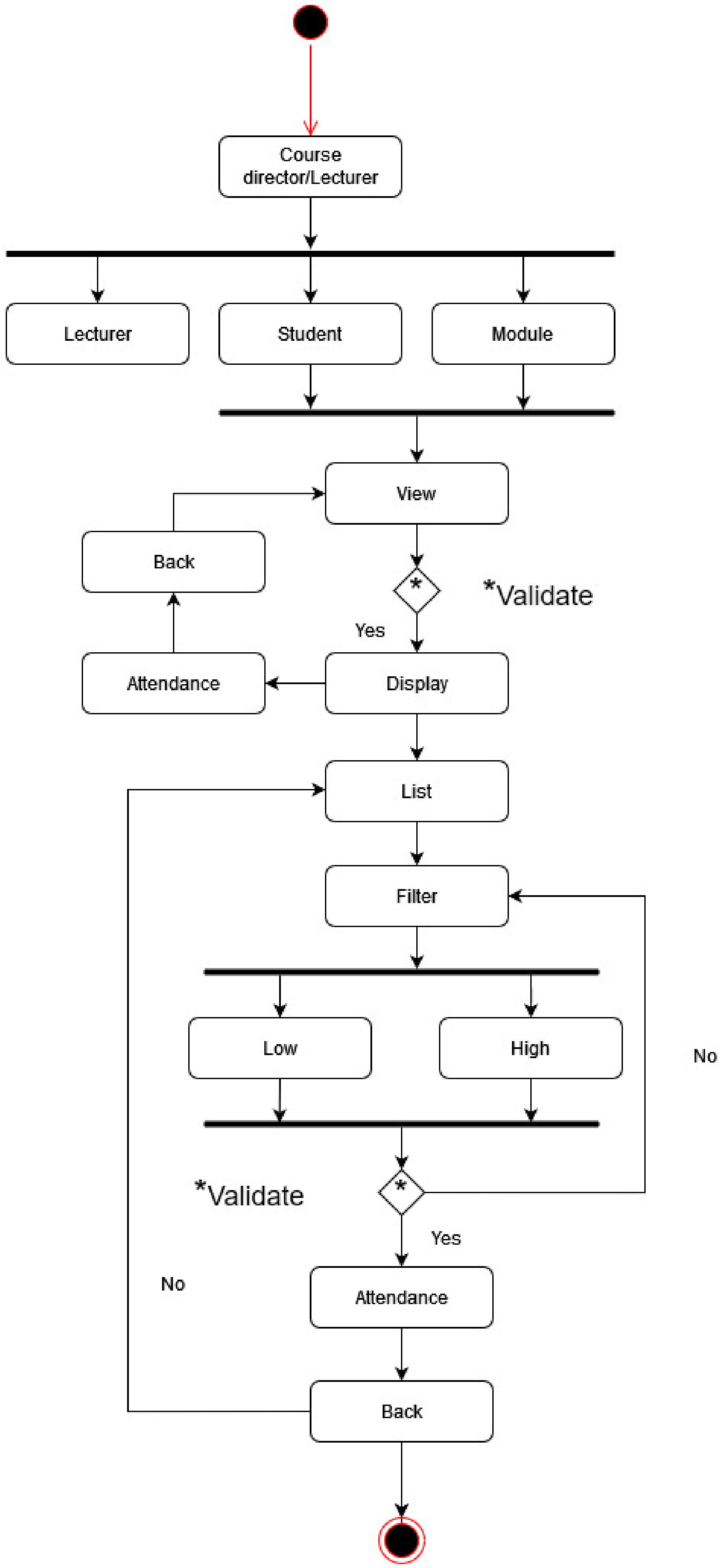


Figure 9 activity diagram for Lecturer and course director

Figure 10 will show the flow of how the admin is available to add, edit and delete a course, user and module. It will show how each step would take to achieve those goals.

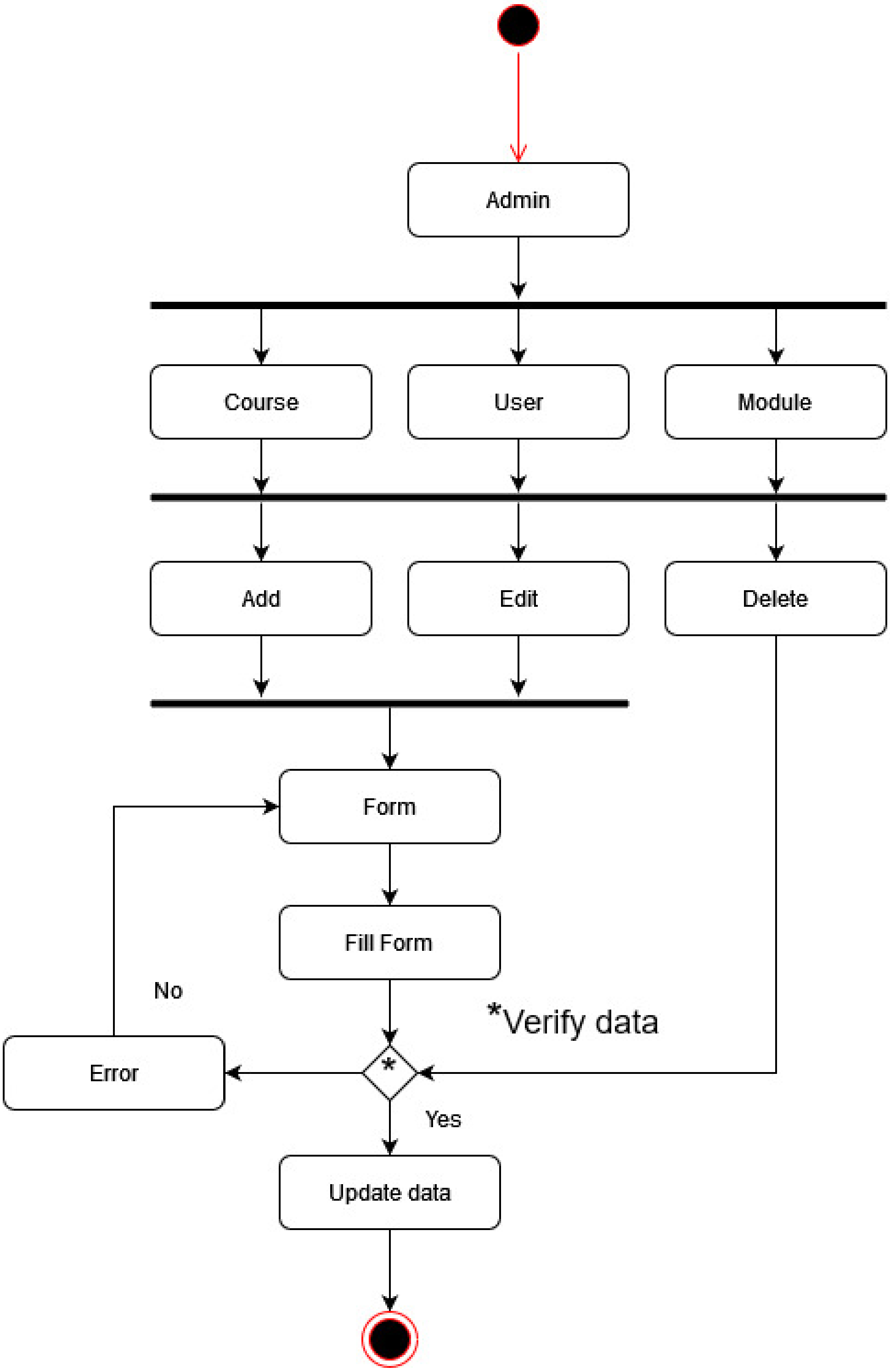


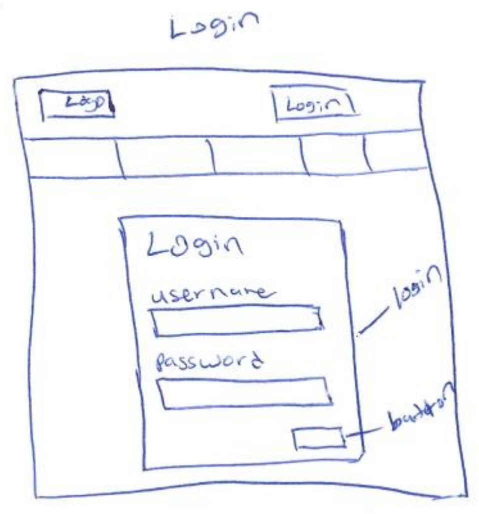
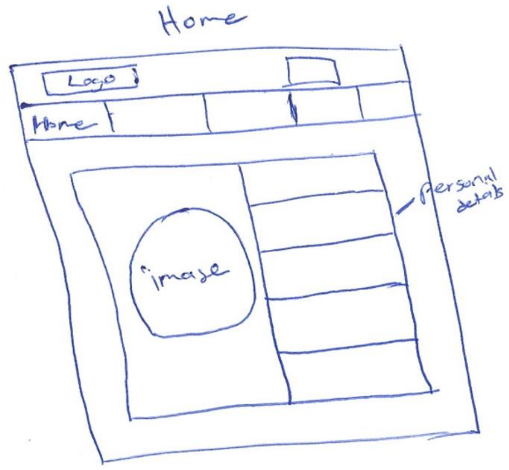
Figure 10 activity diagram for admin to add, edit and delete

4.3 Page design

The page design will demonstrate the application's low to high fidelity design and how they will be used to implement the application.

4.3.1 Initial sketches

The initial sketches are low fidelity that was created while discussing the project, which gives an overview of the system's layout and how each page will look. This will help the client also see what the system could be and allow them to make suggestions to the sketches so that they can be amended before developing the system. The requirements will be used to determine what the user wants to do on the system, which will help in sketching more of the pages.

Login page	**Home page**
 Login Logo Login Login username password login button	
The sketch of the Login form for all the users using the system. The login form is straightforward to ensure that it gives ease of use. Which is only will ask the user to enter their username and password.	The Home page is for all the users, and the users will land on the Home page once they log in successfully into the system. The user type determines the displayed information on the home page. For instance, for the student, it will only display data related to the students, and for another user, the type will have data related to them.

4.3.2 Wireframes

The wireframe is a high-fidelity frame that was designed using the initial sketches, and the client feedback is used to create the wireframes that can be seen below. The frames show what content will be seen on each page and the page's style, font, text size, etc. This demonstrates the system outcome and how each user type will use the system. The colour choice was kept simple to give all the users the ability to read and understand the content on each page and navigate the system.

Figure 11 demonstrate is the login page that all the users will use. The login page will be the first page the user will land, and they will have to enter the login credentials to access the contents of the system.

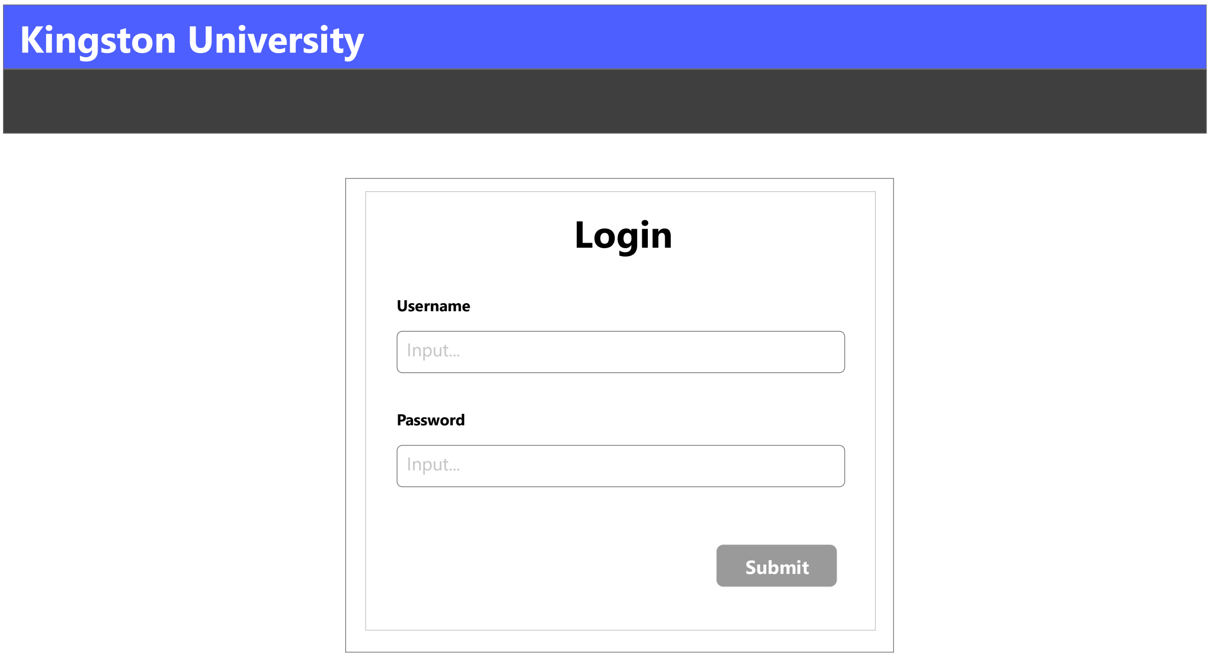
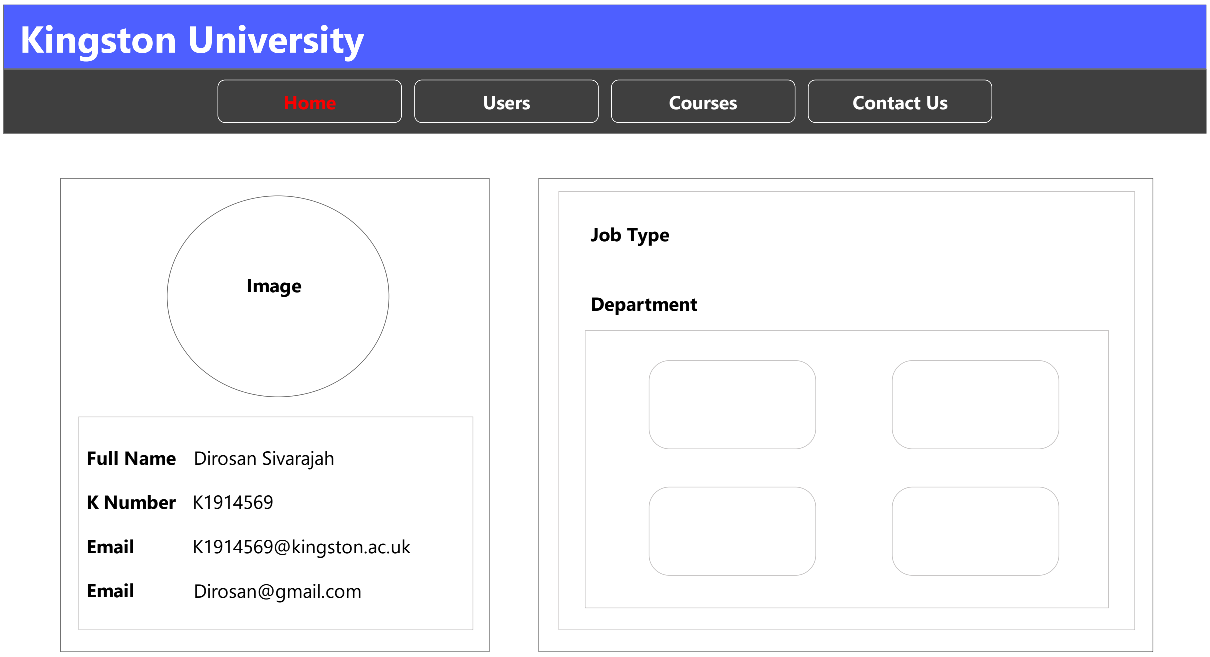


Figure 11 wireframe login

Figure 12 is the home page where it illustrates the user profile and showcases the user's details, which department they are in, and displays all the modules they are part of.



Full Name	Dirosan Sivarajah
K Number	K1914569
Email	K1914569@kingston.ac.uk
Email	Dirosan@gmail.com

Figure 12 wireframe Home page

Figure 13 is for the student users to view their module attendance, and as it shows in the frame, it will hold a pie chart of their induvial attendance of the module sessions (Lecture and Workshop).

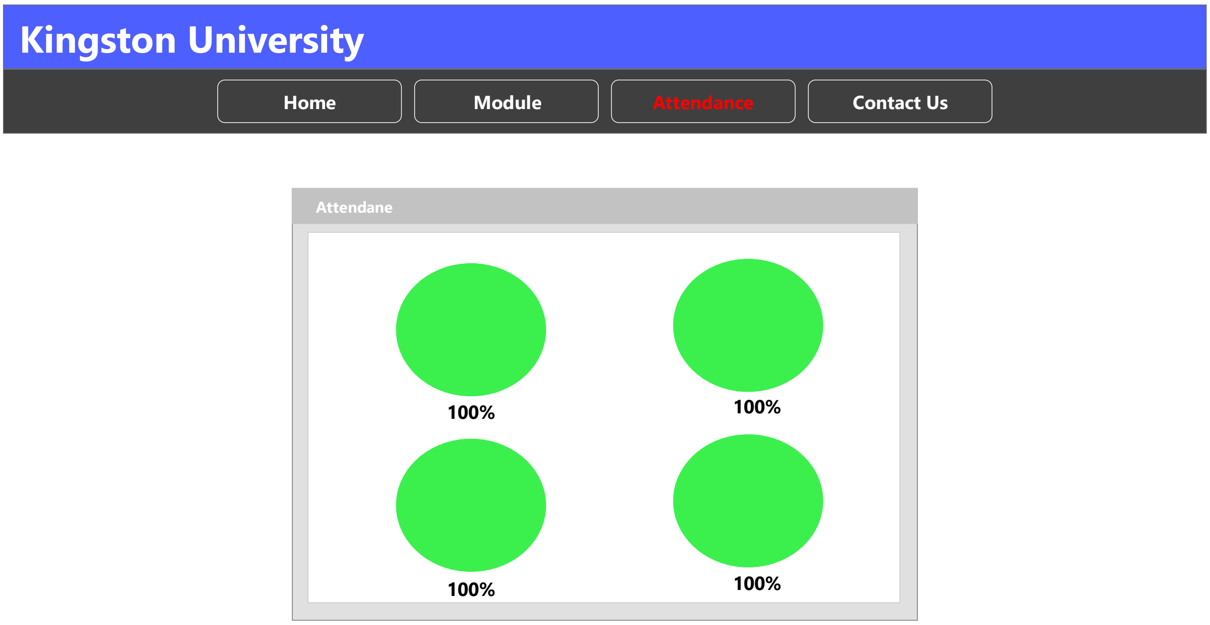


Figure 13 wireframe attendance

4.3.3 Navigation diagram

The navigation diagram helps understand the user options and what they can do on the system. It is one of the most important parts of the system as it showcases the routes of the induvial user and the site content that they will be able to view and access on the system. The routes that the diagram will show the return routes such as when the user Logout, it will take the user back to the login page, where they will have to login again to view the page contents. However, a few functions are similar to all the user types, such as the login, logout, home page, etc. Once they log in, all the users will first land directly on the home page.

The admin has complete control over what the users can see and do. Furthermore, the admin can create, edit, and delete users, courses, modules, reset user passwords and access rights, etc. Figure 2 shows what the admin can see and do on the system. For example, the admin can add new users and assign courses and modules to those users. On the system, each user is defined by their roles: admin, course leader, Lecturer, and student. To see and understand each user's access and view of other users can be found in the APPENDIX.

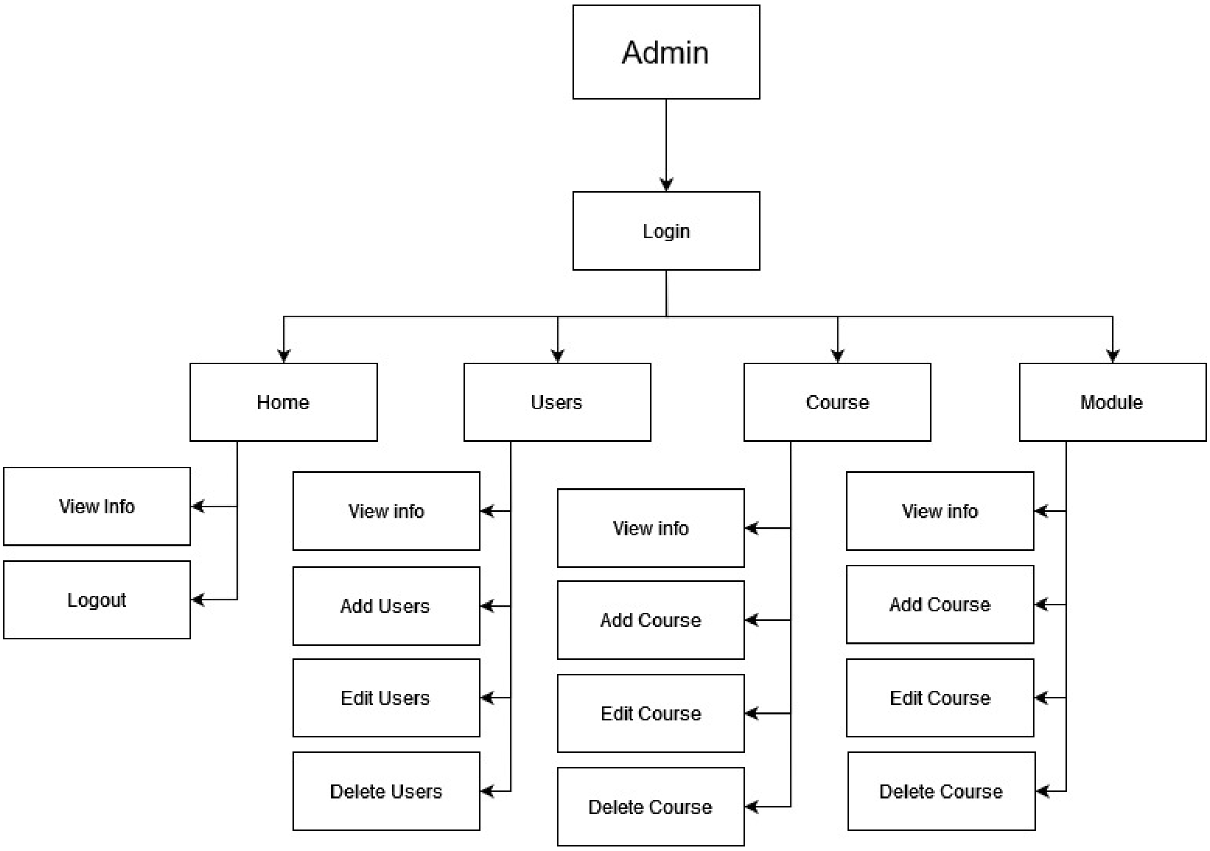


Figure 14 navigation diagram admin

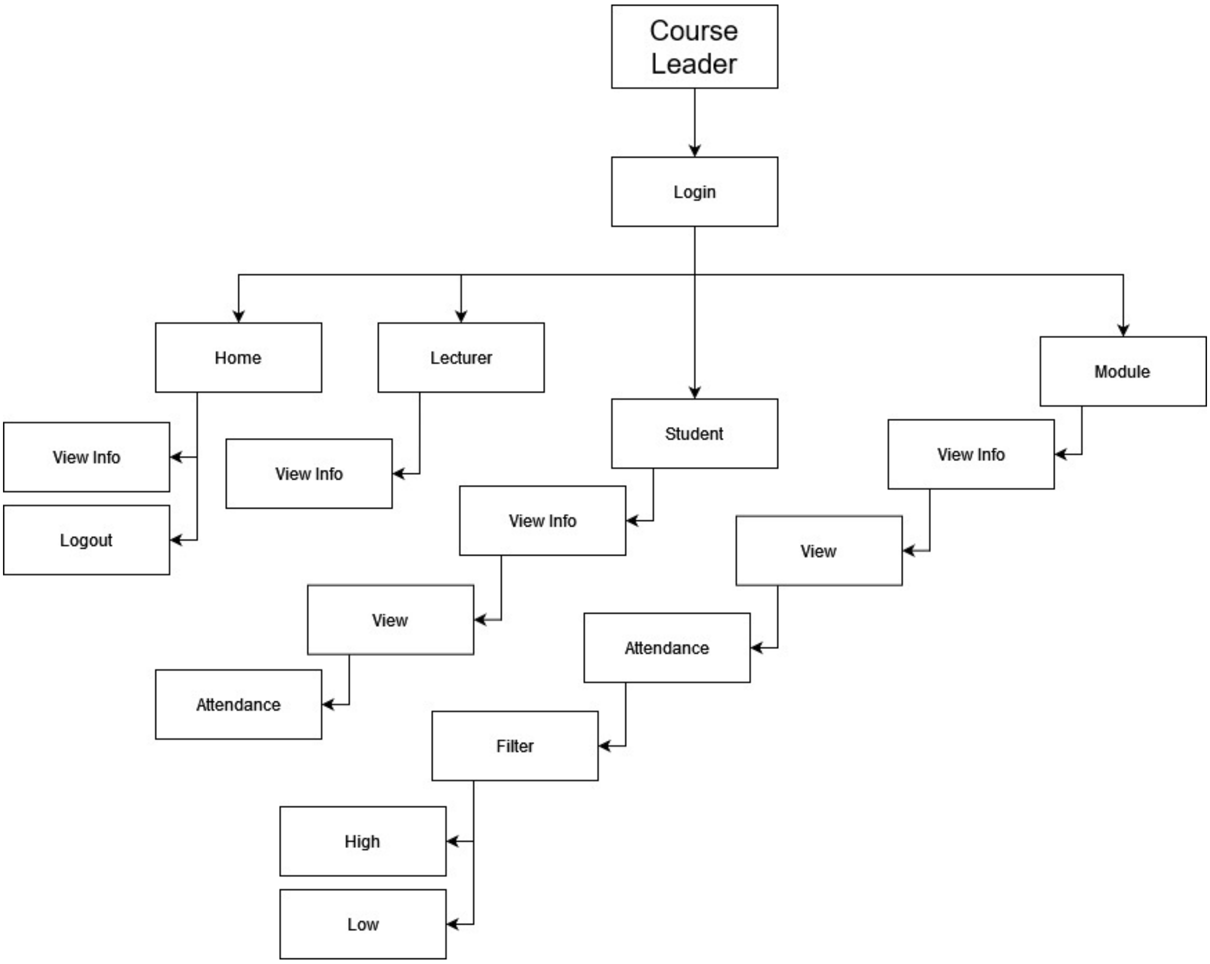


Figure 15 navigation diagram course leader

4.4 Conclusion

The research used in the design chapter to show the outcome of the system for the client and getting feedback from the client helped to make the necessary changes to the design before the implementation phase. Furthermore, it helped redesign the database and resolve the relationship between tables. This would allow the database to be implemented and necessary data to be populated into the database. In the end, the design chapter helped solve any problems that would cause before the implementation phase started, and the Entity-relationship diagram was changed a few times as some of the requirements were changed by the client. This delayed some parts of the design phase, as a few more changes needed to be made on the wireframe, and those changes affected other diagrams, but the problems were solved.

5 Implementation

5.1 Introduction

The implementation segment will demonstrate how the application was developed and various tools, technologies, and languages used to implement a functional application that would allow the users to use to achieve their end goals. The research was conducted to identify which tools, technologies and languages will help to develop the application. Furthermore, the segment will also cover creating the database that will be used in this application store data such as (User, Course, Module, etc.). The sample code of the core functions implemented on the application to meet the user requirement will focus on the backend and front-end of the code.

5.2 Tools, Technologies and Architecture

This report segment will cover a range of tools and technologies that will be explored to implement the application. The tools section will cover all the tools used to develop the application and the platforms that were used while developing the application using the user requirements. The technologies section will cover the coding language used, and architecture will cover the discussions made in the development phase.

5.2.1 Technologies

Visual Studio Code:

It is an open-source and the best platform for an editing tool. It will become beneficial for the developer as Visual Studio Code highlights codes in different colours to differentiate the codes to help understand the purpose of the code. Furthermore, it has various features that can be very useful in the development of the application. For instance, auto-complete is beneficial as it predicts the code and speeds up the coding process. In addition to this, it has another feature as it allows extensions to be installed which could be used in the development of the application.

GitHub:

It helps to keep all the code stored in the GitHub, and it can be accessed on any device long as there is an internet connection. Furthermore, it will help the developer manage the project version. Every time the project code is updated, it can be pushed to the GitHub to be stored. The advantage of using GitHub is that it will hold the code's new and old versions. This will allow the developer to use the old code if the project encounters problems that couldn't be fixed.

phpMyAdmin:

phpMyAdmin will be used to manage the database and all the data stored by the developer. Furthermore, it allows to create/drop tables, create an association between tables, and insert data to tables using MySQL statements. It is beneficial as it will enable the developer to test the SQL query to insert or delete data from a table on the database.

Draw.io:

It is used to create the MUL models such as Use case, Sequence, Activity and (ERD) Entity-Relationship diagrams. It will allow the layout of the database's design and how it communicates with each table.

AdobeXD:

It was used to create high-fidelity wireframes to design the applications and see how each page would look and what data each page would have. It is used by user experience/user interface designers to test all the pages before it is built. It helps to identify any issues and problems that can be fixed before building the web application.

Mockaroo:

It is a website that many users use to generate datasets automatically. It is mainly used to generate dummy data as this project will use dummy data, and using Mackaroo will help generate any data required for this project, for instance, name, address, phone number, etc.

5.2.2 Tools

Node js:

It is an API service that will help connect to the database and allow the application to communicate with the database by retrieving and storing data. The advantage of using node js is that it will allow all the functions to be executed simultaneously.

React js:

It is an open-source JavaScript library that will help create a dynamic web application and help the developer manage. Furthermore, it helps to handle the view layer of the web application and allows reusable comments that can be used in the code. It is straightforward to learn as many recourses are available online that help, for instance, when reusable coding components can be used in the development, and this will stop the developer from repeating the same code again.

MySQL:

It is a query language used to run queries to the database to carry out tasks. The SQL will be used in the Backend to Select, Delete, Update, and Insert data into the database and manage the database using the query.

Express.js:

The express is a framework for Node.js, and it helps build the backend API of the application and helps set up the ports and route handlers for the application to communicate. The express is used to handle the requests which is made on the application(Auderer, 2018).

Axios.js:

It is a Javascript library that is used to make the HTTP requests to (GET, INSERT, DELETE AND EDIT) data—for instance, using the HTTP request to communicate with the backend to send a request that is made by the user(Kollegger, 2018).

5.2.3 Framework

CSS Bootstrap:

It is a framework that helps create a res responsive user interface, and it is an open-source library for the front-end. Furthermore, with the help of Bootstrap, it helps to focus on the main functionality of the system. It holds pre-styled components such as navbar, card, table, colour, button, style, etc. Using all the components will help to give a better user interface and responsiveness to the application.

5.2.4 Architecture

The architecture of this application is shown below, and figure 16 demonstrates how the application will communicate with each other. The database is on a remote server that holds all the dummy data that will be used on this application. For instance, the Lecturer wants to view all the student attendance from the front-end, an HTTP request will be sent to the backend API, and then the Backend API will take the request and then it communicates with the database using the SQL query to fetch all the student and their module attendance. The database then executes the request, fetches all the data related to the request, and sends it back to the Backend API and then it displays on the front end for the Lecturer to view the student attendance.

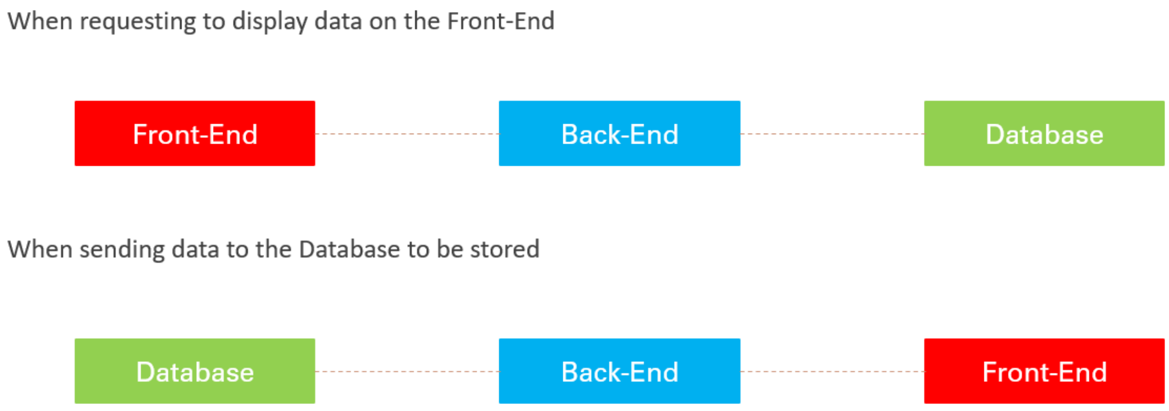

When requesting to display data on the Front-End
Front-End
Back-End
Database
When sending data to the Database to be stored
Database
Back-End
Front-End

Figure 16 architecture

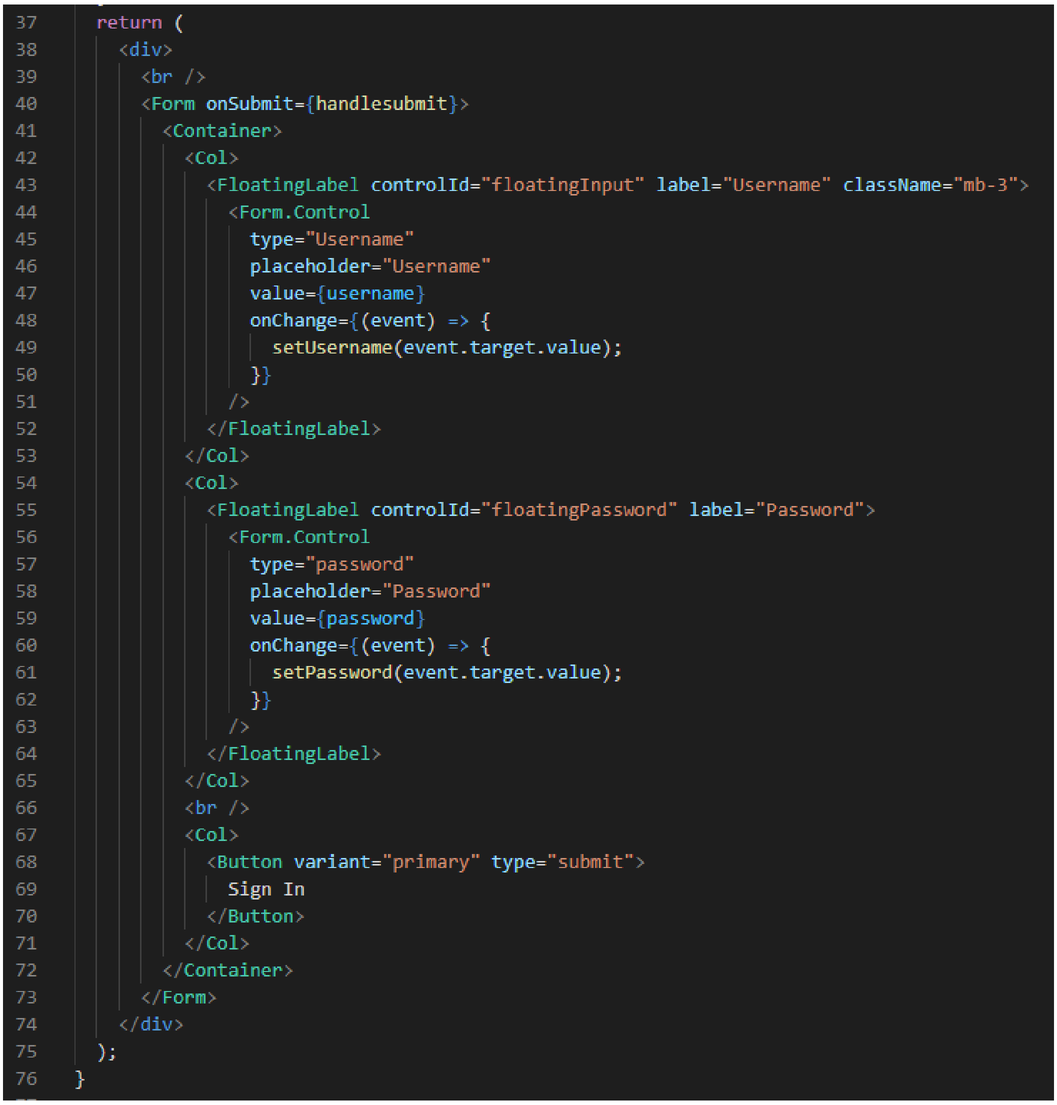
5.3 Frontend implementation

The front-end implementation will focus on the front-end code for the users to interact with the application through the browser.

5.3.1 Login Page

The login was an important part of the application. It acts as a gateway for the users to access their accounts on the application. The login is used to differentiate the user access rights as some contents on the page can't be seen by certain users. The login was one of the must-have requirements for the users to log in to the application. As to meet the stakeholder request, login was implemented. For instance, the admin has full control over the application and has to ability to change user access rights for the users. To access the home page, the user will have to enter the login credentials to access the home page.

The Figure 17 shows that it is a form with two input fields to enter the username and password, and a button is in place to send any request to verify the login credentials. If the user enters valid login credentials, it will automatically send the user to the home page. However, if the login credentials are incorrect, it will display an error message and displays the login page again, asking the user to enter the username and password again.



```
  return (
    <div>
      <br />
      <Form onSubmit={handlesubmit}>
        <Container>
          <Col>
            <FloatingLabel controlId="floatingInput" label="Username" className="mb-3">
              <Form.Control
                type="Username"
                placeholder="Username"
                value={username}
                onChange={(event) => {
                  setUsername(event.target.value);
                }}
              />
            </FloatingLabel>
          </Col>
          <Col>
            <FloatingLabel controlId="floatingPassword" label="Password">
              <Form.Control
                type="password"
                placeholder="Password"
                value={password}
                onChange={(event) => {
                  setPassword(event.target.value);
                }}
              />
            </FloatingLabel>
          </Col>
          <br />
          <Col>
            <Button variant="primary" type="submit">
              Sign In
            </Button>
          </Col>
        </Container>
      </Form>
    </div>
  );
}
```

Figure 17 login form code

The Figure 18 shows the login form the user will be available to view on the browser, and as mentioned before, the form only requires the user to enter their login credentials to access the application.

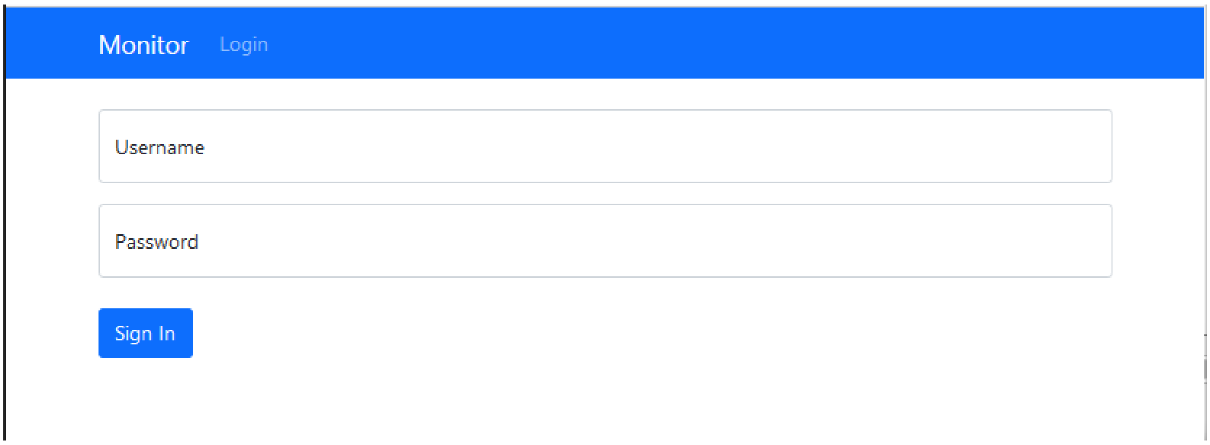
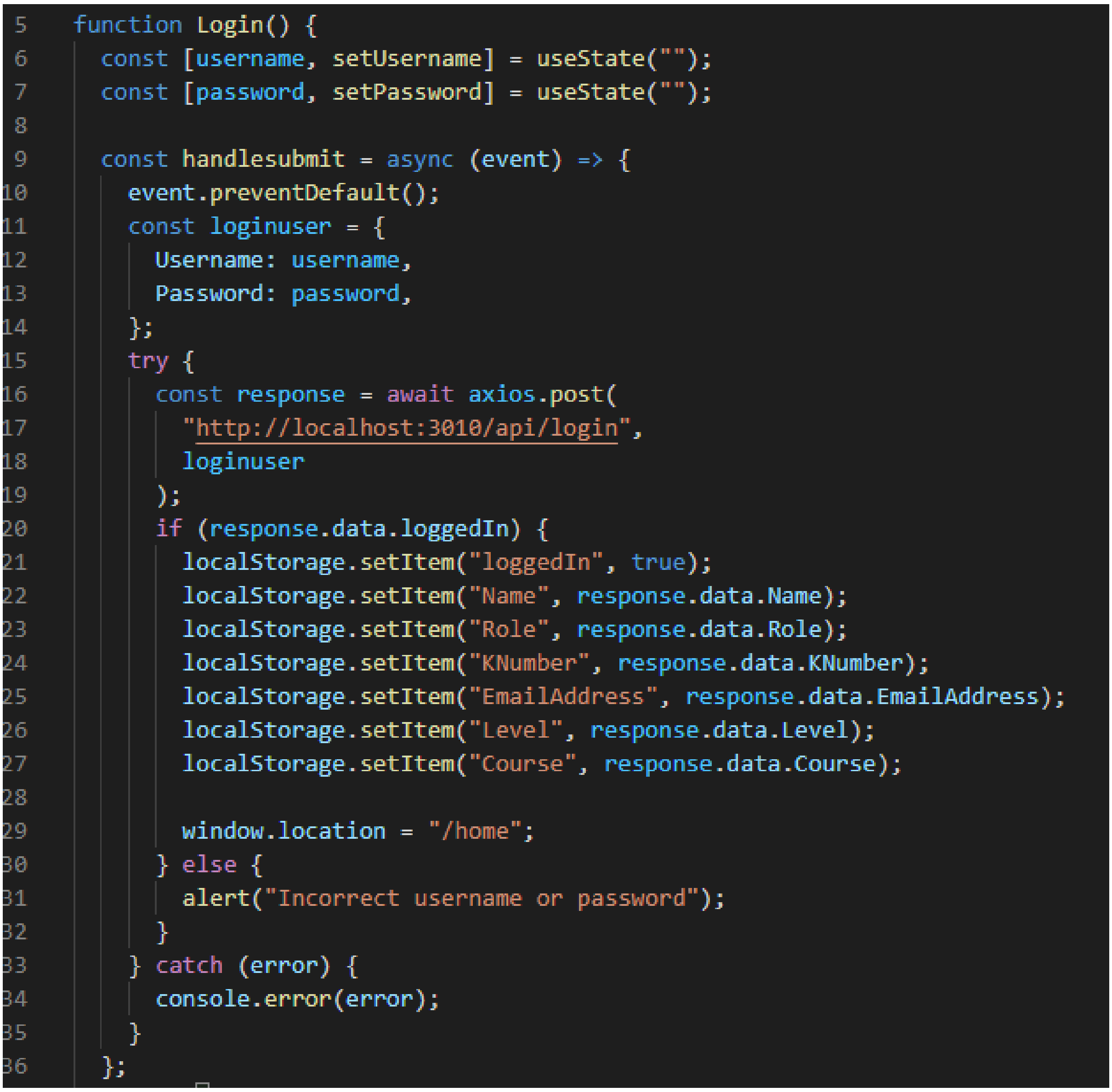


Figure 18 login form

For the login page, a function was created, and within the function, two const will take the username and password. Const is created to handle the request when the Login button is clicked. It sends an HTTP request to the Backend API, with the SQL query, the Backend API communicate with the remote database server to fetch all the requested data when the login credentials are valid and if the login credentials are invalid, it will display an error message on the login page.

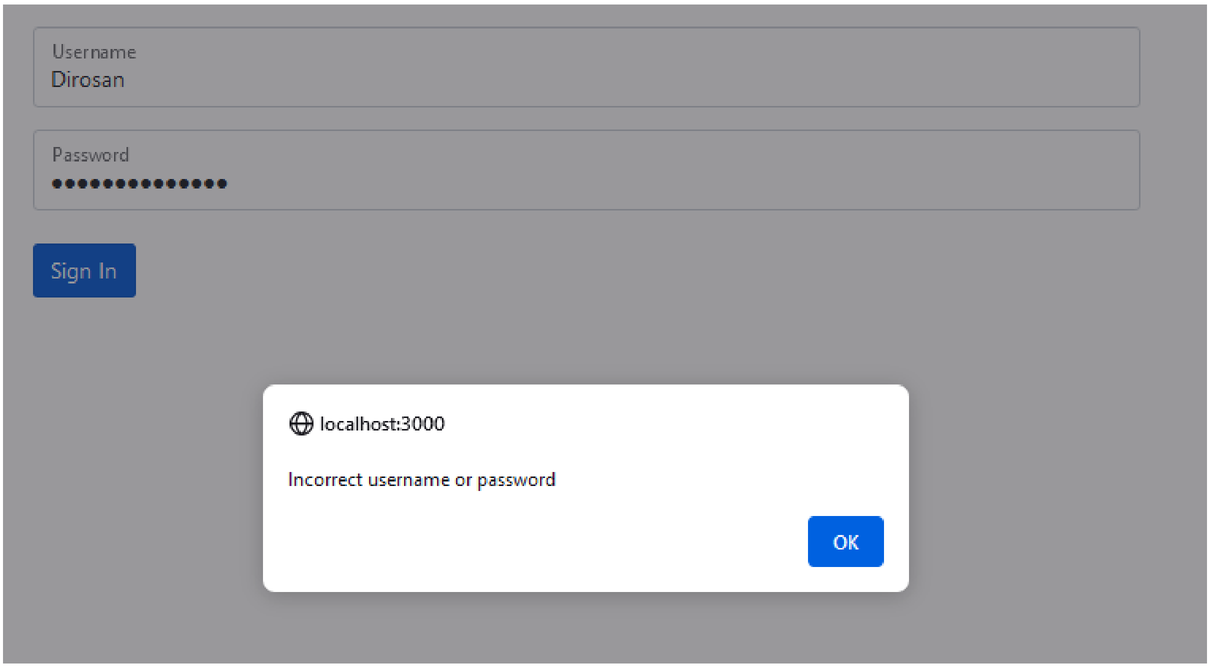


```javascript
function Login() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");

  const handlesubmit = async (event) => {
    event.preventDefault();
    const loginuser = {
      Username: username,
      Password: password,
    };
    try {
      const response = await axios.post(
        "http://localhost:3010/api/login",
        loginuser
      );
      if (response.data.loggedIn) {
        localStorage.setItem("loggedIn", true);
        localStorage.setItem("Name", response.data.Name);
        localStorage.setItem("Role", response.data.Role);
        localStorage.setItem("KNumber", response.data.KNumber);
        localStorage.setItem("EmailAddress", response.data.EmailAddress);
        localStorage.setItem("Level", response.data.Level);
        localStorage.setItem("Course", response.data.Course);

        window.location = "/home";
      } else {
        alert("Incorrect username or password");
      }
    } catch (error) {
      console.error(error);
    }
  };
```

Figure 19 login form function

Error message:

Figure 20 shows that when invalid login credentials are entered, it will display an error message to prevent unauthorised access to the user account on the application.


Username
Dirosan

Password
••••••••••••••

localhost:3000

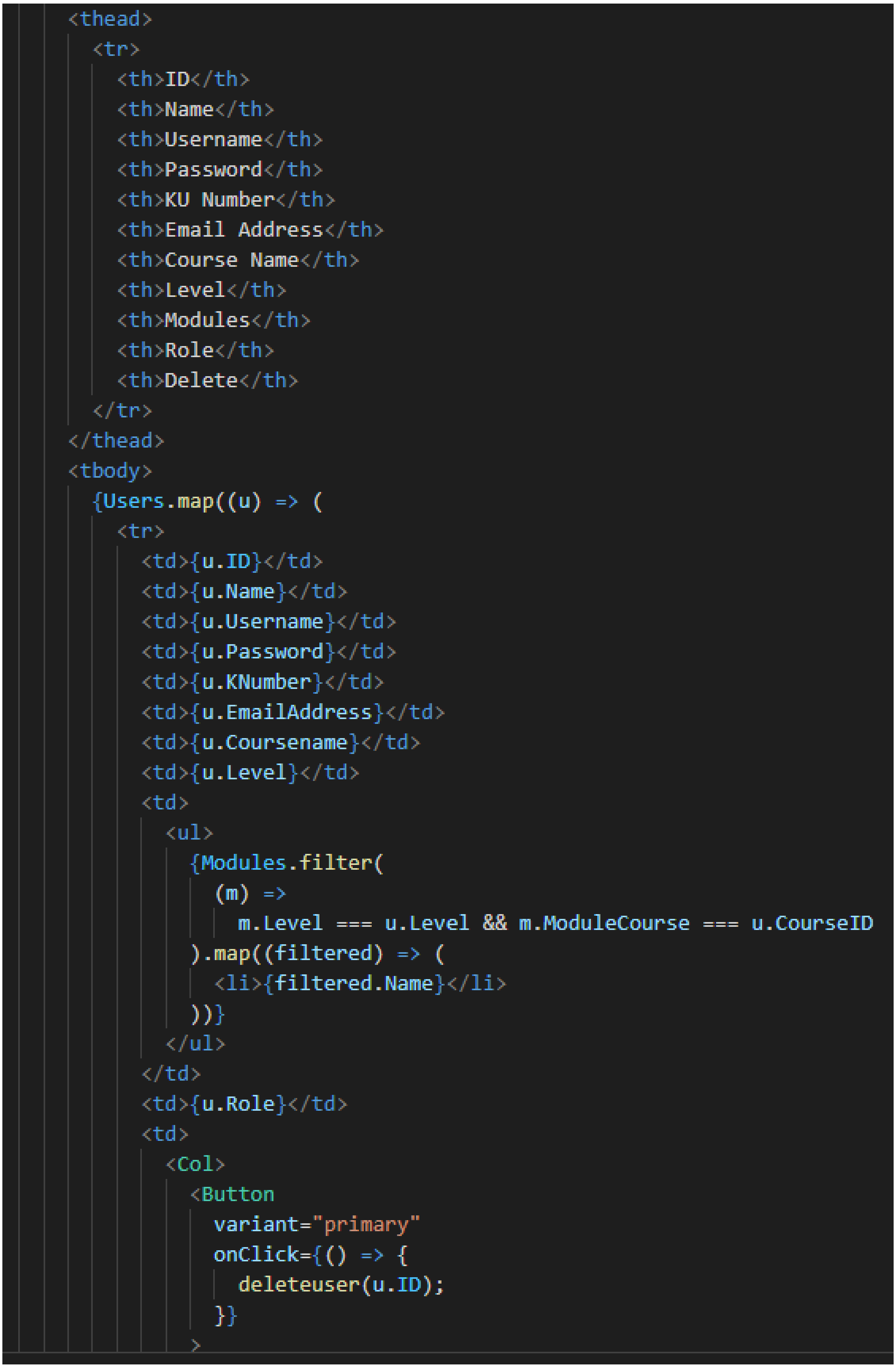
Incorrect username or password

Figure 20 login form error message

5.3.2 Add user page

The add user page is only available for the admin only as the must-have requirement to add users to the application. Multiple functions are implemented on the Add user page to do various tasks such as GET, POST, and DELETE on one page. This was considered to help the admin to manage the users more efficiently as the user data is stored in one place in the database user table.

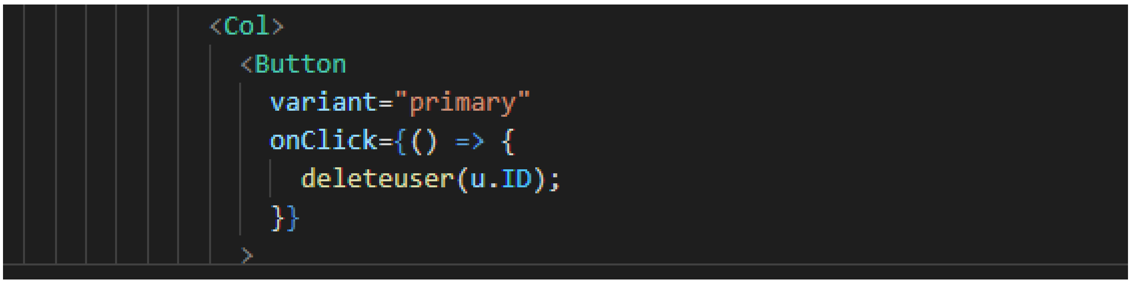
The GET request is to fetch all the User, Module, and Course lists. On the add page, a list of users is displayed in a table, and the table holds all the information about the user. For instance, the admin is able to see the Name, Username, Password, Course, Module, Level etc. The use of map() method created an array by calling all the provided functions on the table.



```
<thead>
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Username</th>
    <th>Password</th>
    <th>KU Number</th>
    <th>Email Address</th>
    <th>Course Name</th>
    <th>Level</th>
    <th>Modules</th>
    <th>Role</th>
    <th>Delete</th>
  </tr>
</thead>
<tbody>
  {Users.map((u) => (
    <tr>
      <td>{u.ID}</td>
      <td>{u.Name}</td>
      <td>{u.Username}</td>
      <td>{u.Password}</td>
      <td>{u.KNumber}</td>
      <td>{u.EmailAddress}</td>
      <td>{u.Coursename}</td>
      <td>{u.Level}</td>
      <td>
        <ul>
          {Modules.filter(
            (m) =>
              m.Level === u.Level && m.ModuleCourse === u.CourseID
          ).map((filtered) => (
            <li>{filtered.Name}</li>
          ))}
        </ul>
      </td>
      <td>{u.Role}</td>
      <td>
        <Col>
          <Button
            variant="primary"
            onClick={() => {
              deleteuser(u.ID);
            }}
          >
```

Figure 21 List of user table

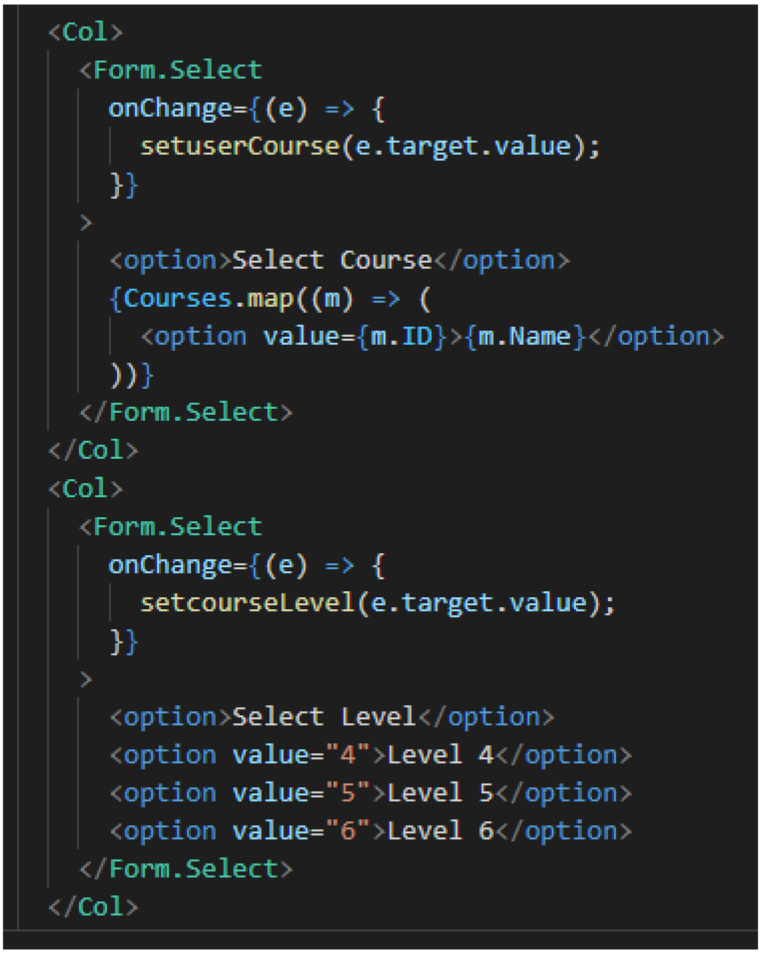
The DELETE request is to delete all the user data from the user table. The purpose of the delete button was implemented to allow the admin to delete users who no longer need access to the application. The DELETE request works by using the user's ID to delete the user from the database. When the Delete button is clicked, it sends an HTTP request to the Backend API, using the SQL query, it cross-checks the UserID with all the UserID on the user table on the database. Once it finds a match, it will delete the user from the database.



```
<Col>
  <Button
    variant="primary"
    onClick={() => {
      deleteuser(u.ID);
    }}
  >
```

Figure 22 Delete button

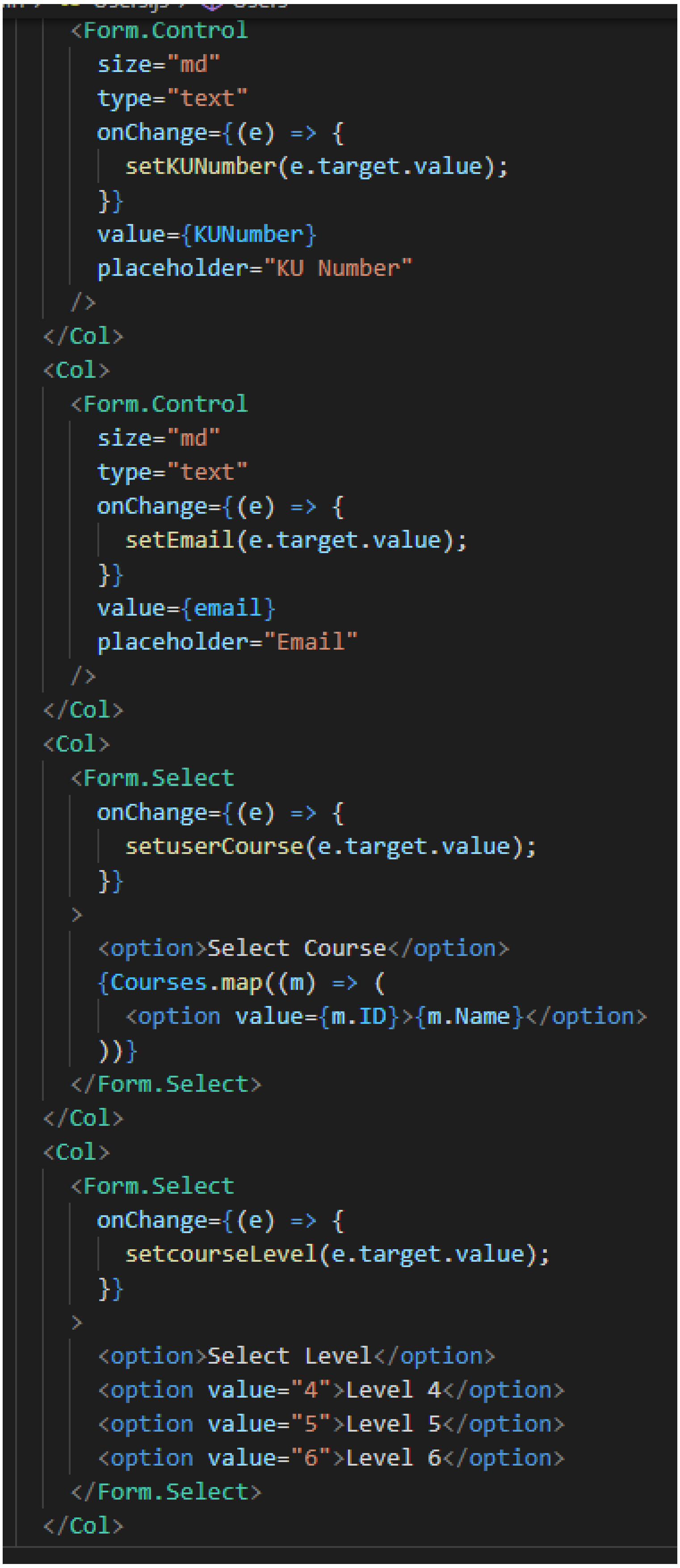
The POST request is to add new users to the database by filling in the user form with all the required input fields with user information. The add user form is not simple as the form has a dropdown option that displays data fetched from the database to display as an option for the admin to select when adding new users using the form. The Figure shows that the list of courses is being fetched from the course table using GET request to display as an option using map() method. This will allow the admin to select the course the student will study, and by clicking the Add user button, it will send a POST request with all the user data to the Backend API. Then using the SQL query function will interest the user information by filling all the attributes on the user table.



```
<Col>
  <Form.Select
    onChange={(e) => {
      setuserCourse(e.target.value);
    }}
  >
    <option>Select Course</option>
    {Courses.map((m) => (
      <option value={m.ID}>{m.Name}</option>
    ))}
  </Form.Select>
</Col>
<Col>
  <Form.Select
    onChange={(e) => {
      setcourseLevel(e.target.value);
    }}
  >
    <option>Select Level</option>
    <option value="4">Level 4</option>
    <option value="5">Level 5</option>
    <option value="6">Level 6</option>
  </Form.Select>
</Col>
```

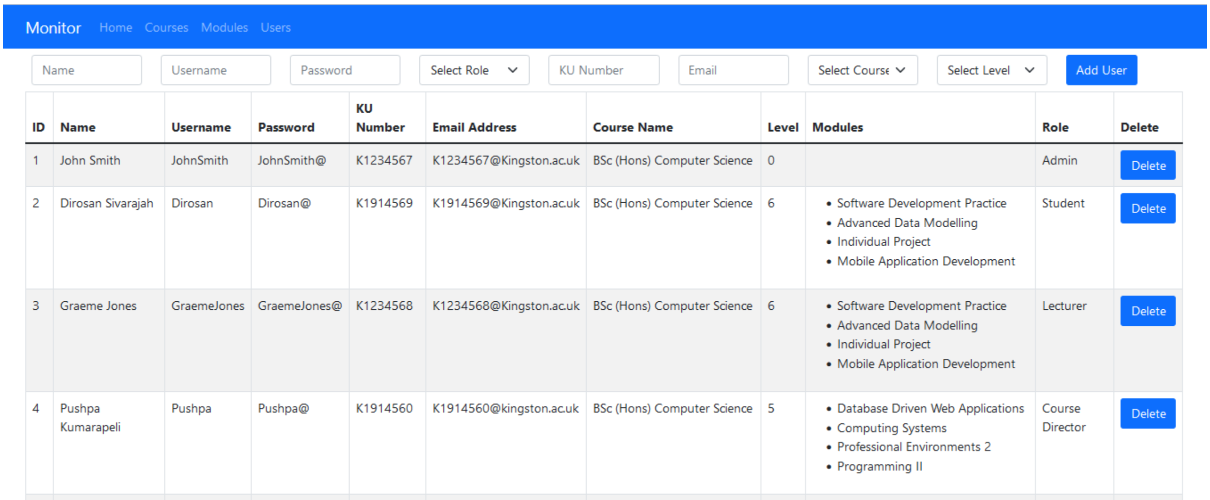
Figure 23 Select option

The add user form is complex as some of the must-haves been implemented to achieve multiple user requirements. This is to ensure that many forms are not required for certain tasks. For instance, the add user form also assigns modules to users when all the form options are correctly inputted. A function was implemented to check the user Course and Level to automatically assign the module to the user when the user is added to the database. The assigned module was a must-have requirement for students to be assigned modules depending on the course and level.



```
  <Form.Control
    size="md"
    type="text"
    onChange={(e) => {
      setKUNumber(e.target.value);
    }}
    value={KUNumber}
    placeholder="KU Number"
  />
</Col>
<Col>
  <Form.Control
    size="md"
    type="text"
    onChange={(e) => {
      setEmail(e.target.value);
    }}
    value={email}
    placeholder="Email"
  />
</Col>
<Col>
  <Form.Select
    onChange={(e) => {
      setuserCourse(e.target.value);
    }}
  >
    <option>Select Course</option>
    {Courses.map((m) => (
      <option value={m.ID}>{m.Name}</option>
    ))}
  </Form.Select>
</Col>
<Col>
  <Form.Select
    onChange={(e) => {
      setcourseLevel(e.target.value);
    }}
  >
    <option>Select Level</option>
    <option value="4">Level 4</option>
    <option value="5">Level 5</option>
    <option value="6">Level 6</option>
  </Form.Select>
</Col>
```

Figure 24 Select option



ID	Name	Username	Password	KU Number	Email Address	Course Name	Level	Modules	Role	Delete
1	John Smith	JohnSmith	JohnSmith@	K1234567	K1234567@Kingston.ac.uk	BSc (Hons) Computer Science	0		Admin	Delete
2	Dirosan Sivarajah	Dirosan	Dirosan@	K1914569	K1914569@Kingston.ac.uk	BSc (Hons) Computer Science	6	• Software Development Practice • Advanced Data Modelling • Individual Project • Mobile Application Development	Student	Delete
3	Graeme Jones	GraemeJones	GraemeJones@	K1234568	K1234568@Kingston.ac.uk	BSc (Hons) Computer Science	6	• Software Development Practice • Advanced Data Modelling • Individual Project • Mobile Application Development	Lecturer	Delete
4	Pushpa Kumarapeli	Pushpa	Pushpa@	K1914560	K1914560@kingston.ac.uk	BSc (Hons) Computer Science	5	• Database Driven Web Applications • Computing Systems • Professional Environments 2 • Programming II	Course Director	Delete

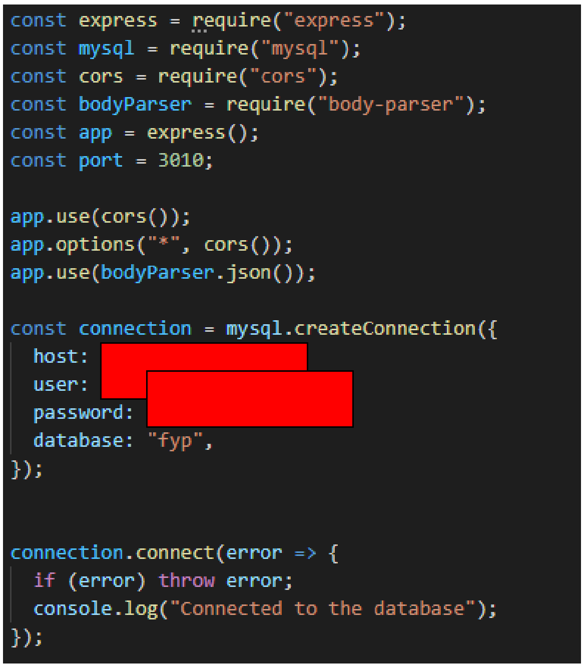
Figure 25 User page

5.4 Backend implementation

The backend API is the application's core as it creates communication between the user and database. This section will go into more depth about how the application communicates with the database to store and retrieve data from the database tables.

5.4.1 Database Connecting

From the backend API, the database connection was created as the first part of the implementation of the application. To create the connection, MySQL library was installed to connect to the database, which allows to creation of the necessary tables that will store the application data on the database tables. (const connection = mysql.createConnection) will have all the details such as the URL of the database server, username, password and database file name. Using all those details, a connection is made from the backend API. When the connection is made, it will print out the message to indicate that the database is connected with the Backend API

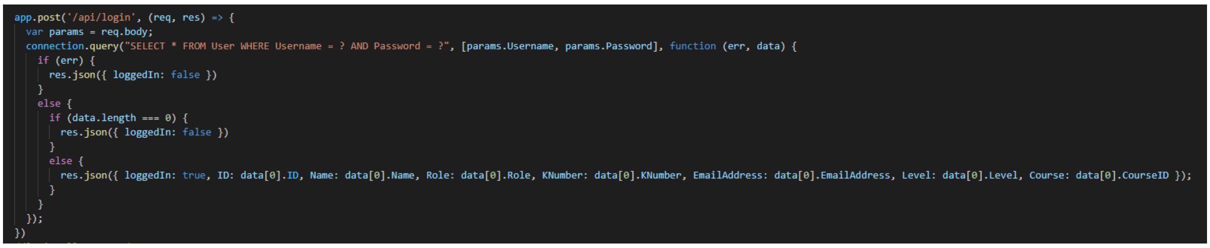


```
const express = require("express");
const mysql = require("mysql");
const cors = require("cors");
const bodyParser = require("body-parser");
const app = express();
const port = 3010;

app.use(cors());
app.options("*", cors());
app.use(bodyParser.json());

const connection = mysql.createConnection({
  host: [redacted]
  user: [redacted]
  password:
  database: "fyp",
});


connection.connect(error => {
  if (error) throw error;
  console.log("Connected to the database");
});
```

Figure 26 database connection

5.4.2 Login

Using POST to select the user information by Username and Password to display all the information related to the user on the dashboard. The IF statement is being used because it is required to make the login work, and when (res.json({logedIn:true}), it will fetch all the data that it is being asked to fetch.

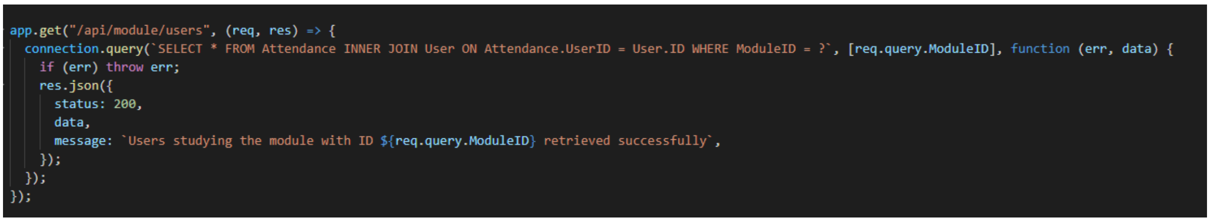


```
app.post('/api/login', (req, res) => {
  var params = req.body;
  connection.query("SELECT * FROM User WHERE Username = ? AND Password = ?", [params.Username, params.Password], function (err, data) {
    if (err) {
      res.json({ loggedIn: false })
    }
    else {
      if (data.length === 0) {
        res.json({ loggedIn: false })
      }
      else {
        res.json({ loggedIn: true, ID: data[0].ID, Name: data[0].Name, Role: data[0].Role, KNumber: data[0].KNumber, EmailAddress: data[0].EmailAddress, Level: data[0].Level, Course: data[0].CourseID });
      }
    }
  });
})
```

Figure 27 login

5.4.3 Attendance

The GET request to fetch all the students' attendance by doing INNER join to fetch the students related to the module, when the lecturer or course director press the view button on the front-end.



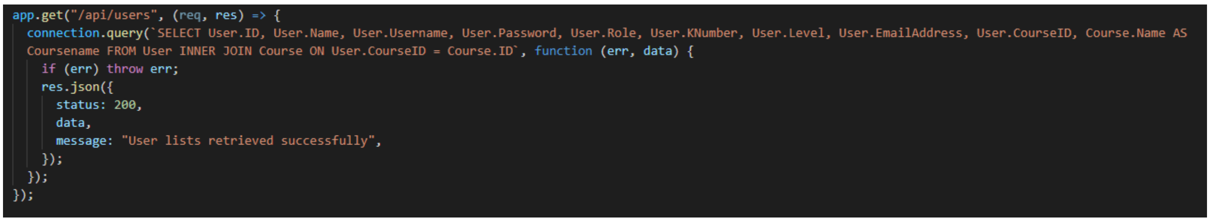
```
app.get("/api/module/users", (req, res) => {
  connection.query(`SELECT * FROM Attendance INNER JOIN User ON Attendance.UserID = User.ID WHERE ModuleID = ?`, [req.query.ModuleID], function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: `Users studying the module with ID ${req.query.ModuleID} retrieved successfully`,
    });
  });
});
```

Figure 28 attendance

5.4.4 User

The user is the core part of this system as it holds all the information about the user on the database and is displayed on the front-end for the admin to view and manage the user details. The front-end displays data such as name, course, module, level, role, login credential, etc. The admin is available to edit and delete users. For instance, one change in the user details will affect the user page. However, as this is a prototype, the system only holds dummy data of the users.

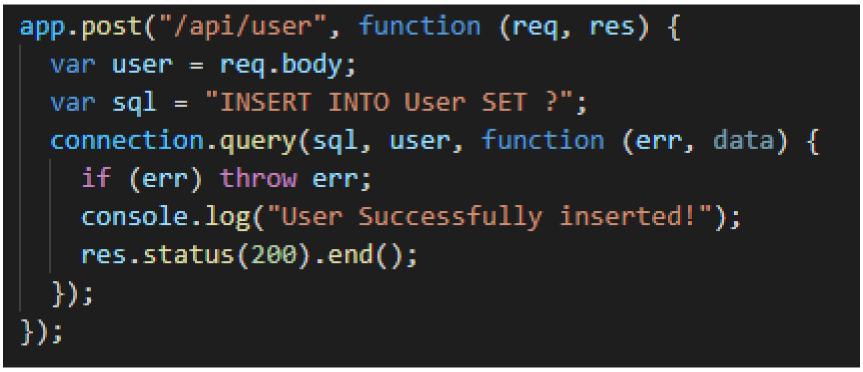
The get request is fetching all the data from the database of the user, and using INNER JOIN allows to get more data from other tables on the data as the tables have a relation with each other. The Figure shows the course and module from different tables on the database using the INNER JOIN SQL statement. The help of the SQL statement allows selecting only the data required from the database to display on the front-end.



```
app.get("/api/users", (req, res) => {
  connection.query(`SELECT User.ID, User.Name, User.Username, User.Password, User.Role, User.KNumber, User.Level, User.EmailAddress, User.CourseID, Course.Name AS
  Coursename FROM User INNER JOIN Course ON User.CourseID = Course.ID`, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "User lists retrieved successfully",
    });
  });
});
```

Figure 29 user

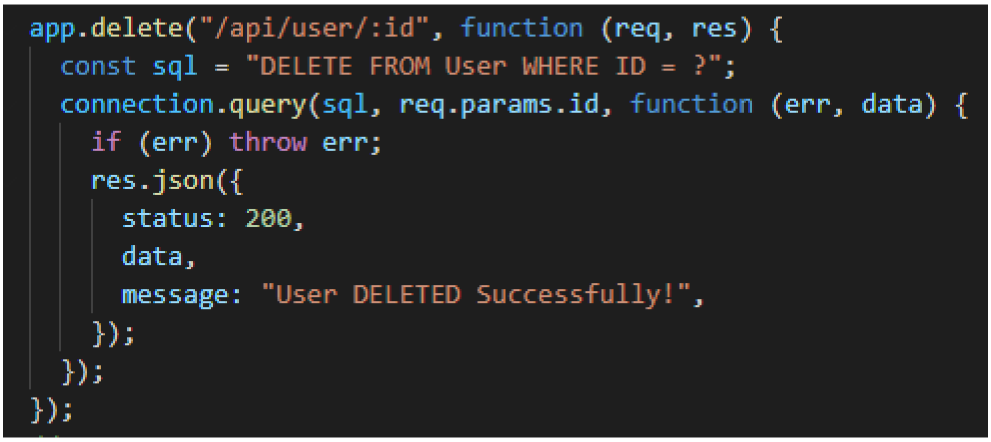
The POST request is communicating with the database to INSERT new user data into the User table.



```
app.post("/api/user", function (req, res) {
  var user = req.body;
  var sql = "INSERT INTO User SET ?";
  connection.query(sql, user, function (err, data) {
    if (err) throw err;
    console.log("User Successfully inserted!");
    res.status(200).end();
  });
});
```

Figure 30 user SQL query

The DELETE request communicates with the database to the DELETE user by the userID.

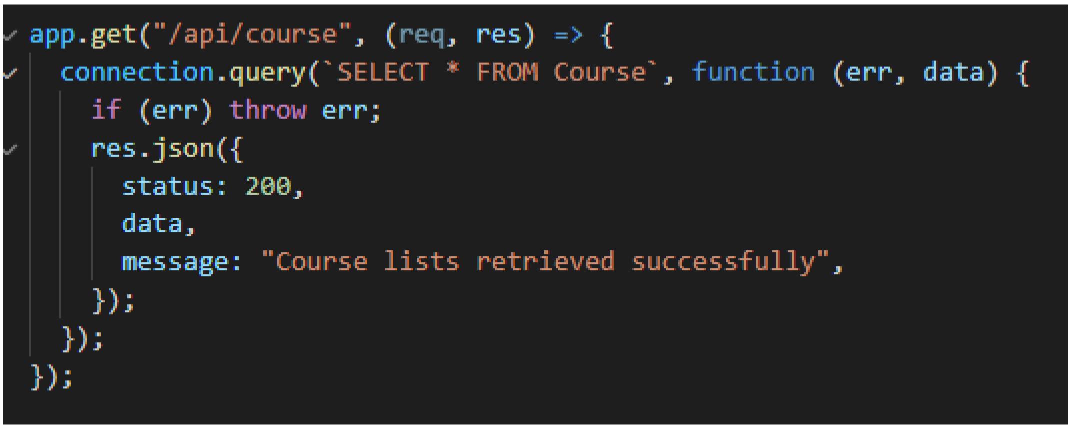


```
app.delete("/api/user/:id", function (req, res) {
  const sql = "DELETE FROM User WHERE ID = ?";
  connection.query(sql, req.params.id, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "User DELETED Successfully!",
    });
  });
});
```

Figure 31 user SQL query

5.4.5 Course

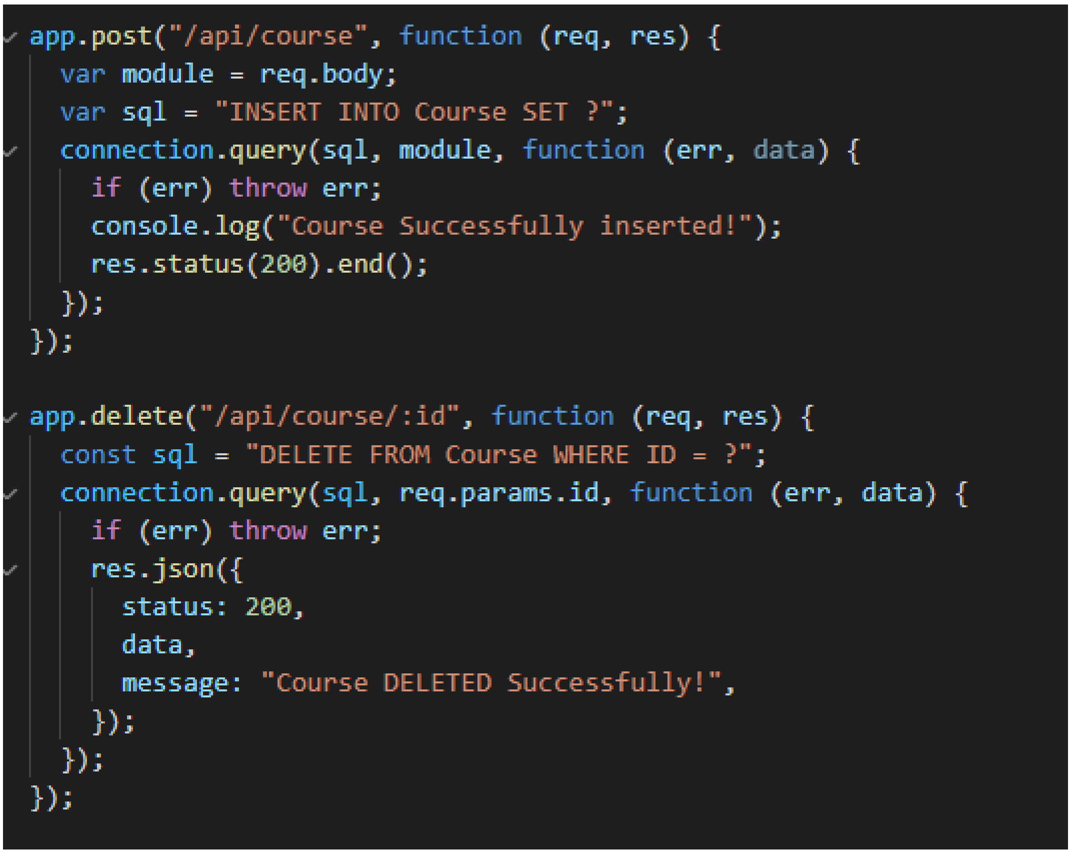
The GET request to fetch all the courses from the course table and display them in the course page.



```
app.get("/api/course", (req, res) => {
  connection.query(`SELECT * FROM Course`, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "Course lists retrieved successfully",
    });
  });
});
```

Figure 32 course SQL query

The POST request to add new course into the Course table and DLETE request to delate the course by CourseID.



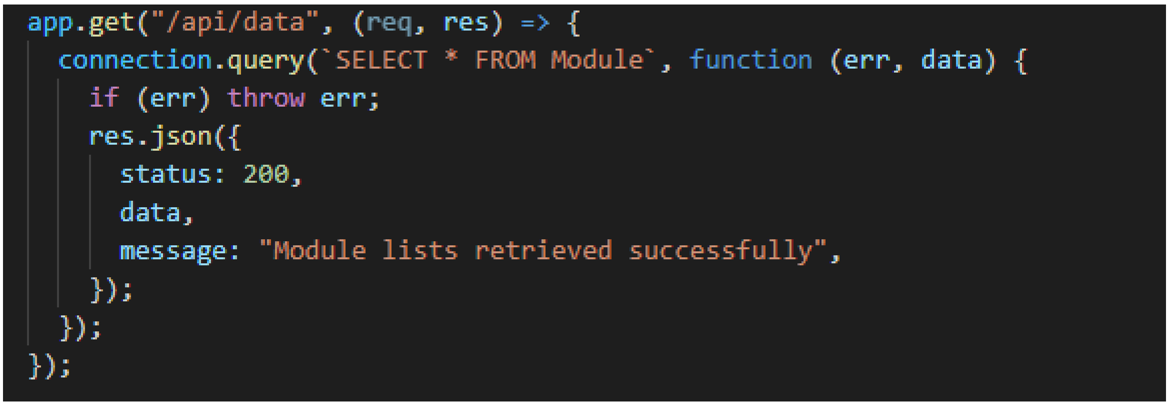
```
app.post("/api/course", function (req, res) {
  var module = req.body;
  var sql = "INSERT INTO Course SET ?";
  connection.query(sql, module, function (err, data) {
    if (err) throw err;
    console.log("Course Successfully inserted!");
    res.status(200).end();
  });
});

app.delete("/api/course/:id", function (req, res) {
  const sql = "DELETE FROM Course WHERE ID = ?";
  connection.query(sql, req.params.id, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "Course DELETED Successfully!",
    });
  });
});
```

Figure 33 course SQL query

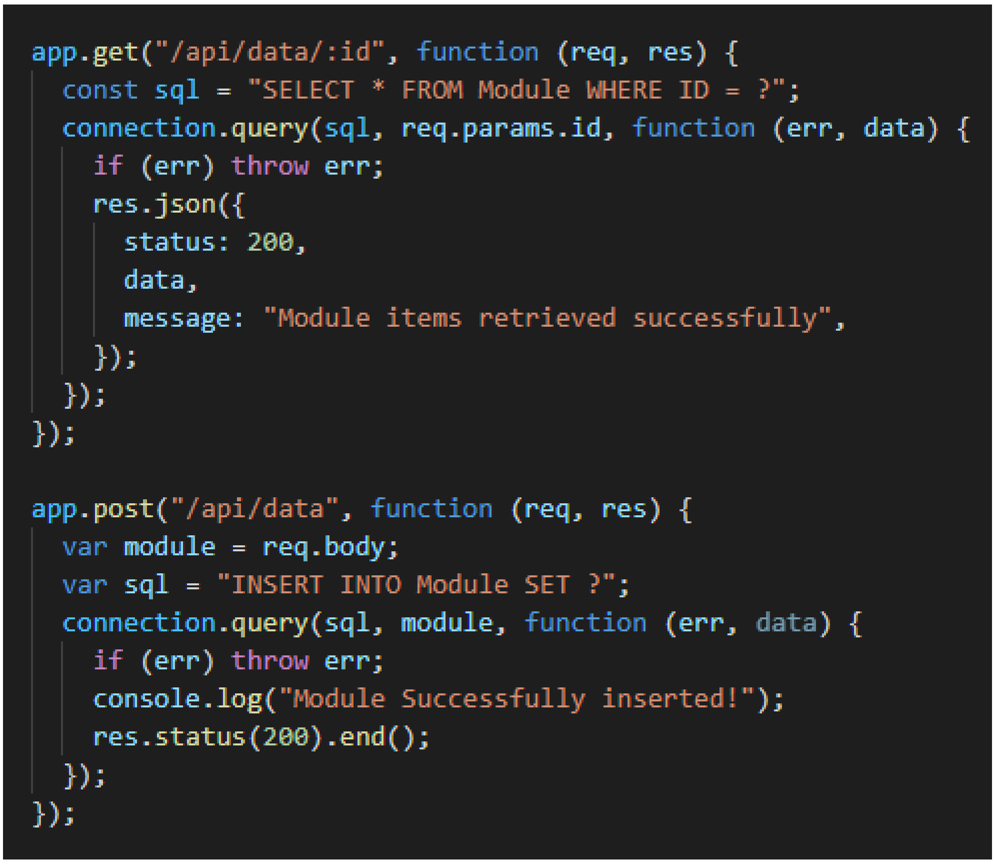
5.4.6 Module

The module is where the admin can Select, Insert, Delete and Update. When the admin alters a module's data, it will automatically update the user. When the admin modifies the data, it will update the database where all the module data is stored. However, the course director and Lecturer only see the module data's name, such as students who do the module, their attendance, and module attendance. However, the student said they could only see the module name and image of the module.

In Figure 34, using get a request to fetch all the module data from the Module table on the database and display it in the front-end for the admin to view and manage and using (Select *) to allow to fetch all the data without mission single data. Once the data is fetched, it will display a message indicating that the module list has been successfully retrieved.



```
app.get("/api/data", (req, res) => {
  connection.query(`SELECT * FROM Module`, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "Module lists retrieved successfully",
    });
  });
});
```



```
app.get("/api/data/:id", function (req, res) {
  const sql = "SELECT * FROM Module WHERE ID = ?";
  connection.query(sql, req.params.id, function (err, data) {
    if (err) throw err;
    res.json({
      status: 200,
      data,
      message: "Module items retrieved successfully",
    });
  });
});

app.post("/api/data", function (req, res) {
  var module = req.body;
  var sql = "INSERT INTO Module SET ?";
  connection.query(sql, module, function (err, data) {
    if (err) throw err;
    console.log("Module Successfully inserted!");
    res.status(200).end();
  });
});
```

Figure 34 module SQL query

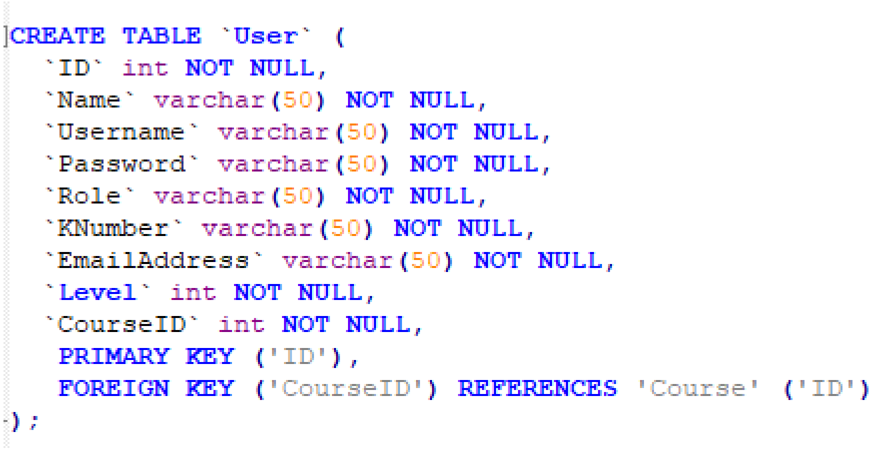
5.5 Database implementation

phpMyAdmin is used to manage the database that will be used to store the data of the application, and it holds various features that will allow creating tables. To create the database tables, the entity-relationship diagram and data dictionary will be used to create all the necessary tables that are required for the application to store and retrieve data. Furthermore, it will ensure that the correct attributes, data type etc., are given appropriately for each table.

The data dictionary will be used to create an SQL statement to create all the tables that are required for the application.

5.5.1 Create User table

The User table will be used to store all the user information when the admin adds all the users to the user page. The user table holds attributes required for the application, and the ID will be the primary key for the user table, and courseID will be a forging key on the user table as the course has one too many users and the user has one course. The course table becomes an association with the user table.

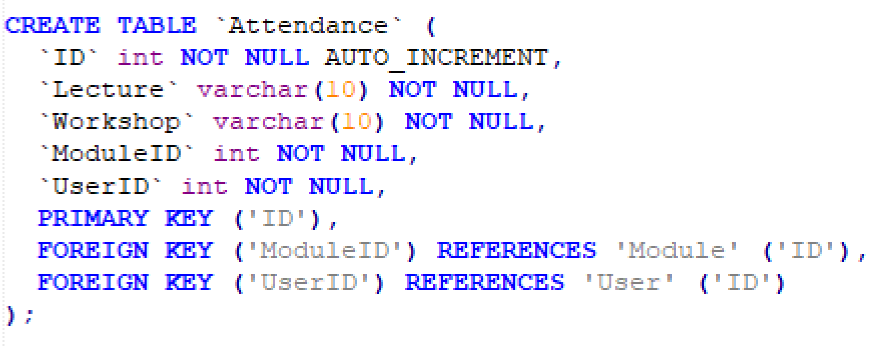


```
CREATE TABLE `User` (
  `ID` int NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Username` varchar(50) NOT NULL,
  `Password` varchar(50) NOT NULL,
  `Role` varchar(50) NOT NULL,
  `KNumber` varchar(50) NOT NULL,
  `EmailAddress` varchar(50) NOT NULL,
  `Level` int NOT NULL,
  `CourseID` int NOT NULL,
   PRIMARY KEY ('ID'),
   FOREIGN KEY ('CourseID') REFERENCES 'Course' ('ID')
);
```

Figure 35 user SQL statement

5.5.2 Create Attendance table

The attendance table is used to store the module attendance of students to allow the course director or Lecturer to identify the student with lower attendance. The attendance table has an association with the module and user table.

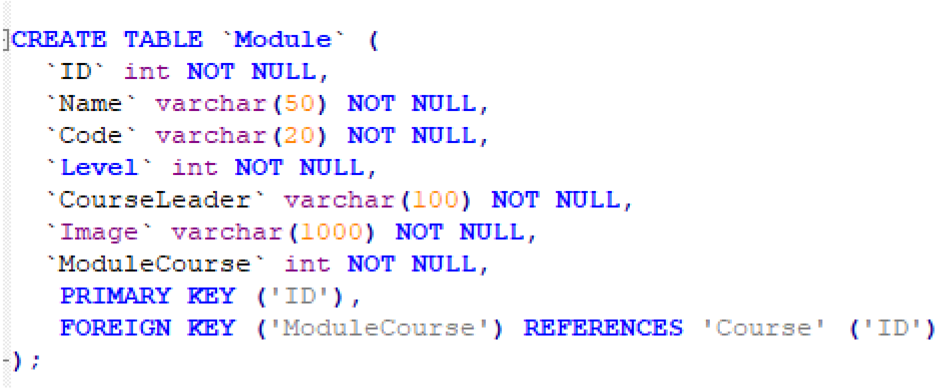


```
CREATE TABLE `Attendance` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Lecture` varchar(10) NOT NULL,
  `Workshop` varchar(10) NOT NULL,
  `ModuleID` int NOT NULL,
  `UserID` int NOT NULL,
  PRIMARY KEY ('ID'),
  FOREIGN KEY ('ModuleID') REFERENCES 'Module' ('ID'),
  FOREIGN KEY ('UserID') REFERENCES 'User' ('ID')
);
```

Figure 36 attendance SQL statement

5.5.3 Create Module table

The module table is used to store all the modules, and the module will have an association with the course table as a module can have one too many courses, and a course can have one too many modules. To solve many to many relations, use a join table to solve the many to many relations.

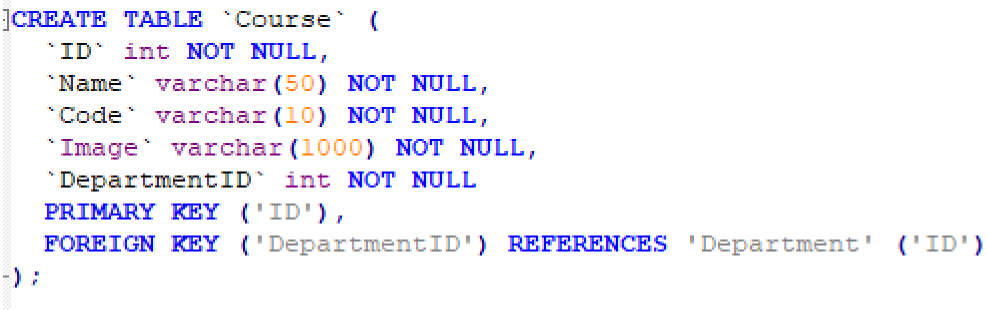


```
CREATE TABLE `Module` (
  `ID` int NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Code` varchar(20) NOT NULL,
  `Level` int NOT NULL,
  `CourseLeader` varchar(100) NOT NULL,
  `Image` varchar(1000) NOT NULL,
  `ModuleCourse` int NOT NULL,
   PRIMARY KEY ('ID'),
   FOREIGN KEY ('ModuleCourse') REFERENCES 'Course' ('ID')
);
```

Figure 37 module SQL statement

5.5.4 Create Course table

The course table is used to store all the courses, and the course will have an association with the department as the department has one too many courses, and the course has one department.



```
CREATE TABLE `Course` (
  `ID` int NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Code` varchar(10) NOT NULL,
  `Image` varchar(1000) NOT NULL,
  `DepartmentID` int NOT NULL
  PRIMARY KEY ('ID'),
  FOREIGN KEY ('DepartmentID') REFERENCES 'Department' ('ID')
);
```

Figure 38 course SQL statement

5.6 Conclusion

The implementation chapter outlines the process of the development of the system and what actions were taken to create a functional system for the user to use. Furthermore, some of the functions and plans were smoothly together to achieve the end goal and the tools and technology that were used helped to layout the system in an easy way to develop. The recourses were used to fix the implementation issues by solving and creating a dynamic system. Bootstrap allows for creating a smooth GUI for the users to interact with the application and the components that Bootstrap provides to create a better navbar, form, text, colour etc.

6 Validation

The validation segment will focus on different testing techniques used to test the functionality of the application to ensure that the application has met all the requirements. Testing is crucial in the software development lifecycle as it helps to ensure that the application functions. There are two test approaches, dynamic and static, which help to focus mainly on the backend API code and dynamic focus on the application's front-end. Furthermore, the testing will be carried out using the box testing method, which will be used to test the application. The purpose of the testing is to identify errors made in the development phase and ensure that all the aspects of the application function correctly.

6.1 Test strategy

In the test, the strategy will determine the testing methods that will be used and how it will be used to test the requirement. The types of testing will be a white box, and black box testing will be used to test the application to ensure that the user requirements have been met and implemented on the application.

6.1.1 White-box testing

The white box testing involves testing the internal structure of the application, for instance, testing the functions implemented in the code, instance, by inserting data to check if the data is successfully stored in the database.

6.1.2 Black box testing

The black box testing can be done by not having the knowledge of the internal structure of the application. For instance, the test can be done on the login to check if the error message displays when incorrect login credentials are entered.

6.2 Testing plan

The testing plan will use the user requirements to test all the features and functions of the application to work properly. Testing can be done on the application's front-end to ensure that all the option and input fields are working and allow the user to navigate around the application without any difficulties. The testing will be done on the application's backend aspects to ensure that functions and queries are working correctly by Get, Update, Delete and Post the data into the database when the request is made from the front-end.

ID	Functional requirements	Description	Expected Results	Pass/Fail
1	The users must be able to view their personal information	Users Click on the 'Home' page	The application should display the user details once the user login to the application	Pass
2	The users must be able to log in to the system using login credentials	User filles in the login form by entering the login credential	The application should check the username and password match the details in the database The system will take all the information of the user using the login details to display on the dashboard	Pass
3	The admin must be able to add users to the system	The admin is available to add users by entering their details in the form	The system must add the user details to the user table in the database	Pass
4	The admin must be able to delete users from the system	Admin select the user and clicks on the delete button	The system must delete the user from the user table on the database	Pass
5	The admin should be able to edit users on the system	The admin edit user by filling the form with updated details and click on the update button	The system must update the user details on user table	Fail
6	The admin must be able to assign courses and related modules by level	Admin assign modules when adding new users	The system takes the request and assign the module by user level	Pass
7	The student must be able to view the course and related modules	The student clicks on the course page and view module	The system should display course and display modules related to the course and the user	Pass

8	The course director must be able to view the modules attendance	The director clicks on the view button on the module to see the attendance	The system should display module attendance	Pass
9	The Lecturer must be able to view the students on the module	The Lecturer clicks on the modules to view student	The system should fetch all the students that study the module	Pass
10	The course director and Lecturer must be able to filter student attendance by low/high	Director clicks on the module and then click view to see the student attendance	The system should display the student and their module and filter the student attendance by low/high	Pass
11	The admin must be able to add a new course to the system	The admin is available to add course by entering their details in the form	The system must add the course details to the course table in the database	Pass
12	The admin must be able to add modules related to the new course	The admin is available to add module by entering their details in the form	The system must add the module details to the module table in the database	Pass
13	The admin must be able to delete course on the system	Admin select the course and clicks on the delete button	The system must delete the course from the course table on the database	Pass
14	The admin should be able to edit module on the system	The admin edit module by filling the form with updated details and click on the update button	The system must update the module details on module table	Fail
15	The admin must be able to delete module on the system	Admin select the module and clicks on the delete button	The system must delete the module from the module table on the database	Pass
16	The admin should be able to edit course on the system	The admin edit course by filling the form with updated details and click on the update button	The system must update the course details on course table	Fail
17	The student should be able to view overall attendance in a graph showing different module	The student click on the module to view attendance	The system must fetch the student attendance and display in the module page	Fail

ID	Non-Functional Requirement	Description	Expected Results	Pass/Fail
1	**Platform**	The system should be cross-platform	The user should be available to access the system on any device long as it has a Browser	Pass
		Log in to the system through Windows PC	The user should be able to access the system	Pass
		Log in to the system through Window Laptop	The user should be able to access the system	Pass
		Log in to the system through iOS	The user should be able to access the system	Pass
		Log in to the system through browser	The user should be able to access the system	Pass
2	**Maintainability**	The admin is available to maintain the system from not running.	The admin should have full control over the system and monitor the actions of users	Pass
3	**Design**	The navigation bar is the same for admin	The system should show all the pages related to admin	Pass
		The navigation bar is the same for the Course leader	The system should show all the pages related to the Course leader	Pass
		The navigation bar is the same for Lecturer	The system should show all the pages related to Lecturer	Pass
		The navigation bar is the same for student	The system should show all the pages related to student	Pass
4	**Performance**	The system runes properly for all users	The system should respond to the user's request	Pass
		The system is available to handle multiple SQL queries	The system should respond to the request in 3 seconds	Pass
5	**Security**	Checking password field	The password should be hashed using an algorithm	Fail
		Creating an account for non-register user	It should allow new users to be registered	Pass
		Accessing the system with correct login details	It should allow the user to access the system	Pass
		Closing the browser while logged in	It should log out the user	Fail
		Entering incorrect password	It should display an error message	Pass

6.3 Conclusion

The Testing chapter is one of the essential parts of the system as it helps to ensure that all the features and functions are working correctly before deploying the system. The testing phase tests the design and implementation to ensure that all the requirements have been met. The testing phase covers all the functional and non-functional requirements as they are the core of this system. However, many of the features and functionally have not been implemented as one person develops this system. There are no time and resources to help implement all the features and functions for the system to run successfully. Furthermore, this will help in the future to plan properly and ensure that all the features and functionality are doable if there was more time, more workforce and more recourses, the system could have been implemented to a higher standard.

7 Critical review and Conclusion

The critical segment will review the entire project and discuss how the project was completed, the problems that were encountered in building the project. This report will give a summary of the overall project journey.

7.1 Achievements

The achievement will explore what worked well in the process of building the application, and this project used the agile methodology to develop the application as it allows to change the requirements. For instance, if the client wants to change or add more features to the application while in the development phase, the agile methodology will change the user requirements and implement necessary changes. To improve the application further, a weekly meeting was scheduled with the client to discuss what changes could be made. However, the outcome of the application has proven that it gives the user what they want to do.

The functional requirements identified with the client's help have helped move forward with the design, where various tools were used to create low to high fealty wireframes to give an insight into the application before implementation. Once all the research was conducted, it was used to implement an application that is functional for the user type to achieve their end goals. The application is responsive and gives the users ease of use to navigate the application smoothly. Overall, the application is functional and has met all the user requirements, which will allow the users to achieve their end goals.

7.2 Problem encountered

While implementing the system, multiple issues kept the project from progressing. One of the issues that kept the project from going forward was a third-party chart library which was causing issues when trying to generate a graph based on the student attendance records from the database. Every time a graph was created using the student's attendance records to show the student's attendance throughout the time, the web page was displaying an empty white page without displaying any error, but after an internet search to find a solution to the problem, it was concluded that the reason why the page was showing empty white page instead of the graph was due to the fact that the library was no longer being updated, so it didn't support the React version that was being used to create the system.

Many problems were encountered while implementing some of the features that were the core of the application to function successfully. Ensuring that the user requirements have been met, various methods were used to implement some of the features to give a better user experience when they are using the application. For instance, one of the features that were difficult to implement was the chat system for the Lecturer to send messages to students with lower attendance, but in the end, the chat system was not implemented as it required a heavy amount of time and resources to implement the chat. However, even if the Lecturer can't message the student, the Lecturer is still available to identify students with lower attendance on the application.

7.3 Future work

In the future, the application can be improved to a high standard to provide better functions and features by researching and gathering more resources that can be used to implement better functions and features the current application has. To give the user an even better experience of the application, making the application multi-platform so that the users can use the application on any device. Hence, making the application multi-platform could help provide more features, such as a QR scanner that will allow the user to scan a barcode, which will take the user's attendance and record the user date and time to ensure that the user is aware is available to view their attendance record.

The design of the application could be improved better to allow the user to change the themes of the application to give a better experience for users with impaired eyesight so that they could use the function and features as other users.

7.4 Conclusion

This project achieved its aim and objective to allow lecturer and course director to monitor student attendance to identify engaged and non-engaged students. Overall, this project has given me many knowledge and skills that will be useful for my future career, and new skills were developed in the project's progress. For this project, many of the requirements were implemented successfully, and in the testing chapter, all the must-have requirements were passed, and the project was successfully completed.

8 References

Auderer, M. (2018) *How To Get Started with Node.js and Express | DigitalOcean*. Available at: https://www.digitalocean.com/community/tutorials/nodejs-express-basics (Accessed: 12 April 2022).

Castagna, R. (2021) *What is GDPR? An Overview of GDPR Compliance and Conditions*, *WhatIs.com*. Available at: https://www.techtarget.com/whatis/definition/General-Data-Protection-Regulation-GDPR (Accessed: 10 April 2022).

IBM (2021) *Database Security: An Essential Guide*. Available at: https://www.ibm.com/uk-en/cloud/learn/database-security (Accessed: 10 April 2022).

Information Commissioner's Office (2021) *Penalties*. ICO. Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-le-processing/penalties/ (Accessed: 10 April 2022).

Information Commissioner's Office (2022) *72 hours - how to respond to a personal data breach*. ICO. Available at: https://ico.org.uk/for-organisations/sme-web-hub/72-hours-how-to-respond-to-a-personal-data-breach/ (Accessed: 10 April 2022).

Kollegger, E. (2018) 'What is Axios.js and why should I care?', *Medium*, 14 May. Available at: https://medium.com/@MinimalGhost/what-is-axios-js-and-why-should-i-care-7eb72b111dc0 (Accessed: 12 April 2022).

legislation.gov.uk (no date a) *Computer Misuse Act 1990*. Statute Law Database. Available at: https://www.legislation.gov.uk/ukpga/1990/18 (Accessed: 10 April 2022).

legislation.gov.uk (no date b) *Data Protection Act 2018*. Queen's Printer of Acts of Parliament. Available at: https://www.legislation.gov.uk/ukpga/2018/12/part/3/chapter/2/enacted (Accessed: 10 April 2022).

McCallion, J. (2022) *What is the Computer Misuse Act?*, *IT PRO*. Available at: https://www.itpro.co.uk/it-legislation/28174/what-is-the-computer-misuse-act (Accessed: 10 April 2022).

Smith, T. (2013) 'Sony fined £250,000 after breaching Data Protection Act', *Tech Monitor*, 24 January. Available at: https://techmonitor.ai/technology/data-centre/sony-fined-250000-after-breaching-data-protection-act-240113 (Accessed: 10 April 2022).

9 Appendix: Use cases

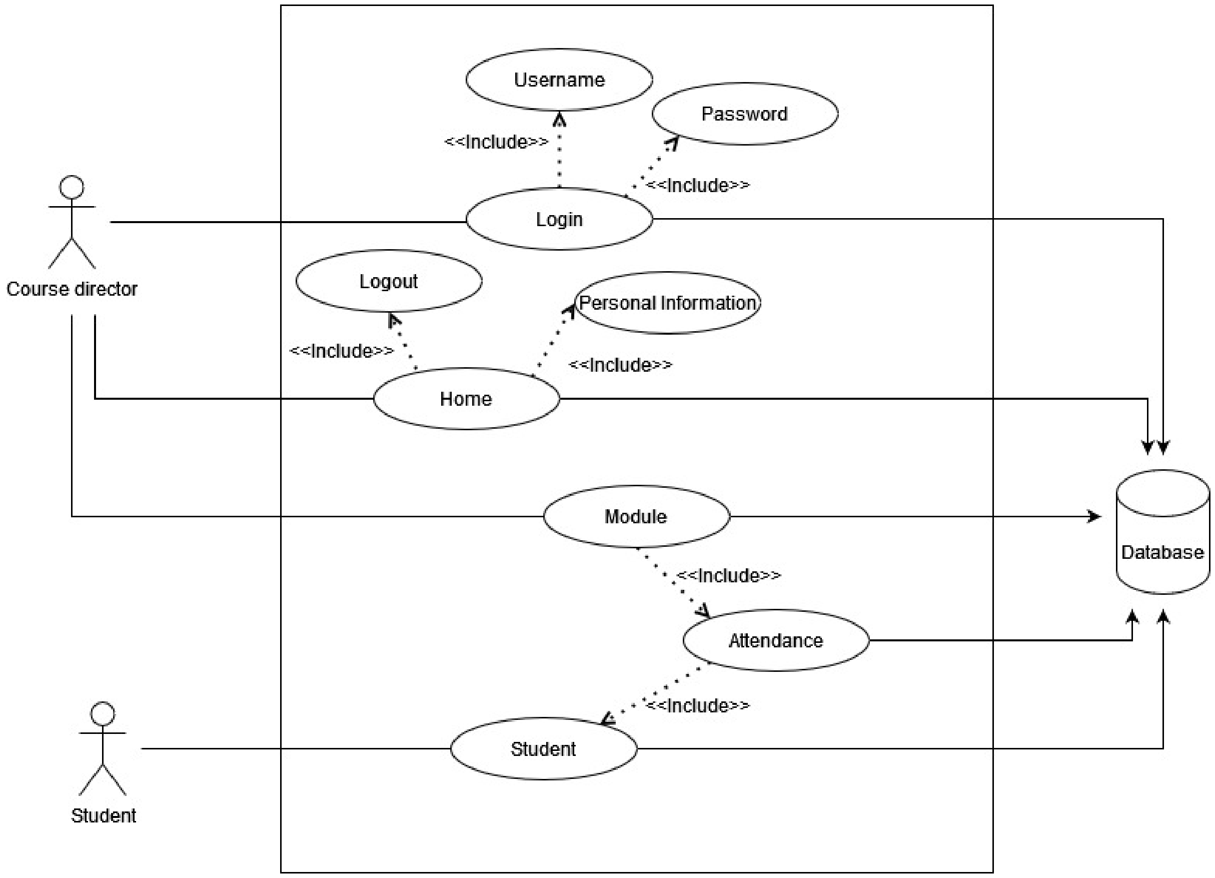


Figure 4 Use case for Course director

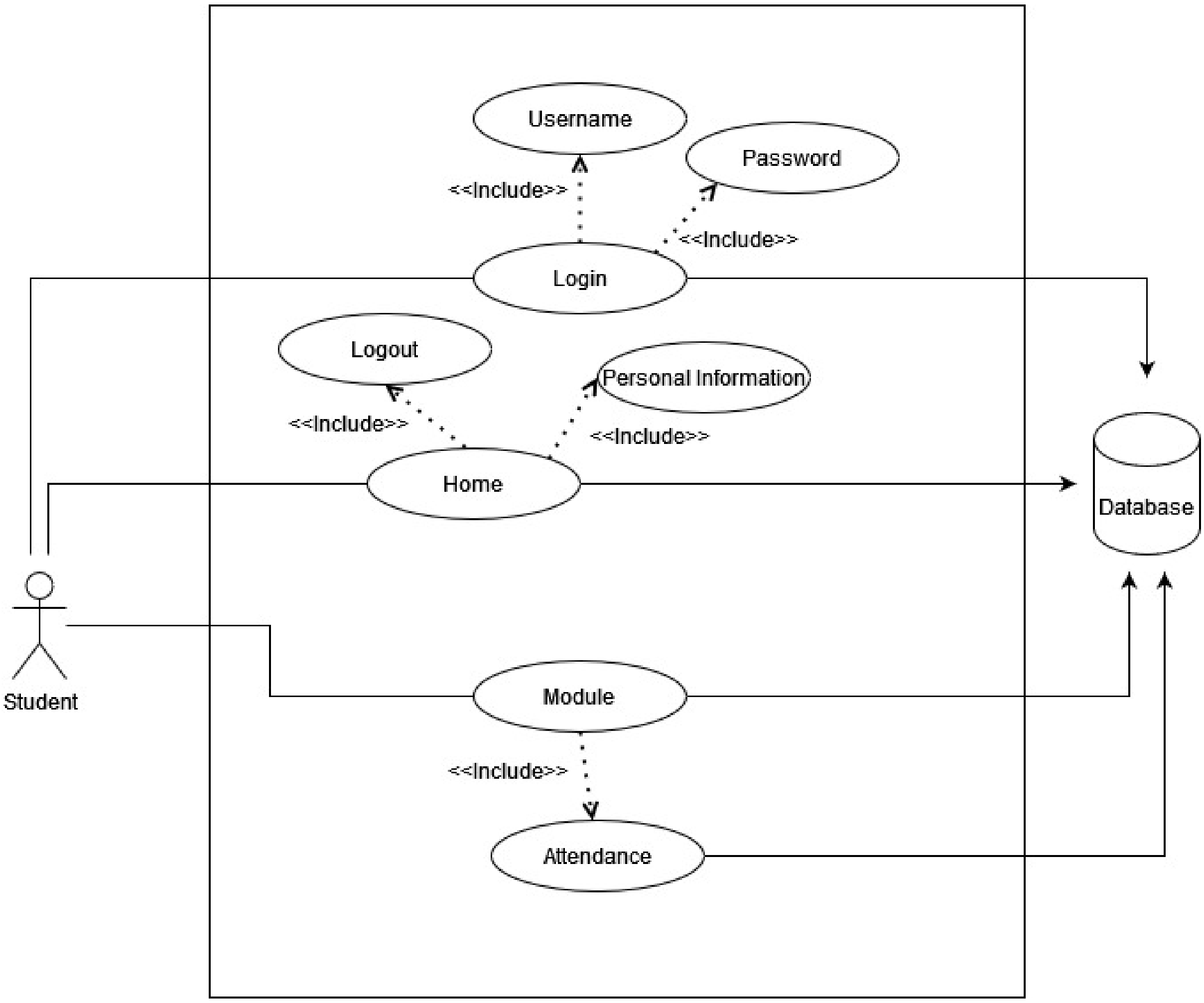


Figure 5 Use case for Course director

10 Appendix: Use case text

Use case 1	Login	
Actor	Admin	
Goal	Able to login to the system	
Condition	Login	
Main Flow	**Step**	**Action**
	1	Admin must be able to Login
	2	Admin must enter the username and password
	3	Admin press login button
	4	Admin must be able to view the content once logged in
Alternative Flow	**Action**	
MoSCoW	Must	

Table 2 Admin use case text Login

Use case 2	Personal Information	
Actor	Admin	
Goal	Able to view Personal information	
Condition	View Personal information	
Main Flow	**Step**	**Action**
	1	Once logged in

	2	Admin must be able to view the content
	3	Personal information displayed
	4	Admin must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 2 Admin use case text personal information

Use case 3	Create Student	
Actor	Admin	
Goal	Able to add new students	
Condition	Students should be enrolled on a Course with Modules	
Main Flow	**Step**	**Action**
	1	Admin clicks on the user
	2	Admin fill form with student details
	3	Admin enter student name
	4	Admin assign course, level to the student
	5	Admin submit a request to add to the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 3 Admin use case text create student

Use case 4	Create Lecturer	
Actor	Admin	
Goal	Able to add new Lecturer	
Condition	Lecturers should be enrolled on a Course with Modules that they are teaching, and the students who will be doing the module	
Main Flow	**Step**	**Action**
	1	Admin clicks on the user
	2	Admin fill form with lecturer details
	3	Admin enter lecturer name
	4	Admin assign courses and module they teach
	5	Admin submit a request to add to the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 4 Admin use case text create Lecturer

Use case 5	Create Course director	
Actor	Admin	
Goal	Able to add new Course director	
Condition	Adding new Course Leader	
Main Flow	**Step**	**Action**
	1	Admin clicks on the user
	2	Admin fill form with course director details
	3	Admin enter course director name
	4	Admin assign courses and module they manage
	5	Admin submit a request to add to the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 5 Admin use case text create course director

User case 6	Create Course	
Actor	Admin	
Goal	Able to add new Course	
Condition	Adding new Course	
Main Flow	**Step**	**Action**
	1	Admin must enter the details of the Course
	2	Admin must enter the Course code
	3	Admin must add related modules to the Course
	4	Admin must assign which department is doing the Course
	5	Admin submit a request to add to the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 6 Admin use case text create course

User case 7	Create Modules	
Actor	Admin	
Goal	Able to add new Module	
Condition	Adding new Module	
Main Flow	**Step**	**Action**
	1	Admin must enter the details of the Module
	2	Admin must enter the Module code
	3	Admin must add the module to the related Course
	4	Admin submit a request to add to the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 7 Admin use case text create module

Use case 1	Login	
Actor	Student	
Goal	Able to Login	
Condition	Login	
Main Flow	**Step**	**Action**
	1	Student must be able to Login
	2	Student must enter the username and password
	3	Student press the login button
	4	Student must be able to view the content once logged in
Alternative Flow	**Action**	
MoSCoW	Must	

Table 8 Student use case text login

Use case 2	Personal Information	
Actor	Student	
Goal	Able to view Personal information	
Condition	View Personal information	
Main Flow	**Step**	**Action**
	1	Once logged in
	2	Admin must be able to view the content
	3	Personal information displayed
	4	Student must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 9 Student use case text personal information

Use case 3	Course	
Actor	Student	
Goal	Able to view Course	
Condition	View Course	
Main Flow	**Step**	**Action**
	1	Students must be able to view their Course detail
	2	Students must be able to view the content
	3	Students should be able to see the Course code
	4	Students should be able to see their Modules under Course
	5	Students must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 10 Student use case text course

Use case 4	Module	
Actor	Student	
Goal	Able to view Module	
Condition	View Module	
Main Flow	**Step**	**Action**
	1	Students must be able to view their Module detail
	2	Students must be able to view the content
	3	Students should be able to see the Module code
	4	Students should be able to see their Modules and Course
	5	Students must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 11 Student use case text Module

Use case 5	Attendance	
Actor	Student	
Goal	Able to view Attendance	
Condition	View Attendance	
Main Flow	**Step**	**Action**
	1	Students must be able to view their Attendance detail
	2	Students must be able to view the content
	3	Students should be able to see Attendance code
	5	Students must be able to see induvial module attendance
	5	Students should be able to see their overall Attendance
	6	Students must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Should have	

Table 12 Student use case text Attendance

Use case 1	Login	
Actor	Lecturer	
Goal	Able to Login	
Condition	Login	
Main Flow	**Step**	**Action**
	1	Lecturer must be able to Login

	2	Lecturer must enter the username and password
	3	Lecturer press the login button
	4	Lecturer must be able to view the content once logged in
Alternative Flow	**Action**	
MoSCoW	Must	

Table 13 Lecturer use case text Login

Use case 2	Personal Information	
Actor	Lecturer	
Goal	Able to view Personal information	
Condition	View Personal information	
Main Flow	**Step**	**Action**
	1	Once logged in
	2	Admin must be able to view the content
	3	Personal information displayed
	4	Lecturer must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 12 Lecturer use case text personal information

Use case 3	Course	
Actor	Lecturer	
Goal	Able to view Course	
Condition	View Course	
Main Flow	**Step**	**Action**
	1	Lecturers must be able to view their Course detail
	2	The Lecturer must be able to view the content
	3	The Lecturer should be able to see the Course code
	4	Lecturers should be able to see the course they are teaching in
	5	The Lecturer must be able to see the Module under Course
	6	The Lecturer must be able to see the students that are doing the Course
	7	The Lecturer must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 13 Lecturer use case text course

Use case 4	Module	
Actor	Lecturer	
Goal	Able to view Module	
Condition	View Module	
Main Flow	**Step**	**Action**
	1	Lecturers must be able to view their Module detail
	2	The Lecturer must be able to view the content
	3	The Lecturer should be able to see the Module code
	4	The Lecturer should be able to see the Modules and the course is under
	5	The Lecturer must be able to see the students that are doing the Module
	6	The Lecturer must be able to see the student attendance on the Module
	7	Lecturer must send a request for data from the database

Alternative Flow	**Action**
MoSCoW	Must

Table 14 Lecturer use case text module

Use case 5	Attendance	
Actor	Lecturer	
Goal	Able to view Attendance	
Condition	View Attendance	
Main Flow	**Step**	**Action**
	1	The Lecturer must be able to view Attendance detail
	2	The Lecturer must be able to view the content
	3	The Lecturer should be able to see Attendance code
	4	The Lecturer must be able to see induvial module attendance
	5	The Lecturer must be able to view induvial student attendance
	6	The Lecturer should be able to see the overall Attendance
	7	The Lecturer must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 15 Lecturer use case text attendance

Use case 1	Login	
Actor	Course Director	
Goal	Able to Login	
Condition	Login	
Main Flow	**Step**	**Action**
	1	Course Director must be able to Login
	2	Course Director must enter the username and password
	3	Course Director press login button
	4	Course Director must be able to view the content once logged in
Alternative Flow	**Action**	
MoSCoW	Must	

Table 16 Lecturer use case text login

Use case 2	Personal Information	
Actor	Course Director	
Goal	Able to view Personal information	
Condition	View Personal information	
Main Flow	**Step**	**Action**
	1	Once logged in
	2	Admin must be able to view the content
	3	Personal information displayed
	4	Course Director must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 17 Lecturer use case text personal information

Use case 3	Course	
Actor	Course Director	
Goal	Able to view Course	
Condition	View Course	
Main Flow	**Step**	**Action**

	1	Course Director must be able to view their Course detail
	2	Course Director must be able to view the content
	3	Course Director should be able to see the Course code
	4	Course Director should be able to see the course they are managing
	5	Course Director must be able to see the Module under Course
	6	Course Director must be able to see the students that are doing the Course
	7	Course Director must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 18 Lecturer use case text course

Use case 4	Module	
Actor	Course Director	
Goal	Able to view Module	
Condition	View Module	
Main Flow	**Step**	**Action**
	1	Course Director must be able to view their Module detail
	2	Course Director must be able to view the content
	3	Course Director should be able to see the Module code
	4	Course Director should be able to see the Modules under Course
	5	Course Director must be able to see the students that are doing the Course and the Module they chose to do
	6	Course Director must be able to see the student attendance on the Module
	7	Course Director must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 19 Lecturer use case text module

Use case 5	Attendance	
Actor	Course Director	
Goal	Able to view Attendance	
Condition	View Attendance	
Main Flow	**Step**	**Action**
	1	Course Director must be able to view Attendance detail
	2	Course Director must be able to view the content
	3	Course Director should be able to see Attendance code
	4	Course Director must be able to see induvial module attendance
	5	Course Director must be able to view induvial student attendance
	6	Course Director should be able to see the overall Attendance
	7	Course Director must send a request for data from the database
Alternative Flow	**Action**	
MoSCoW	Must	

Table 20 Lecturer use case text attendance

11 Appendix: Data Dictionary

Relation	Attribute	Data Type	Size	Key PK/FK	Not Null	Constraints	Referential Relation
User	ID	INT		PK		AUTO-INCREMENT	
	Name	VARCHAR	50				
	Username	VARCHAR	50				
	Password	VARCHAR	50				
	Role	VARCHAR	50				
	KNumber	VARCHAR	50				
	Emailaddress	VARCHAR	50				
	Level	INT					
	CourseID	INT		FK			Course
	DepartmentID	INT		FK			Department

Table 24 User Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Student	ID	INT		PK		AUTO-INCREMENT	
	UserID	INT		FK			User
	AttendanceID	INT		FK			Attendance

Table 25 Student Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Lecturer	ID	INT		PK		AUTO-INCREMENT	
	UserID	INT		FK			User

Table 26 Lecturer Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Department	ID	INT		PK		AUTO-INCREMENT	
	Name	VARCHAR	50				
	Image	VARCHAR	1000				

Table 27 Department Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Course	ID	INT		PK		AUTO-INCREMENT	
	Name	VARCHAR	50				
	Code	VARCHAR	10				
	Image	VARCHAR	1000				
	DepartmentID	INT		FK			Department

Table 28 Course Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Module	ID	INT		PK		AUTO-INCREMENT	
	Name	VARCHAR	50				
	Code	VARCHAR	10				
	Level	INT					
	CourseLeader	VARCHAR	50				
	Image	VARCHAR	1000				

Table 29 Module Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
Attendance	ID	INT		PK		AUTO-INCREMENT	
	Lecture	INT	10				
	Workshop	INT	10				

Table 30 Attendance Data Dictionary

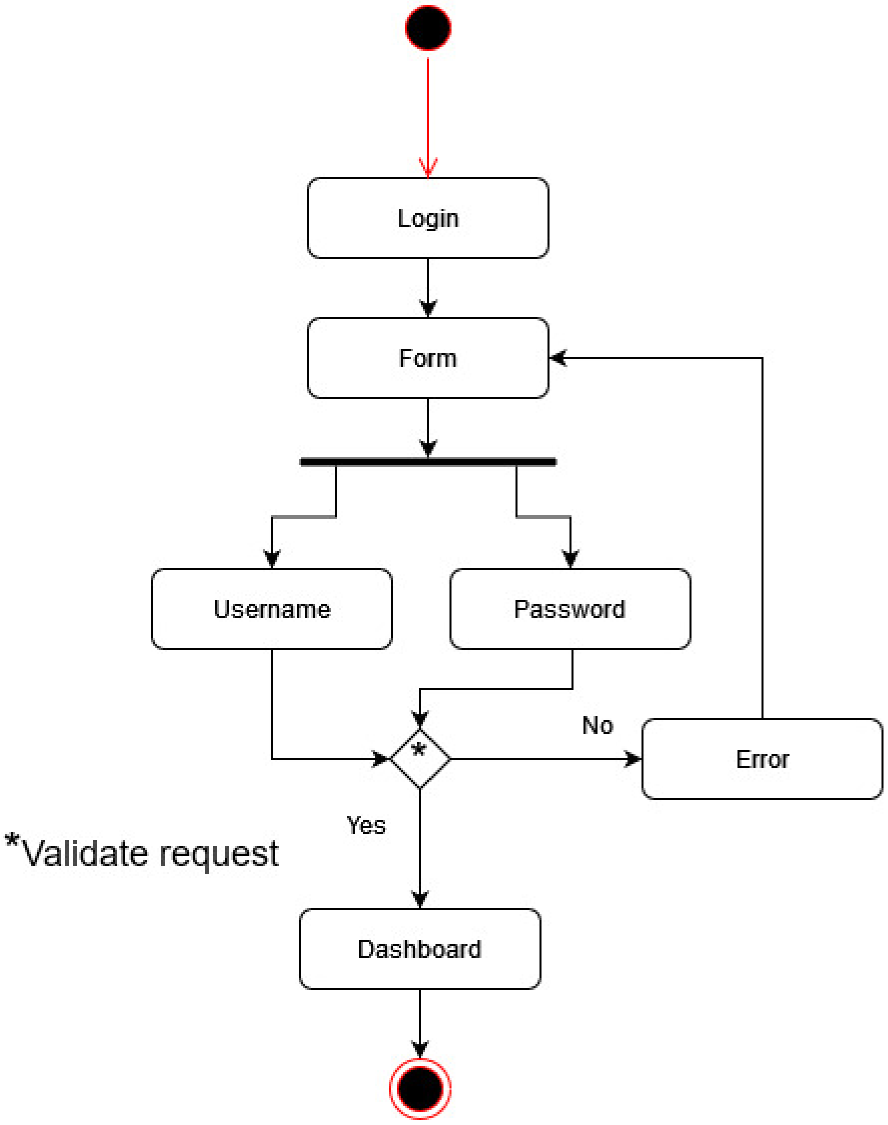
Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
CouseModule	ID	INT		PK		AUTO-INCREMENT	
	ModuleID	INT		FK			Module
	CourseID	INT		FK			Course
	AttendanceID	INT		FK			Attendance

Table 31 CourseModule Data Dictionary

Relation	Attribute	Data Type	Size	PK/FK	Not Null	Constraints	Referential Relation
ModuleLecturer	ID	INT		PK		AUTO-INCREMENT	
	ModuleID	INT		FK			Module
	LecturerID	INT		FK			Lecturer

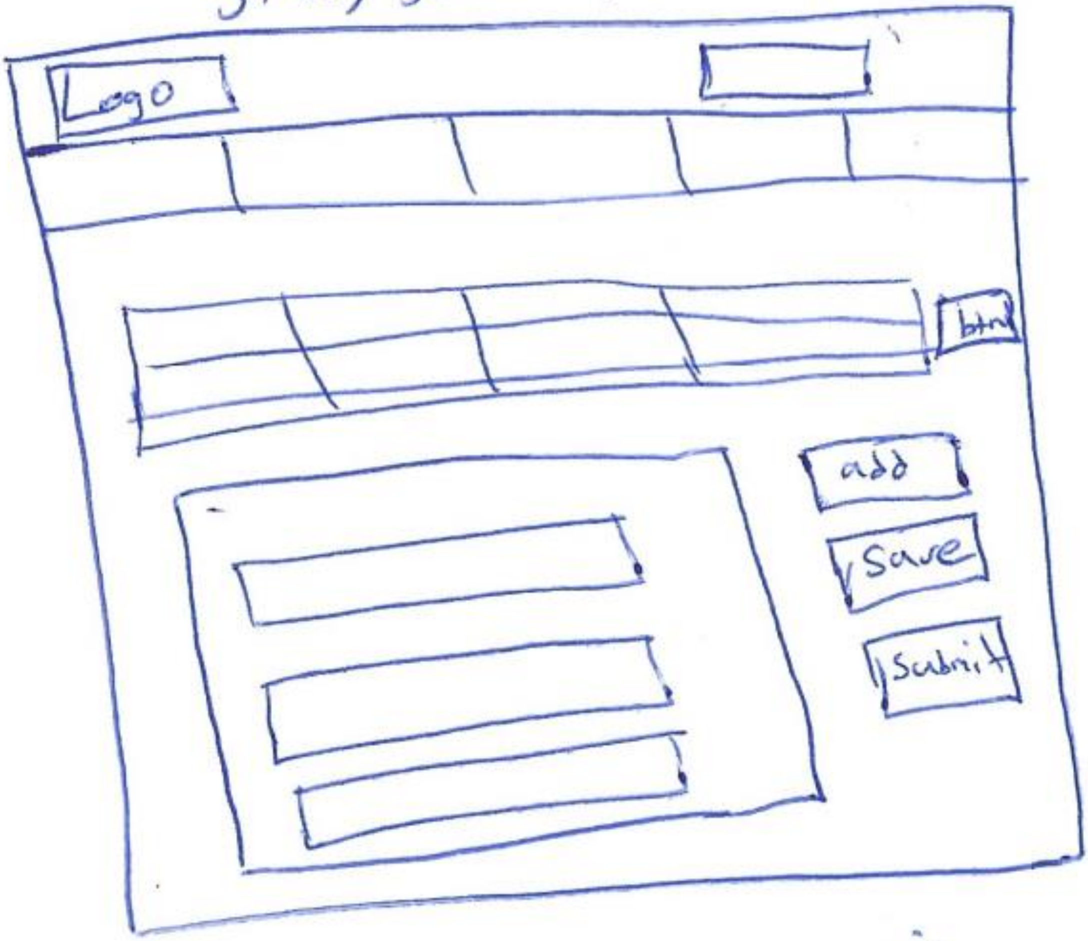
Table 32 ModuleLeader Data Dictionary

12 Appendix: Activity diagram

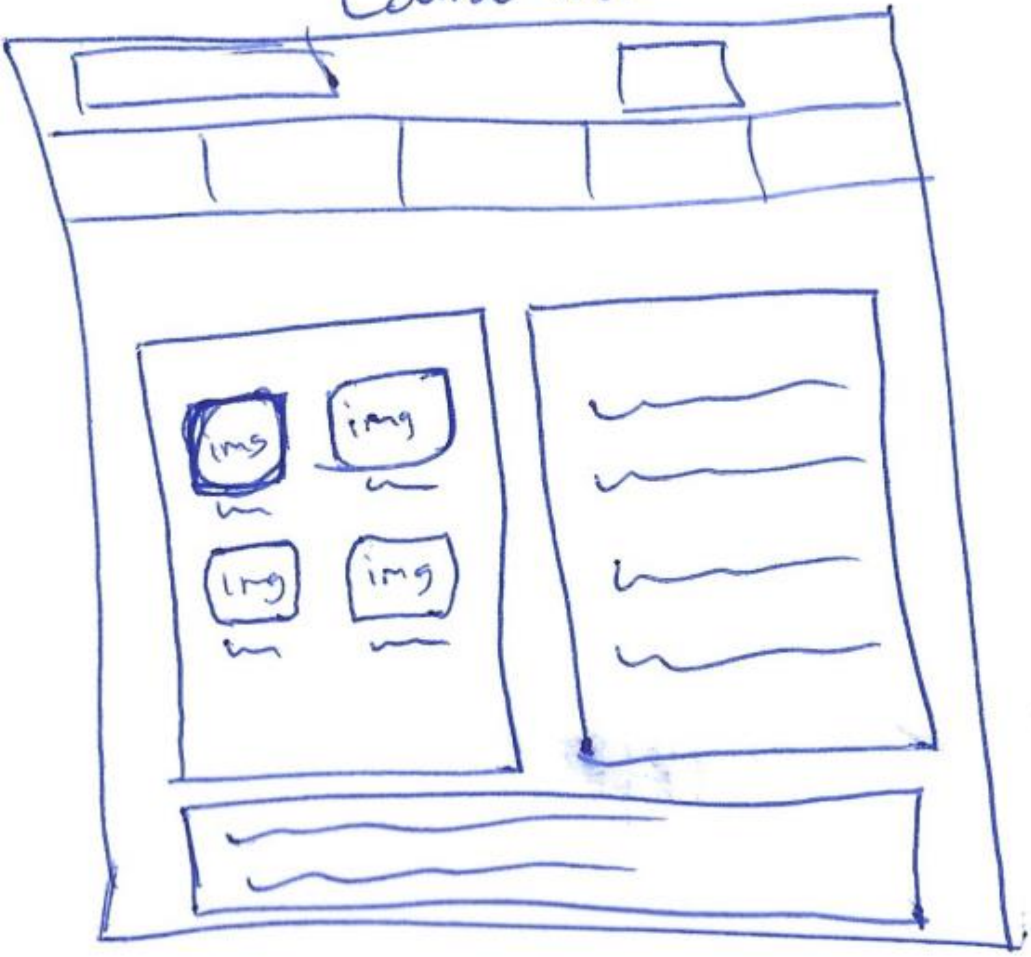


13 Appendix: Design

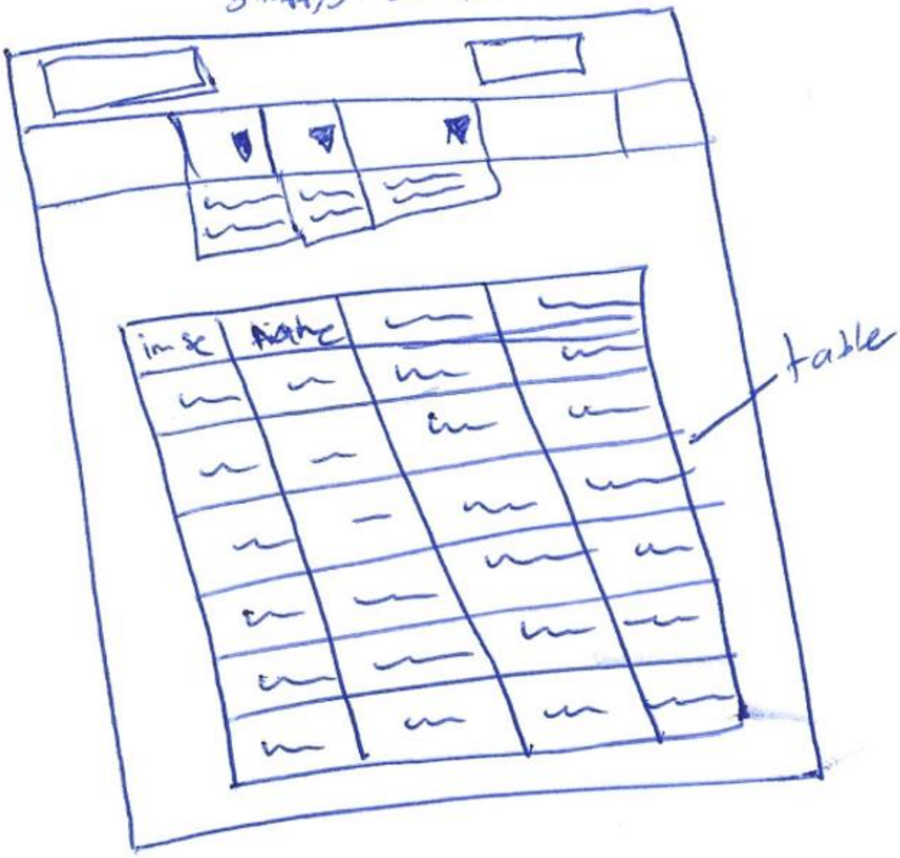
Staff, student, course, module



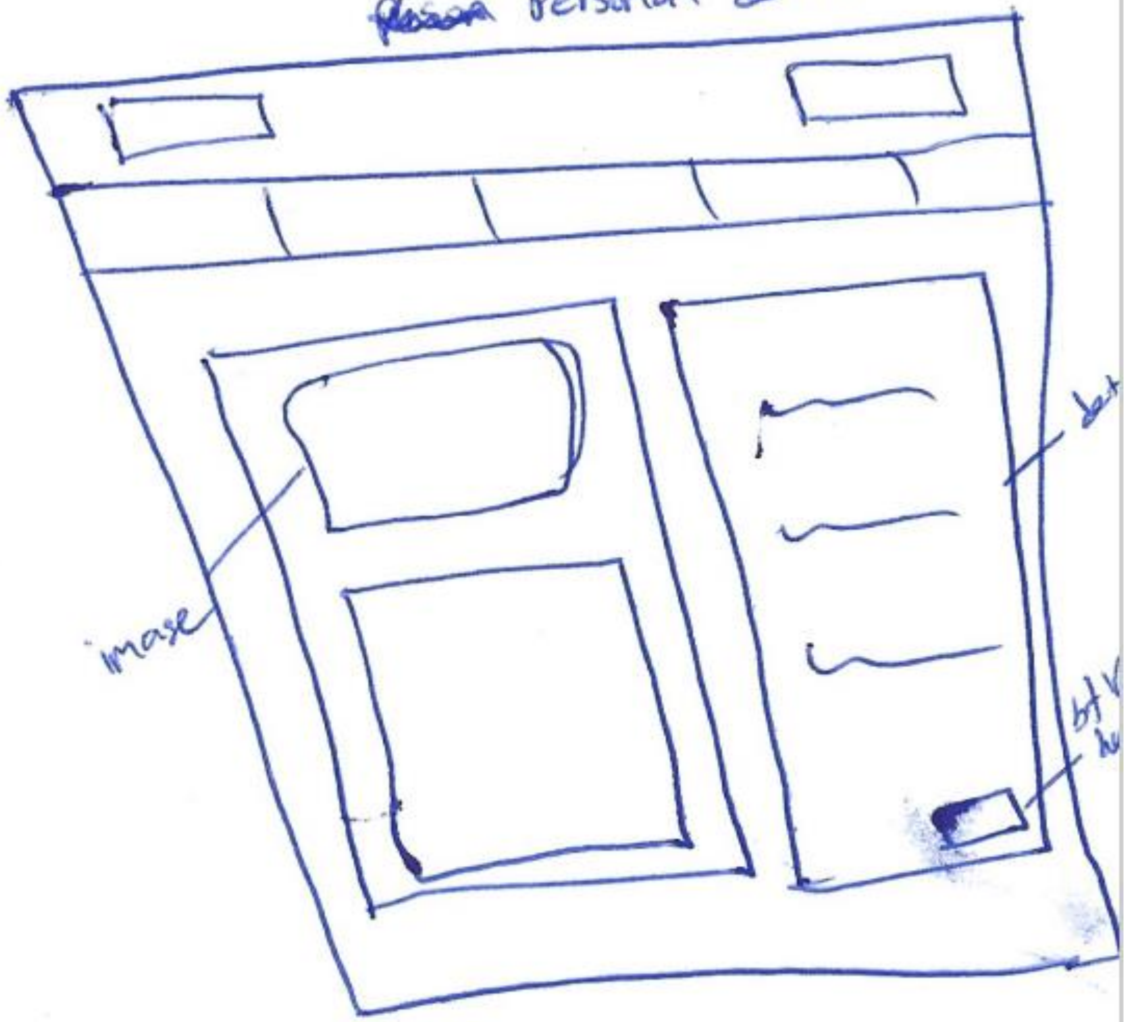
Course info

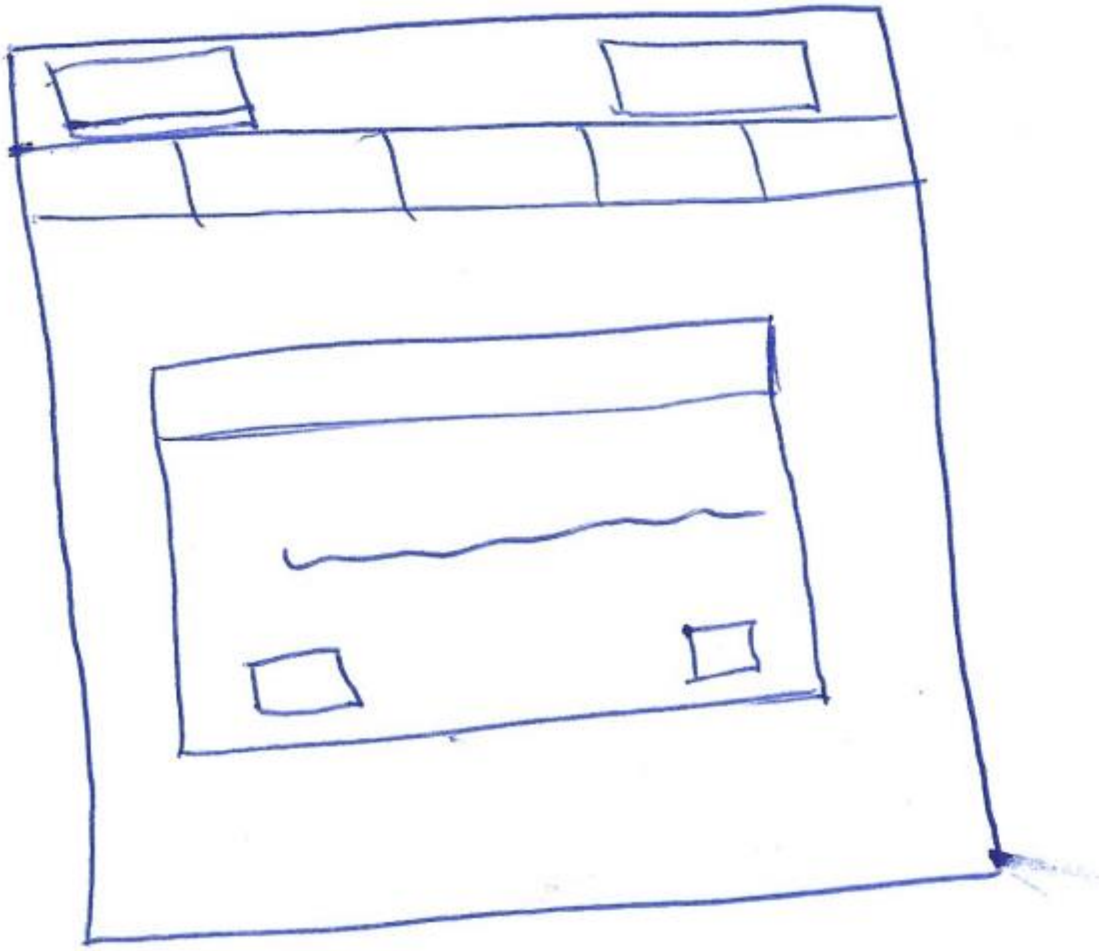


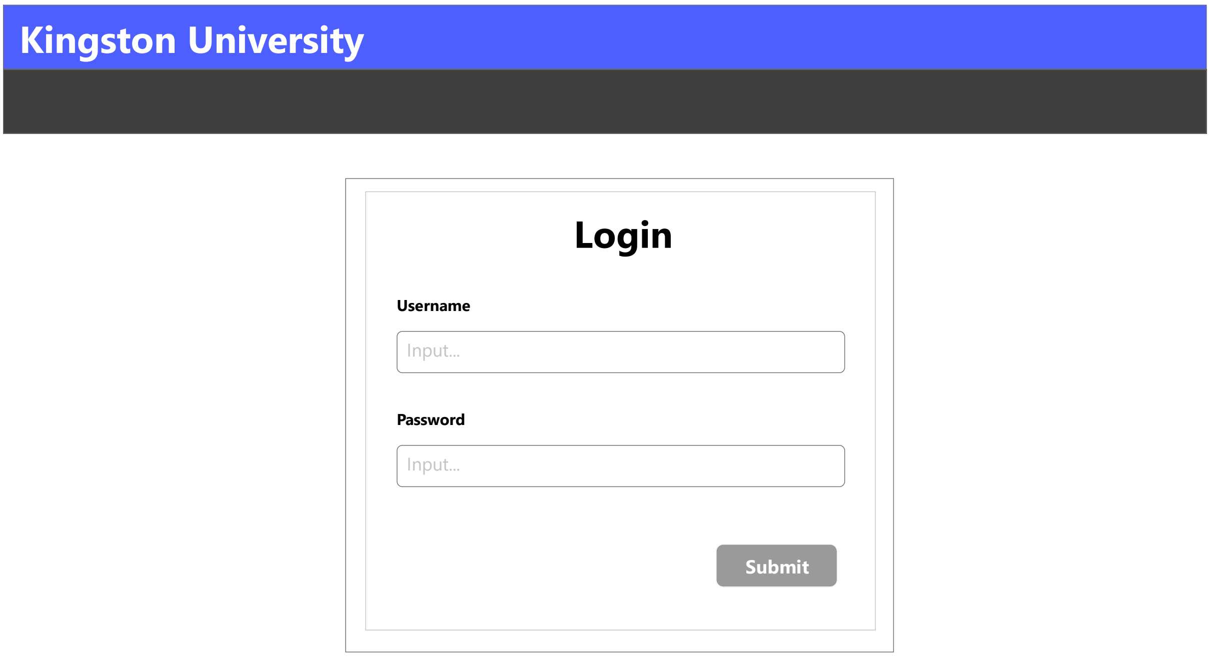
Staff, Student, ~~Course~~ Course /module

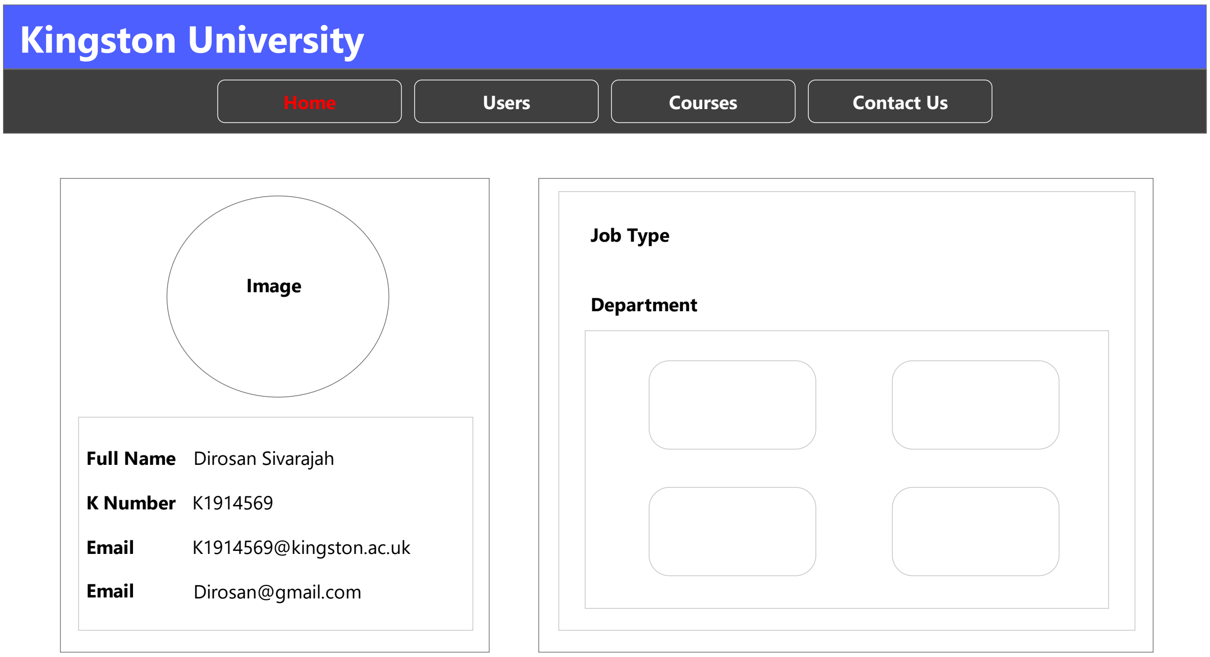


Personal detail



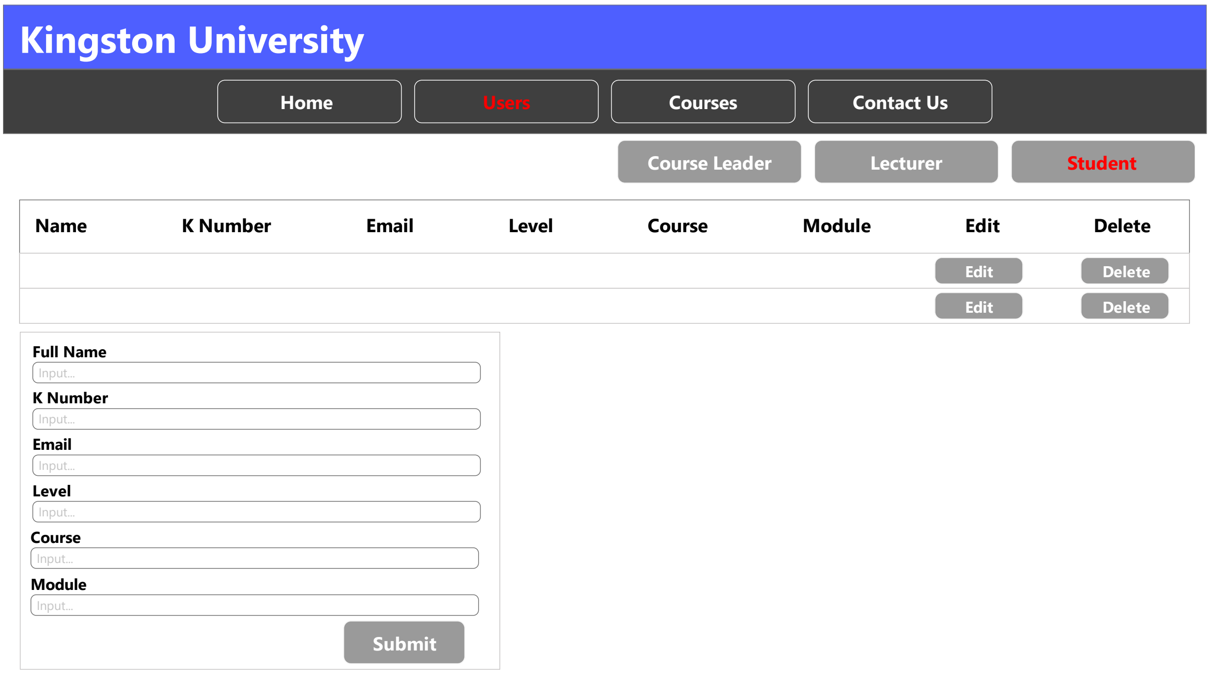



Kingston University
Login
Username
Input...
Password
Input...
Submit



Kingston University

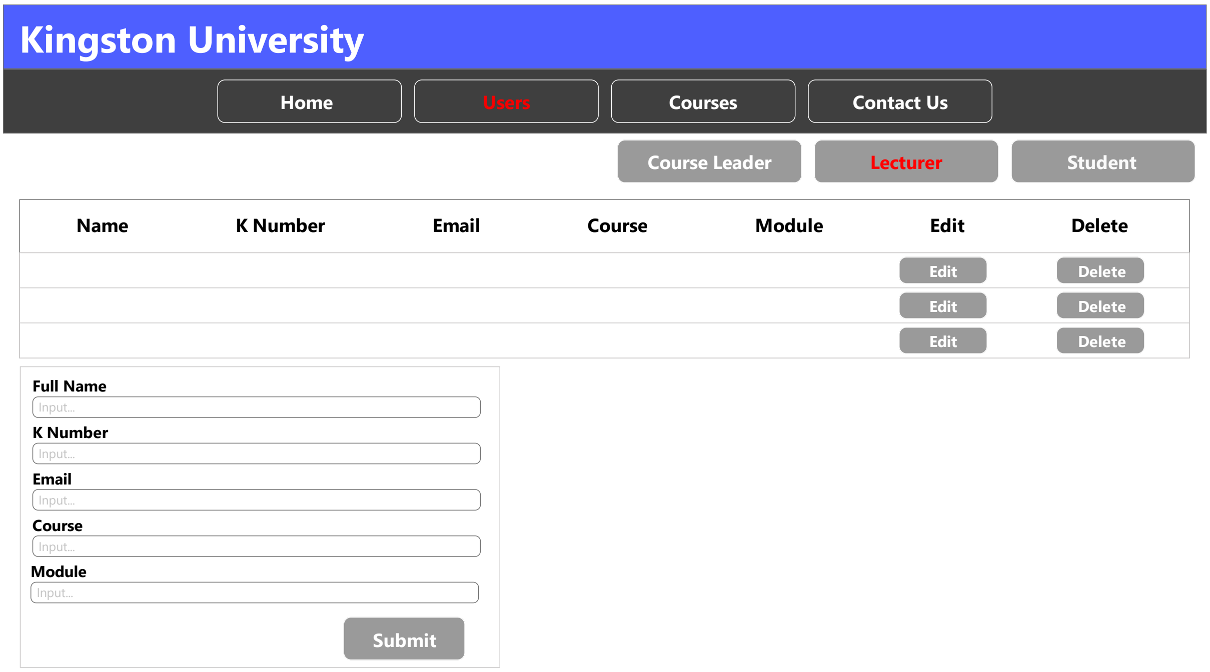
Full Name	Dirosan Sivarajah
K Number	K1914569
Email	K1914569@kingston.ac.uk
Email	Dirosan@gmail.com



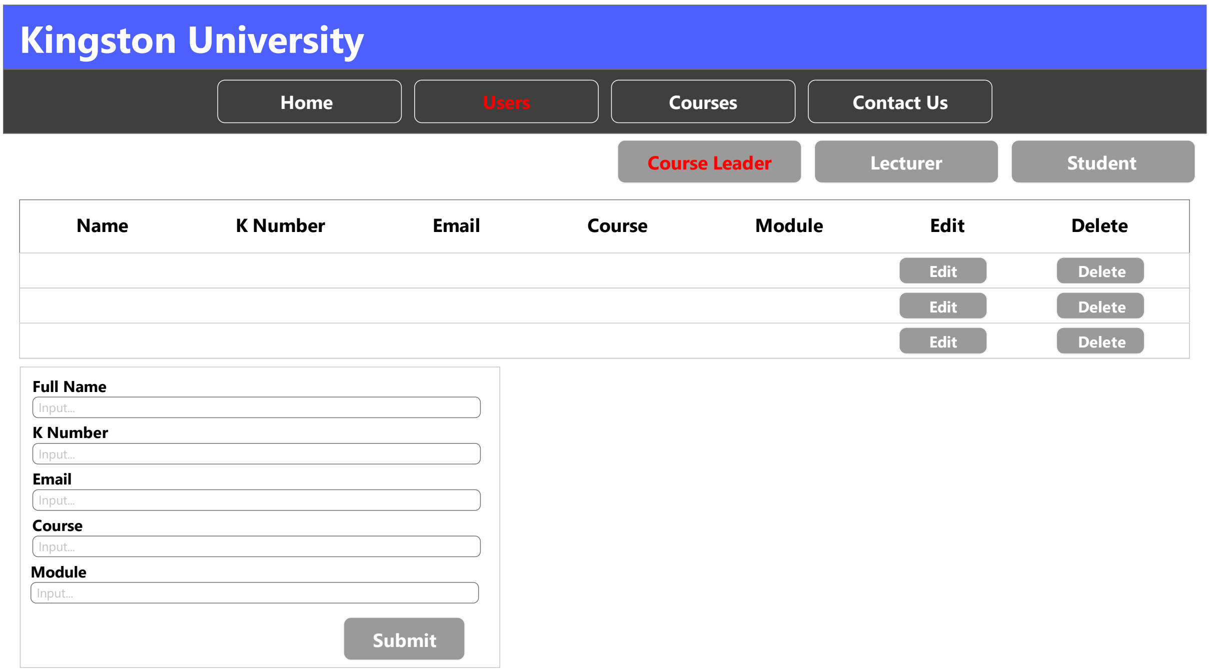
Kingston University

Name	K Number	Email	Level	Course	Module	Edit	Delete
						Edit	Delete
						Edit	Delete

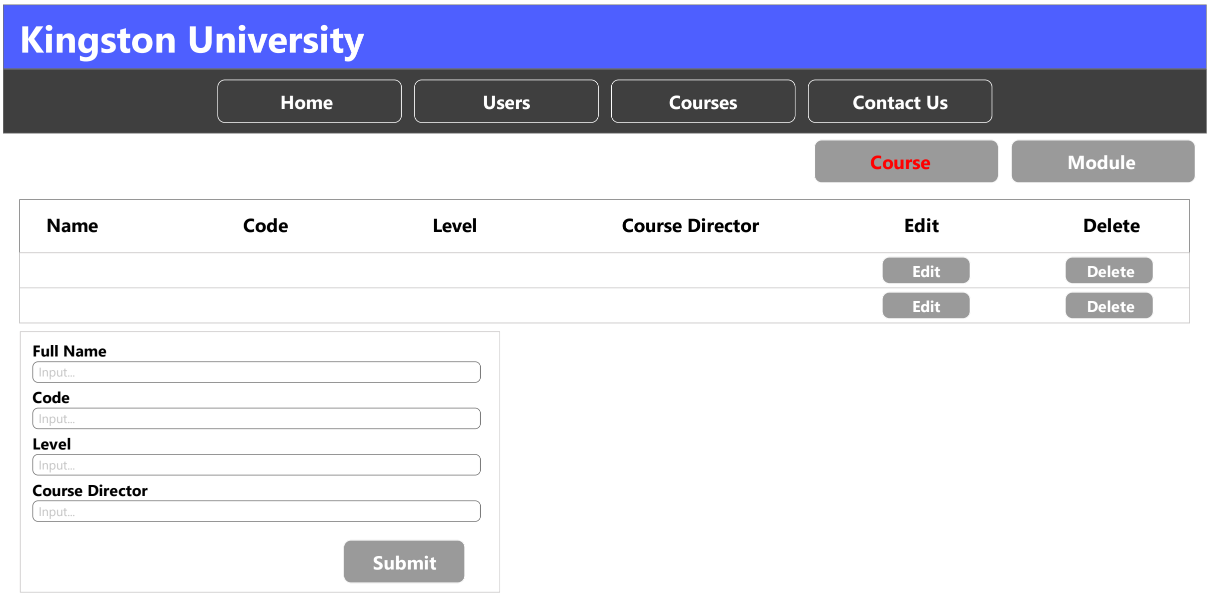
Full Name
Input...
K Number
Input...
Email
Input...
Level
Input...
Course
Input...
Module
Input...



Name	K Number	Email	Course	Module	Edit	Delete
					Edit	Delete
					Edit	Delete
					Edit	Delete

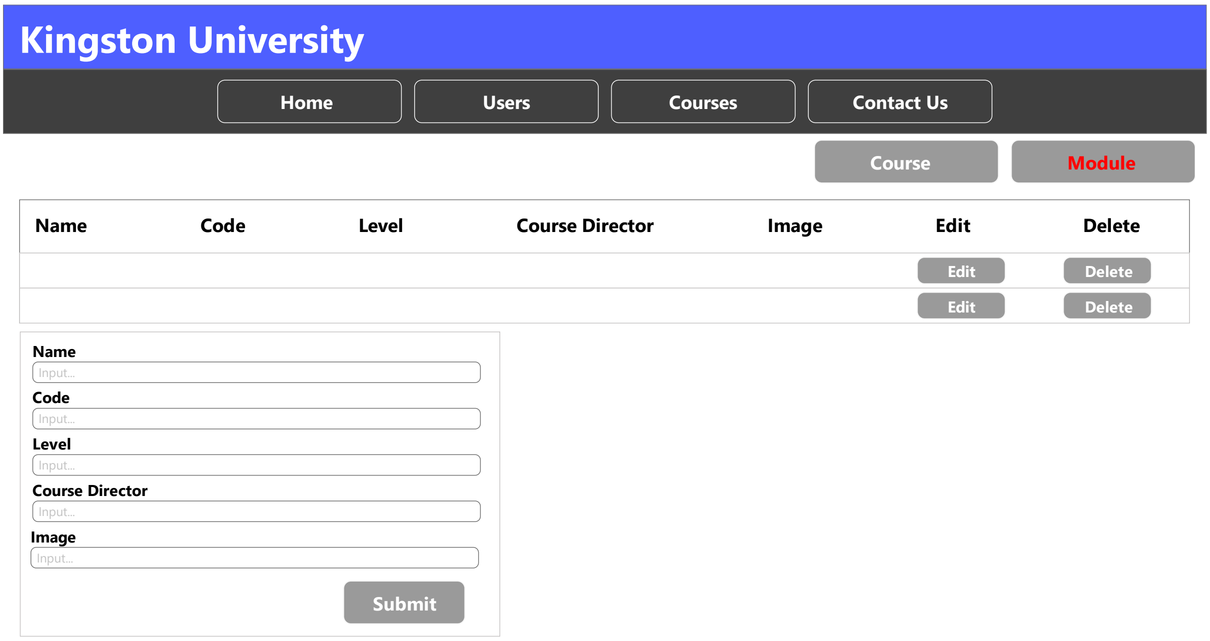


Name	K Number	Email	Course	Module	Edit	Delete
					Edit	Delete
					Edit	Delete
					Edit	Delete



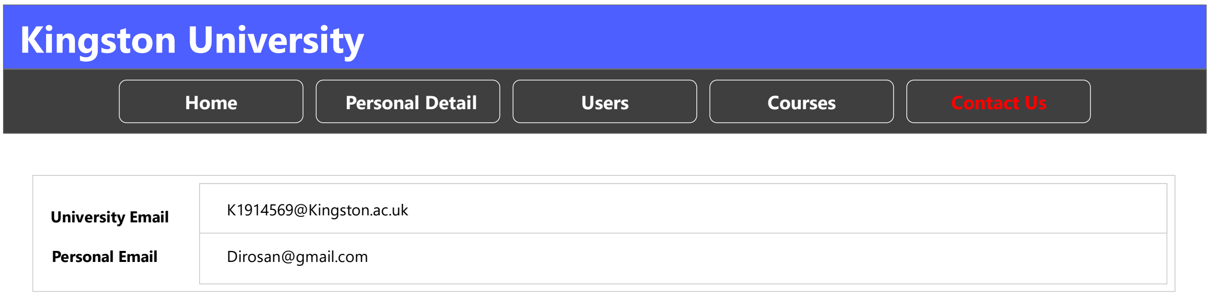
Name	Code	Level	Course Director	Edit	Delete
				Edit	Delete
				Edit	Delete

Full Name
Input...
Code
Input...
Level
Input...
Course Director
Input...



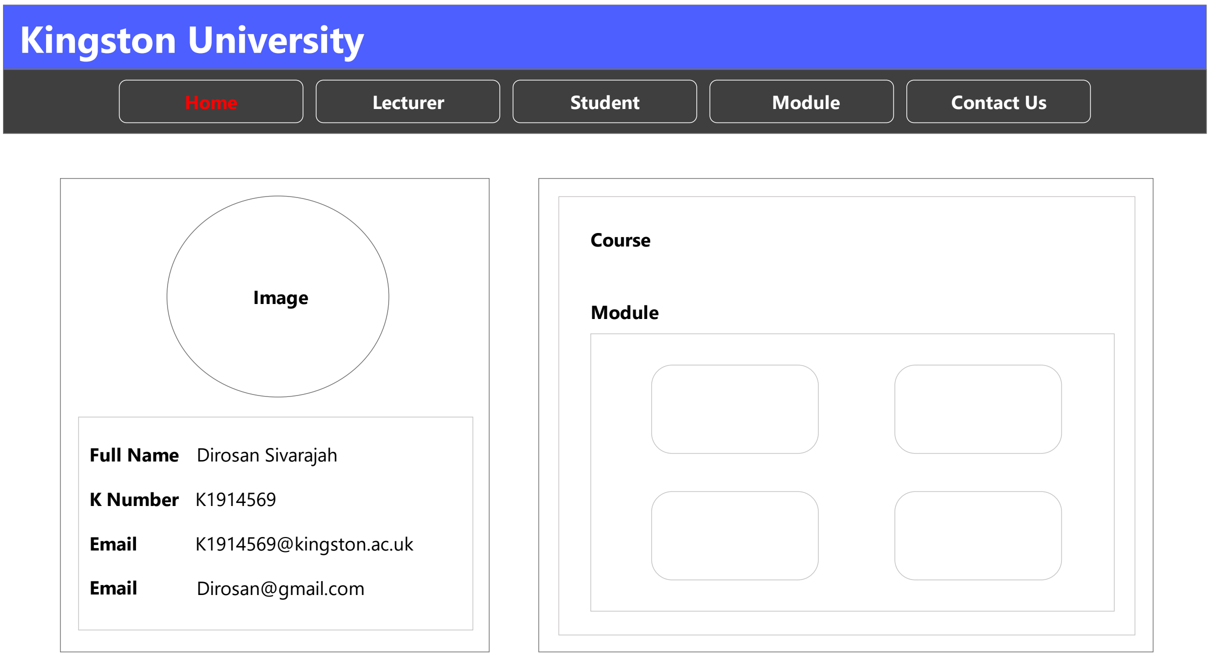
Name	Code	Level	Course Director	Image	Edit	Delete
					Edit	Delete
					Edit	Delete

Name
Input...
Code
Input...
Level
Input...
Course Director
Input...
Image
Input...



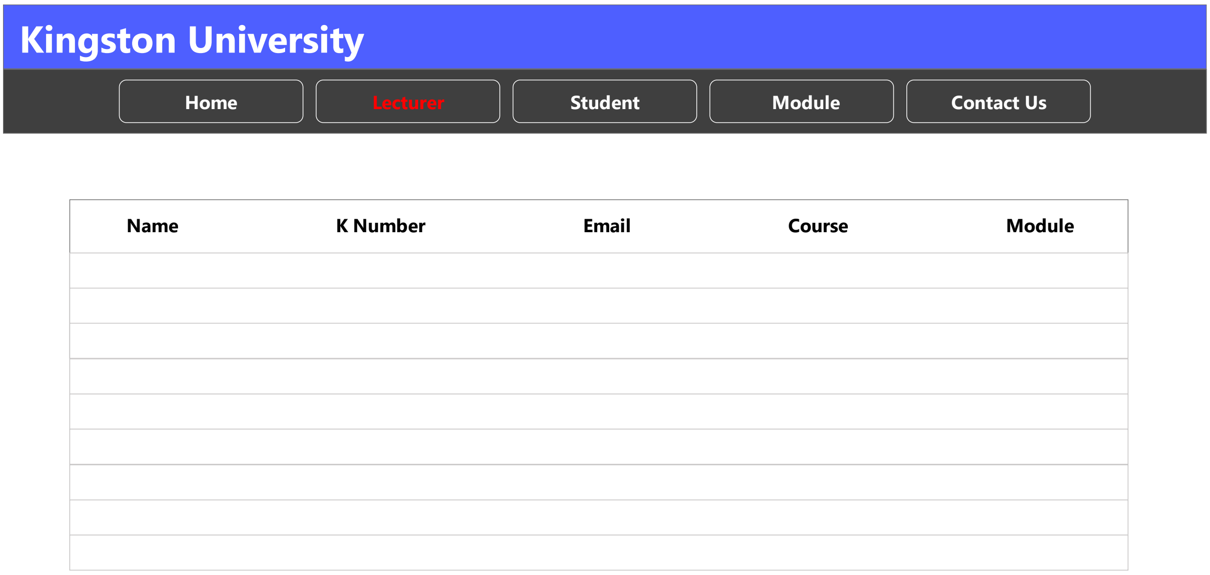
Kingston University

University Email	K1914569@Kingston.ac.uk
Personal Email	Dirosan@gmail.com

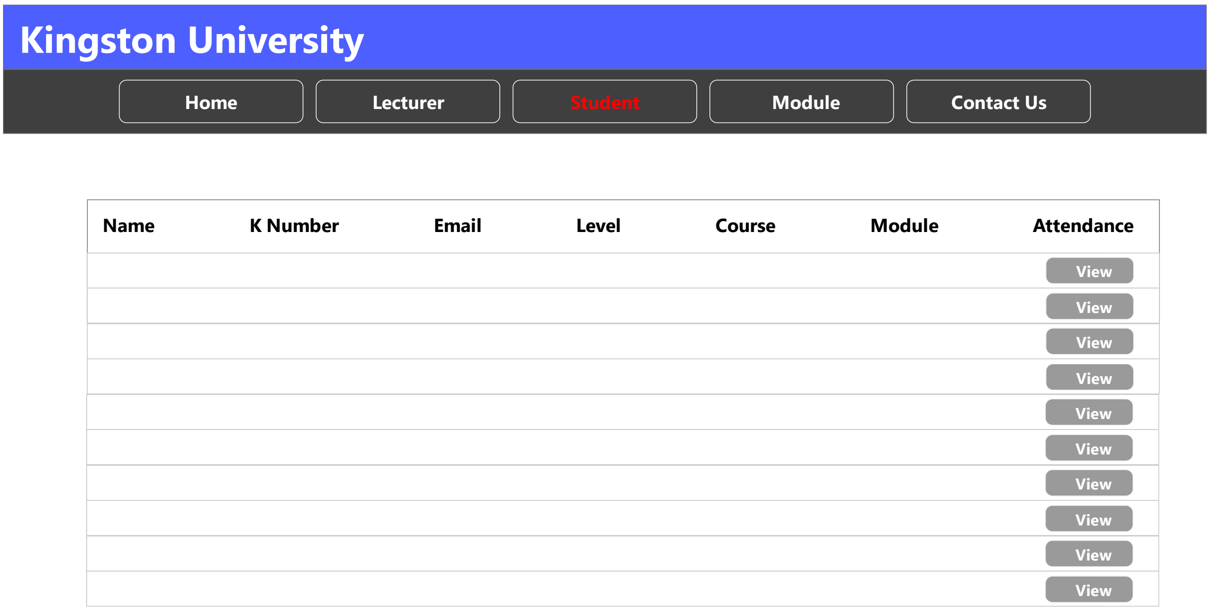


Kingston University

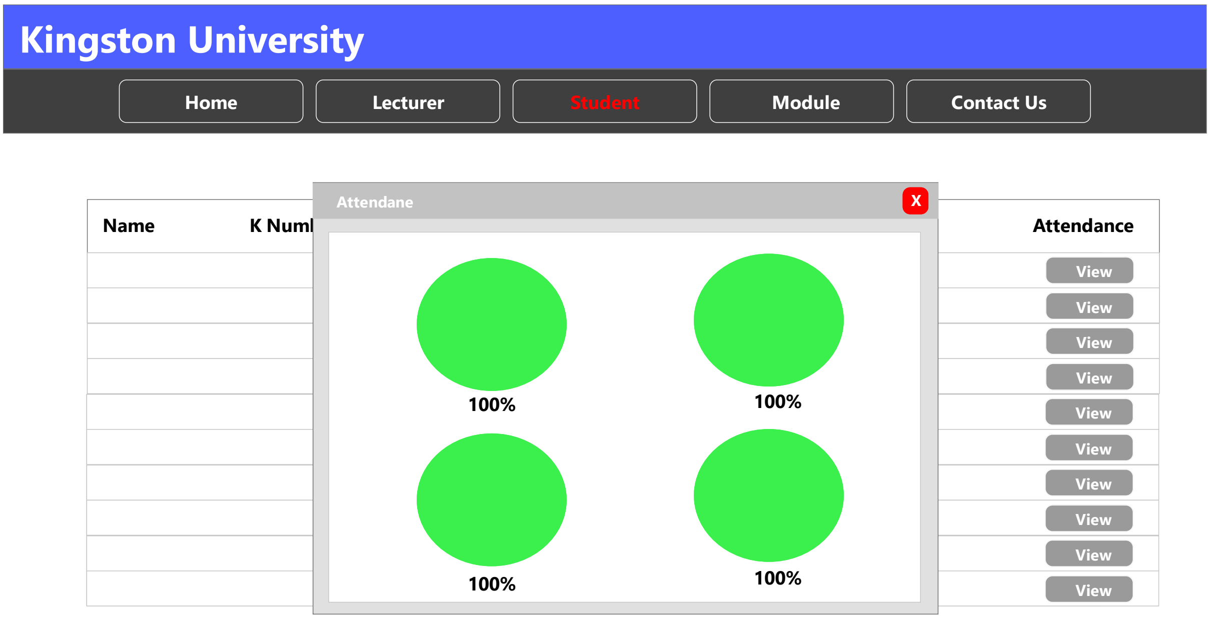
Full Name	Dirosan Sivarajah
K Number	K1914569
Email	K1914569@kingston.ac.uk
Email	Dirosan@gmail.com

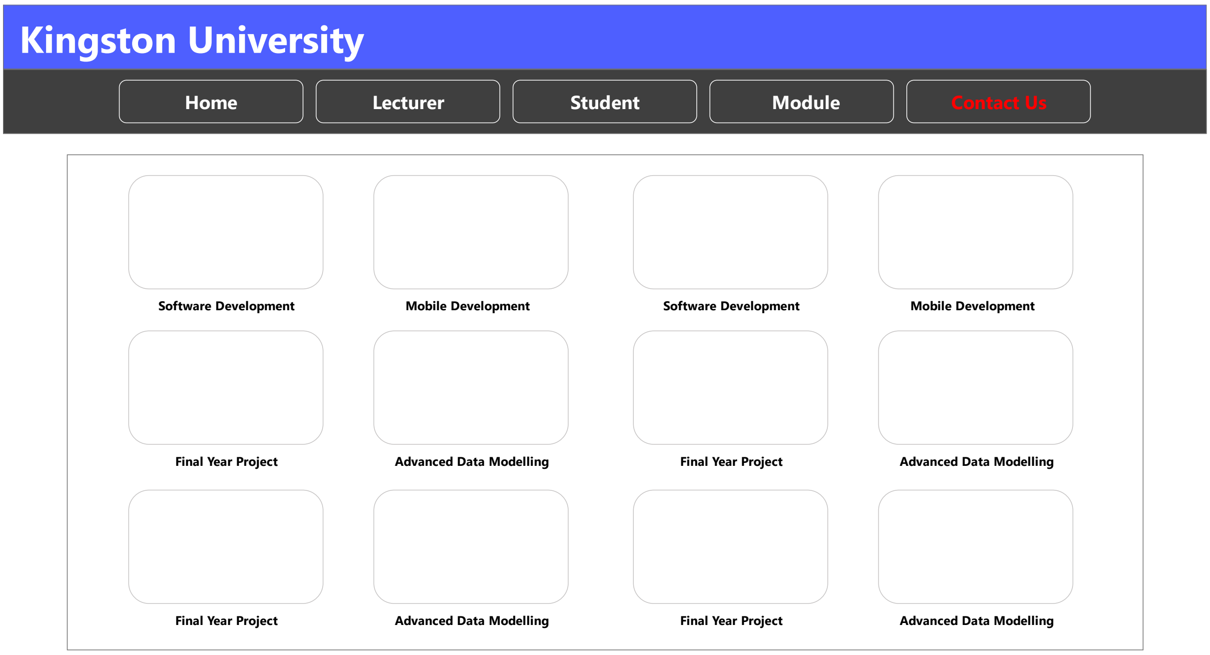


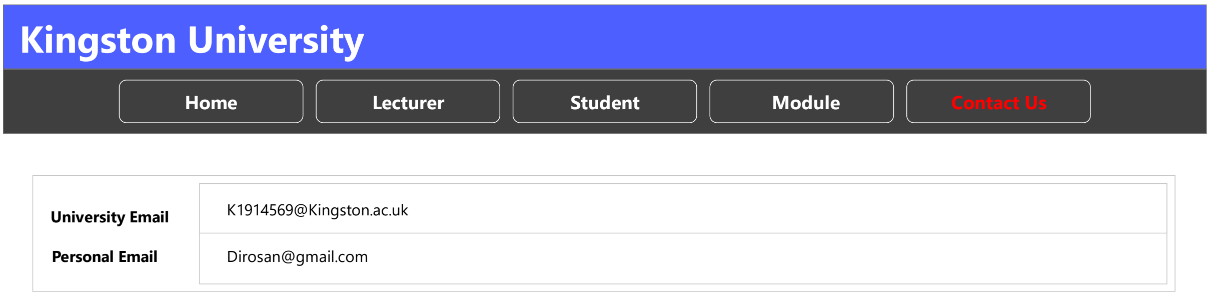
Name	K Number	Email	Course	Module



Name	K Number	Email	Level	Course	Module	Attendance
						View
						View
						View
						View
						View
						View
						View
						View
						View
						View

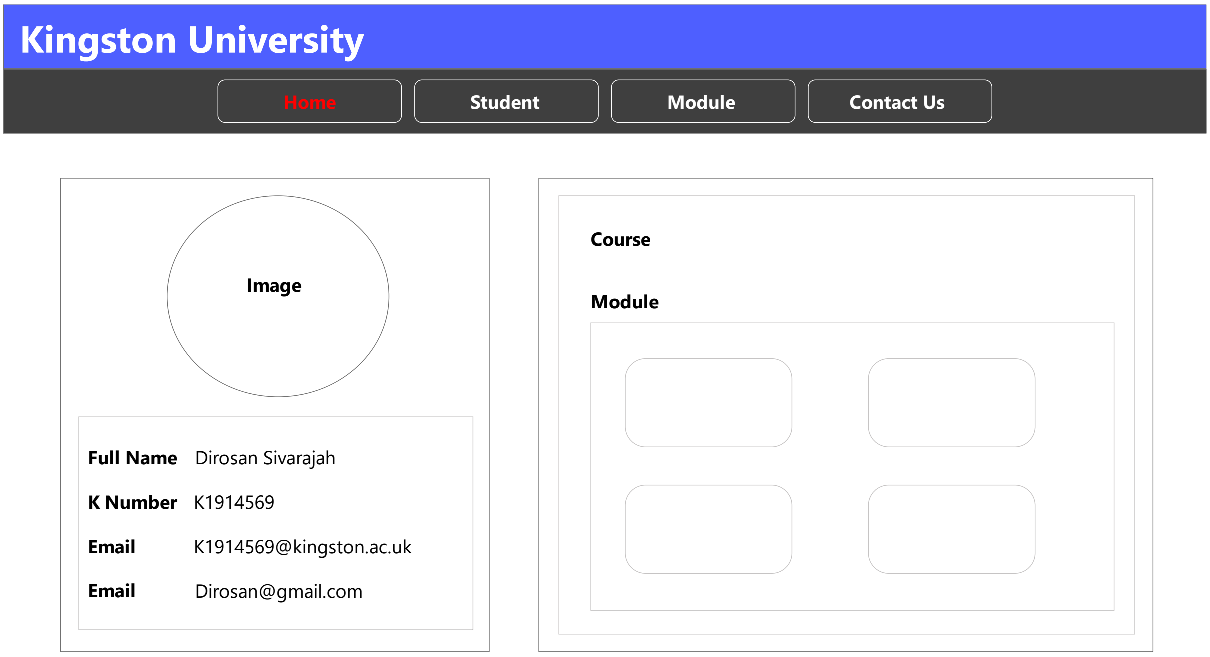

Kingston University
Home
Lecturer
Student
Module
Contact Us
Name
K Numl
Attendance
Attendane
X
100%
100%
100%
100%
View
View
View
View
View
View
View
View
View
View


Kingston University
Home
Lecturer
Student
Module
Contact Us
Software Development
Mobile Development
Software Development
Mobile Development
Final Year Project
Advanced Data Modelling
Final Year Project
Advanced Data Modelling
Final Year Project
Advanced Data Modelling
Final Year Project
Advanced Data Modelling



Kingston University

University Email	K1914569@Kingston.ac.uk
Personal Email	Dirosan@gmail.com

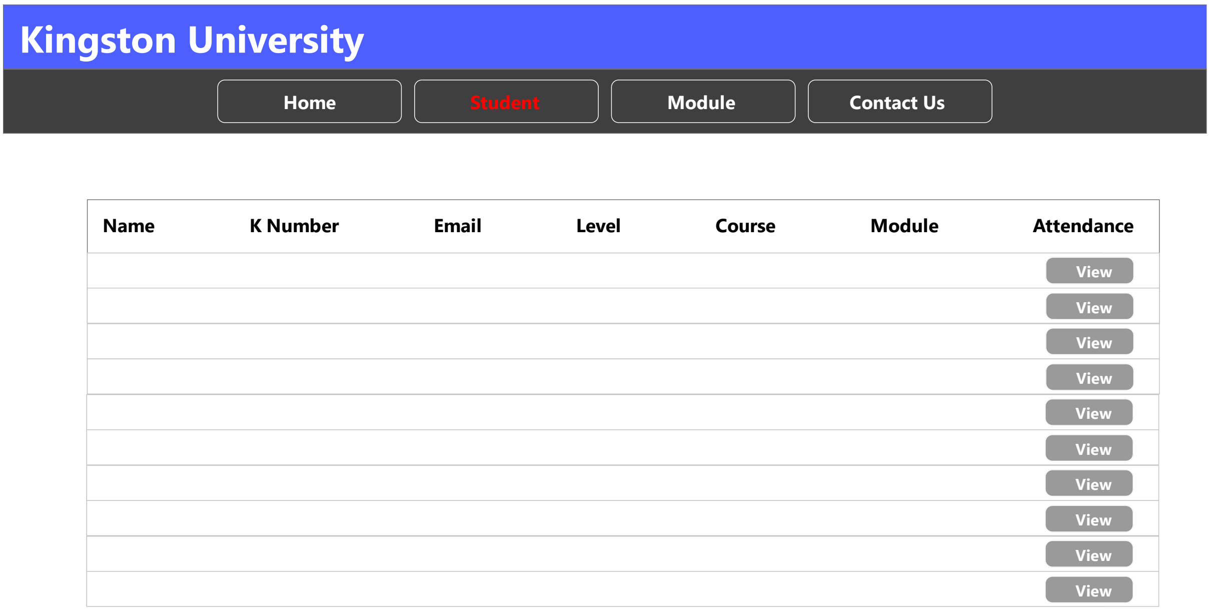


Kingston University

Full Name	Dirosan Sivarajah
K Number	K1914569
Email	K1914569@kingston.ac.uk
Email	Dirosan@gmail.com

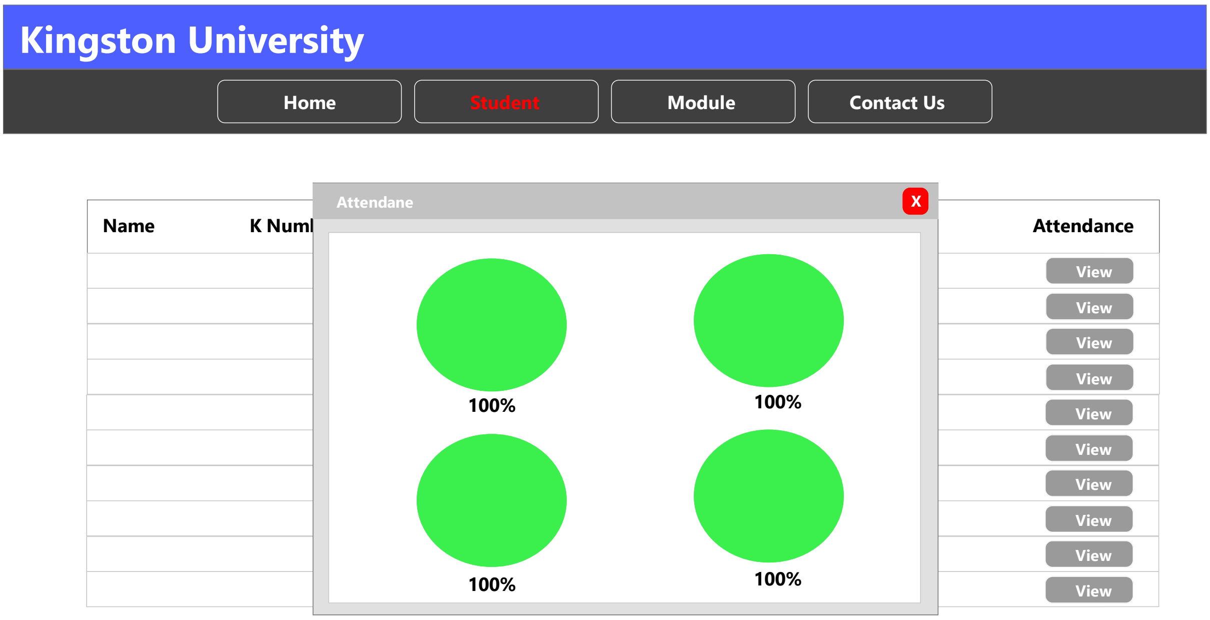
Course

Module



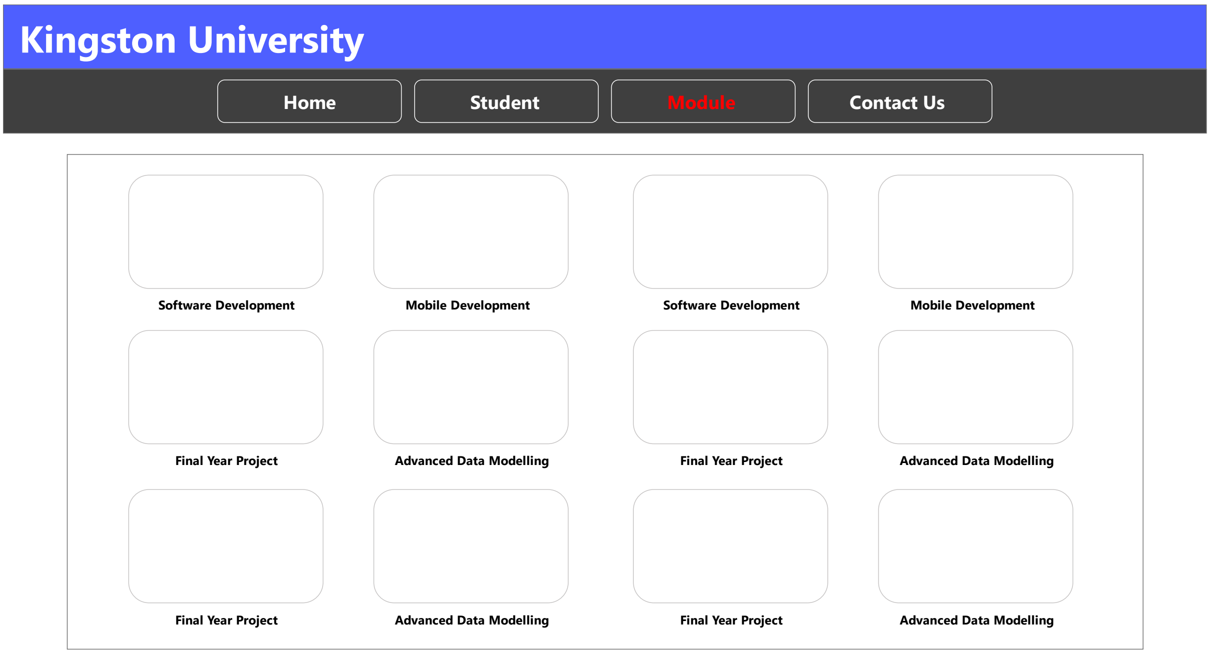
Kingston University

Name	K Number	Email	Level	Course	Module	Attendance
						View
						View
						View
						View
						View
						View
						View
						View
						View
						View



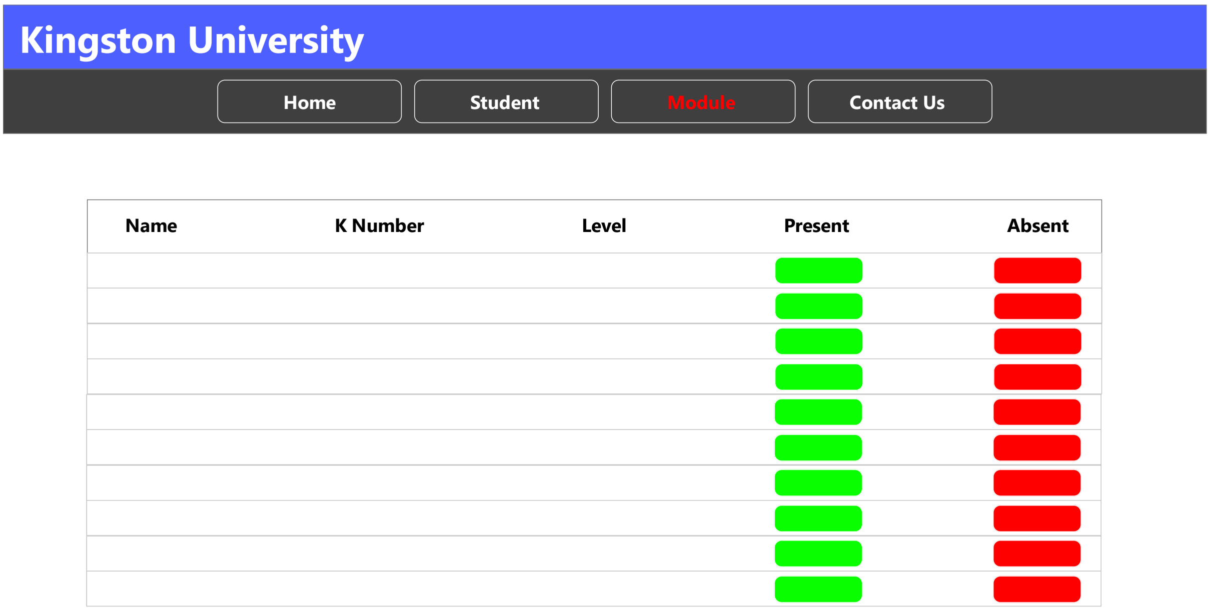
Kingston University

Name	K Numl	Attendance
		View
		View
		View
		View
		View
		View
		View
		View
		View
		View



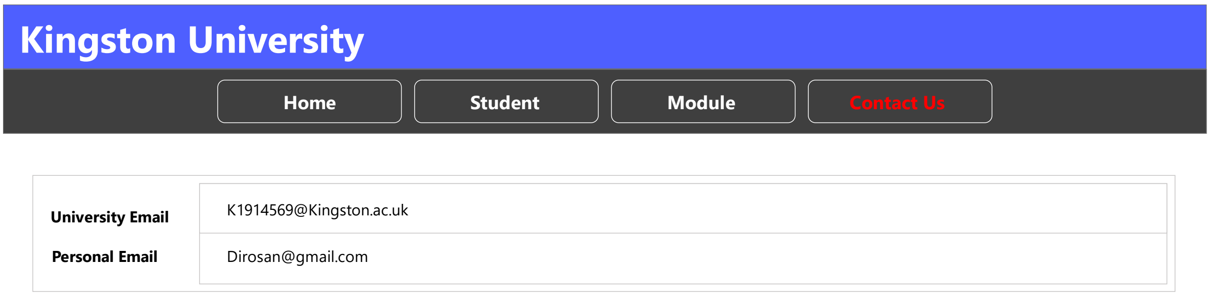
Kingston University

Software Development	Mobile Development	Software Development	Mobile Development
Final Year Project	Advanced Data Modelling	Final Year Project	Advanced Data Modelling
Final Year Project	Advanced Data Modelling	Final Year Project	Advanced Data Modelling

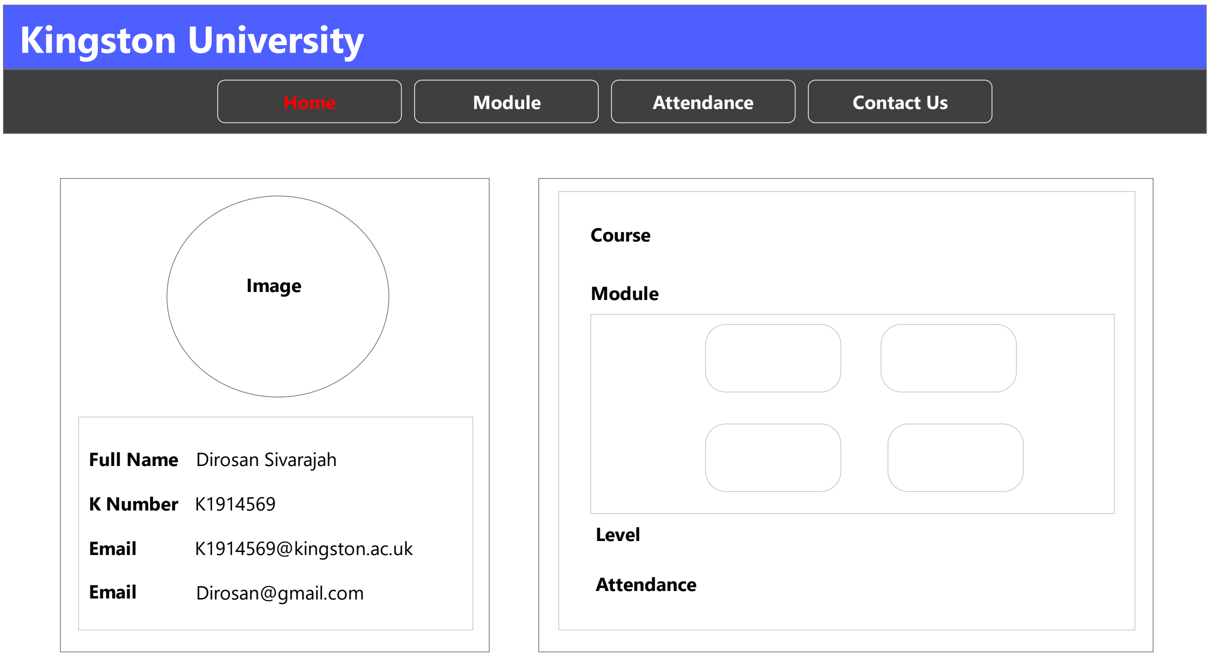


Kingston University

Name	K Number	Level	Present	Absent
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥
			🟩	🟥



University Email	K1914569@Kingston.ac.uk
Personal Email	Dirosan@gmail.com



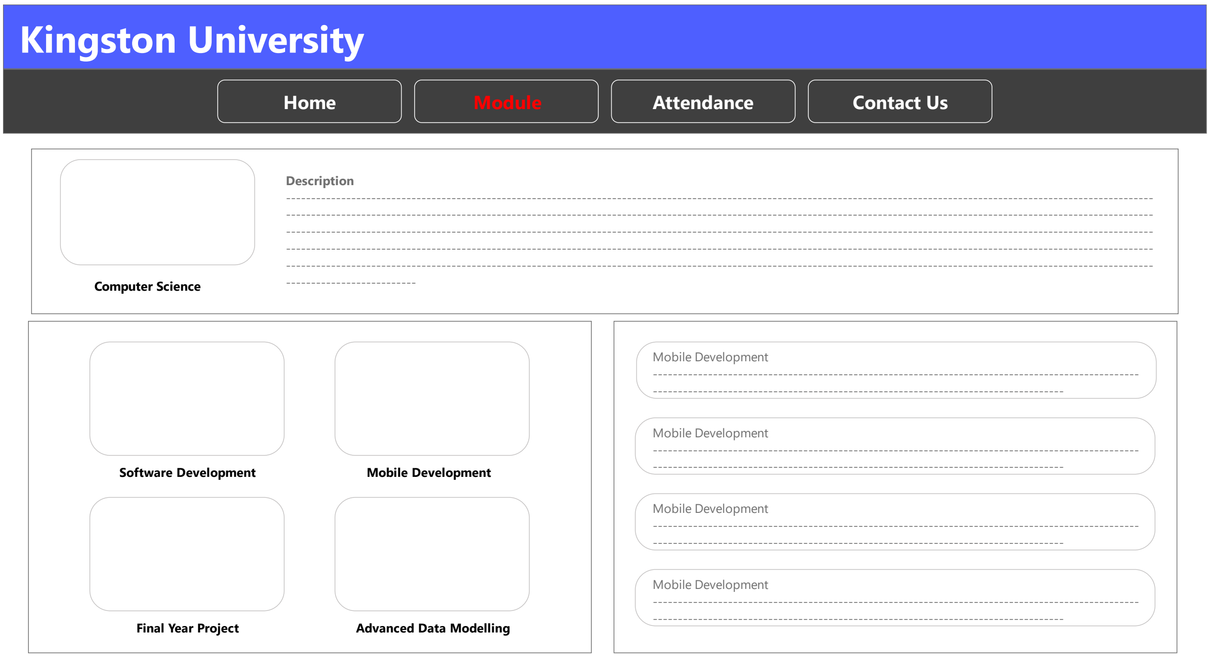
Full Name	Dirosan Sivarajah
K Number	K1914569
Email	K1914569@kingston.ac.uk
Email	Dirosan@gmail.com

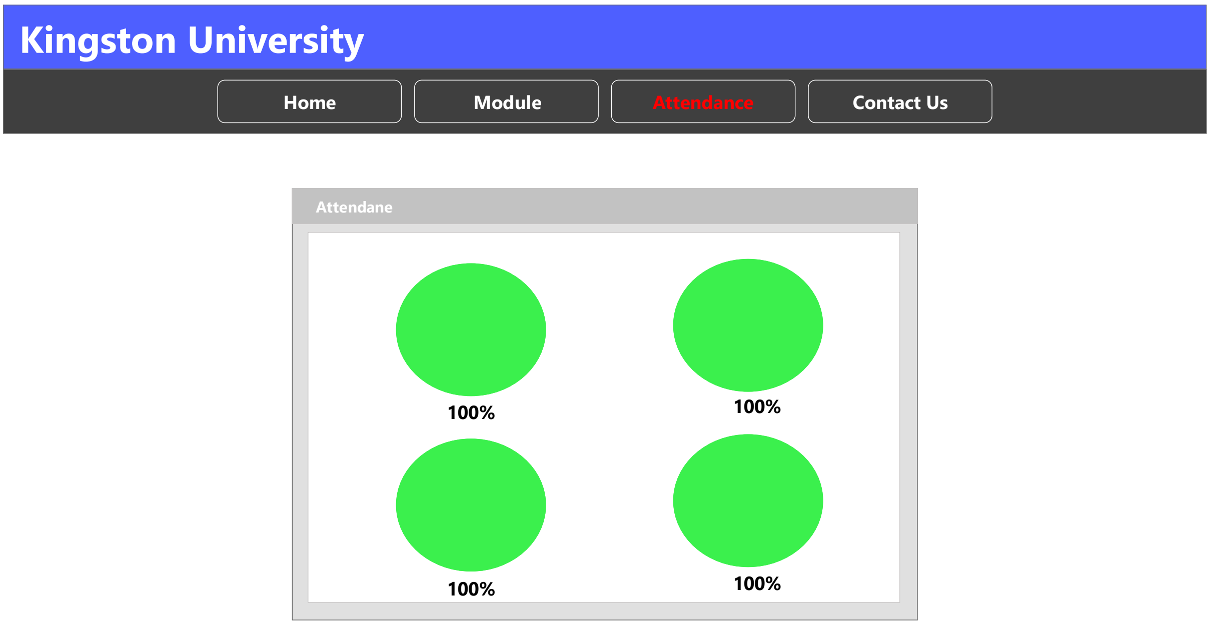
Course

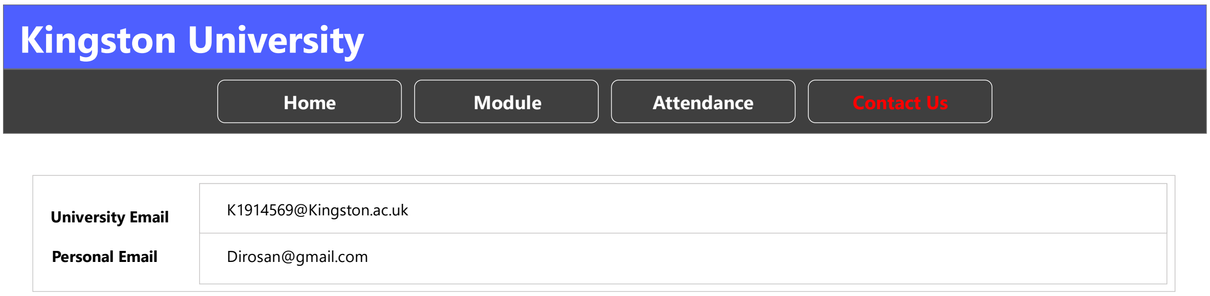
Module

Level

Attendance


Kingston University
Home
Module
Attendance
Contact Us
Description
Computer Science
Software Development
Mobile Development
Final Year Project
Advanced Data Modelling
Mobile Development
Mobile Development
Mobile Development
Mobile Development


Kingston University
Home
Module
Attendance
Contact Us
Attendane
100%
100%
100%
100%


Kingston University

University Email	K1914569@Kingston.ac.uk
Personal Email	Dirosan@gmail.com

14 Appendix: Navigation diagram

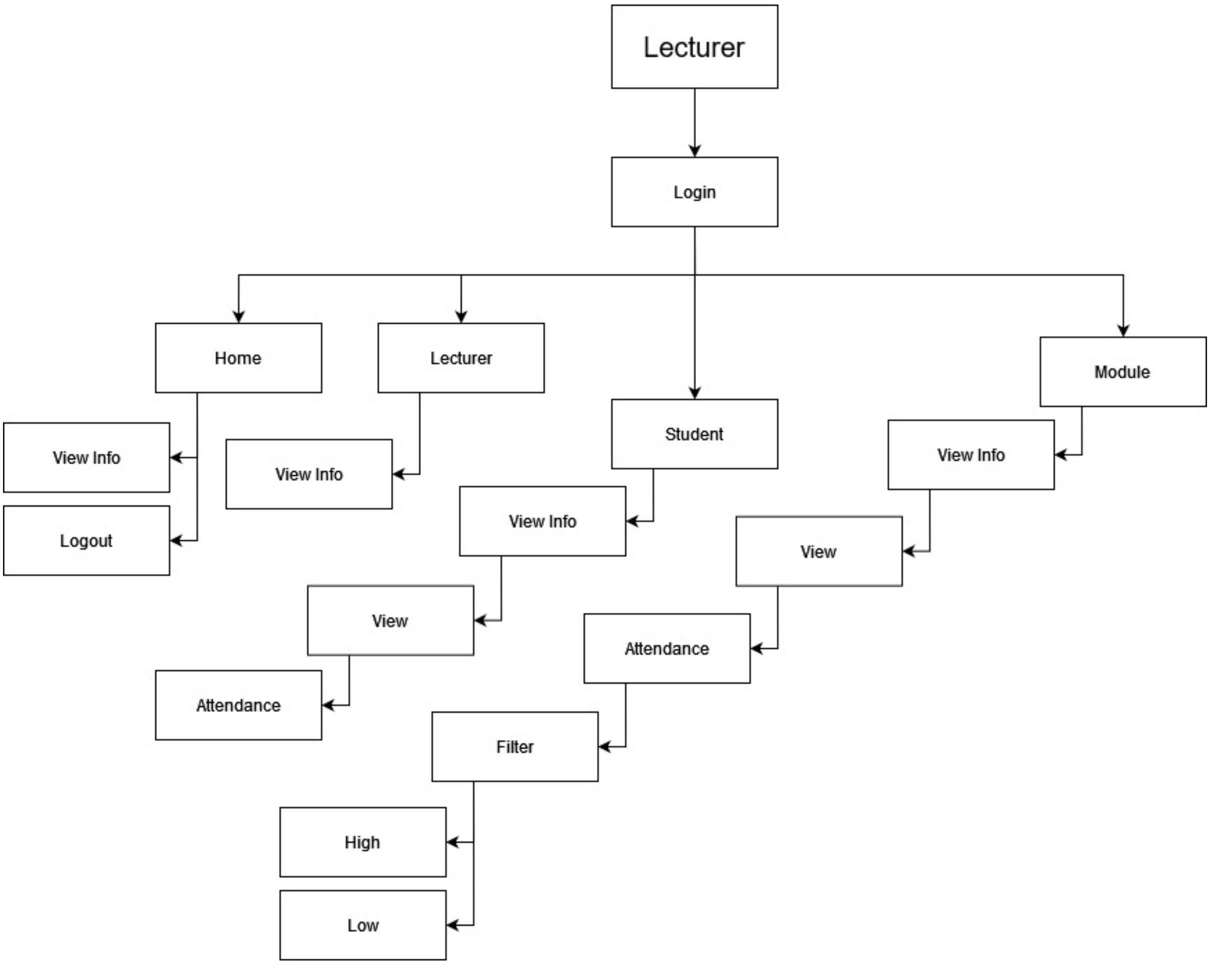


Figure 16 navigation diagram lecturer

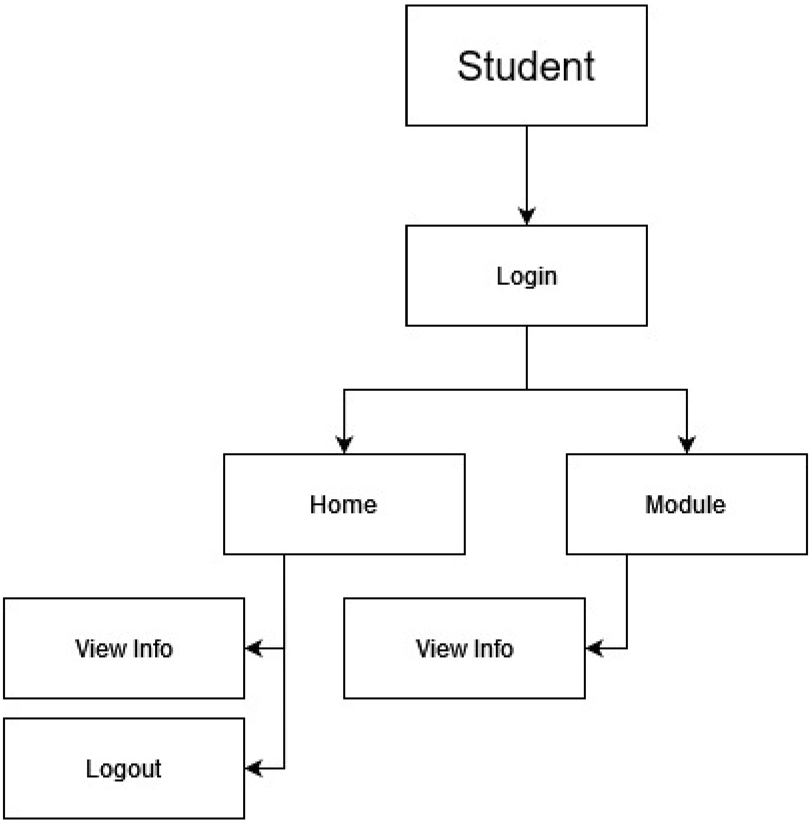


Figure 17 navigation diagram Student