

FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computing Science and Mathematics*

BSc DEGREE IN COMPUTER SCIENCE PROJECT DISSERTATION

Thomas McLean

K1038383

Project Deliverable Monitoring System

Project Type: Build

Date: 24/04/16

Supervisor: Graeme Jones

URL: <http://unipdms.co.uk>

Word count: 17,970

Kingston University London

Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

Declaration

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

Date: 23/04/2017

Signature

A handwritten signature in black ink, appearing to be 'T. W. Jones', written over a horizontal line.

Table of Contents

Chapter 1: Introduction and Literature Review	4
Introduction	4
Literature Review	5
1.1 Introduction	5
1.2 Law	5
1.3 Data Protection Act 1998.....	5
1.4 Database	7
1.5 Database Security	7
1.6 Data Encryption.....	8
1.7 HTTPS/SSL	9
1.8 Data protection and this project.....	9
1.9 Conclusion	10
Chapter 2: SWOT and Methodology	11
2.1 Introduction	11
2.2 SWOT Analysis.....	11
2.3 Software Development Methodology	12
Chapter 3: Requirements Analysis.....	14
3.1 Introduction	14
3.2 Stakeholder Analysis	14
3.3 Use Case Analysis	16
3.4 Requirements.....	22
3.5 Moscow	23
3.6 Conclusion.....	25
4 Design Stage	25
4.1 Introduction	25
4.2 Database Design.....	25
4.2.3 Data Dictionary	29
4.4 User Interface Design.....	30
4.4.1 Introduction	30
4.4.2 Accessibility.....	31
4.5 Activity Diagrams	31
4.6 Wireframes	34
4.7 Icon/ Logo	36

Chapter 5 Implementation Chapter.....	37
5.1 Introduction	37
5.2 Tools and Technologies.....	37
5.3 Architecture/ Structure.....	39
5.4 Examples of Functional Requirements Implemented.....	52
5.5 User Interface	56
5.6 Database Implementation	59
5.7 Conclusion.....	61
Chapter 6 Testing Strategy.....	62
6.1 Code Testing Methodologies	62
6.2 Security Testing in this Project.....	62
6.3 Test Cases.....	63
6.3.1 Functional Test Cases.....	63
6.3.2 Non-Functional Test Cases.....	64
6.5 Conclusion.....	64
Chapter 7 Critical Review	65
7.1 Introduction	65
7.2 Implementation Issues.....	65
7.3 Future Development.....	65
7.4 Lessons Learned	65
7.5 Conclusion	65
References	66
Bibliography	66
Appendixes.....	68
Use Case Texts	
Wireframes	
Activity Diagrams	
Data Dictionary	
Example data in tables	
Student_Deliverable Table.....	
Student_Meetings.....	
Supervisor Meetings Table	
Testing.....	
Functional testing	
Non-Functional Test Cases.....	

Chapter 1: Introduction and Literature Review

Introduction

This report will discuss the process which is going to be taken to deliver the project for the user. Supervisors at Kingston University currently have no online system to monitor how their students are performing in their final year projects. This project will look to overcome this by creating a student deliverable monitoring system for final year projects to give supervisors the necessary tools to support their students through an online application.

The deliverables will be gathered from the stakeholders and the requirements. There are currently no similar systems in use at Kingston University which will set students deliverables which were agreed for them to create a successful final year project.

Technologies must be planned and recommended based on which will be best for the project. Being an online system it will use programming languages like PHP, HTML and JavaScript to deliver the application.

This project scope can be viewed in a Gantt Chart. On the Gantt chart, each phrase and key dates can be viewed.

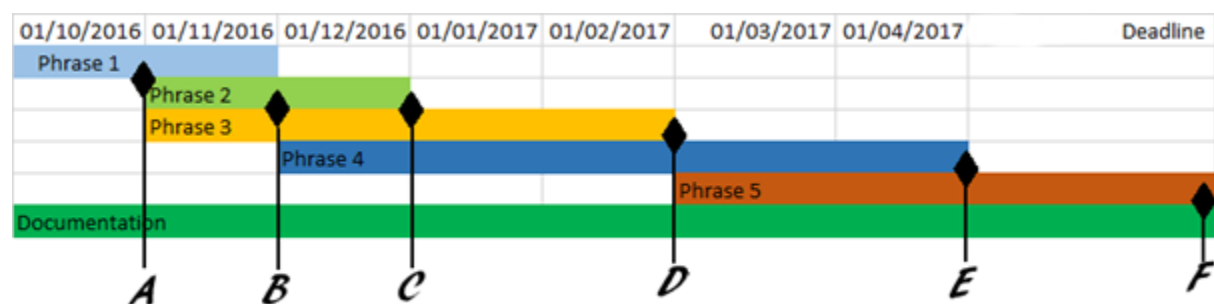


Figure 1 Gantt Chart

This report will discuss each stage the project went through to get to its final state. This involves producing a Literature review, analysis and requirements, design process, implementation and testing. Each stage will be reviewed and documented as the project develops through each one.

Literature Review

1.1 Introduction

This chapter will discuss the data protection law. It will highlight the laws surrounding the data protection act and review how a database can be developed to support the application in staying within the laws. The Data Protection Act (1998) will be researched and described below on how this law will affect the project when storing personal data. This includes The Law, Data Protection Act, the eight principles and data protection including data encryption and security. The findings will then be used to ensure the data of the users are stored safely and kept within the law.

1.2 Law

Start with the data protection act 1998 was introduced in the UK to ensure the protection of different types of personal data being stored and accessed by businesses. As this project is based in the UK, other Laws such as the European Law, will not be described in detail as this will not affect my current project although is something to keep in mind for the future development of the project if it were to expand.

1.3 Data Protection Act 1998

“The Data Protection Act controls how your personal information is used by organisations, businesses or the government” (*Data protection*, 2016). The data protection act includes data stored remotely such as on computers as well as data that is paper based.

“The **1998 Data Protection Act** was passed by Parliament to control the way information is handled and to give legal rights to people who have information stored about them.” (BBC, 2006). The Data Protection Act was introduced to control the way companies stored and processed personal information to stop the data being misused or accessed by unauthorised users. If unauthorised users gain access to the database this could lead to the serious danger of people personal information. The data protection act gives the users within the database the legal rights to obtain access to their data stored about them and can ask for their data to be destroyed.

The Eight Principles

The data Protection Act has eight principles which must be followed. This is to ensure that all the data is stored and used correctly within the data protection act. The eight principles are:

Principle One: Processed Fairly and Lawfully

“Personal data shall be processed fairly and lawfully and, in particular, shall not be processed unless:

- at least one of the conditions in Schedule 2 is met
- in the case of sensitive personal data, at least one of the conditions in Schedule 3 is also met.” (Information Commissioner’s Office, 2017).

This principle ensures the data is used correctly and is not used for anything against the law. This is important as, if this data got into the wrong hands, the data would be misused which could affect people’s everyday lives and cause major problems (like fraud).

It also instructs that the data can be processed if two of the following are met. One of these is if schedule two is met. Schedule 2 is “Conditions relevant for purposes of the first principle: processing of any personal data” (Participation, 1998). This may be for many different reasons:

- The user has given consent for the processing to happen, which many websites may include a tick box at the bottom of a form or in the terms and conditions.

- The processing must happen (e.g. administration of justice).
- To protect the interests of the user data.

The other reason for the data to be processed under the act is if schedule 3 is met for sensitive data. Schedule 3 is “The data subject has given his explicit consent to the processing of the personal data.”

Principle Two: Personal data obtained

“Personal data shall be obtained only for one or more specified and lawful purposes, and shall not be further processed in any manner incompatible with that purpose or those purposes.” (Information Commissioner’s Office, 2017). This ensures that personal data is only obtained for the reasons it is intended for.

Principle Three: Relevant Personal data

“Personal data shall be adequate, relevant and not excessive in relation to the purpose or purposes for which they are processed.” (Information Commissioner’s Office, 2017). This ensures that only the necessary data is obtained and stored by the company or website. This stops unnecessary amounts of data about users being kept and stored in a business.

Principle Four: Accurate Personal data

“Personal data shall be accurate and, where necessary, kept up to date.” (Information Commissioner’s Office, 2017). This is to ensure that the information being stored by the company is relevant and can be used for the purposes needed and is not outdated and not being used in which case it should not be being kept at all.

Principle Five: Personal data Processed

“Personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes.” (Information Commissioner’s Office, 2017). This ensures that the data is not kept and stored for longer period than what it is needed for.

Principle Six: Personal data processed according to the rights

“Personal data shall be processed in accordance with the rights of data subjects under this Act.” (Information Commissioner’s Office, 2017). This makes sure that the data of the user is processed correctly and ethically. This enforces the user's rights as well.

Principle Seven: Appropriate measures

“Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.” (Information Commissioner’s Office, 2017). This once again enforces the seriousness of storing personal data and ensuring that it does not get into the wrong hands or lost. This can be from SQL injection or someone gaining unauthorised login into the database.

Principle Eight: Personal data transfer rules

“Personal data shall not be transferred to a country or territory outside the European Economic Area unless that country or territory ensures an adequate level of protection for the rights and freedoms of data subjects in relation to the processing of personal data.” (Information Commissioner’s Office, 2017). This states the important rules for global businesses which may transfer data around the

world. This should not affect this project, as it is only UK based project in one location, however, is an important point for the potential future of the project or other similar projects.

1.4 Database

A database is a software which contains records which are stored in a table or class. In this project, a database will be required to store the data. 'A database is defined in the legislation as "a collection of independent works, data or other materials which are arranged in a systematic or methodical way and are individually accessible by electronic or other means." '(Database rights: The basics, 2015). This definition covers a wide range of data storage including being in paper or electronic form.

A database is vital these days for a business to have and is used worldwide to store information about their users. This may be customers for a shopping market to readers of a blog or even a collections of product types. A database is required to store their information. Databases can hold a massive amount of data. There are different ways database can be created. The most popular ways are using MySQL or Oracle. It is also vital to ensure that the database meets the UK law of the Data Protection Act to keep personal data safe.

1.5 Database Security

Database security should be taken very seriously as this is the last line of defense against someone trying to gain access to the data. Database security is measures put in places to protect and secure the database from illegal use or malicious threats and attacks. This is achieved by using available tools and methodologies to the business which increases security within the database, to cover all aspects.

The different aspects which should be covered in a database are:

- Data
- Server
- DBMS (Database management system)
- Database workflow applications

Database Security vulnerabilities

While databases are very secure when implemented correctly with the right security, there are many vulnerabilities. "The most common cause of database vulnerabilities are a lack of due care at the moment they are deployed." (Osborne, 2013). Many database administrators will install and create the database through different ways, and then test the functionality is working correctly. While this is important to check, it has been proved that fewer checks are made on the database, to check it is not doing anything it shouldn't be. This may include allowing incorrectly values or users with more permissions than they should have. This is how most company's databases can be accessed or attacked because of these flaws which were missed in the deployment stage. This leads to big issues with the data protection which can cause the company to become a part of legal battles which could lead to issues which can't be controlled, including data being miss used. Back in 2011, Sony released a statement that "names, addresses and other personal data of about 77 million people with accounts on its PlayStation Network (PSN)" had been stolen. (Quinn and Arthur, 2011). This caused a massive upset to the company and affected sales, shutting down the online service to avoid any more hacks into their database system, which was storing personal data about their users. It was shut down for just under a week losing the customers trust and reputation.

Another big security vulnerability is simply not keeping the database security up-to-date. "The SQL Slammer worm of 2003 was able to infect more than 90 percent of vulnerable computers within 10

minutes of deployment, taking down thousands of databases in minutes. “(Osborne, 2013). This was a bug at the time in the Microsoft’s SQL Server database software a year before but due to the lack of updates within the database security being done, a massive security risk was still available within 90% of the databases which used the server.

One of the biggest problems with databases being vulnerable is SQL injections. This is when a hacker writes SQL in a form to be sent to the database. This may be a delete statement in the first name field which will be sent by the application thinking this is innocent data but once it reaches the database server, it performs a completely different type of statement than what the developer intended. This is very popular for hackers to part take in, to attack the database and gain or delete data. This can be controlled through SQL rights managements and code implemented before the SQL are ran, to check and ensure the SQL is going to do what it was intended to do.

These three forms of vulnerabilities are due to human errors. The database security is only as good as it has been programmed to be, so to ensure it is at its highest level, updates should be done as soon as possible, it should be tested to ensure it can’t be hacked due to lack of security and the developers should use up to date code to protect the database This can all be done if developed correctly, with the right research and knowledge. Another way data can be protected is from data encryption.

1.6 Data Encryption

“Data encryption translates data into another form, or code so that only people with access to a secret key (formally called a decryption key) or password can read it.” (Lord, 2017). This is currently an effective data security method. Data encryption protects important data like passwords or personal data using algorithms and includes a security authentication.; integrity and non-repudiation. Each one of these has a different job. Authentication stores the message origin for verification, the integrity “provides proof that message contents have not changed since it was sent” (Lord, 2017).

There are two ways in which data encryption exist. They are asymmetric and Symmetric encryption.

1.6.2 Asymmetric Encryption

Asymmetric encryption uses two keys as a pair. One is a public key and the second is a private. The public key is available for anyone that wishes to get in contact with the user. The private key is kept for the user. With this, any message or files that are encrypted by the public and sent to the user can only be decrypted by using the same algorithm by using the users private key. The same thing works the other way around with the private key, anything sent by the user is encrypted using the private key and can only be decrypted by using the matching public key, both users have. This is a very strong way of encrypting and keeping data secure but there are disadvantages of doing it this way. The problem with asymmetric encryption “is that it is slower than symmetric encryption. It requires far more processing power to both encrypt and decrypt the content of the message.” (Microsoft, 2017). This means that while this is a secure way of encrypting and keeping data safer, it will slow down the application which may cause a bad user experience compared to other application.

1.6.3 Symmetric Encryption

Symmetric encryption is the oldest and best-known technique (Microsoft, 2017). Symmetric uses only one key, called a secret key. This can be a word, random letters or a number which is applied to the string in the message to change it in a way which cannot be read by humans. It is then decrypted by both sender and recipient knowing the secret key. This approach has proven successful and

effective but with newer approaches like Asymmetric Encryption, this is no longer the most secure way of encrypting data.

1.7 HTTPS/SSL

While encryption is a great way for the developers to protect the applications personal data, there is more the developer can do to ensure the data is secure. One way is Hypertext Transfer Protocol Secure (HTTPS). HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the Web server (Rouse, 2008). Developed by Netscape, it was used to keep data from getting into the wrong hands while it was being sent over the network.

It was a known technique for hackers to listen in on what was being sent over a network to gather information about a person. Another technique is known as 'Middleman attacks'. This is when the HTTP has been interrupted by a middleman who receives the data and then lets the HTTP continue to its intended destination.

HTTPS works by encrypting the data being sent to the server to ensure it was untouched and can't be read on the way. This one technique is used by many online applications sending and receiving personal data. An issue with HTTPS is that while the data is secure being sent and received, once decrypted the data is only as secure as the client computer. HTTPS uses the Secure Socket Layer (SSL)

SSL is another way developers keep the data secure between the users and the server. SSL is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser, or a mail server and a mail client (What is SSL (secure sockets layer), 2003). SSL are protocols that provide secure communications over a network. The protocols are found in many different applications including web browsing, emailing and instant messaging. SSL also protects personal information like credit card details to be transmitted securely. SSL protects millions of people's data every day on the internet, making it the most well-known and most commonly used.

Both can be seen in the URL when they are being used. Most browsers will display a message in the URL displaying HTTPS//or a green box with the SSL certificate in use. Both HTTPS and SSL support the use of X.509 digital certificates from the server so that, if necessary, a user can authenticate the sender (Rouse, 2008). You need a certificate to use an SSL protocol and these must be purchased.

Personally, recommend both to be considered for this project to ensure all the data is protected to and from the server.

1.8 Data protection and this project

Data protection will be an important aspect of this project to think about when developing and testing my project. This project will be storing personal data about the students and supervisors like their legal first name and surname. To store personal details, this project must meet the Data Protection Act set by the law. It is recommended for this project that all the below are achieved:

- Complying with the law.
- Ensuring the necessary tools are in place to protect and secure the personal data and passwords
- Use Symmetric encryption for passwords and other personal data
- To use HTTPS or purchase an SSL certificate
- The latest version of MySQL is installed and updated continually to the latest version.
- Test database for possible attacks and threats.

- Use up to date web development technologies
- Use specific ways to ensure SQL Injections is avoided.
- Ensure the latest security tools are available to use.

1.9 Conclusion

Data protection is an important matter and should be taken seriously. The Data Protection act was put in place by the UK Government to ensure businesses comply with rules about handling personal data. This can be on a paper or systematic approach using a database. With the eight principles written within the Act, it provides businesses with a straightforward way of understanding what the Law expects anyone handling this data to do. As a database, will be used in this project, it is important to make sure the database security is the latest version and the right actions are taken to protecting the database. This included encryption, verify the code and testing to ensure the application and the database only does as it is intended to do.

Overall, this was an interesting topic to review and one which will help with not only this project but all projects undertaken in the future.

Chapter 2: SWOT and Methodology

2.1 Introduction

This chapter will look preform a Strength, Weakness, Opportunity and Threats (SWOT) analysis and discuss software development methodologies.

2.2 SWOT Analysis

The SWOT analysis is a well-known tool to explain the value and feasibility of the project. The SWOT diagram below displays and highlights the strengths, weaknesses, opportunities and threats. It is essential for these to be thought out and compared against each other to justify if the project should go ahead (or not).

<ul style="list-style-type: none">• Gap in the market for the system as it is in demand by a current supervisor at a university.• Support from the users/ client is in need for the system.• Can be developed and maintain by all open source tools.• Based on a system already being used by supervisors but on paper/ office tools.	<ul style="list-style-type: none">• May need continuous development to keep the data secure from threats in the future
<ul style="list-style-type: none">• Possibility to develop and expand the user base to millions of users• Challenging, exciting project• Build a system required for a university• To build a bespoke system	<ul style="list-style-type: none">• Competition from systems also in development at this current stage• Requires a reliable external hosting to host the web based system• Client changed it mind about the functionality. Slow down the process• Loss of data

The strengths highlight some strong arguments, showing the project can be developed using open source software. It also shows that the current supervisors need a system to be developed, which highlights there is a demand for this project.

The opportunities have arguments towards justifying to go ahead with the project. There is a great possibility that the system can be developed to expand the user base to meet the needs of lots of educational institutions in the UK but also has the potential to become more bespoke application to a university where it is continuing to be developed to meet their individual needs.

While the strengths and opportunities put forward some strong arguments to going ahead with the project, the weakness and the threats provide reasons to not go ahead. The weakness is being outweighed by the strengths and highlight the fact that the system will need to be maintained in the future to help ensure the system stays secure from threats including hacking by encrypting passwords to the latest form of encryption.

Four threats have been highlighted but with the correct mitigations put in place, they can be overcome. Mitigation is putting plans to ensure threats are reduced and opportunities are increased.

The SWOT analysis justifies going ahead with the project with the strengths and opportunities being far stronger than the threats and weaknesses. With the threats being highlighted and analysed, these can now be carefully considered to construct a plan to reduce the chances of these happening and if they do there will be a plan in place to make sure the project continues to run. This project can be highly successful and is worth going ahead with.

2.3 Software Development Methodology

Software development methodology is the “framework that is used to structure, plan, and control the process of developing an information system.” ("Software Development Methodologies", 2015)

There are many methodologies which could be taken up by this project but finding the one which best fits this project is important. The requirements will be provided at the beginning by my client. There is two software development methodology which stands out, to use on this project. They are Waterfall and an Agile approach using DEDM, Scrum or extreme programming.

Waterfall

Waterfall model is a sequential design process. Each phase of the product lifecycle is in a sequence. These phases are Feasibility, Plan, Design, Build, Test, Production, Support.

The advantages of the waterfall method are:

- Potential issues are found out before the coding begins
- The structure of the method is easy to understand compared to other methods, making it easy for a team of developers to follow and complete.
- Requirements and design work are completed before any coding happens.
- Good for Client who knows exactly what they want.

The disadvantages of waterfall method are:

- Clients don't always know what they need up front
- Not all problems can be foreseen before the coding begins
- Waterfall can be poor with changes in requirements
- Can be costly
- Hard to go back and make changes

Agile

Agile methodologies are great for requirements which are most likely going to change over time. The requirements are prioritised based on the business or user value, meaning the most important task will get completed first.

Different Agile approaches include:

- DSDM
 - Follows 8 principles to make sure the project is delivered successfully
- Scrum
 - Concentrates on how to manage tasks in a team-based environment
 - Sprints (ALLIANCE®, 2016).
- XP
 - Focuses on programming the project with a little planning before. About getting the function available to the user as quickly as possible.

Decision

All the Agile approaches provide a great relation to the client by keeping closely together but on the other hand, my client understands what they require as they have been doing this system on paper and Microsoft office for many years. For this reason, the methodology which has been chosen to be used is the waterfall method with Prince 2 because the requirements are unlikely to change and will provide clear deadlines for each stage which must be hit for the critical path to not change. Prince 2 will provide a series of processes which help control activities and ensure they meet the standard required which meeting the target deadlines.

Chapter 3: Requirements Analysis

3.1 Introduction

The purpose of this chapter is to derive the requirements which shape what will be delivered by the deadline. Functional requirements define the specific behaviors of the application. Non-functional requirements define constraints on how the system will achieve the requirements. To generate these requirements, the stakeholders will need to be identified and summarised. This will help to create a list of requirements. This will be done by creating a use case diagram and then with user stories, identify the functional and non-functional requirements.

3.2 Stakeholder Analysis

3.2.1 Introduction

It is vital to identify the stakeholders as they play an important role in the success of this project. Stakeholders are the group of people which can affect the project progress and outcome. There are four categories the stakeholders are a part of depending on the power and interest they have over the project. They are 'Keep Satisfied', 'Manage Closely', 'Monitor' and 'Keep Informed'. Each of the stakeholders will be a part of at least one of these categories.

To find the stakeholders who are involved and can be affected by the project, they were identified by researching into the main users and grouped. Contact was made to the client who helps to highlight who may be affected and have some power or interest over the project. After working from each many users (Supervisors and Students), the stakeholders came to light. It was important to make sure all aspects were covered internally and externally and that their duties are discussed.

3.2.2 Stakeholder Analysis Table

Role/Stakeholder	Categories	Internal/External	Duties
Students	Keep Satisfied	Internal	The student will be using the system and will become a vital part of their final year project when communicating with the supervisor. Ensuring it is user-friendly is important and all the most important functionally is in place.
Supervisor	Manage Closely	Internal	The supervisors will be the main users of the system. A close relationship will need to be kept and consistent feedback. This is so the stakeholder can oversee the system and the system will be exactly what required.
Module Leaders	Keep Satisfied	Internal	Module Leaders will be overseeing the Supervisors so it important this system meets the module leader's needs and keeps them satisfied. They will have power over the system but unlikely to show this power unless the supervisors are not happy.
Supporting Staff	Keep Informed	Internal	These stakeholders may use the system but do not have a high priority. They unlikely to be affected by the system.
Parents	Monitor	External	Parents will not have usage of the system but are important to monitor as if they feel the system does not support the students, this can cause issues for the system in the

			long term which could end the project in failure. I will aim to address this by keeping the students satisfied, which in turn should keep the Parents satisfied too.
Computer Societies for Educations	Monitor	External	This stakeholder does not have a direct impact on the system but can influence the success. If the system impresses this stakeholder group, it will in the long term bring success to the project as hopefully, they will recommend the system to university and supervisors.

The stakeholders all play a role in the success of the project. Some have high power and others have a higher interested. Covering each stakeholder has not safeguarded this project and hopefully, by keeping each of these stakeholders happy, the project will be successful.

3.2.3 Stakeholder Prioritisation

The stakeholders which have been identified will now be prioritise. The Stakeholder Prioritisation highlights and demonstrates the most important stakeholders to the project (PDMS).

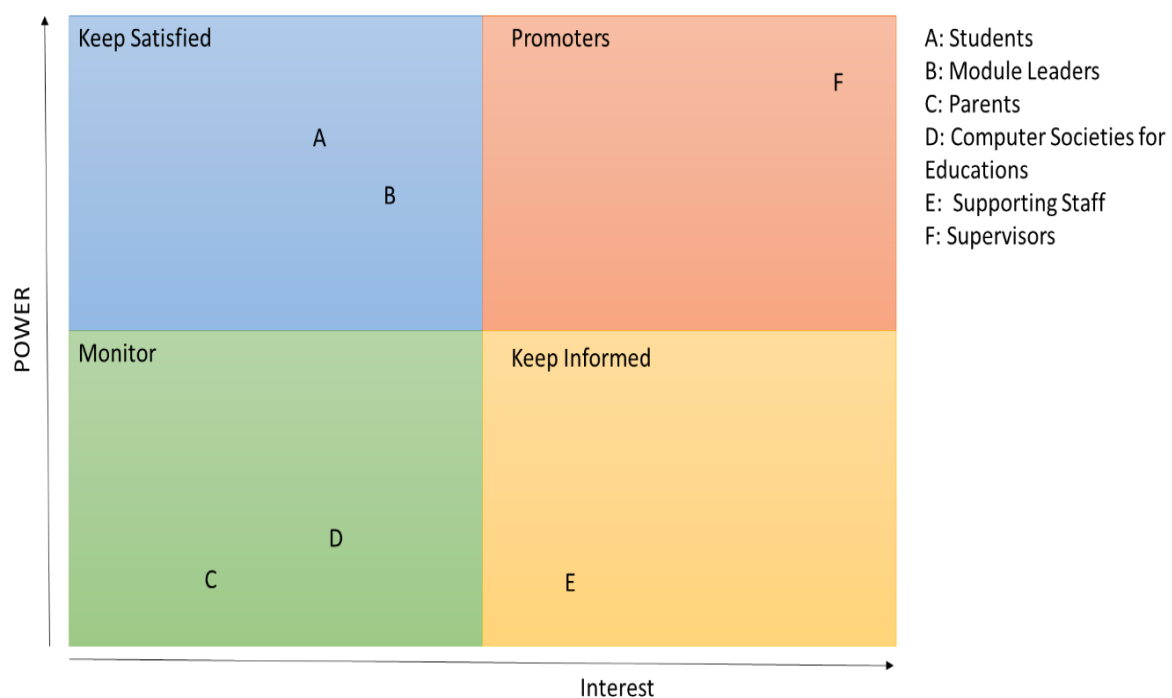


Figure 2 Stakeholder Prioritisation

The stakeholder prioritisation shows a clear view of the stakeholders which will need to be kept closely and informed regularly. Supervisors have the most power and the most interest in the project and should be addressed first. By meeting the needs and wants of the supervisors, while controlling requirements, will help make this project successful. Module leaders and students need to be kept satisfied. This can be done by showing these stakeholders the benefits of the system and why it will make a difference to them. The rest of the stakeholders should be monitors or kept informed. This pleases the stakeholders which helps to ensure the project will be successful.

3.2.4 Conclusion

With the stakeholders now recognised and prioritised, this project has a high chance of being successful. Throughout the next few stages these stakeholders will be kept in mind to ensure they are addressed correctly based on their prioritisation.

3.3 Use Case Analysis

3.3.1 Introduction

The use case analysis demonstrates the functional requirements for each user. The functional requirements show the functionality which each system must be able to achieve. Some of the requirements are extended from other requirements which highlight that the requirement cannot be completed without the previous being implemented first. To gain these user requirements, user stories will be produced first.

3.3.2 User Stories

User stories are important to be written out as it help shows the functionalities which will be needed to achieve different outcomes. Three user stories can be seen below, featuring users which will be using the system the most.

Student

Jamie is now a month into his final year. He is struggling with some of his deliverables and not sure what deliverable to complete next on his final year project and needs some guidance. He remembers his Supervisor put a list of deliverables on the PDMS. Jamie logs in and clicks on Deliverables. The system produces a list of deliverables he agreed on at the beginning of the year. He sees that the wireframes are late. Jamie is confused about this requirement so clicks on 'Messages'. Jamie fills out the message input box and clicks send. He is worried that he might not get a reply in time so looks on the system to see when the next meeting is. Jamie logs off and goes to his next lecture.

Supervisor

Ben has been working at Kingston University for many years now and struggles to keep track of his students' progress that he is supervising in their final year projects. Ben decides to have a quick check and clicks on a report. This brings up multiple reports including meetings, deliverables progress overview and more. He clicks on deliverables progress overview. The system loads up a graph displaying how all the students are performing. He sees Jamie has fallen behind with some of his deliverables and needs to be alerted. Ben is happy to see the rest of the students are doing well. Ben creates a new meeting which the system records into the database. Ben invites Jamie to the meeting and types a message explaining why. Ben logs off and waits for Jamie to accept the invite.

Module Leader

Roy is the module leader on the final year project module. At the beginning of the year, Roy needs to set the student and set them a supervisor. Roy adds the student onto the system and then select which supervisor the student will have. Roy works he way through the list.

3.3.3 Use Case Diagram

A use case diagram has been created. This displays the actors and the goals they wish to achieve when using the system. These have been created from the stakeholder's analyses and the user stories.

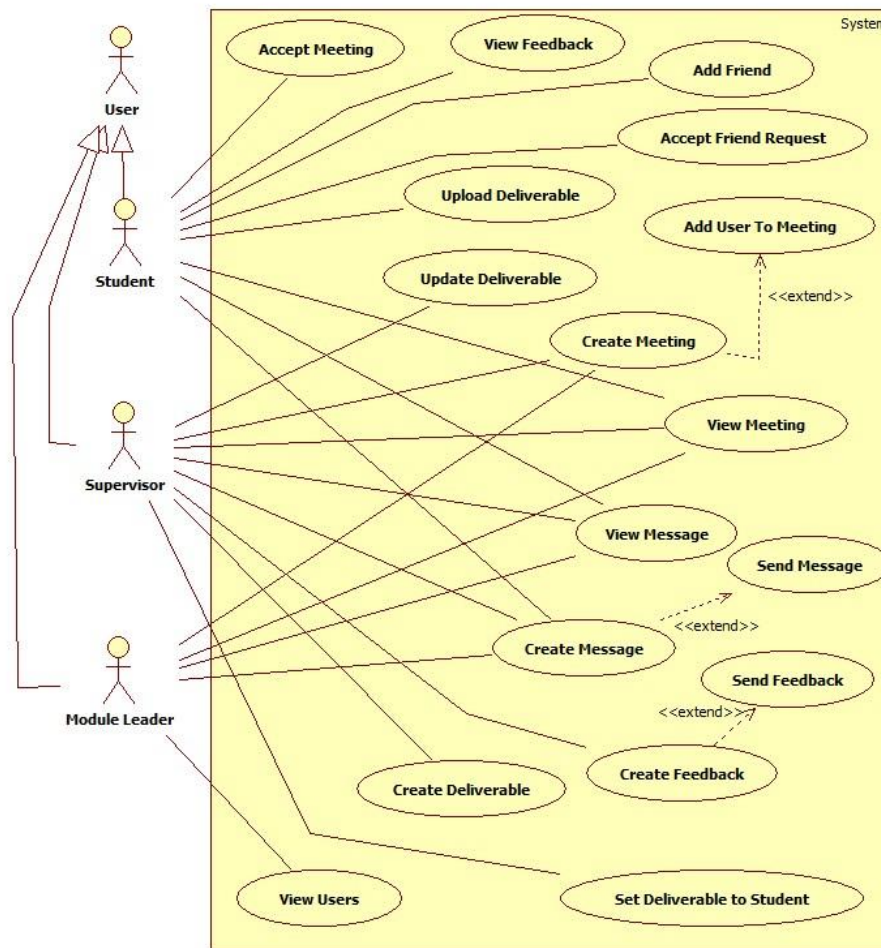


Figure 3 Use Case Diagram

As can be seen from the diagram, the requirements are required from some of the same actors, while some are for an individual user. These requirements give a great overview of the functionality which is required for the system to do. It highlights the difference requirements between the users and the system needs to ensure this is kept when being developed.

3.3.4 Use Cases

Use cases have been produced to give an in depth and clear description of each functional requirement in the use case diagram.

The use cases are broken up into titles which provide a part of the deliverable. Below each row is broken down to show the row title and a description to why this plays apart in the table.

- Use Case Name
 - A clear name of the deliverable to give a direct understanding of what the deliverable is.
- Use Case ID

- A unique identifier.
- Description
 - A small, clear description highlighting what the functionality is.
- Level
 - The level states how important the functionality is.
 - A High-Level Goal is an important functionality which must be completed. The system will not meet the user needs or work without it being implemented.
 - A Low-Level Goal is a functionality which is required by the system but is not vital to the project succeeding.
- Primary Actor
 - This is who will be requiring the functionality of the use case.
- Post-Condition
 - Which pre-actions the actor did below the having access to the user case.
- Main Success Scenario
 - This is the process the actor takes to successfully complete the use case functionality.
- Alternative
 - Replacement steps to still successfully complete the scenario on certain steps.
- Exceptional Flows
 - A response the system will give if the main success scenario is failed to be completed on a certain step.

Each of these plays an important part in each use case to help define the functionality of the system and the success outcome of the user. An example of two use cases is provided.

Create Deliverable

The purpose of this use case is to give the primary actor the functionality to produce a new deliverable to be completed by the students for their final year project. This will give the primary actor the functionality to create deliverables which must be completed by the students they wish to set this too. This will be a powerful functionality which will allow the supervisor to have an excellent overview of the work each student has been set and completed for their Final Year Project.

Use Case Text 1		
Use Case Name		Create Deliverable
Use Case ID		1001
Description		Insert a new deliverable into the system.
Level		High-Level Goal
Primary Actor		Supervisor
Post-Condition		User successfully logs in to their account
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Deliverables	System loads a new user page
2	User clicks on ‘Create New Deliverable’	System loads a new page containing a form
3	User enters fills in the required form	
4	User clicks on save	System creates a new deliverable and takes the user back to the deliverable screen
Alternative		
Steps	User Interaction	System Response
2a	User clicks on a drop down from the navbar and selects ‘Create New Deliverable’	System Loads the form for the required form in a new view
Exceptional Flows		
Steps	User Interaction	System Response
4a	User doesn’t fill in the required inputs	System highlights the required form and does not create the new deliverable

Figure 4 Use Case Text 1

This use case provides details on the primary actor creating a new deliverable for the students to complete on the system. The main success scenario provides four steps on how this can be completed in a straight forward way. There shouldn't be much confusion on completing this goal but if the user forgets to fill in the required fields, the system will highlight these fields and won't let the user progress to the next step. This use case gives all this information and more to make the deliverable clear to understand and execute by any developers with skills in this area.

Create Meeting and Invite

The goal of the primary actor is to create a new meeting and invite the relevant users. This will be mainly used by supervisors, setting a time and location, to invite their students to a meeting. This will be a key feature as students are meant to be meeting up with their supervisors every week. This will ensure both user types understand the agreed location and time.

Use Case Text 10		
Use Case Name		Create Meeting and Invite
Use Case ID		4002
Description		To create a new meeting and send an invite to a user
Level		High-Level Goal
Primary Actor		Supervisor/ Module Leader
Post-Condition		User successfully logs in to their account and clicks on meetings
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on 'New Meeting'	Loads new page for the user to view with required fields
2	User fills out the required form	
3	Selects the user they wish to send the meeting too	User adds user id to the form
4	User clicks 'Save	The system creates new meeting and links to the correct user(s). Returns user to Meetings view
Exceptional Flows		
Steps	User Interaction	System Response
4a	Not all Required fields are filled in.	System highlights forms required to fill in.

Figure 5 Use Case Text 10

The goal is high level because it is essential that this functionality is in place to comply with the recommended amount of times a supervisor and student should be meeting up every week. Once again, the exceptional flows are if the actor forgets to fill in one of the required forms. To stop null values going into a non-null cell in the database, the system will have checks in place on the front-end and back-end. This use case covers the main success scenario which should be simple to complete because of the simplicity of the steps to achieve the goal.

3.3.5 Summary of use cases

Table 3.3 shows the Case ID, Description, and Actor of all the use cases listed in Appendix A.

Case ID	Description	Level	Actor
1001	Insert a new deliverable into the system.	High-Level Goal	Supervisor
1002	Select a deliverable and add a student.	High-Level Goal	Supervisor
1003	Update an already made Deliverable	High-Level Goal	Supervisor
1004	Upload a completed deliverable	High-Level Goal	Student
2001	Add feedback to a student deliverable	High-Level Goal	Supervisor
2002	To look at the user feedback on deliverables	High-Level Goal	Student
3001	To look at messages sent to the user	High-Level Goal	Supervisor/ Student
3002	To create a new message to the user	High-Level Goal	Supervisor/ Student
4001	To look at the user meetings	High-Level Goal	Supervisor/ Student
4002	To create a new meeting and send an invite to a user	High-Level Goal	Supervisor
4003	To accept user invite to a meeting	High-Level Goal	Supervisor
5000	User selects another user as a friend to follow	Low-Level Goal	Student
5001	User accepts a friend request from another fellow student	Low-Level Goal	Student
6001	Show all users on the system linked to the module leader	Low-Level Goal	Module Leader
6002	Create a new user and link to a supervisor	High-Level Goal	Module Leader

Figure 6 User Requirements

3.4 Requirements

3.4.1 Introduction

The functional and non-functional requirements for all users have been listed out. Some of the requirements will be shared between the users as these users may need the same functionality. Non-functional requirements are gathered from deciding on what the users and stakeholders would require from the system which is not functional. These are requirements which the system must provide to create a good user experience but also a trustworthy application.

3.4.2 Functional Requirements

1. Log in
2. Log out
3. Change Password
4. Add new deliverable
5. Edit new deliverable
6. View deliverable
7. Create feedback
8. Read Feedback
9. View Message
10. Create Message and Send
11. View Meeting
12. Create Meeting and Invite
13. Accept Meeting
14. View All Users
15. View Attendance
16. Update Attendance
17. Add Friend
18. Accept Friend Request
19. Upload Deliverable
20. Delete Deliverable

3.4.3 Non-Functional Requirements

- Performance
 - Browser Compatibility
 - Handle many Users
 - Large amount of data
 - Quick response speed
- Security
 - Password Encrypted
 - Stop SQL Injection
 - User Privileges
- Usability
 - User actions
 - User training
 - Effective simple layout
 - Response speed
 - Responsive design

3.5 Moscow

3.5.1 Introduction

To ensure the system requirements are prioritised and the most important tasks are achieved first, the Moscow technique will be used. The Moscow technique is an priority system which can be used for each requirement. This will enable the important requirements to be achieved first. There are four types of priority used in the Moscow technique:

- Must
 - These are the requirements which must be achieved for the system and project to be successful
- Should
 - These requirements are not required for the system to be successful but should be implementing after the Must requirements have been completed.
- Could
 - These requirements will benefit and improve the experience and workflow of the system. They are not vital but would complement the 'Must' and 'Should' requirements.
- Won't
 - These requirements will not be in the final implementation but may be useful to be used in the future.

The list of the functionality in a Moscow format can be seen below.

3.5.2 Functional Requirements with Moscow

Functional Requirements defines the functions of the system which needs to be implemented. Some of these functions will have a higher priority than others for the system to be a success for the users. To ensure the most important requirements are implemented first, the Moscow technique will be used. The functional requirements are as shown below.

Requirement	Description	Priority
Log in	Users can log into the system using their User Id and Password. User Id and Password is stored in the Database	MUST
Log out	The user can log out of the system at any time. The system will make sure the session is destroyed.	MUST
Change Password	The user must be able to change their password from the system.	MUST
Add new deliverable	User must be able to add a new deliverable	MUST
Edit deliverable	User must be able to edit a deliverable	MUST
View deliverable	User must be able to view their deliverable(s)	MUST
Create Feedback	User type 'Supervisor' can create feedback to send to a student deliverable.	MUST
View Feedback	User can look at their feedback they have received/ sent	MUST
View Message	User will be able to view their messages they have received through the system	MUST
Create Message and Send	User can create a new message and send it to the correct user	MUST
View Meeting	User can see their meetings arranged coming up in an ordered table	MUST
Create Meeting	The user can create a new meeting and invite the correct	MUST

and Invite	user(s).	
Accept Meeting	User will be able to accept or decline meetings that have been sent to them	MUST
View All Users	User Type Module Leader can view all users on their system	MUST
View Attendance	User can view their attendance to meetings	MUST
Update Attendance	User Type 'Supervisor' can update the attendance of meetings	MUST
Add Friend	User type 'Student' can add a friend so they can see the progress of the student	SHOULD
Accept Friend Request	User type 'Student' can accept friend request sent to them	SHOULD
Upload Deliverable	User type 'Student' can upload their deliverable to their 'Supervisor'	SHOULD
Delete Deliverable	User type 'Supervisor' can remove a deliverable if it is not set to any students (not being used in the system).	SHOULD
Email Alert	User to receive an email when something has changed on the system. Example being a new deliverable or message.	COULD

3.5.3 Non-Functional Requirements with Moscow

Non-functional Requirements is criteria the system should meet to be successful. These are different to functional requirements because the non-functional requirements are the behavior the system should show.

Once again I have priorities these and the non-functional requirements are shown below.

Type	Requirement	Description	Priority
Performance	Browser Compatibility	The system needs to work across at least 2 main browsers (Firefox and Chrome or Edge)	MUST
Performance	Handle many Users	The system must be able to achieve high performance with many people on the system working with the server	MUST
Performance	Large amount of data	System will be able to deal with large amount of data efficiently	MUST
Security	Password Encrypted	The database must encrypt passwords	MUST
Security	Code Security	The code across the system must be secure in ensuring the data is being transferred and used correctly.	MUST
Security	Stop SQL Injection	Make sure the SQL is secure and to put the code in place to stop SQL Injection.	MUST
Security	User Privileges	They system must give the users to correct access to certain data.	MUST
Usability	User actions	To handle the user actions efficiently to ensure a good user experience	MUST
Usability	Train users	User should be taught how to use the system be one on one or through a help video	COULD
Usability	System Support	The system will support the users in getting around the system and achieving their goal	MUST
Usability	Speed	The system will perform actions to an acceptable speed without having to make the user wait longer than expected	MUST

3.6 Conclusion

The requirements have been identified which will be delivered for the deadline of the project. These requirements have been prioritised to ensure the most important requirements are delivered first, giving the users the most essential functionality to meet their goals.

4 Design Stage

4.1 Introduction

The design chapter will discuss the design plans of the application based on finding previously. The design will focus on meeting the needs of the users and ensuring the database has a strong set up to support these needs and requirements. As well as ensuring the back end has a strong design, this chapter will also show the design decisions made on the front end. The front end is what the users will see and interact with. The user experience needs to be to a high standard and this will be met by producing wireframes on design decisions discussed throughout this chapter. The design chapter will discuss and include the following in this chapter:

- Database design
- Data Dictionary
- User Interface design
- Activity diagrams
- Wireframes
- Logo

4.2 Database Design

4.2.1 Introduction

The database is an essential part of developing this application. The application must be able to contain information about its users and data relevant for this application to successfully work. Universal Modelling Language (UML) is an excellent way to design the database to ensure the database meets the requirements and needs. This is because it allows the database to go through different stages of design and forms to remove issues.

There are different database forms which show which state the database is in. They are known as first, second and third normal form. To achieve each form, the form before has to have been met. These forms can be confusing so a simple definition of each has been included.

First form is when there is no repeating data in the database. This is completed by splitting up data in the table so any tables which need to reference the same data can point to one table by a unique foreign key.

Second normal form is when all non-key attributes are fully functional on the primary key of the table.

Third normal Form is when all columns rely upon another column within the table.

Excellent databases are looked upon as they should be in Third normal form. While this is mostly true, third normal form can cause the database to be slower because it requires the queries to join more tables together to retrieve the data. Many developers decide to keep their databases in Second normal form to keep the transactions to and from the database as quick as possible. When developing the database for this application, it was aimed to design and create a database which was quick at retrieving and sending the data to and from the application to create an excellent

As can be viewed, the ERD contains seventeen tables which all interlink with each other. The ERD has improved greatly from the first design, which needed many to many relationships resolved and all repeating columns were removed.

As there is a lot going on in this ERD, it has been broken up into three main areas. They are User Types with message, meetings, projects with deliverables. Each part is discussed into detail to show what is being achieved by this design.

Users Types with message

With three different users being able to access the system the database needed to be designed around this. The decision was made to have a user table containing all the fields in which all the other tables will share. This removes reoccurring data and helped move the design into second normal form. The user table will be the first table the application will access. This will handle the login page and get the user the correct permissions based on the user type. The user type entity points the application to the right table. The user login in, can be a student, supervisor or module leader. This structure can be seen below.

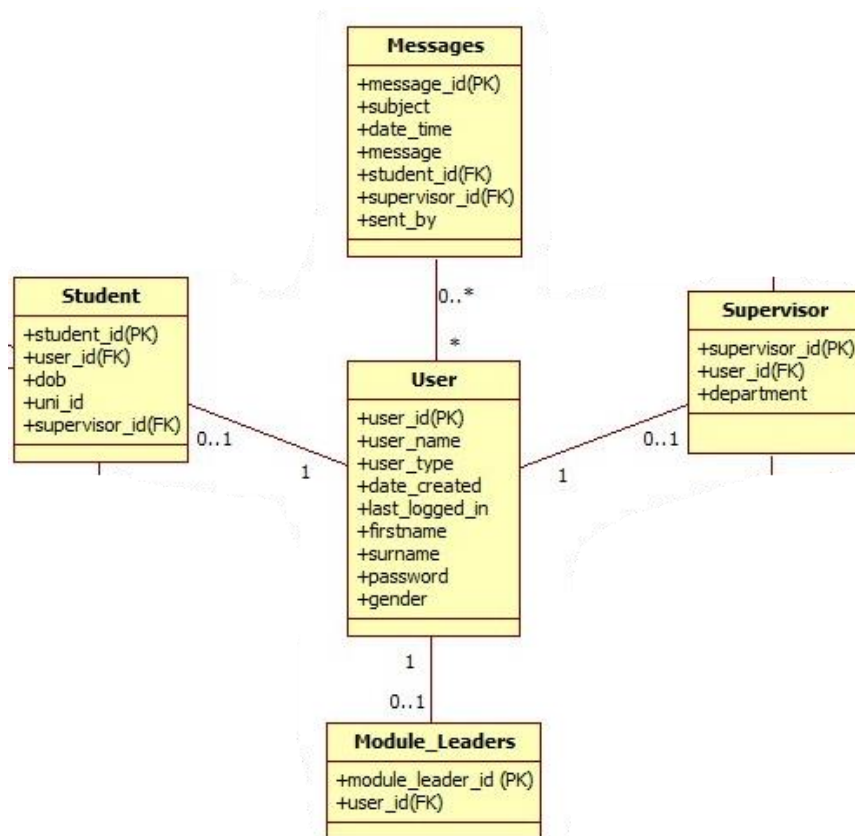


Figure 8 Users Types with message

As can be seen from figure 8 the user has a none to one relationship with student, supervisor and module leaders. This is because the user id can only be one type of user whereas the student, supervisor and modules have a must (1) relationship to user because every type of user, student, supervisor or module leader, must be a user.

Messages can also be seen in this diagram. Messages provides a table which allows the different user types to interact with each other by sending messages. This may be away for the supervisors to chase up students or the other way around. The relationship from the user types to messages is

none to many and messages to the user types is none to one. This shows that all users may or may not send any messages but can send as many as the like in the system where as every message must contain a student id and supervisor id. This is enforced by foreign keys.

The decision was made that module leaders will have an overview of the system by being able to access different user accounts to see how they are progressing and because of this the messaging system will be left to be used between supervisors and students for now.

Meetings

Meetings is another section of the design. Supervisors are required to make meetings with their final year students. As can be seen there is a main meeting table called 'meetings' which holds the main bulk of data. As many students of supervisor could attend the same meeting, two new tables were introduced called Student_Meetings and Supervisor_Meetings. This design allows many of the same user types to attend the same meeting by containing foreign keys student_id and meeting_id. Each meeting will need a location of where it is being held too. As there are only so many locations around the university, these are stored in a table called location. Location has a relationship with buildings where each building is stored with an ID to the location.

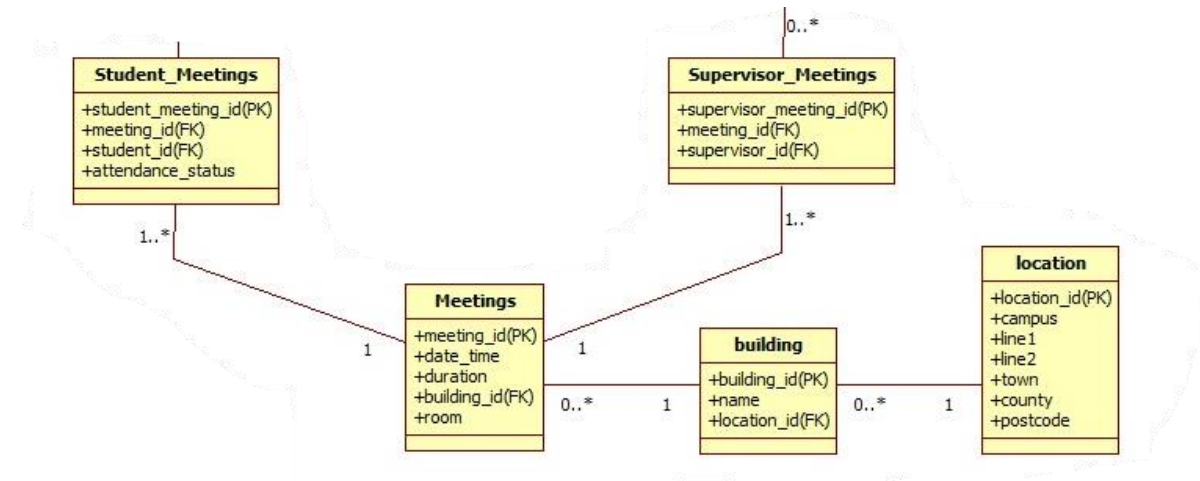


Figure 9 Meetings

The Meeting table has a one to many relationship with Students_Meetings and Supervisor_Meeting while these tables must have a meeting stored in the meeting table. This is to ensure all meetings have one to many students and supervisors attending the meeting.

Project with Deliverables

Deliverables is a big part of the functionality in the system and the database design needed to cover this. This is a similar layout to the last two where the data about the deliverable is stored within the Deliverable table. This gives the supervisor the functionality to be able to set the deliverable to many students.

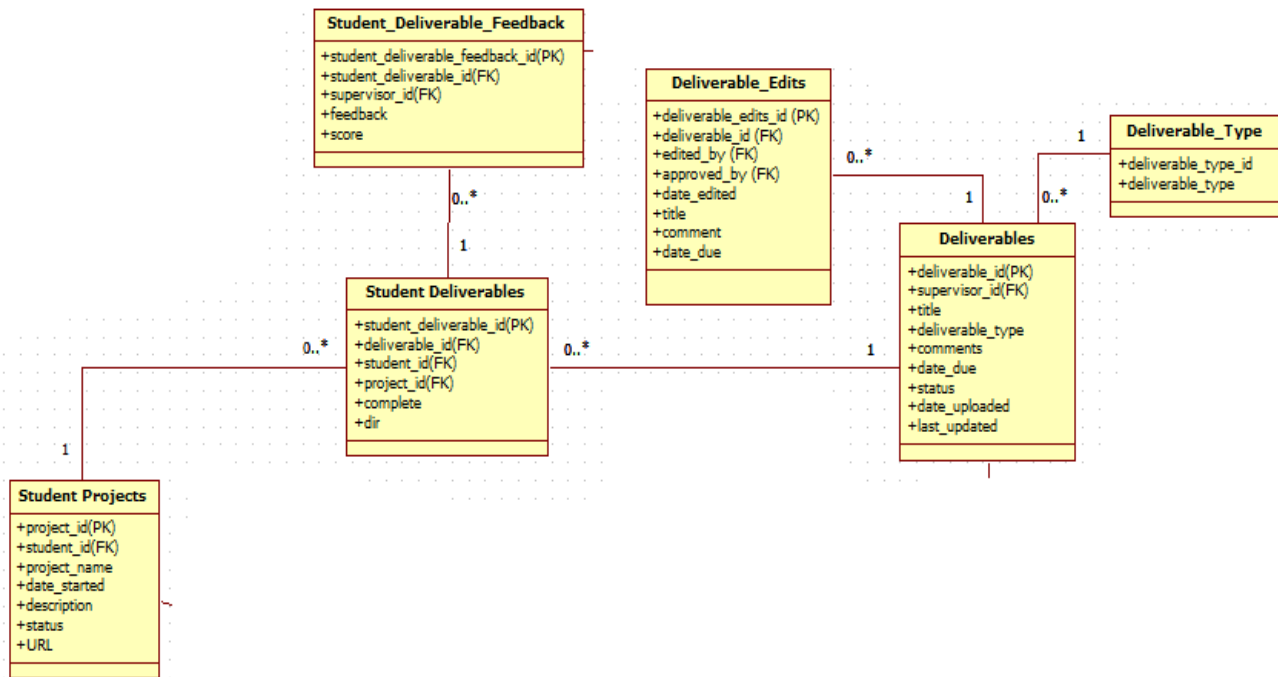


Figure 10 Project with Deliverables

Every student will have a project which will be set deliverables from the supervisor. These deliverables may be set to a student project which would be stored in the Student Deliverables table. Once they are completed and submitted to the supervisor, they can give feedback to the student. Some deliverables may be edited which is set up for the idea that the application could alert the set students so they can see the edited and original version.

4.2.3 Data Dictionary

The Data dictionary contains information about each table in the database. The data contains allows the developers to easily create each table as all the information is ready and available to use.

The Data Dictionary contains the following columns:

- Relation Name
 - The name of the Table
- Attribute Name
 - The name of the column
- Data Type
 - How the data/information will be stored in the attribute. This is what the Attribute will expect the type to be.
 - VARCHAR is a string
 - CHAR is a number
 - Tiny Int is 1 or 0 (true or false).
- Length
 - The maximum amount of characters
- Key Type
 - Primary Key or Foreign Key.
- NOT NULL
 - Not Null means that the attribute is not allowed to be empty.
- Other Constraints

- Other rules the attributes will follow.

User Relations

This is the student project relation (table) in the Data Dictionary (displayed below).

Relation Name	Attribute Name	Data Type	Length	Key Type	NOT NULL	Other Constraints
User	User_id	CHAR	10	PK	NOT NULL	
	Date_created	DATE			NOT NULL	
	Last_logged_in	DATE				
	firstname	VARCHAR	255		NOT NULL	
	surname	VARCHAR	255		NOT NULL	
	password	VARCHAR	255		NOT NULL	Must be encrypted

The relation has six attributes. This relation contains information about the user stored in the system. This is where the log in page will check to see if the user belongs to the system already. With the user id, the system will be able to look in the student and supervisor table to see which type the user is. Both these relations can be seen below. As all these fields are important to the user, they are mostly NOT NULL which means an empty value cannot be inserted into these attributes. This is to keep the integrity of the relation.

Student and Supervisor Relations

Relation Name	Attribute Name	Data Type	Length	Key Type	NOT NULL	Other Constraints
Student	Student_id	CHAR	10	PK	NOT NULL	
	User_id	CHAR	10	FK	NOT NULL	
	dob	DATE			NOT NULL	
	Supervisor_id	CHAR	10		NOT NULL	
Supervisor	Supervisor_id	CHAR	10	PK	NOT NULL	
	User_id	CHAR	10	FK	NOT NULL	
	department	VARCHAR	255			

These relations are both extended from the user table. With the User Id being the foreign key, each row can easily be matched to the row it belongs to in the user relation. As you can see, they have been split up because they contain slightly different information about both users, with students having a DOB and supervisor Id and the supervisors have a department. They have been split from the User table to make sure no NULL values are having to be put into each row as a student does not have a department.

4.4 User Interface Design

4.4.1 Introduction

The user interface is designed to meet the requirements of the project and be accessible to its users. This covers the platform on which it shall be used, user experience and ensuring the website is accessible to users with accessible issues like vision impairment.

4.4.2 Accessibility

It is law to ensure your applications on the web is accessible to everyone including people seen with disabilities. This is a challenge many web developers have when developing websites and sadly this is often not considered by many. A table has been created to show different accessibility issues users may have and how the application will help deal with these issues.

Type of Accessibility	Description	Website functionality to help
Vision impairment (Colour)	User is unable or struggles to see the difference between certain colours.	Different shades of the colour blue have been proven to help users see the difference between each one. There shall be a functionality which colour codes certain areas of the websites where different colour scales are important to ensure they have no issues because of vision impairment.
Vision impairment (Sight)	User struggles to see small text.	User will be able to increase the text size. Small text will be avoided.
Dyslexic	User struggles to read and write.	The system will keep to the HTML guidelines and include HTML alt tags which will allow screen readers to read out the text.
Epilepsy	Flashing white lights are harmful to the user which can lead to serious problems.	No flashing lights will be used in this application.

By covering these accessibility issues and ensuring the correct functionality is in place and with the specialise software on the web, it will enable all users to part take in this application. This will increase the user experience with the application and provide a great experience overall.

4.5 Activity Diagrams

4.5.1 Introduction

The Activity Diagram shows how the flow of the system will work. The Activity Diagrams shows the actions the use can take to complete their goal but also highlights where the system will go back on the flow if something is not completed correctly. The Activity Diagram gives the developer a clear direction of how the system works.

4.5.2 Activity Diagram

The activity diagram has key shapes which show different actions by the user or system.

- Oval shows an 'Action'
- Diamond is a 'Decision' (if)
- Circle filled in is the 'Start'
- Circle filled with an outer circle is the 'Finish'
- A solid rectangle is an 'OR' which shows two different paths to achieve the same outcome.

All the Activity Diagrams can be viewed in the Appendix's.

View and Send Message

The system will allow users to view messages. The user may want to reply to the message or create a new message. The diagram below shows how this can be achieved and the progress which needs to be followed.

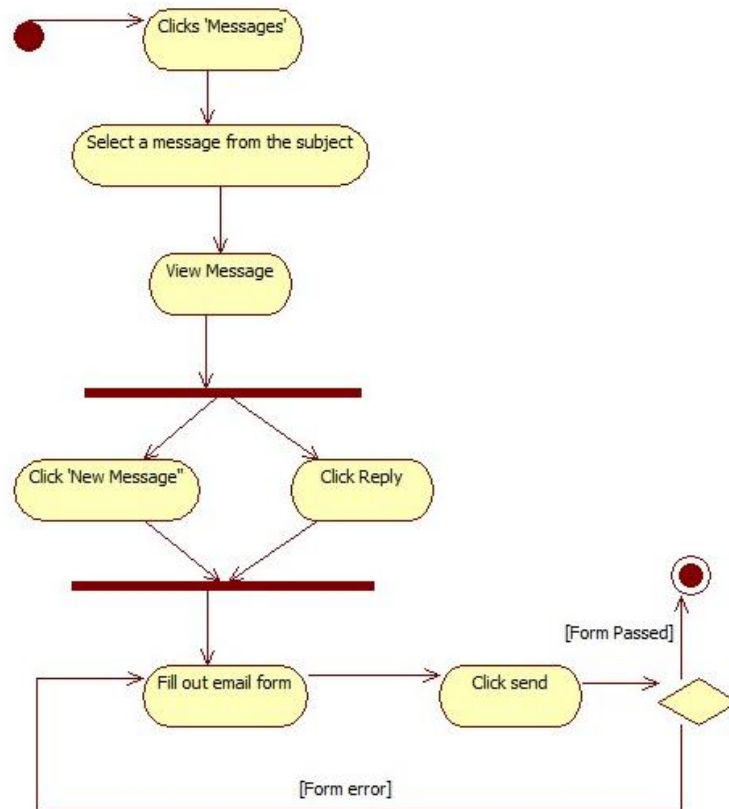


Figure 11 Activity Diagram 1

This is an interesting activity diagram because the user has the option to pick two different actions to achieve the same outcome of replying to a message. They can select 'New Message' or 'Reply'. Reply will fill out part of the form for them saving the user time. The system has a decision to make at the end of the process (if the data is correct or not) below allowing the goal to be met.

Log in, User Permission and Change Password

This Activity diagram shows the process of a user logging in and receiving the right user permission from the system. The user then goes on to change their password.

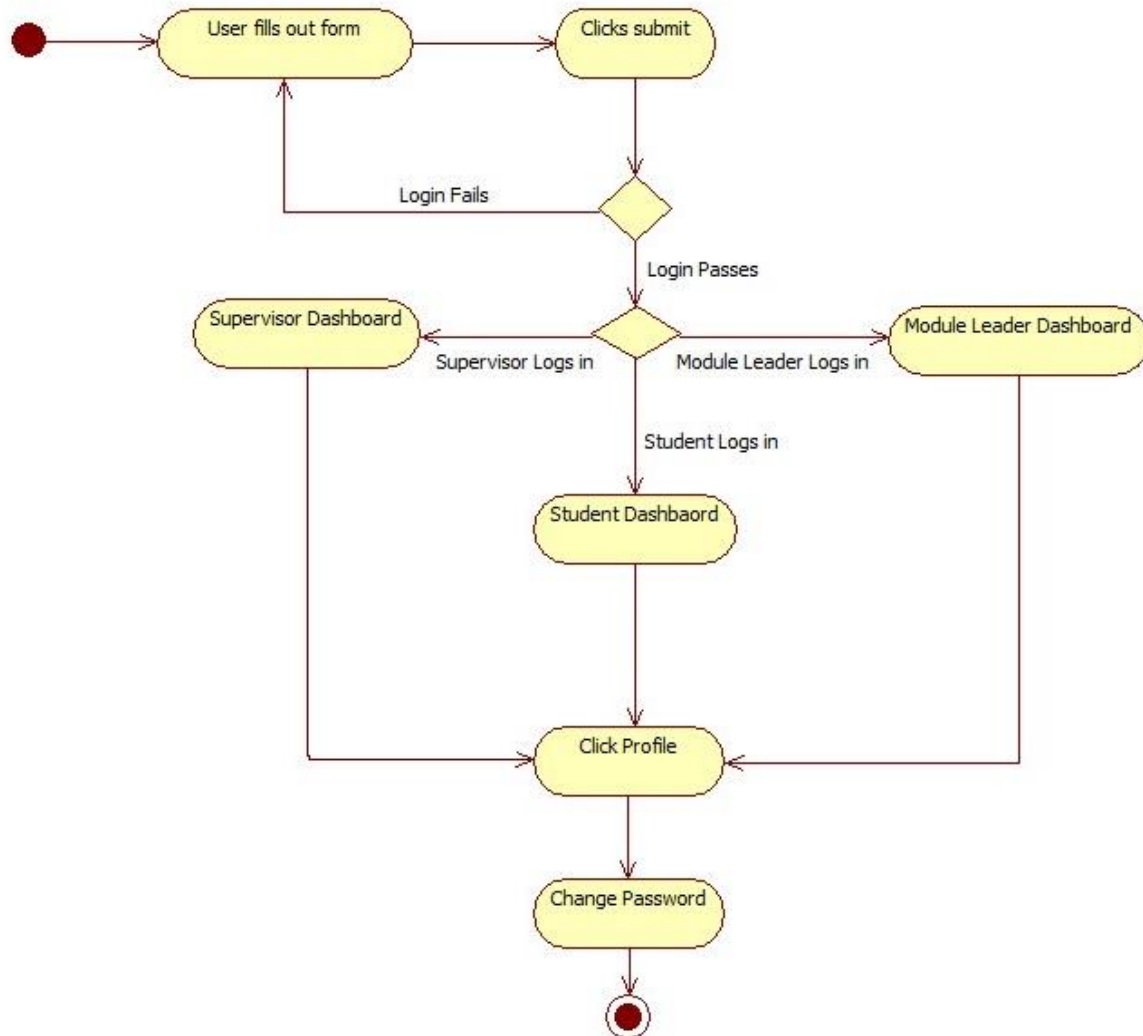


Figure 12 Activity Diagram 2

It is important that the user gets the correct access to the system depending on what user type the user is and this diagram demonstrates this. It shows that three different dashboards can be shown depending on which user type logs in. From this, each user will be able to take the same actions to change their password.

4.6 Wireframes

4.6.1 Introduction

After drawing out sketches for the layout of the webpages in this application, they have been developed to produce wireframes clearly showing the structure of each page. Each wireframe has been produced with the technologies in mind. The aim when designing these wireframes were to develop each page mobile first and then enlarge it up to desktop. Because of this, the desktops look clear and simple with extra data shown with a click.

Index Login Page.

The index login page will be the first page the user will be greeted by. The information type in here will be sent to the server to determine which user type the user is and redirected them to the correct location (student, supervisor or module leader index).

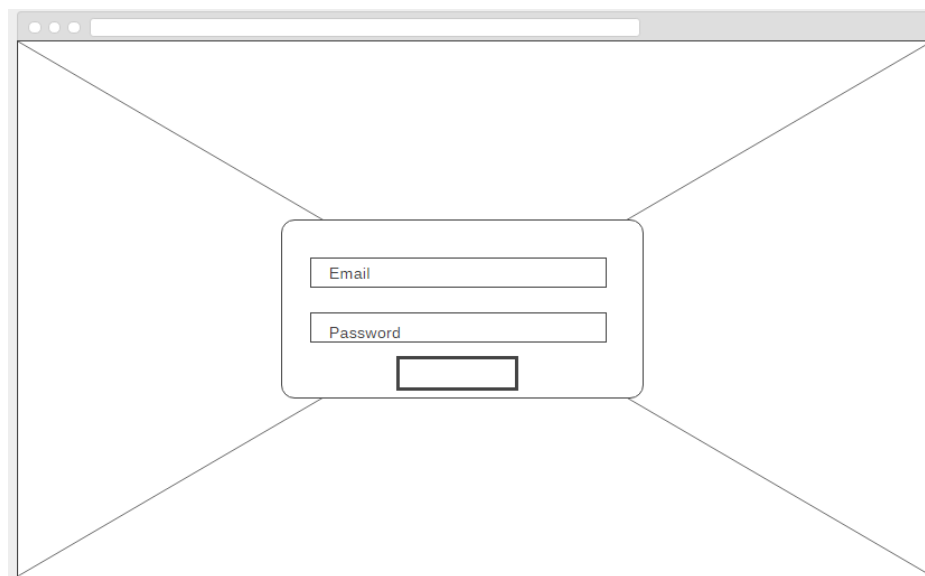


Figure 13 Wireframe Login

The wireframe clearly shows the login form, which the user will need to enter email and password to enter the application. There will be a background image (in a form of a box with an X) to remove the white space behind the form. This will be a simple but attractive image that will look inviting to the user. The user experience has been thought of when designing this wireframe, as there is only one way to progress to the next screen. This removes any confusion from the users, keep the application simple and less time consuming to the user.

Supervisor Dashboard

Supervisor Dashboard is an important page. The Supervisor Dashboard will give the user a complete overview on how their students are succeeding or failing. This wireframe was designed to address this, providing a clear view to the user looking at the information.

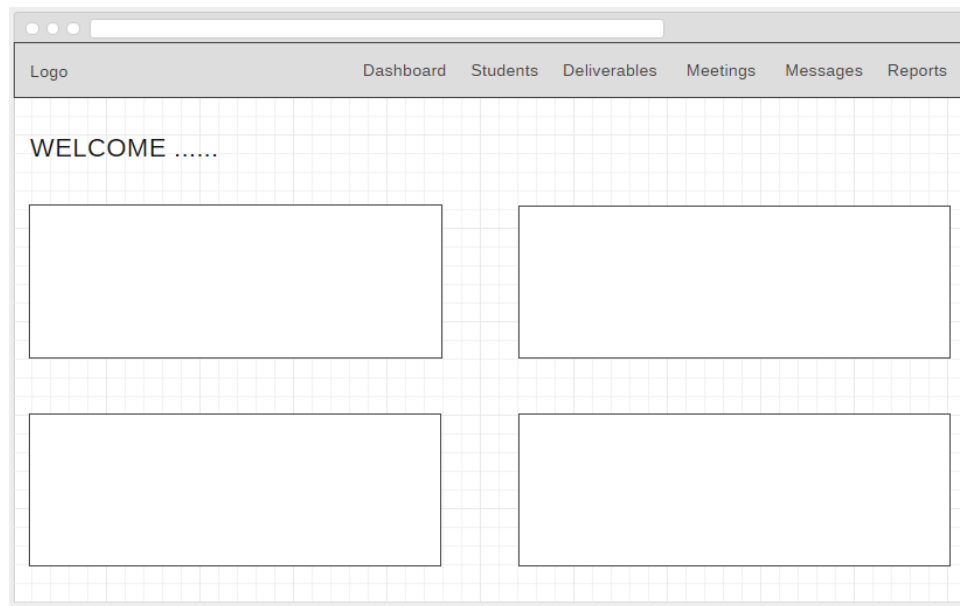


Figure 14 Wireframe Supervisor Dashboard

This screen will be broken up into four sections each displaying different information about different parts of the system. The dashboard will show due deliverables, upcoming meetings, messages and students falling behind. This will be a very powerful screen to quickly highlight any upcoming issues or reminders without the user having to scroll or click a button to get there. This should give the supervisor users a clear view and direct information they are looking for to interact with their students. In mobile these blocks will appear on top of each, putting the most important ones towards the top of the screen, appearing before the fold.

Supervisor Students

Another powerful screen in the Supervisor system is the students page. This page quickly shows all the students they are reasonable for in a clear and simple layout.

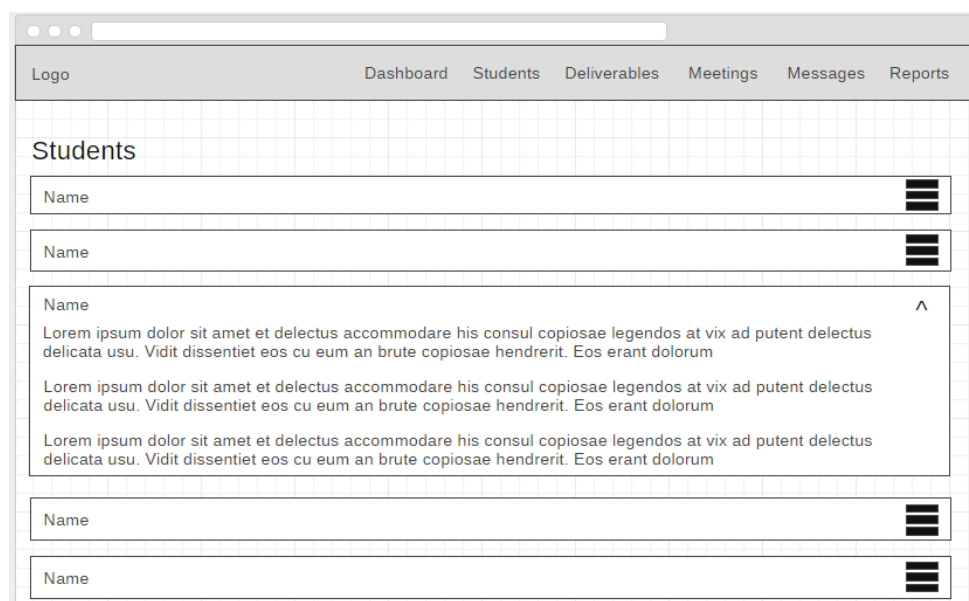


Figure 15 Wireframe Supervisor Students

The user will be able to click on the student names to open the box to show all the information about the students. A lot of the screen are designed in this way to provide a simple, attractive design by not overloading the user with lots of information everywhere. When the screen size reduces to mobile, it will allow the name and slider bar to come closer together, keeping the same feel about the application on any device it is being used on (Desktop, tablet, mobile).

4.7 Icon/ Logo

Designing the icon has become an important design on the application. The icon gives instant creditability and personal touch between the user and the product. The icon allows users to know they are at the right place and they slowly gain trust in the brand or in this case, the applications.

After doing research and speaking with graphical designers, the modern icons are a simple basic but clear image. Facebook and Twitter are good examples of this. They are simply one letter or icon but with a bold present

that as soon as you see it, you know which company it is related too.

Another good example of a business changing their icon to stay modern is Gumtree. They have recently changed their icon from having the word Gumtree to a simple tree which is bold and unique. Still, a lot of businesses have still got their name there like Microsoft or Intel as this is the logo they are recognised for.

With this on mind, of both cases, two different logos to represent the business was put together. It is clear to the user who they are with.



Icon Designs



Design one and two both have the same sun as the background. Sun is normally associated with good times (holidays) and being happy and successful. A simple sun in the background of the icons is eye-catching. The first one is better at achieving its aim, to be recognised as it contains the first letters from the application name. This would suit much better sitting on the website within the website.

The third icon was supposed to represent each letter from the application. Blue being the 'P', yellow being the 'D' and red being the 'M' and the 'S' within the shape. While this was a good idea, the outcome didn't succeed to expectation and it would look out of place on the system.

The final is very simple but effective for the prototype and shows the users what system they are using straight away. This is instantly recognisable and fits the application design well.

For this project, an artistic design will not look right on the application and do not fit the feel of the application very well. This application is here to support the students and give the supervisor the tools and support to make a difference so the icon should stand clear of what they are using. Therefore 'PDMS' should be displayed clear at the very top of each page in the nav. Either the first or

last icon (both highlighted by a blue ring) will be brought forward to the application, with the final decision will be made when the application is being build and both can be tested.

Chapter 5 Implementation Chapter

5.1 Introduction

The implementation chapter will show the use of libraries and tools used in this application, explaining the structure, choice of methodologies and highlighting important aspects of the code. There will be a discussion on how the file structure has changed as my knowledge of design principles are expanded. After these have been explained, this chapter will then show the technical challenges with this project when implementing and developing.

5.2 Tools and Technologies

The decision to build this application on web technologies gave the project a lot of freedom to pick technologies that suited the application best. The programming languages which were used were jQuery, PHP, SQL, JSON, CSS using the bootstrap framework and HTML 5. These different languages gave the application the power it required to make a modern web application. The decision to uses these technologies are discussed below.

jQuery

jQuery is a popular library which allows the user to produce JavaScript with less code which can results in excellent results quickly. JavaScript by itself can be time-consuming and slow to achieve the same outcome so jQuery to overcome this. JQuery also provides a simple AJAX call function which is used frequently in this application.

PHP

The application required a technology which could communicate with the server to pass requests and data. PHP is a famous technology for doing this and has an excellent support online. While Python was considered, PHP provided personal benefits over using it. With experience and understanding the basics of PHP, it allows the application to progress at a better rate with a smaller learning curve instead of using Python.

The decision was made to switch from using PHP SQLI to PHP PDO with using an MVC model. PDO had never been used before but it provides a good set of tools to produce object orientated code as well as providing straight forward SQL function to stop SQL injection. SQL injection is when the query is altered to perform an action it wasn't intended to by the form forms being purposely containing a string which is shaped like a query. PDO stops this without the need for extra functions when using SQLI. This was a great learning curve and one which benefited the application to have good results.

My SQL

The decision to use MySQL instead of Oracle was due to the costing factor. For this prototype, MySQL provides an excellent database system to store all the records which are required. It is installed on most c-panel and provides a great collection of tools to set up the database and adding constraints.

AJAX/JSON

Ajax will return JSON to for instant responses to the actions the user performs on the page without the need to refresh the whole page. This is a powerful technology with allows the application to interact with the server, getting a response back in JavaScript to then update the current web page straight away. This technology has become the standard for any modern web application and one

which will be used a lot to provide a powerful application to the user in the model which has been used.

Bootstrap 3/ CSS

CSS is a powerful programming language to give the application a unique look and feel. The CSS provides a style sheet to style the HTML to what is required.

As CSS takes a lot of code to style everything to exactly the way which is required can be very time consuming so to overcome this, the Bootstrap framework is used. Bootstrap provides a very quick way to add style to your web page with lots of already defined CSS. This is a popular framework with lots of companies using it to quickly build a website and application for their business. This framework will allow me to focus on the functionality of the application and less time on defining styles, which allows the application to be quickly build to the wireframes and the functionality and testing to be performed at its best for the final prototype.

HTML 5

HTML 5 is the backbone of all web pages. This technology provides the structure of each page which all browsers understand. HTML 5 will be used as it is the most up to date HTML version providing new features like being able to embed a video straight into the document and being able to use local storage.

Database

The database is an essential part of the project which will hold the data in a structure of the application. The database was designed using UML to develop the tables and attributes. After many implementations of the design, the final one was developed using MySQL. MySQL was chosen over Oracle because MySQL provides all the features and tools required and is open source. Open Source means this tool is free to use and is popular with many developers.

Host

The hosting for the application has been a bigger issue than it should have been. Originally using Smart Hosting, they had excellent reviews and 99% up time. They looked like a brilliant option to host this application but as time went on their service starting to fall below expectations where the site would be down for 24 hours, not even enabling myself to FTP. Because of this, a new hosting company needed to be considered. It was important this project was hosted on two hosting sites so if one fails, the other can be used. The decision was made to use GoDaddy and Kingston student net would be the backup for emergencies.

Microsoft Expression/ Notepad++

Both text editors have been used to code the application. Both can provide an FTP connection and upload and download live files. Both have their different advantages when programming but overall Microsoft Expression was used the most as it provided a better FTP system and more advanced interface.

SQL Workbench

SQL Workbench is a basic database FTP. This provided a quick and easy access to the database once set up without having to go log into a C-panel each time. This allowed me to test SQL and see the data being stored as testing took place.

5.3 Architecture/ Structure

5.3.1 Introduction

The architecture and structure of the application are important to get right and planning is vital to program a well-made application. In this section, the structures which were used will be discussed and demonstrate to show the flow of the system and the important changes which were done to improve the structure.

5.3.2 Structure and implementing

The structure of the application code went under a change half way through. This can be seen between the supervisor access and the student access to the system. These are two of the three user's types on the system. The third one is module leaders.

When a user types in the URL, they are greeted with a login page. By the user logging in, they system can give the user the right privileges depending on what user type they are. The application on the server will then set up the user's sessions and directed them to the correct location in the application for their user type. Part of the code can be seen in figure 16

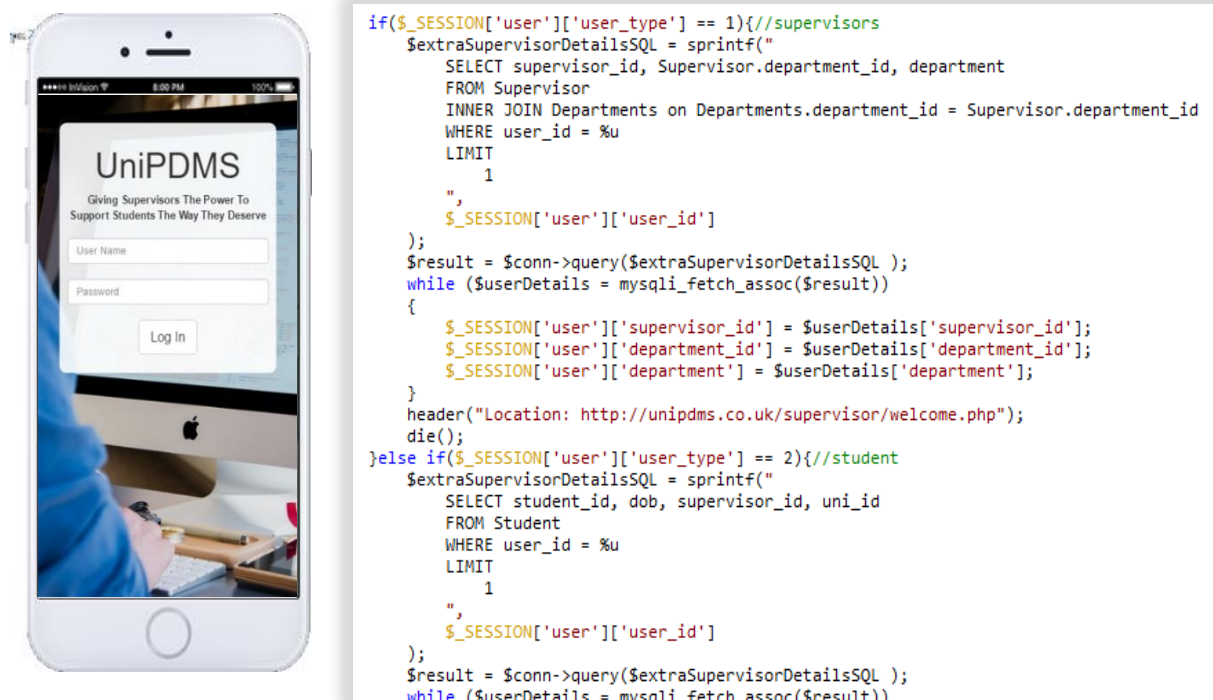


Figure 16 Login Code with Screen

Depending on which user type logs in, the header location will be different. This is where the different structure of code happens. When the system first entered the implementation stage of the project lifecycle, a one-page structure was undertaken.

One page structure is where all the HTML is stored on one page and the data is requested straight away by AJAX and returned from the server in JSON. This data is then used to build the different partial HTML views using jQuery. Implementing the application in this way provides many advantages which makes the user experience more enjoyable. Response time is quick as the user switches between the different views. As all the views are on one page, they are hidden and shown

depending on what the user clicks on. A view of this one-page structure code can be seen in figure 17. The other structure which was then used was MVC. Both these structures are shown and discussed in more detail below as to why this decision was made.

5.3.3 Supervisors Application: One Page Structure

The supervisor side of the application was developed first and because of this, it was built using the one-page structure. With past experience of this structure, this was the first choice of structure to use in the development for this application. The application was designed in mind to use this layout effectively and produce an application which had clean code, was responsive to user actions and using AJAX to provide quick updates to the system without needing to refresh the page.

For the one-page structure to work, all the partial views, scripts, and CSS needed to be put into one file. This file can be seen below in figure 17. Please note that the bootstrap and jQuery files have

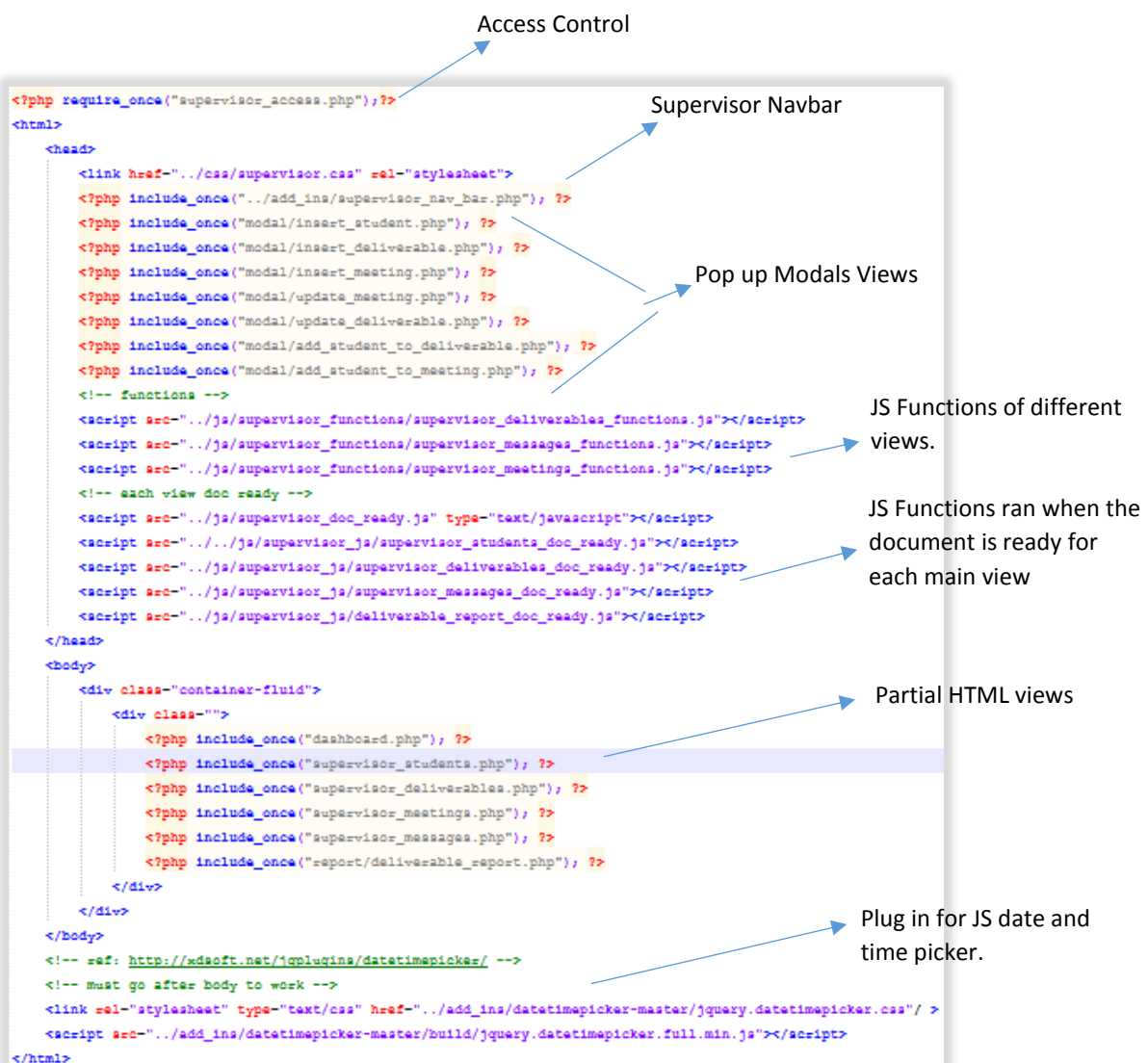


Figure 17 One Page Layout

already been linked to the system by the index page on the login.

This file contains everything to do with this application under the supervisor access to the system. This is ensured that only supervisors can access the system by the access control file at the top of the file. This is done to split up the different views across the system and give better control of the code behind the application.

5.3.4 User type Supervisor implementation example

The following code will demonstrate how this works by giving an example of the process the application works through between the different languages to generate the views and initialising the clicks. Supervisor Students will be used as the example.

First, the supervisor-student partial view is built from the HTML. This is before the JS is ran to allow the structures of each page to be built. The JS is controlled by document ready functions. Document ready functions will wait till the document has been created before running the JavaScript.

The view uses bootstrap to get the layout style from the CSS already created. Each view has an Id to ensure this view can be accessed when required from languages like jQuery to get to an element.



Figure 18 HTML View

As can be seen from the figure 19 the view has been created but contains no data. This is because there are no technologies injecting or editing the DOM just yet. Once the view has been created the jQuery document ready tag will run (figure 20).

```
$(document).ready(function(){
    console.log('supervisor students doc
    student_list_data = ''; //public
    load_student_data();

    //clicks
```

```
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if($_SESSION['user']['user_type'] != 1){
    echo 'access not allowed redirecting.....';
    header("Location: http://unipdms.co.uk");
    die();
}

//print_r($_SESSION);
?>
```

Figure 19 Access Control

Figure 20 Document Ready function

This function sets a console log to display a message that this functions is underway. This is good practice when testing. It then creates a variable which will be public as it is required to be used later in the application. This is set in the load student data function and continue the document ready function at the same time. JavaScript is programmed to run multiple lines at once, unlike PHP which will run each line individually.

The load student data will go find the function which is stored in a different file, and start running it.

```
function load_student_data(){
  console.log('requesting student data');
  $.get("../data_requests/supervisor/get_supervisor_student_data.php", {}, function(data, status){
    data = JSON.parse(data);
    student_list_data = data;
    console.log('student_list_data array is complete');
    var student_table = '';
    var firstName = '';
    var lastName = '';
    var projectTitle = '';
    var dob = '';
    var gender = '';
    var uniId = '';
    var projectDescription = '';
  });
}
```

Figure 21 JavaScript Function

This function starts off by doing a GET request via AJAX to the server. This AJAX call does not have any data sent over with it. This is because the data required is stored in the SESSION of the PHP from when the application logged the user in. The function will then wait for the server to return data before it continues.

```

<?php require_once('.../config.php');?>
<?php
$supervisorId = $_SESSION["user"]["supervisor_id"];
//sql
$sql = sprintf("
    SELECT
        Student.student_id, User.firstname as fn, User.surname as sn, User.gender
    FROM
        Student
    INNER JOIN
        User on (Student.user_id = User.user_id)
    LEFT OUTER JOIN
        Student_Projects on (Student_Projects.student_id = Student.student_id)
    WHERE
        Student.supervisor_id = %u
    ORDER BY
        User.surname
    ",
    $supervisorId
);
//perform query
$result = $conn->query($sql);
//set data inside an array from what was returned
$i = 0;
$rows = array();
while($r = mysqli_fetch_assoc($result)) {
    $rows['student'][$i] = $r;
    //set an english date
    $rows['student'][$i]['dob'] = date("d F Y", strtotime($r['dob']));
    $i++;
}
//return the data back as JSON.
echo json_encode($rows);
?>

```

Figure 22 PHP with SQL statement

The AJAX call points to a file on the server. These files are all PHP which handles the SQL queries and returns the data. There are config files at the top of every page again to ensure only users with the correct privileges can access the file. The SQL is then built in a PHP variable using a sprintf function which helps reduce the chances of SQL injection. This works by displaying key letters with a percent sign in the SQL where values need to be inserted into the query. Two key letters are used throughout this application; %u and %s. %u means only a number can be entered this space and %s means only a string value can be put here. The SQL query is the first permeator of this function and the second permeator is the values to be inserted.

Once the PHP has run on the server, it returns the data in JSON to the application on the client side. This AJAX call then finishes off with running a function. This then entered a loop to create the rows for the supervisor students table.

```

for(x in data.student){
    firstName = data['student'][x]['fn'];
    lastName = data['student'][x]['sn'];
    projectTitle = data['student'][x]['title'];
    dob= data['student'][x]['dob'];
    gender= data['student'][x]['gender'];
    uniId= data['student'][x]['uni_id'];
    projectDescription = data['student'][x]['description']
    if(projectTitle == null){
        projectTitle = '<font color="red">No project created</font>';
    }
    student_table += '<tr id="'+data['student'][x]['student_id']+'" class="student_row" >';
    student_table += '<td class="student_name student_row_click" colspan="2">'+firstName+'&nb';
    student_table += '<td align="right" class="student_row_click" colspan="2"><button class="';
    //student_table += '<td align="right"><button id="a" class="testingAdd" value="'+data['stuc
    student_table += '</tr>';

    student_table += '<tr class="student_row_edit" style="display:none;">';
    student_table += '<td class="student_details_title" colspan="4">';
    student_table += '<div class="col-xs-6 col-sm-3">';
    student_table += 'Project Title: ';
    student_table += '</div>';
    student_table += '<div class="col-xs-6 col-sm-9">';
    student_table += projectTitle;
    student_table += '</div>';
    student_table += '<div class="col-xs-6 col-sm-3">';
    student_table += 'Project Description: ';
    student_table += '</div>';
    student_table += '<div class="col-xs-6 col-sm-9">';
    student_table += projectDescription ;
    student_table += '</div>';
    student_table += '<div class="col-xs-6 col-sm-3">';
    student_table += 'DOB: ';
    student_table += '</div>';
    student_table += '<div class="col-xs-6 col-sm-9">';

```

Figure 23 JavaScript HTML build

In the loop, each value is put into a private variable which will be used for the rest of the loop. The table is then built by the JavaScript by putting HTML code inside a JavaScript variable.

```

$('#supervisor_student_table_body').html(student_table);

$('.student_row_click').click(function(){
    var rowId = $(this).parent('tr').attr('id');
    $(this).parent().next('tr').slideToggle();
    $('#'+rowId+' .icon').toggleClass('fa-angle-double-down').toggleClass('fa-angle-double-up');
});
});

```

Figure 24 jQuery adding Click Function

Once the data has been looped through and the table is built, it is then injected into the table body on the view by using jQuery. After the HTML has been injected into the HTML DOM the clicks are initialised to within the newly built HTML. The click in figure 24 toggles between two different rows in the table. One being the view row and the next being the view edit row.

5.3.5 Issues with this structure

While this works and does achieve in structuring the files, there was no real control over the data and functions and the project was going to hit complications in the future if the application continued to be built in this way. Another issue with the file structure coding was that it did not separate the different technologies which are not an issue in the short term but when a newer, exciting technology comes out which can perform better than the current latest technologies being used, it will be hard to quickly swap the technologies over without affecting the whole application. Further research into modern architectures which could be used to build next user group application.

After learning about and liking the idea of MVC (Model, View Controller), this was used as the structure for the future users.

5.3.6 Change to MVC

Starting off the application code was not in a great structure or in the best of layouts. The folders and code started to get messy as no real structure was being followed. This is where MVC came in. Researching and learning why this architecture is popular and the benefits, the decision was made to continue making the application using an MVC structure. This allowed for a better control over the code. Another reason for switching to MVC was to apply my knowledge from my skillset which has been expended from this year's modules in my Computer Science course.

Before an example of an MVC structure is discussed, it is important to show the file structure layout differences between the two. (figure 25 and 26)

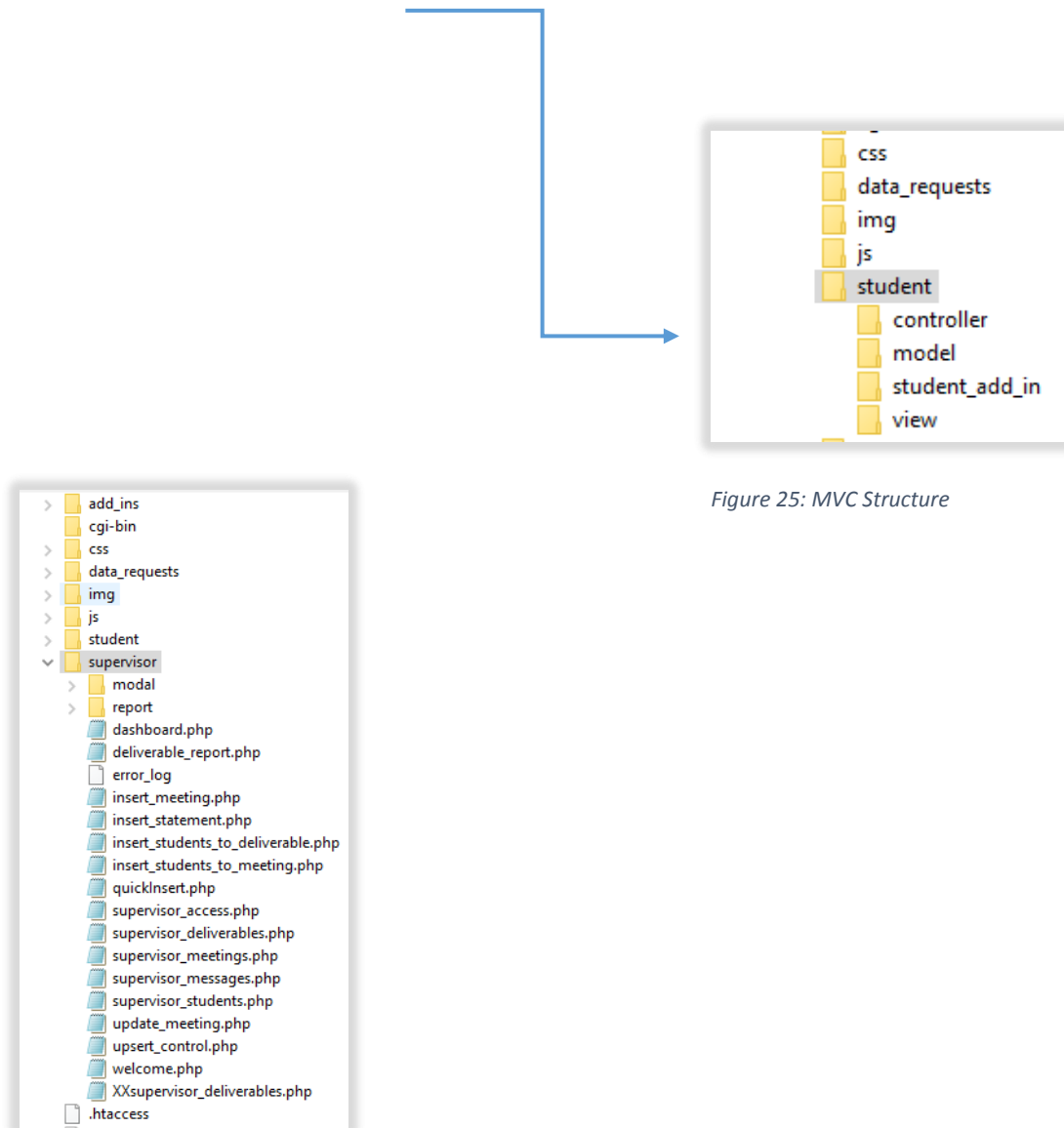


Figure 25: MVC Structure

Figure 26: One Page Layout

As can be seen the MVC structure is much cleaner than the one-page approach that was previously done. The one structure is messy and untidy and while this may be due to my programming and structuring as the main reason was no official methodology was followed. The MVC methodology gave a structure (which is Object Orientated) to be followed. This was helpful and progressed this part of the application quicker, producing better tidier code.

5.3.7 PHP Type Changed

As the structure of the application was changing to use MVC across the student application, the PHP was also switched from PHP SQLI to PHP PDO. This is because MVC is an object orientated programming structure and so is PDO. PHP PDO works with the models which will be created as part

of this new structure and therefore switching PHP is vital for this new structure to work successfully. PDO also offers more benefits such as providing protection against hackers using SQL injection because of the way the queries are put together. Very like the sprintf function which was discussed earlier on in this report.

5.5.8 User type Student: MVC implementation example

Following on from the supervisor application and the decision to change the structure of the code to an MVC. This was a new concept which was learnt in university and further research. MVC is all about splitting up your code across the application to keep the structure clear and tidy. It also allows a technology to be replaced easily as the files mainly only contain one form of technology.

An example of this structure will now be discussed in more detail. Students Meetings will be used for the example.

View requirements

Every time a new page is loaded it will start by accessing the view. The view is what the user sees and interacts with. At the start of each view, the file uses PHP which requests the controller for the page and the header.

```
<?php require_once ('../controller/meetingsController.php'); ?>
<?php require_once("../student_add_in/header.php"); ?>
<h1>Meetings</h1>
<div class="meetings div">
```

Figure 27 PHP Requires

PHP Controller

The controller is the brains behind each page. This controller is created in PHP to work on the server. This controller is vital for the MVC methodology to work successfully.

```
<?php
require_once '../model/config.php';
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
require_once '../model/meeting.php';
require_once '../model/meeting_functions.php';
$result = getAllmeetings($_SESSION['user']['student_id']);
?>
```

Figure 28 PHP Controller

This controller is built to control what model data can be accessed on this view. In this case, it is the meeting model. It first requires the config file where the connection to the database and the access control is located.

Config

The config file is the connection to the database. As PHP PDO is now being used, a new config file was created for the student application.


```

<?php
$servername = "unipdms.co.uk";
$username = ""; //removed for security reasons
$password = ""; //removed for security reasons
$dbname = "unipdms_db";

try {
    $pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}
?>

```

Figure 29 Config

This file sets the PDO variable which will be used across the application when performing a query to the database. If the connection fails, it will throw an error and no data will be able to be received from the database. This is to stop any unauthorised access to the data.

Model

The model is a class which contains all the attributes which are associated with the model along with any functions. The model is the principle of the data which is about to be received from the database and only the attributes set can be obtained from the view. This can be seen in the figure 30.

```

<?php
class Meeting implements JsonSerializable{
    private $meeting_id;
    private $date_time;
    private $Title;
    private $duration;
    private $description;
    private $building_id;
    private $room;
    private $name;
    private $location_id;
    private $status;

    private $campus;
    private $line1;
    private $line2;
    private $town;
    private $country;
    private $postcode;
    private $attendance_status;

    function __get($name) {
        return $this->$name;
    }

    function __set($name,$value) {
        $this->$name = $value;
    }

    /* JSON serialize function */
    public function jsonSerialize(){
        return get_object_vars($this);
    }
}
?>

```

Data can be
 Private Attributes
 Getters and setters
 Returns Data back as JSON

Figure 30 Modal (Class)

This structure is built around being Object Orientated and this model provides the attributes which can be used. Private means that the attributes can only be changed by the class itself, which gives control over how the attributes are updated. The class setter and getters give the class the functionality to update the attributes. In PHP, it is only required to create one setter and getter as it is used for all the attributes in the model which is unlike many other programming languages including JAVA where each setter and getter must be created for every attribute.

PHP Controller data request.

The PHP controller is built to then be programmed to fetch the data. Only the value is passed in by the parameters of the function which are going to be used within the SQL query. The query is then created.

```
<?php
function getAllmeetings($studentId){
    global $pdo;
    echo $studentId;
    $statement = $pdo->prepare("SELECT *
                                FROM Student_Meetings
                                INNER JOIN Meetings on Student_Meetings.meetings_id = Meetings.meeting_id
                                INNER JOIN Building on Building.building_id = Meetings.building_id
                                INNER JOIN Location on Location.location_id = Building.location_id
                                WHERE Student_Meetings.student_id = ?
                                ORDER BY date_time desc
                                ");
    $statement->execute([$studentId]);
    $result = $statement->fetchAll(PDO::FETCH_CLASS, "Meeting");
    return $result;
}
```

Figure 31 PHP Function. Select Statement

The difference between PDO and SQLI is how the SQL query is performed with the returning data from the database. When the SQL is executed, PDO will enter the data into the model, creating an object by using the PDO::FETCH_CLASS in the result. Once this is returned to the application as JSON (because the model is serializable), the application will create the view.

Header

After this, the header is required. The header is created as a partial view file so it can be displayed on every page. This stops repeating code on every page. This contains the navbar and all the CSS and JavaScript files along with the access control file.

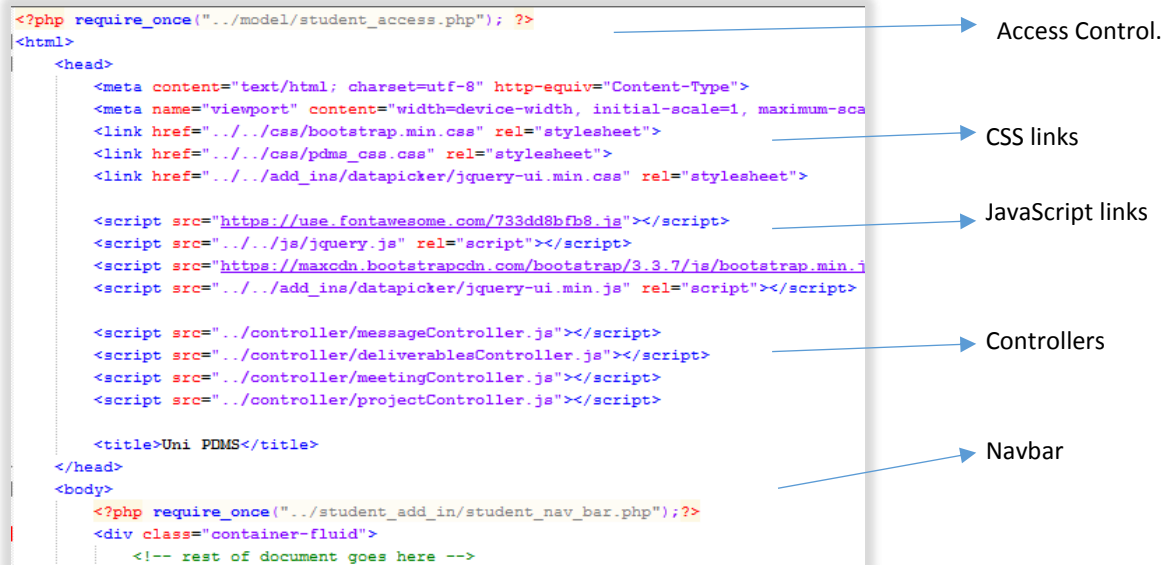


Figure 32 View

The header is coded to hold all the essential links within the application which each page requires like the CSS and JS. It also holds all the controllers too so they are available for each page.

After this, the view is built. This is done by HTML, CSS, and PHP to create the layout, design and data.

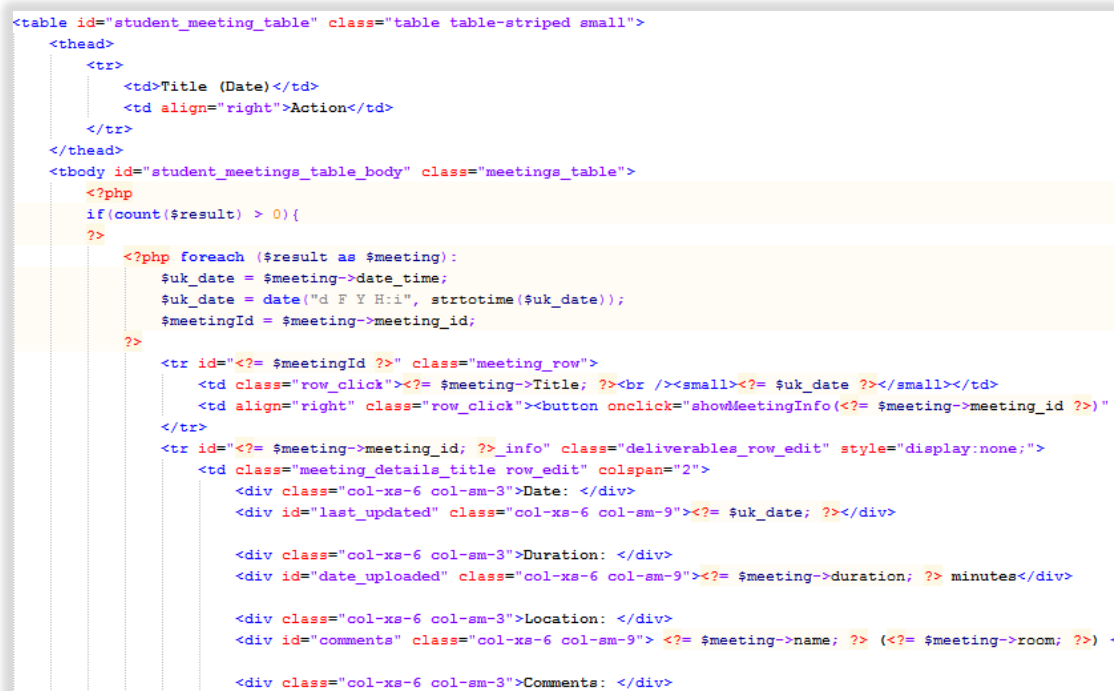


Figure 33 View 2

The PHP is a server side programming language that the server will program before sending the file over the internet to the client. This means as PHP is used in this view, only the complete document will be sent over unlike the one-page layout which uses JS. The PHP loops through the data retrieved from the query to complete the table HTML depending on how many objects are returned.

JS Controller

The JavaScript functions are running when an action is done by the user on the view. The JavaScript (using jQuery) is performing the action the user would expect from the action they performed. The JavaScript code can be viewed in the figure 34.

```
function showMeetingInfo(rowId){
    console.log('im here');
    //show detail row
    $('#student_meetings_table_body #' + rowId).next('tr').slideToggle();
    $('#'+rowId+' button .icon').toggleClass('fa-angle-double-down').toggleClass('fa-angle-double-up');
}

function acceptMeeting(meetingId){
    //show detail row
    $.get("../controller/getMeetingService.php",{meetingId:meetingId, callType:'acceptMeeting'}, function(data, status){
        $('#'+meetingId+'_info .accept_meeting_choices').html('<i class="fa fa-check fa-2x text-success" aria-hidden="true">');
    });
}

function declineMeeting(meetingId){
    //show detail row
    $.get("../controller/getMeetingService.php",{meetingId:meetingId, callType:'declineMeeting'}, function(data, status){
        $('#'+meetingId+'_info .accept_meeting_choices').html('<i class="fa fa-times fa-2x text-danger" aria-hidden="true">');
    });
}
```

Figure 34 JavaScript Controller

There are three different JavaScript functions in this controller which perform different actions. The first one hides and shows the details screen on each meeting and the second and third functions accept/ declines a meeting by sending a get request over the network to run an SQL to update the model.

5.3.9 Discussion on both Structures used (One-page and MVC structure)

Both structures have their advantages over each other and provide benefits to the application which the other can't in their current structure. A table has been created to highlight the advantages of each structure.

	One Page (Supervisor)	MVC (Student)	Comments
Responsiveness between each view	✓		As the data, already been requested by the one-page layout, the responsiveness between each page is instant.
Reusing Code		✓	For this, both can be just as powerful at doing this. MVC structure forces you to more meet the layout.
Object Orientated code		✓	MVC is OO whereas the current one-page layout is not.
File Layout		✓	MVC had a much cleaner layout when comparing the both together.
Controlling updates		✓	MVC restricts what can be accessed by its structure of the system
Modifying Code		✓	Personally, found the code in the MVC easier to follow and edit
Maintaining Code		✓	Code was easier to follow and therefore easier to maintain
User Experience	✓	✓	Due to the structure of the one page layout, the application experience feels that little better as it more responsive but if an update has not performed correctly, this can cause issues so

			therefore the MVC structure would be better.
--	--	--	--

As can be seen the MVC structure provides more benefits over the old structure which was being used. This was mainly because the code was easier to follow and maintain making the development stages quicker to progress.

5.3.10 Conclusion

Building the structure was an exciting goal of the project and lots of time was spent to get this right. From studying at university and learning about MVC, changes a personal view on how a modern application can be developed. In the future, it would be important to learn to use an MVC structure on a one-page layout. This would be very effective and would gain all the advantages discussed about both structures.

5.4 Examples of Functional Requirements Implemented

In this section, code is discussed on some of the functional requirements which were implemented.

5.4.1 Adding Students to a Deliverable

Adding students to a deliverable was one of the required functionality of the requirements. This allows the supervisor to create a deliverable then click on the icon to add students. A modal would pop up with a list of the students the supervisor supervises. From this list the supervisor will click all the students they wish to add to the deliverable. The process can be seen in the figure 35.

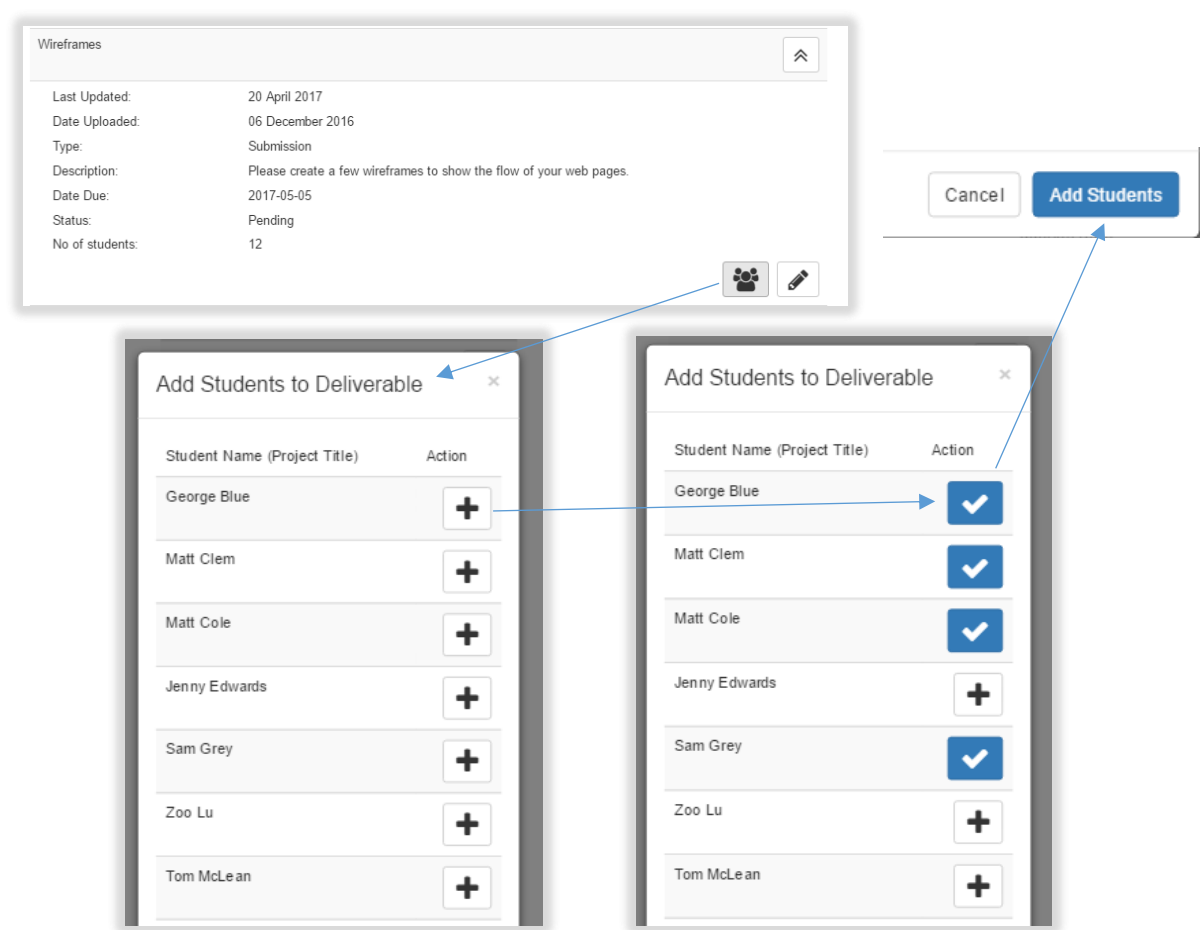


Figure 35 Adding Student to deliverable process

The programming to make this action work is effective and efficient. Once the submit button is clicked the JavaScript function sends data over to the PHP on the server to update the database. The JavaScript creates an array to store the project IDs and then loops through each element in the body of the modal and finds all the students which have been clicked. It knows if it has been clicked by the colour of the button in the class.

```
var project_id_array = [];
var i = 0;
$('#table_add_student_deliverable_body').find('.btn-primary').each(function() {
    project_id_array[i] = $(this).attr('value');
    i++;
})
```

Figure 36 jQuery part 1

Buttons with this style class (dark blue) contain a value with a project id. These are stored in the array. When this is complete the application is asked if it found any values. If it did not find any, the application will alert an message saying “Please Add Students”. If it does find students, it will post a request to the server to update the database.

```
if(deliverable_id > 0 && project_id_array.length > 0){
    $.post("insert_students_to_deliverable.php",{deliverable_id:deliverable_id, project_id_array: project_id_array}, func
    $('#submit_add_students_to_deliverable').val('');
    $('#modal_add_student_deliverable').modal('hide');
    var current_count = $('#'+deliverable_id+'#count_students_set').html();
    var new_count = (Number(current_count)+Number(numberOfNewStudents));
    $('#'+deliverable_id+'_info #count_students_set').html(new_count);
}
});
}else if(deliverable_id < 1){
    alert('Error With System');
}else{
    alert('Please Add Students');
}
```

Figure 37 jQuery part 2

The PHP will now loop through the array and insert these values into the table. It does this by getting the project id out of each element in the array in the for each loop.

```
$project_id_array[] = $_POST['project_id_array'];
$deliverable_id = $_POST['deliverable_id'];
$i = 0;
foreach($project_id_array[0] as $count => $project_id) {
    //echo $project_id;
    setStudentToDeliverable($deliverable_id, $project_id, $conn);
    $i++;
}
echo json_encode($i);
```

Figure 38 PHP for each loop

This code demonstrates how effective code behind the scenes can create an excellent user experience. This goal is made up of simple loops to collect the data and quickly update the database when the user says so. This improves the user experience because it is a quick process for the user to quickly go through and do a one-click process though all the students.

5.4.2 Upload Deliverables

Uploading deliverables were a challenging task in this project. The first decision was how to store the documents being uploaded. This could be done by storing the deliverables documents in a database or in a web dictionary.

	Database	Web Dictionary	Comments
Storage Allowance		✓	Storing it in a database would increase the amount of storage required and could push costs far higher than required.
Performance of the application		✓	Storing it in a database could slow down the application because of the amount of space it requires over many years.
Uploading the document	✓		Web dictionary requires more coding to cover all aspect of uploading the document. If the path changes, this would cause big issues.

As can be seen, the two ways of storing both have their advantages but due to keeping the performance of the application to the highest standard, the web dictionary was used.

In order for a student to upload a deliverable, they go to the deliverable view and click on the deliverable row they wish to view more of. The deliverable icon is within this view. This button shows the upload modal which allows the user to select the file and upload the document. The next file that uploads the document to the dictionary using the code from below.

```
//check to see if user folder already exists or not
if (!is_dir("../../deliverables_submitted/".$_SESSION['user']['user_id'])) {
    mkdir("../../deliverables_submitted/".$_SESSION['user']['user_id']);
}
//get location ready
define("UPLOAD_DIR", "../../deliverables_submitted/".$_SESSION['user']['user_id']."/");
if (!empty($_FILES["myFile"])) {
    $myFile = $_FILES["myFile"];

    if ($myFile["error"] != UPLOAD_ERR_OK) {
        echo "<p>An error occurred.</p>";
        exit;
    }
    // ensure a safe filename
    $name = preg_replace("/[^\A-Z0-9_\-]/i", "_", $myFile["name"]);
    // don't overwrite an existing file
    $i = 0;
    $parts = pathinfo($name);
    $name = $title . "." . $parts["extension"];
    // preserve file from temporary directory
    $success = move_uploaded_file($myFile["tmp_name"],
        UPLOAD_DIR . $name);
    // echo $myFile["tmp_name"];
    if (!$success) {
        echo "<p>Unable to save file.</p>";
        exit;
    } else {
        $project_id = $_SESSION['user']['project_id'];
        //need to update database
        $dir = UPLOAD_DIR . $name;
        $result = storeUploadDetails($dir, $deliverable_id, $project_id);
        //go back to deliverablesView.php
        header("Location: ../view/deliverablesView.php");
    }
}
```

Figure 39 PHP Upload deliverable

The PHP function will upload the file as the same name as the deliverable. This is to allow the system to know which file matches to each deliverable in the file and so the old deliverable can be overwritten. After this is complete, the URL needs to be stored in the database.

This is completed by an update function.

```
function storeUploadDetails($dir, $deliverable_id, $project_id){
    global $pdo;
    try{
        //prepare statements
        $updateStatement = $pdo->prepare("UPDATE Student_Deliverables
                                         SET dir = ?, Complete = 1
                                         WHERE deliverable_id = ? AND project_id = ?
                                         ");
        $updateStatement->execute([$dir, $deliverable_id, $project_id]);
        return 'success';
    }
    catch(PDOException $e){
        echo $sql . "<br>" . $e->getMessage();
    }
}
```

Figure 40 PHP function

5.4.3 Header Location

Once the user is logged in, a session is created containing information about the user. One of the attributes in the session is user_type. This comes in useful throughout the application to make sure they have the right permissions to access certain parts of the application. Another place this is used is on the login screen. If the user goes back to the logging screen, the application will check to see if they have already logged by the session data.

```
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if(isset($_SESSION['user']['user_type'])){
    if($_SESSION['user']['user_type'] == 1){//supervisor
        header("Location: http://unipdms.co.uk/supervisor/welcome.php");
        die();
    }else if($_SESSION['user']['user_type'] == 2){//student
        header("Location: http://unipdms.co.uk/student/view/dashboardView.php");
        die();
    }
}
```

Figure 41 Headers in Login

If the user type is set, the application will be redirected them back to the right location of the application until they log out. This means if the user never logs out, they will always be pointed to their dashboard.

5.4.3 Password Encryption

Passwords must be encrypted to hide these details for the public eye and even the developers who have access to the system. To do this the passwords are encrypted in the PHP before being inserted into the database. To make sure the user passwords are encrypted from the public eye, two forms of encryption is used. This can be seen in figure 42.


```
$hashPassword = crypt($password, '<tm!65>');
$password = hash('sha1', (get_magic_quotes_gpc() ? stripslashes($hashPassword) : $hashPassword));
```

Figure 42 Password Encryption

This is a strong way to store a password as it encrypted twice. The first encryption puts a string on the end of the password and this password is encrypted using a hash function. The hashing used is sha1. This provides a way to hash the password on top of the added characters. This adds extra protection to the passwords which should prevent the password from being stolen and used.

5.5 User Interface

The user interface is an important factor in getting the user experience right. The following screens will show the application running on different devices to highlight the responsiveness and the simple clear interface which was developed to produce a brilliant user experience. Examples are shown below.

Navbar

The navbar is simple displaying the only relevant headings to the user. Bootstrap has its own navbar which can be customised to meet the website design. The text is used instead of icons to make the navbar instantly usable for the user because they do not need to learn/ work out what each icon does and remember it.

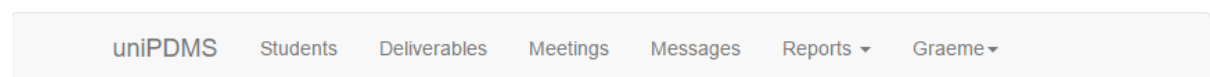


Figure 43 Supervisor Navbar

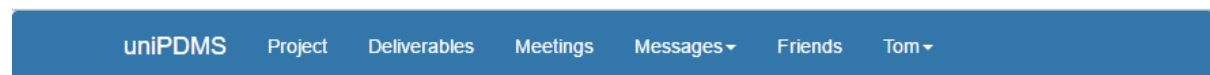


Figure 44 Student Navbar

Interface Views

Each interface was designed with mobile first in mind. Mobile first is designing the application for mobile then expanding the screens out for the desktop. This means that everything that works on a desktop computer should work the same as on mobile. The second goal when developing the interface was to allow the user to achieve their goal with the fewest steps possible to create a great user experience. By keeping the view similar it keeps the application consistent. These interfaces displayed below help demonstrate this while keeping a smart, clean design.

Main Views Interface

The main views within the application are displayed in a clear table. Each screen has a clear title at the top with the supervisors having an added functionality of having a search bar. Buttons appear at the top under the heading of any interface which requires something new to be created (like a new message or new deliverable). By building this on a table and using my own and bootstrap CSS, it creates a clear simple screen which looks modern and eye catching on any device.

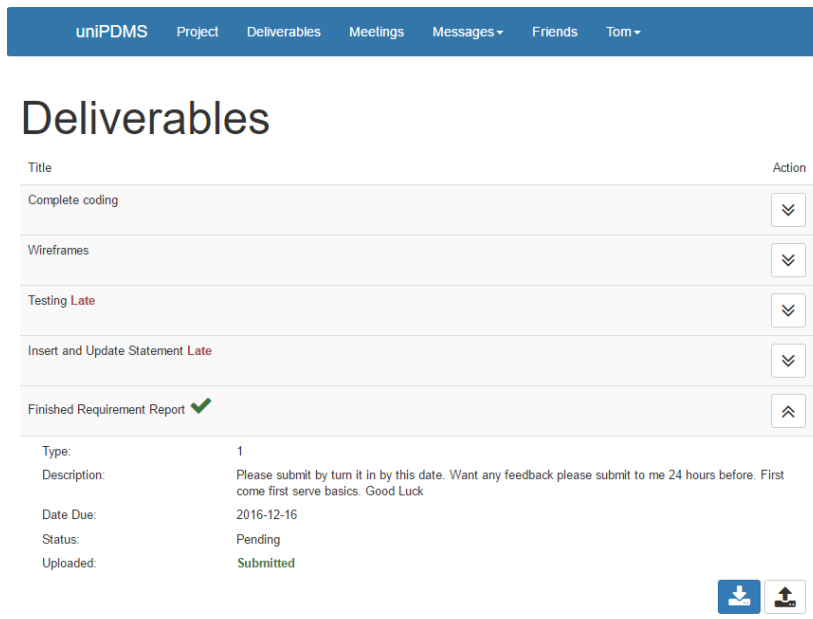


Figure 45 Student Deliverable Interface

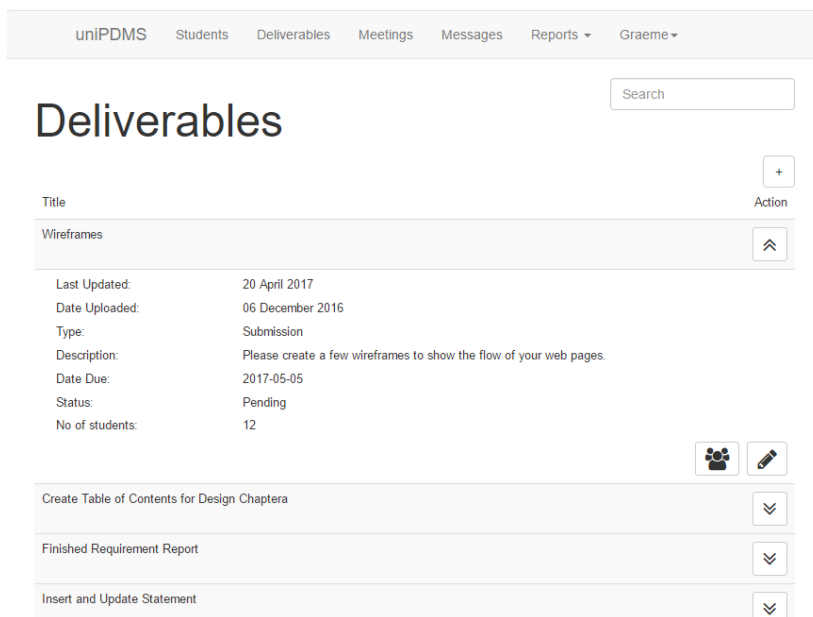
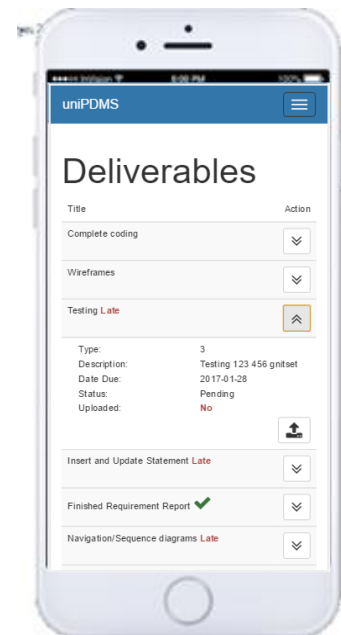


Figure 46 Supervisor Deliverables Interface



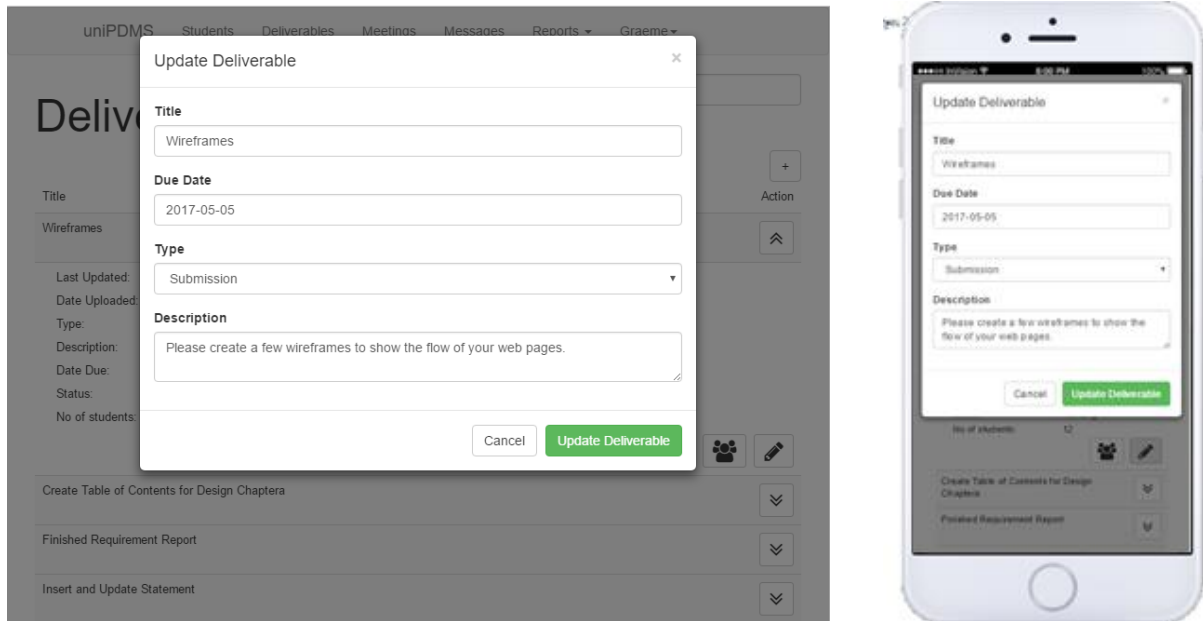
As can be seen in the figure 45 and 46, the screens look different for both users given each user different functionality to meet their requirements. They both are just as useable on a mobile device as it is on a desktop. The toggle button on each row moves professionally, opening and closing. This is achieved by using jQuery. This code can be seen in figure 47.

```
function showMeetingInfo(rowId) {
    console.log('im here');
    //show detail row
    $('#student_meetings_table_body #' + rowId).next('tr').slideToggle();
    $('#'+rowId+' button .icon').toggleClass('fa-angle-double-down').toggleClass('fa-angle-double-up');
}
```

Figure 47 jQuery 3

Edit Interface

All the edit interface is all displayed a modal. A modal is a pop-up screen which lays over the top of the current screen. This keeps the user in the same view without having to reload information. This is done via ajax and JavaScript. An example of the modal can be seen in



the figure 48 and 49.

Figure 48 Edit interface Upload Deliverable

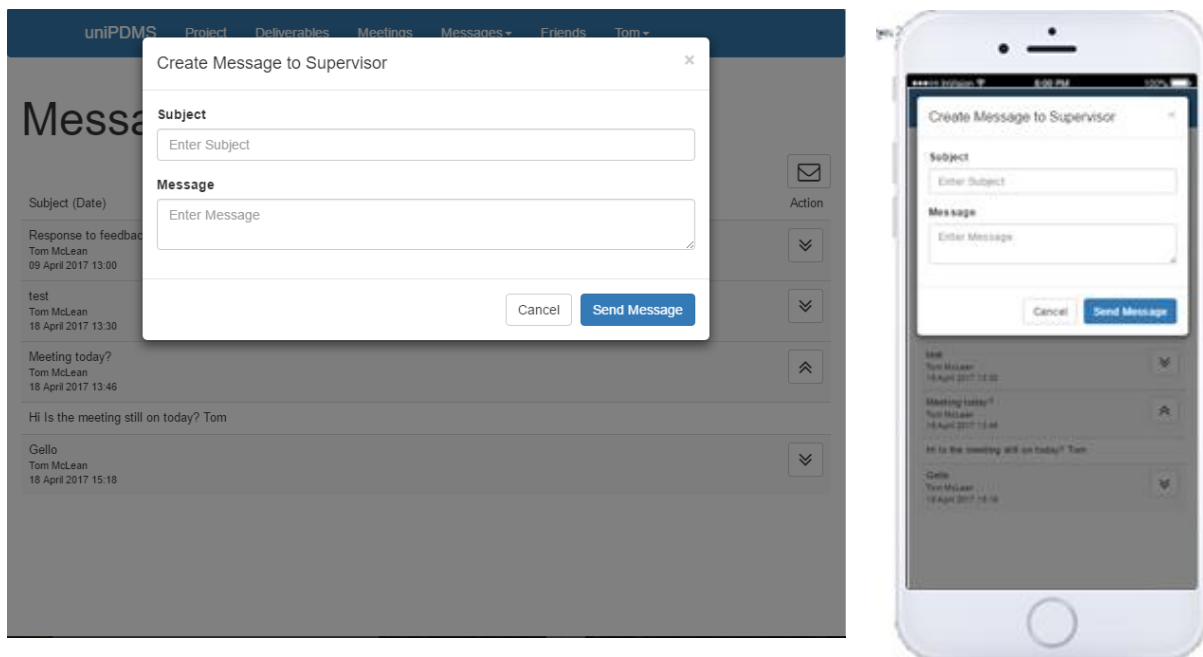


Figure 49 Edit interface Create Message

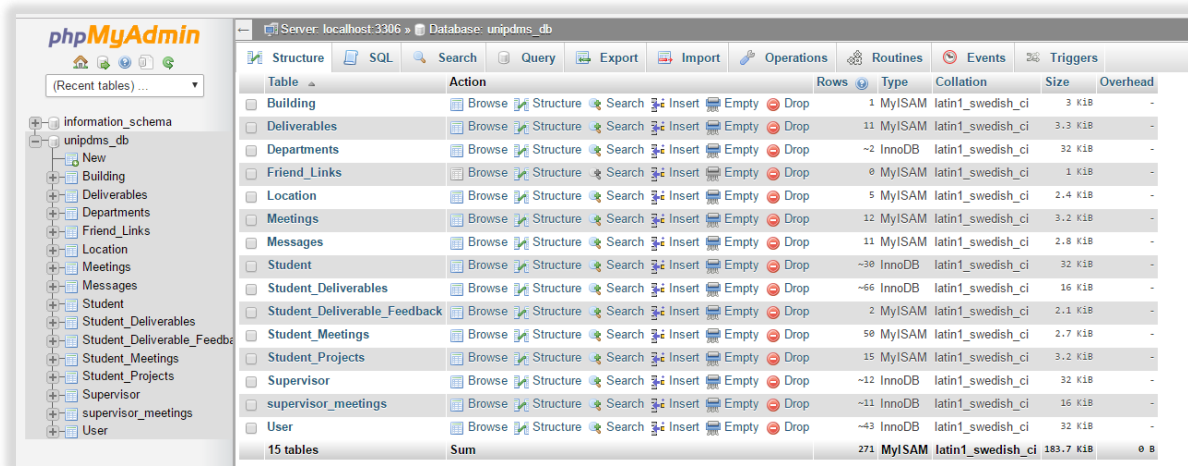
With all the edits being done through a modal it keeps the user experience consistent. The title is always at the top of the goal they are going to achieve by submitting the form. The forms are always in the middle and the goal buttons are at the bottom.

By having the different screen in this layout, it creates an environment where the user feels in control of what they are doing and the more they use the system, the greater confidence they get.

5.6 Database Implementation

5.6.1 Introduction

With the database structure and design already taking place in the design stage, this made for the database to be implemented quickly. phpMyAdmin was used as this is an open source program which is included on many c-panels. The database implemented can be seen below. This example shows all the database tables to the left with an example of the attributes from the User table.



The screenshot shows the phpMyAdmin interface for a database named 'unipdms_db'. On the left, a tree view lists the tables: Building, Deliverables, Departments, Friend_Links, Location, Meetings, Messages, Student, Student_Deliverables, Student_Deliverable_Feedback, Student_Meetings, Student_Projects, Supervisor, supervisor_meetings, and User. The main panel displays a table structure view for 15 tables. Each table row includes a checkbox, the table name, an 'Action' column with icons for Browse, Structure, Search, Insert, Empty, and Drop, and columns for Rows, Type, Collation, Size, and Overhead.

Table	Action	Rows	Type	Collation	Size	Overhead
Building	Browse Structure Search Insert Empty Drop	1	MyISAM	latin1_swedish_ci	3 K1B	-
Deliverables	Browse Structure Search Insert Empty Drop	11	MyISAM	latin1_swedish_ci	3.3 K1B	-
Departments	Browse Structure Search Insert Empty Drop	~2	InnoDB	latin1_swedish_ci	32 K1B	-
Friend_Links	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci	1 K1B	-
Location	Browse Structure Search Insert Empty Drop	5	MyISAM	latin1_swedish_ci	2.4 K1B	-
Meetings	Browse Structure Search Insert Empty Drop	12	MyISAM	latin1_swedish_ci	3.2 K1B	-
Messages	Browse Structure Search Insert Empty Drop	11	MyISAM	latin1_swedish_ci	2.8 K1B	-
Student	Browse Structure Search Insert Empty Drop	~30	InnoDB	latin1_swedish_ci	32 K1B	-
Student_Deliverables	Browse Structure Search Insert Empty Drop	~66	InnoDB	latin1_swedish_ci	16 K1B	-
Student_Deliverable_Feedback	Browse Structure Search Insert Empty Drop	2	MyISAM	latin1_swedish_ci	2.1 K1B	-
Student_Meetings	Browse Structure Search Insert Empty Drop	50	MyISAM	latin1_swedish_ci	2.7 K1B	-
Student_Projects	Browse Structure Search Insert Empty Drop	15	MyISAM	latin1_swedish_ci	3.2 K1B	-
Supervisor	Browse Structure Search Insert Empty Drop	~12	InnoDB	latin1_swedish_ci	32 K1B	-
supervisor_meetings	Browse Structure Search Insert Empty Drop	~11	InnoDB	latin1_swedish_ci	16 K1B	-
User	Browse Structure Search Insert Empty Drop	~43	InnoDB	latin1_swedish_ci	32 K1B	-
15 tables	Sum	271	MyISAM	latin1_swedish_ci	183.7 K1B	0 B

Figure 50 Database

As you can see the model contains thirteen tables which are all used. They all have connections between each other which are created by foreign keys with a unique attribute established by Primary Keys too.

5.6.2 Table Statement Example

An Example of a create statement in SQL can be viewed below.

```
CREATE TABLE Student (  
    student_id int(10) NOT NULL,  
    user_id int(10) NOT NULL,  
    dob date NOT NULL,  
    supervisor_id int NOT NULL,  
    uni_id varchar(100),  
    PRIMARY KEY (student_id),  
    FOREIGN KEY (user_id) REFERENCES User(user_id),  
    FOREIGN KEY (supervisor_id) REFERENCES Supervisor(supervisor_id)  
);
```

Figure 51 Create Table Statement

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
student_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
user_id	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
dob	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
supervisor_id	INT(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
uni_id	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'Not Set'
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 52 Student table created

As can be seen, the Insert statement has created the table with the correct column names. This was completed for the rest of the tables.

5.6.3 Connecting to the database.

To connect to the database from the application, two different config files were created. One was in PHP PDO and the other was PHP SQLI.

```
<?php
    $servername = "unipdms.co.uk";
    $username = ""; //removed for security reasons
    $password = ""; //removed for security reasons
    $dbname = "unipdms_db";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }
?>
```

Figure 53 Config SQLI

```
<?php
    $servername = "unipdms.co.uk";
    $username = ""; //removed for security reasons
    $password = ""; //removed for security reasons
    $dbname = "unipdms_db";

    try {
        $pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        // set the PDO error mode to exception
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    catch(PDOException $e)
    {
        echo "Connection failed: " . $e->getMessage();
    }
?>
```

Figure 54 PHP PDO config

As can be seen, the config files are similar but have a few differences in the way it stores the connection. SQLI uses \$conn while PDO uses \$pdo. Both these files connect the application to the database to create a personal application for the user.

5.6.4 Joining tables

Joining tables within the SQL is an essential to receive all the data required in one query. This allows the query to get data from one or more tables by joining the tables up where attributes are the same.

```
e("SELECT *  
FROM Student_Meetings  
INNER JOIN Meetings on Student_Meetings.meetings_id = Meetings.meeting_id  
INNER JOIN Building on Building.building_id = Meetings.building_id  
INNER JOIN Location on Location.location_id = Building.location_id  
WHERE Student_Meetings.student_id = ?  
ORDER BY date_time desc  
");
```

Figure 55 Joining tables

This SQL query in figure 55 joins three tables to get all the data required for the application. By joining the tables together this statement can get data about the meeting, the building and location which it is held in.

5.6.5 Update

The application performs many possible database updates depending on the user actions. Database Update edits the data already stored in the table.

```
("UPDATE Student_Projects  
SET title = ?, description = ?  
WHERE project_id = ? AND student_id = ?  
");
```

Figure 56 Update Statement

In figure 56 the update statement updates the title and description in the student projects table. This statement only updates where it meets the where clause.

5.7 Conclusion

Implementing this application has been demanding and challenging but with a great outcome. The experience gained from this stage has given great knowledge in many areas when implementing. From using both different types to structure for the application, MVC came out on top. Even though One page has some good advantages, the MVC gave me more control over the project. The application progressed well with all the user requirements being completed.

Chapter 6 Testing Strategy

Testing will be an essential part of building the application. Different kinds of testing will be used to ensure the quality of the project is perfected. Different coding test methodologies will be explained and be used along with test cases and a usability test towards the end of the project.

6.1 Code Testing Methodologies

There are many different types of code testing methodologies. They all aim to improve and address issues with the application which may not have been picked up. There is no formally recognised body of testing methodologies (Overbaugh, 2009). Some of the most common methodologies are Unit testing, Acceptance testing, functional testing, system testing, performance testing and security testing. It is also common for software testers to use approaches to these, such as boundary testing or combinatorial testing.

Three types of user testing methodologies stand out which should be adapted for most projects. These are Unit testing, functional and non-functional testing, and Security testing.

5.1.1 Unit Testing

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use (*ISTQB exam certification*, 2015). Unit testing tests a part of the code to ensure it is performing as expected and the right values are being returned. The program is built to automatically test the software before it is released or updated. This is a good testing methodology to learn as it is usually used for software developers working in large teams. The advantages of doing unit testing are to find issues at an early stage which reduce man hours spent on the project and helps the project to run smoothly by spotting any mistakes before it goes live.

5.1.2 Functional and non-functional Testing

Functional testing is a software testing process used within software development in which software is tested to ensure that it conforms with all requirements (Divestopedia and Institute, 2017). The user stories written in the requirement section will be followed to ensure the application contains all the functionality written in the stories. This will ensure the project is on target for its functional and non-functional requirements.

Black box testing

Black box testing is the functional testing. It is to ensure the application is performing as expected from a programming perspective.

White box testing

White box testing is the non-functional testing. It ensures the application is delivering as expected to a normal user with no programming experience. This is to check the user experience is correct and meets the user needs.

6.2 Security Testing

Security testing is a software testing method that focuses on application security, protecting the application from attacks including SQL injection. Making sure the application has a good standard of security is essential. The code will be tested for to ensure it can stop attacks like SQL. This will take place on a testing environment to make sure no tables or data is lost.

6.2 Security Testing in this Project

Testing will be an important aspect of this project to ensure the requirements are being met and the application is performing as expected. To ensure this is the case, this project will adapt to use

Functional Testing and Security testing by using use case testing. Test cases provide a clear list of everything that been tested and the outcome. This is then used to refer to when ensuring everything has been tested and performing the way expected. The choice has been made not to use unit testing by creating code to test the project as for a web application this can be difficult as the result may be consistently changing. Also because of the time frame and presenting the prototype, it is important to ensure each part has been tested by a user who can also give feedback on the user experience.

6.3 Test Cases

Test cases will be used to ensure each part of the system is tested manually. This will be a process that will happen throughout the development stage, to ensure the functionality has been programmed correctly. The headings which will be used are test case Id, steps, expected result, pass/fail and comments. An example of both can be viewed below while the rest can be viewed in the appendixes.

6.3.1 Functional Test Cases

Student System

Test Case ID	Functional Requirement	Steps (Description)	Expected Results	Pass/Fail	Comments
3.1	Edit Project	1. Click on project 2. Click edit 3. Edit form 4. Click update	Application will update student project	Pass	
3.2	View Meetings Details	1. Click on meetings 2. Click on details button	Application will show details about the meeting	Pass	
3.3	Accept or decline meeting	1. Click on meetings 2. Click on details button 3. Click the tick or cross	The system accepts or rejects meeting	Pass	
3.4	View Deliverable Details	1. Click on deliverables 2. Click on details button	The application shows the deliverables	Pass	
3.5	Upload Deliverable	1. Click on deliverables 2. Click on details button 3. Click upload	Uploaded document can be viewed in the folder	Pass	
3.6	View Inbox Message details	1. Click on inbox 2. Click on details button	The inbox messages can be seen and show the message	Pass	
3.7	View Outbox Message details	1. Click on outbox 2. Click on details button	The outbox messages can be seen and show the message	Pass	
3.8	Send Message	1. Click on messages 2. Click on new 3. Fill in form 4. Send message	Message appears in supervisor inbox	Pass	
3.9	Add Friends	1. Click on Friends 2. Click on Add 3. Add friend	Friend request is sent	Pass	

3.10	Accept Friend Request	1. Click on friends 2. Accept friend Request	Friends appears under friends on both accounts	Pass	
-------------	-----------------------	---	--	------	--

6.3.2 Non-Functional Test Cases

User with no programming experience were invited to test the system. A table has been created to show which goals they were asked to test with a success or fail.

Student

Only students were invited to take part in the student application test.

User one		
Goal	Success / Fail	Comment
Edit the project description	Success	User Edited the project description quickly and easily
Upload a deliverable	Success	Deliverable was uploaded successfully
Accept the meeting 'Weekly meeting 2'	Success	Meeting was found quickly and the correct description was read back.
Accept a friend request from Tom	Success	Friend request accepted
Read message with on the date of 11/12/2017	Success	Message read successfully
Send message	Success	Message sent

User one found the application easy to use with a high success rate. They commented on how easy the system was to use and liked the layout. The rest of the non-functional test cases can be seen in the appendixes.

6.5 Conclusion

Testing completed ensured the application is working as expected. The functional testing also gave additional benefits of making sure all the functional requirements are implemented and the application database is performing as expected with storing the data from inserts and updates. The non-functional testing gave good results for the user experience, to make sure the application performs as the user would expect.

Chapter 7 Critical Review

7.1 Introduction

In this chapter, a critical review will be discussed on how the project went with a perspective on lessons learned with insightful comparisons with existing systems being used. The project will be discussed critically.

7.2 Implementation Issues

While implementing the application, issues arose which slowed down the development of the coding which resulted spending more time than planned on researching the best implementation mythologies. This is due to the nature of wanting to develop the best application that was possible. This was a great learning curve but did hold up the project from producing more features for the project. But on the other hand, this decision to change was due to issues with the structure of the current layout and how a lack of following a popular methodology was causing the application development to slow down. So overall while it was an issue of having to research and learn a new way of structuring the code, it had a great effect on the future of the application.

7.3 Future Development

Future development would be an interesting as there are many ways the project could go. In the future, it be important to get charts onto the site using chart.js. This would provide an effective way to show statistics of the how the students are performing for a supervisor. The supervisor would be able to see which students are struggling. Charts could also be used under student's friends to have a mini competition on who has competed the most work.

Another development which could be taken further would be to provide a teaching environment where it could be used for primary schools too. This would give the application a wider audience.

An interesting development which could be achieved with continuous progression is to release a full bespoke system which meets the demands of many universities. This could be then used by universities around the UK for a small prescription fee per year. This is possible but not in its current developmental stage.

7.4 Lessons Learned

This project has been a brilliant personal experience and personal skill set has been improved greatly. The biggest success was developing through all the stages to produce a full working system with all the requirements. It is great to have to application working in an object ordinated way using MVC. It has been a great progression for personal confidences and with the lessons learnt from this project, it can be applied my future career.

7.5 Conclusion

Overall this has been an exciting project which has grown improved personal confidences and taught lots about delivering a project for a client. It has increased knowledge in getting requirements, designing and implementing technologies have grown and can now be used in future projects.

References

- BBC (2006) *The need for the data protection act*. Available at: [Online] <http://www.bbc.co.uk/schools/gcsebitesize/ict/legal/0dataprotectionactrev1.shtml> (Accessed: 18 February 2017).
- Outlaw (2017) Database rights: The basics*. Available at: [Online] <https://www.out-law.com/page-5698> (Accessed: 20 February 2017).
- Government (2016) *Data protection*. Available at: [Online] <https://www.gov.uk/data-protection/the-data-protection-act> (Accessed: 18 February 2017).
- Legislation (No Date) Data protection act 1998, c.* Available at: [Online] <http://www.legislation.gov.uk/ukpga/1998/29/schedule/2> (Accessed: 18 February 2017).
- Commissioner's Office (2017) *Data protection principles*. Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/data-protection-principles/> (Accessed: 18 February 2017).
- Microsoft (2017) *Description of Symmetric and Asymmetric Encryption*. Available at: [Online] <https://support.microsoft.com/en-gb/help/246071/description-of-symmetric-and-asymmetric-encryption> (Accessed: 23 February 2017).
- Osborne, C. (2013) *The top ten most common database security vulnerabilities*. Available at: [Online] <http://www.zdnet.com/article/the-top-ten-most-common-database-security-vulnerabilities/> (Accessed: 21 February 2017).
- Quinn, B. and Arthur, C. (2011) *PlayStation Network hackers access data of 77 million users*. Available at: [Online] <https://www.theguardian.com/technology/2011/apr/26/playstation-network-hackers-data> (Accessed: 21 February 2017).
- Rouse, M. (2008) What is HTTPS (HTTP over SSL or HTTP secure)? - definition from WhatIs.Com. Available at: [Online] <http://searchsoftwarequality.techtarget.com/definition/HTTPS> (Accessed: 4 March 2017).
- "Software Development Methodologies". Itinfo.am. N.p., 2017. Web. 19 Apr. 2017.
- The rights of individuals (principle 6)* (2017) Available at: [Online] <https://ico.org.uk/for-organisations/guide-to-data-protection/principle-6-rights/> (Accessed: 18 February 2017).
- What is SSL (secure sockets layer)? (2003) Available at: [Online] <https://www.digicert.com/ssl.htm> (Accessed: 4 March 2017).

Bibliography

- Carey, P.L.I.M. and Russel, C. (2000) *Data protection in the UK*. London: Blackstone Press.
- Copyright (2017) *Data protection principles*. Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/data-protection-principles/> (Accessed: 4 March 2017).
- Guide to data protection (2017) Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection> (Accessed: 4 March 2017)
- Heather Rowe (1999) *Data Protect Act 1998: A Practical Guide* (Accessed: 1 March 2017).
- Lynskey, O. (2015) *The foundations of EU data protection law*. Oxford, United Kingdom: Oxford University Press.

Bibliography: Overbaugh, b. (2009) *What are the top software testing methodologies?* Available at: <http://searchsoftwarequality.techtarget.com/answer/What-are-the-different-software-testing-methodologies> (Accessed: 26 January 2017).

ISTQB exam certification (no date) Available at: <http://istqbexamcertification.com/what-is-unit-testing/> (Accessed: 26 January 2017).

Divestopedia and Institute, S. (2017) *What is functional testing? - definition from Techopedia.* Available at: <https://www.techopedia.com/definition/19509/functional-testing> (Accessed: 26 January 2017).

Appendixes

Use Case Texts

The purpose of this appendix is to show all the use cases created to meet the functional requirements of the system. The use cases are discussed in Chapter 3.

Use Case Text 1 is Create Deliverable. This will mainly be used by the actor supervisor and enables the actor to insert a new deliverable into the system for students to then complete.

Use Case Text 1		
Use Case Name		Create Deliverable
Use Case ID		1001
Description		Insert a new deliverable into the system.
Level		High-Level Goal
Primary Actor		Supervisor
Post-Condition		User successfully logs in to their account
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Deliverables	System loads a new user page
2	User clicks on 'Create New Deliverable'	System loads a new page containing a form
3	User enters fills in the required form	
4	User clicks on save	System creates a new deliverable and takes the user back to the deliverable screen
Alternative		
Steps	User Interaction	System Response
2a	User clicks on a drop down from the navbar and selects 'Create New Deliverable'	System Loads the form for the required form in a new view
Exceptional Flows		
Steps	User Interaction	System Response
4a	User doesn't fill in the required inputs	System highlights the required form and does not create the new deliverable

Use Case Text 2 is Set Deliverable to Student. The primary actor is a Supervisor. The goal is to select a deliverable and set it to one of their students who they supervisor.

Use Case Text 2		
Use Case Name		Set Deliverable to Student
Use Case ID		1002
Description		Select a deliverable and add a student.
Level		High-Level Goal
Primary Actor		Supervisor
Post-Condition		User successfully created a deliverable and is linked to his student
Main Success Scenario		
Steps	User Interaction	System Response
1	The user clicks on the deliverable	System loads a new page
2	User clicks on the ‘Deliverable’ name	System loads a new page containing information about the deliverable including current students linked to it
3	User selects ‘Link to Student’	System loads a new view containing students
4	User select student	System links student to deliverable and brings user back to the deliverable page
Alternative		
Steps	User Interaction	System Response
1a	User clicks on student and selects add ‘Deliverable’	The system takes the user to the deliverable page.

Use Case Text 3 is View and Edit Deliverable. The primary actor is a Supervisor and Student. This goal will be achieved by both primary actors and is an important goal which must be met.

Use Case Text 3		
Use Case Name		View and Edit Deliverable
Use Case ID		1003
Description		Update an already made Deliverable
Level		High-Level Goal
Primary Actor		Supervisor/ Student
Post-Condition		User successfully created a deliverable
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Deliverables button	System loads a new page
2	User clicks on the deliverable they wish to update	System loads a new page containing a form with the data of the deliverable
3	User edits the fields	
4	User clicks on update	System updates the deliverable and takes the user back to the deliverable screen
Exceptional Flows		
Steps	User Interaction	System Response
4a	User doesn't fill in the required inputs	System highlights the required form and does not update the deliverable

Use Case Text 4 is Upload Deliverable. This goal will allow the actor to upload his deliverable once they have completed it. This is a way to show the Supervisor that the deliverable has been completed.

Use Case Text 4		
Use Case Name		Upload Deliverable
Use Case ID		1004
Description		Deliverable is completed
Level		High-Level Goal
Primary Actor		Student
Post-Condition		User successfully upload deliverable
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Deliverables button	System loads a new page
2	User clicks on the Deliverable Title they wish to upload	System loads the deliverable details in a new view
3	User clicks upload deliverable	System opens up file explorer
4	User selects file	System closes explorer and save the file path to the form
5	User clicks ‘Upload’	The system inserts the file path into the database and uploads the file to a document.
Exceptional Flows		
Steps	User Interaction	System Response
5a	No file selected. User is asked to select a file	System highlights the error.

Use Case Text 5 is Create Feedback. This goal is for a supervisor to give feedback and guidance to their students they are supervising.

Use Case Text 5		
Use Case Name		Create Feedback
Use Case ID		2001
Description		Add feedback to a student deliverable
Level		High-Level Goal
Primary Actor		Supervisor
Post-Condition		Student completed Deliverable
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on ‘Student’ button	System loads a new page
2	User clicks on ‘Student deliverable’ they wish to give feedback on	System loads a new page containing a form
3	User fills in form to submit feedback	System inserts feedback into database and returns user to the student page
Exceptional Flows		
Steps	User Interaction	System Response
3a	User doesn’t fill in the required inputs	System highlights the required form and does not update the deliverable

Use Case Text 6 is View Feedback. This goal is for a student to view the feedback from their users. This is a high-level goal as it is an important aspect of students improving their work and understanding where they could improve.

Use Case Text 6		
Use Case Name		View Feedback
Use Case ID		2002
Description		To look at the user feedback on deliverables
Level		High-Level Goal
Primary Actor		Student
Post-Condition		User has created a deliverable and is waiting for feedback
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Feedback	System loads a new user page
2	User views the latest feedback	System loads the Feedback data

Use Case Text 7 is View Message. This goal allows the actors to look at their messages which they have received.

Use Case Text 7		
Use Case Name		View Message
Use Case ID		3001
Description		To look at messages sent to the user
Level		High-Level Goal
Primary Actor		Supervisor/ Student/ Module Leader
Post-Condition		User successfully logs in to their account
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Messages	System loads a new user page
2	User view list of messages subjects	System loads the Messages table
3	User clicks on message	System lesions out for a user action
4	User view the message	System loads the message
Exceptional Flows		
Steps	User Interaction	System Response
3a	No messages for the user to view	System display a message saying there are no messages to view

Use Case Text 8 is View Message. This goal allows the actors to send a message to another user. This will be a simple messaging tool in the system.

Use Case Text 8		
Use Case Name		Create Message and Send
Use Case ID		3002
Description		To create a new message to the user
Level		High-Level Goal
Primary Actor		Supervisor/ Student/ Module Leader
Post-Condition		User successfully logs in to their account and clicks on messages
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on ‘New’	Loads new page for the user to view with required fields
2	User fills out the required form	
3	Selects the user they wish to send the message too	User adds user id to the form
4	User clicks ‘Send’	The system creates a new message in the table linked to the correct user. Returns user to Messages
Exceptional Flows		
Steps	User Interaction	System Response
4a	Not all Required fields are filled in.	System highlights forms required to fill in.

Use Case Text 9 is View Meeting. This goal allows the actors to view the meetings which they have created or be invited/ agreed to attend. All user types will have this goal.

Use Case Text 9		
Use Case Name		View Meeting
Use Case ID		4001
Description		To look at the user meetings
Level		High-Level Goal
Primary Actor		Supervisor/ Student/ Module Leader
Post-Condition		User has meetings already in the database
Main Success Scenario		
Steps	User Interaction	System Response
1	The user selects Meeting	System loads a new user page
2	User views the list of meetings	System loads the Meetings coming up and in the past

Use Case Text 10 is Create Meeting and Invite. This goal allows the actors to create a new meeting and invite users along who they supervise/ friends with.

Use Case Text 10		
Use Case Name		Create Meeting and Invite
Use Case ID		4002
Description		To create a new meeting and send an invite to a user
Level		High-Level Goal
Primary Actor		Supervisor/ Module Leader
Post-Condition		User successfully logs in to their account and clicks on meetings
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on ‘New Meeting’	Loads new page for the user to view with required fields
2	User fills out the required form	
3	Selects the user they wish to send the meeting too	User adds user id to the form
4	User clicks ‘Save	The system creates new meeting and links to the correct user(s). Returns user to Meetings view
Exceptional Flows		
Steps	User Interaction	System Response
4a	Not all Required fields are filled in.	System highlights forms required to fill in.

Use Case Text 11 is Accept Meeting. This goal is to be able to accept a meeting when the user has been invited.

Use Case Text 11		
Use Case Name		Accept Meeting
Use Case ID		4003
Description		To accept user invite to a meeting
Level		High-Level Goal
Primary Actor		Supervisor/ Student
Post-Condition		User has meetings waiting already in the database
Main Success Scenario		
Steps	User Interaction	System Response
1	The user clicks on Meeting	System loads a new user page
2	User clicks on ‘Accept Invite’ on the meeting	System reloads the Meetings page and inserts the update into the DB

Use Case Text 12 is Add Friend. This actor goal is to be able to add a friend who is on the system. This may be someone their supervisor also supervises or could be their friend from the same course.

Use Case Text 12		
Use Case Name		Add Friend
Use Case ID		5000
Description		User selects another user as a friend to follow
Level		Low-Level Goal
Primary Actor		Student
Post-Condition		User successfully logs in to their account
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on ‘Friends’	Loads new page for the user to view with required fields
2	User clicks on ‘Add A Friend’	Loads a new view containing students
3	Selects the user they wish to send a friend request too	System gets data ready to send to the server
4	User clicks ‘Send’	The system creates a friend request in the DB. Returns user to friend screen

Use Case Text 13 is Accept Friend Request. The actor will want to be able to accept a friend request from other students on their course. This is a low-level goal because it does not affect the aim of the project if it is not completed/ implemented by the deadline.

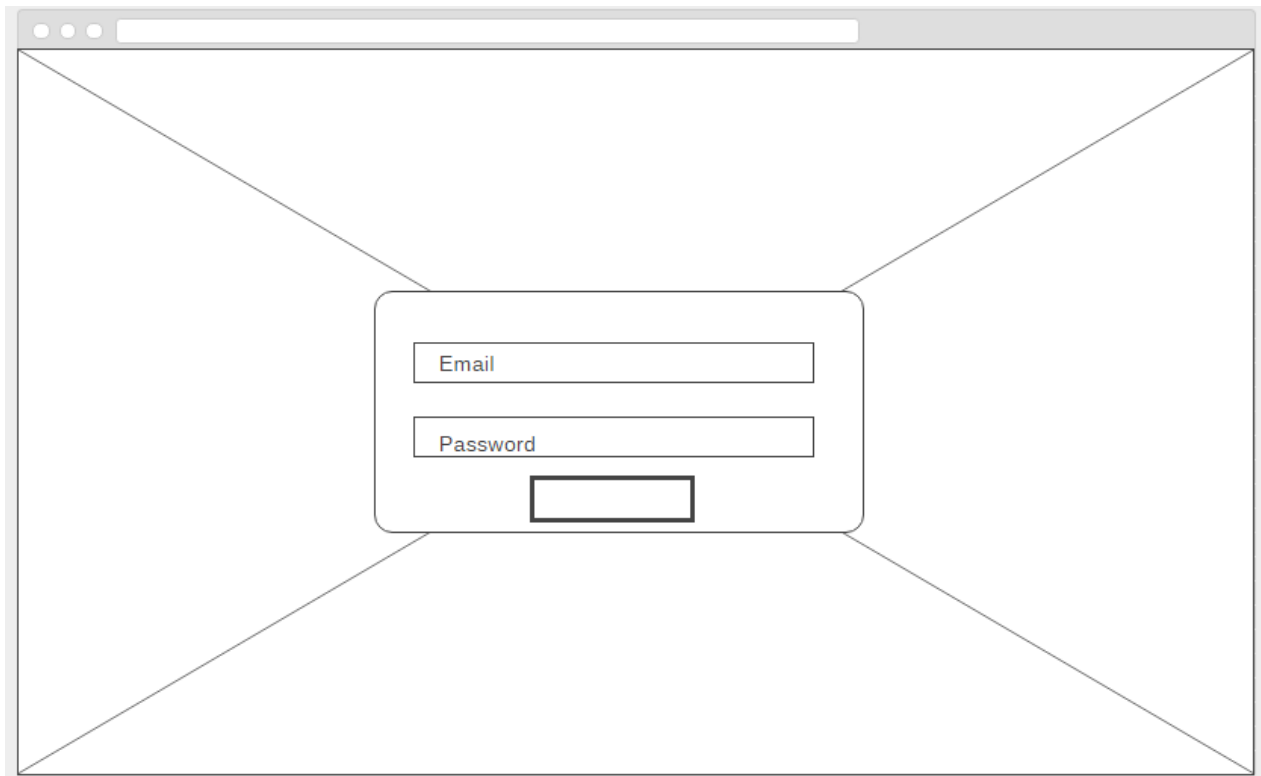
Use Case Text 13		
Use Case Name		Accept Friend Request
Use Case ID		5001
Description		User accepts a friend request from another fellow student
Level		Low-Level Goal
Primary Actor		Student
Post-Condition		User successfully logs in to their account and has a friend request sent
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on 'Friends'	Loads new page for the user to view
2	User clicks on 'Friend Request'	Loads a new view containing request(s)
3	Clicks on Request the User wants to accept	The system creates a link in DB.

Use Case Text 14 is view users. The actor will want the goal of being able to see every supervisor and student who is on the system and have access to view information about each.

Use Case Text 14		
Use Case Name		View Users
Use Case ID		6001
Description		Show all users on the system linked to the module leader
Level		Low-Level Goal
Primary Actor		Module Leader
Post-Condition		User successfully logs in to their account
Main Success Scenario		
Steps	User Interaction	System Response
1	User clicks on 'Users List'	System loads all users

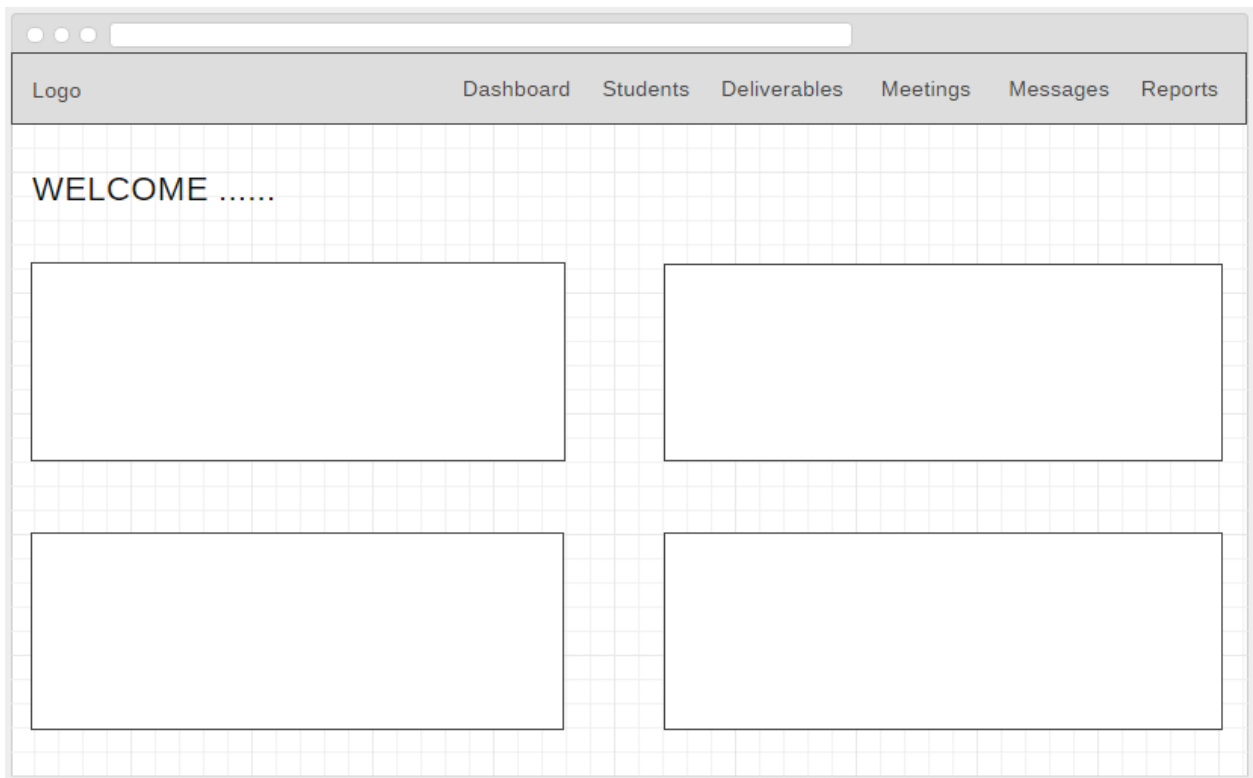
Wireframes

Index login page



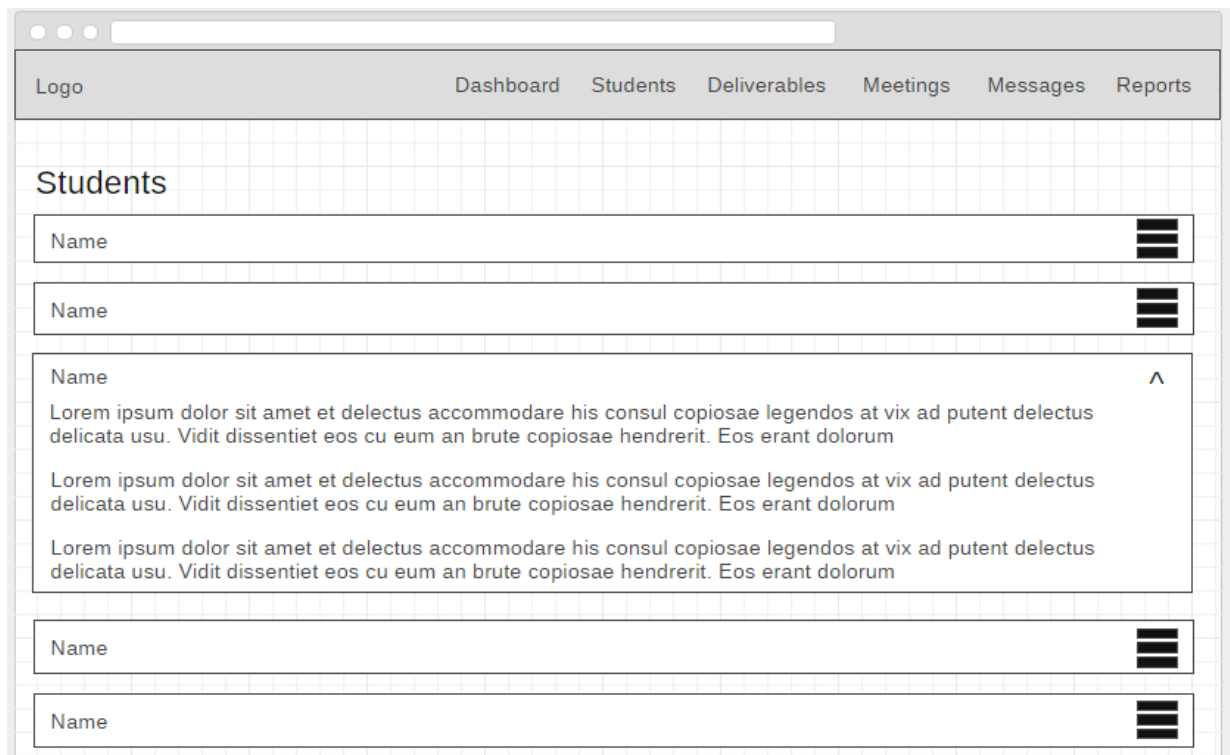
A wireframe of a login page within a browser window. The page features a central rounded rectangle containing three input fields: 'Email', 'Password', and a smaller, empty rectangular box below the password field. The browser window has a standard header with three dots and a search bar.

Supervisor Dashboard

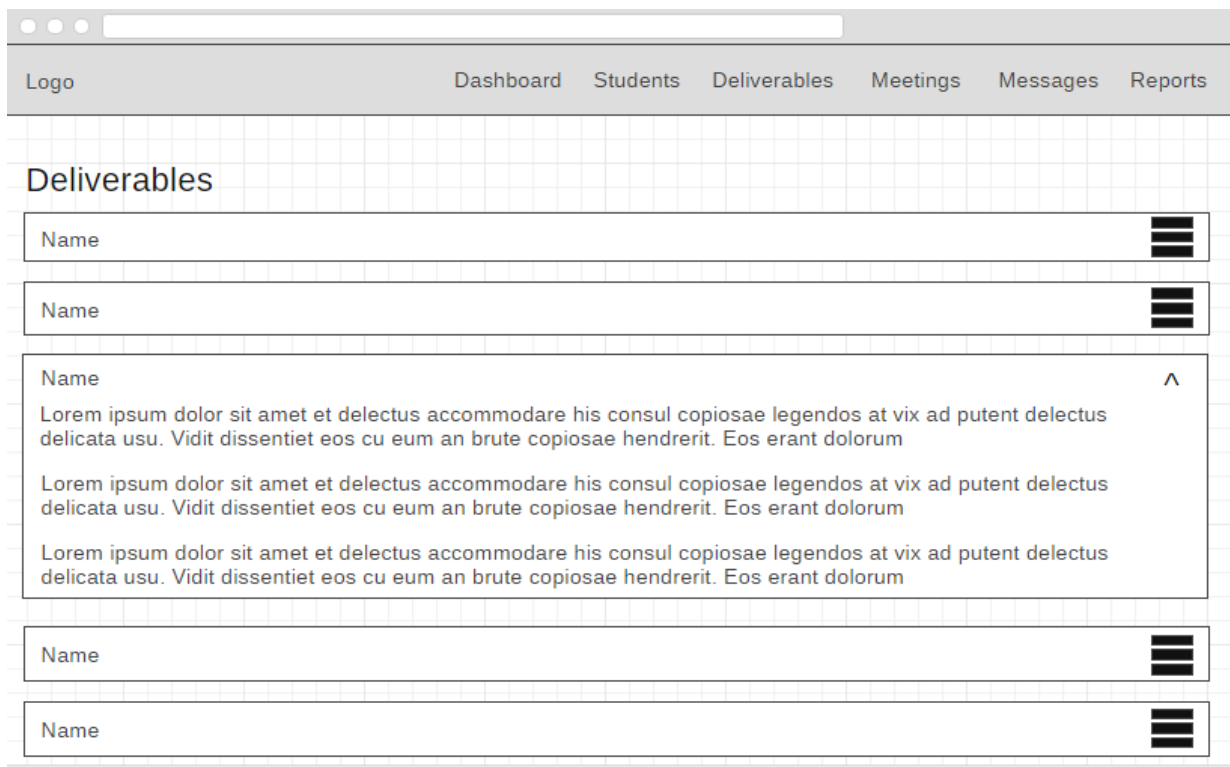


A wireframe of a Supervisor Dashboard within a browser window. The dashboard has a header bar with a 'Logo' on the left and navigation links for 'Dashboard', 'Students', 'Deliverables', 'Meetings', 'Messages', and 'Reports' on the right. Below the header, the text 'WELCOME' is displayed. The main content area is a grid with four large, empty rectangular boxes arranged in two rows and two columns.

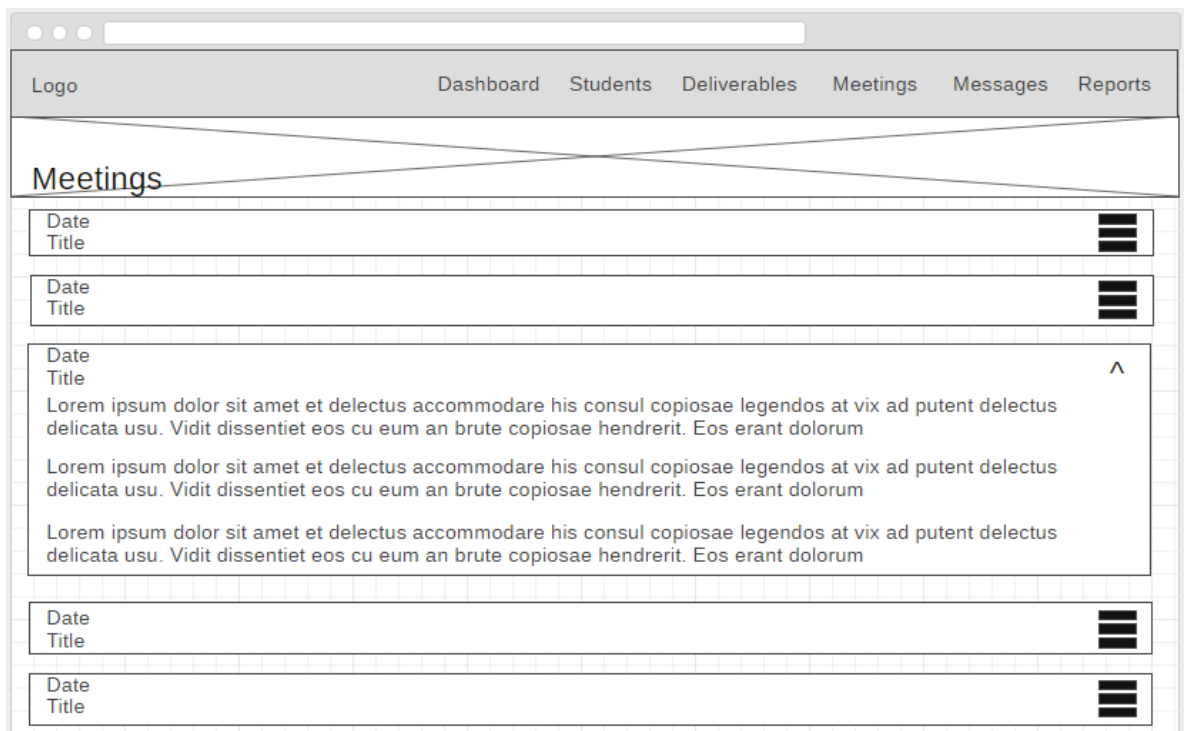
Supervisor Students



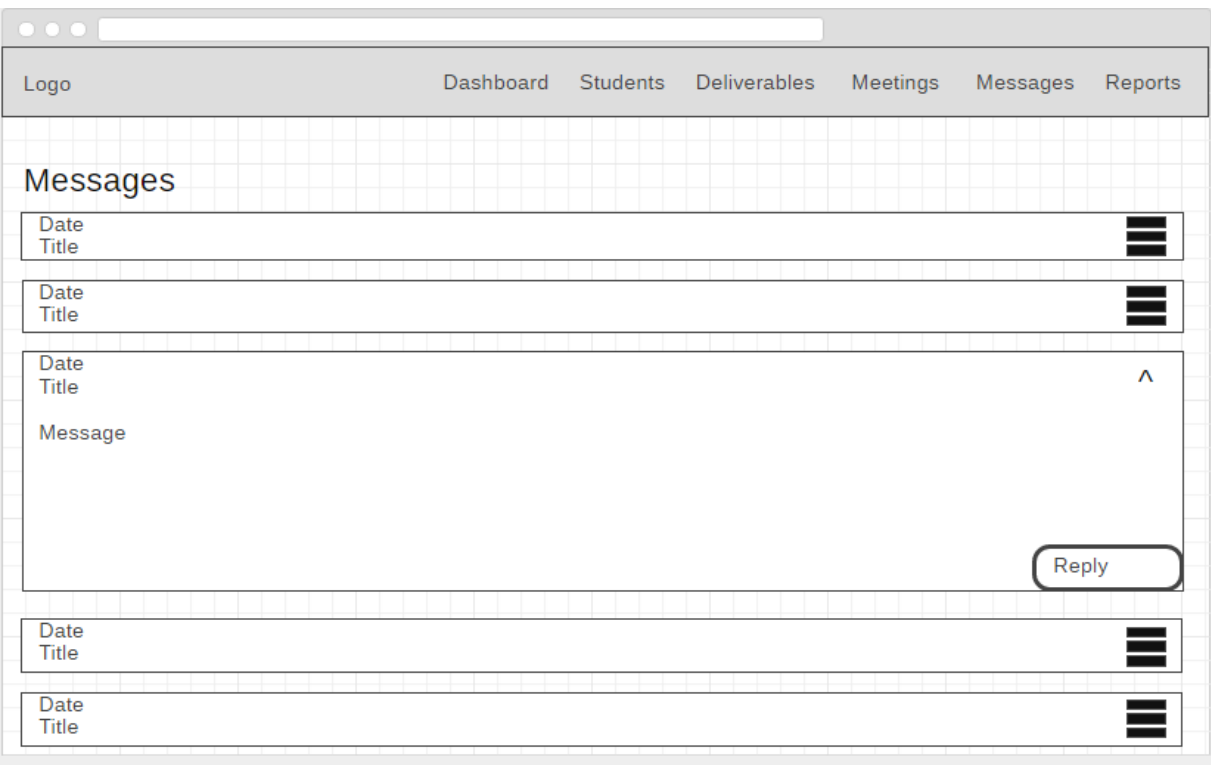
Supervisor student deliverables



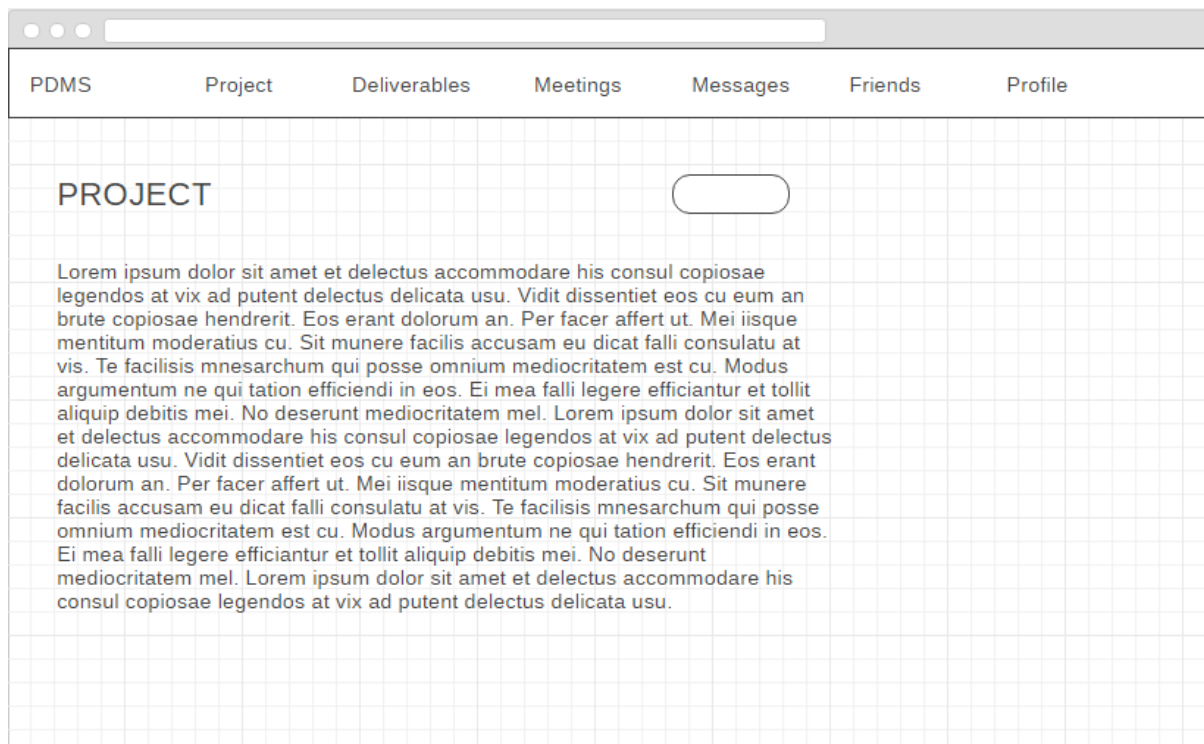
Supervisor meetings



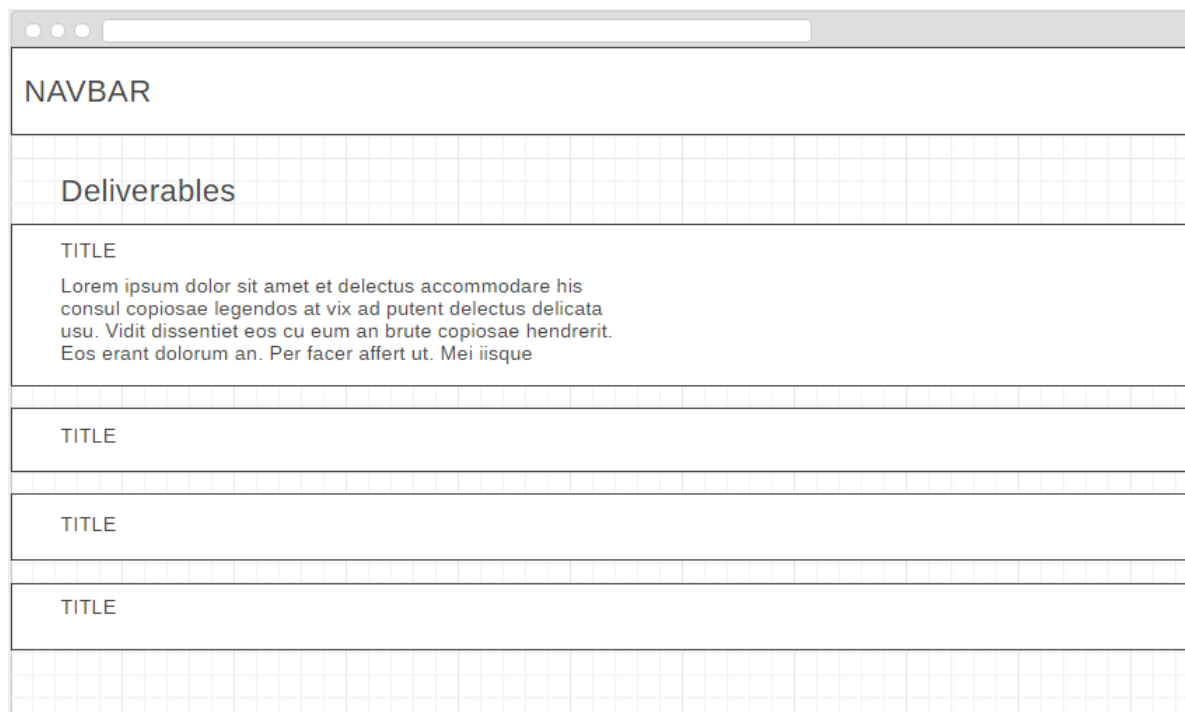
Supervisor messages



Student Dashboard



Student Deliverables



Student Meetings

NAVBAR

Meetings

TITLE

Lorem ipsum dolor sit amet et delectus accommodare his
consul copiosae legendos at vix ad putent delectus delicata
usu. Vidit dissentiet eos cu eum an brute copiosae hendrerit.
Eos erant dolorum an. Per facer affert ut. Mei iisque

TITLE

TITLE

TITLE

Student Inbox

NAVBAR

Messages - Inbox

Subject

Lorem ipsum dolor sit amet et delectus accommodare his
consul copiosae legendos at vix ad putent delectus delicata
usu. Vidit dissentiet eos cu eum an brute copiosae hendrerit.
Eos erant dolorum an. Per facer affert ut. Mei iisque

Subject

Subject

Subject

Student Outbox

● ● ●

NAVBAR

Messages - Outbox

Subject

Lorem ipsum dolor sit amet et delectus accommodare his
consul copiosae legendos at vix ad putent delectus delicata
usu. Vidit dissentiet eos cu eum an brute copiosae hendrerit.
Eos erant dolorum an. Per facer affert ut. Mei iisque

☐

Subject

☐

Subject

☐

Subject

☐

Student Friends

● ● ●

NAVBAR

Friends

Name

Lorem ipsum dolor sit amet et delectus accommodare his
consul copiosae legendos at vix ad putent delectus delicata
usu.

☐

Name

☐

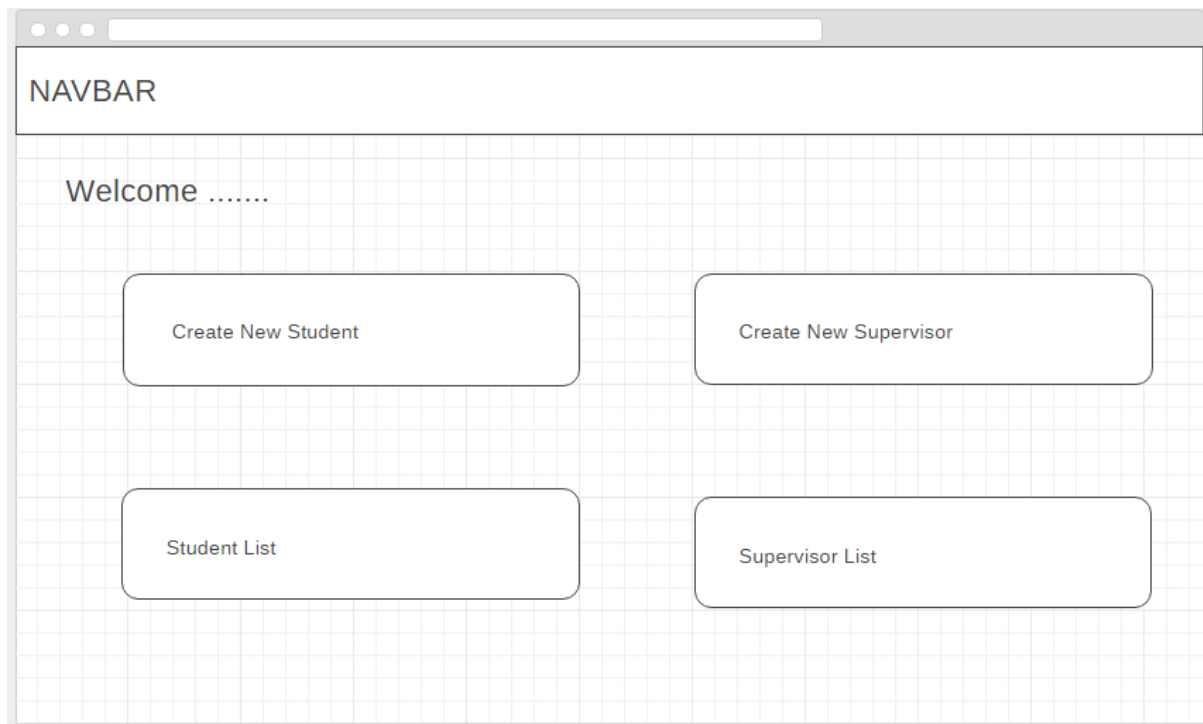
Name

☐

Name

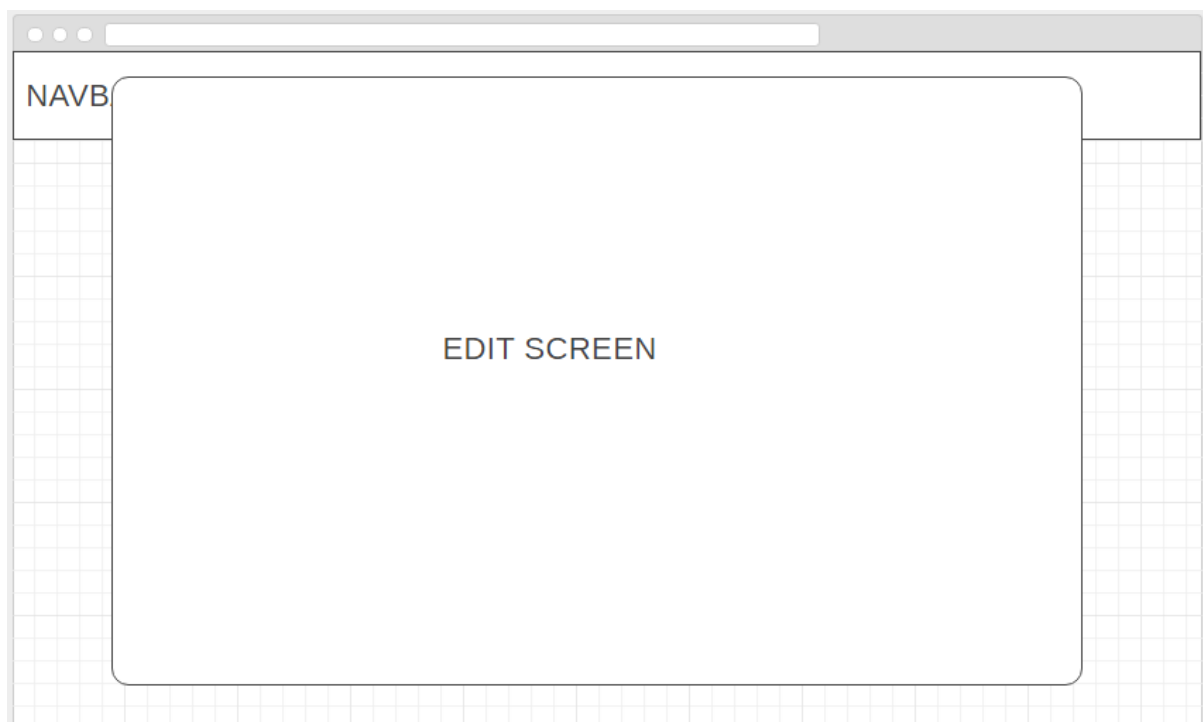
☐

Module Leader Dashbaord



Edit Screens

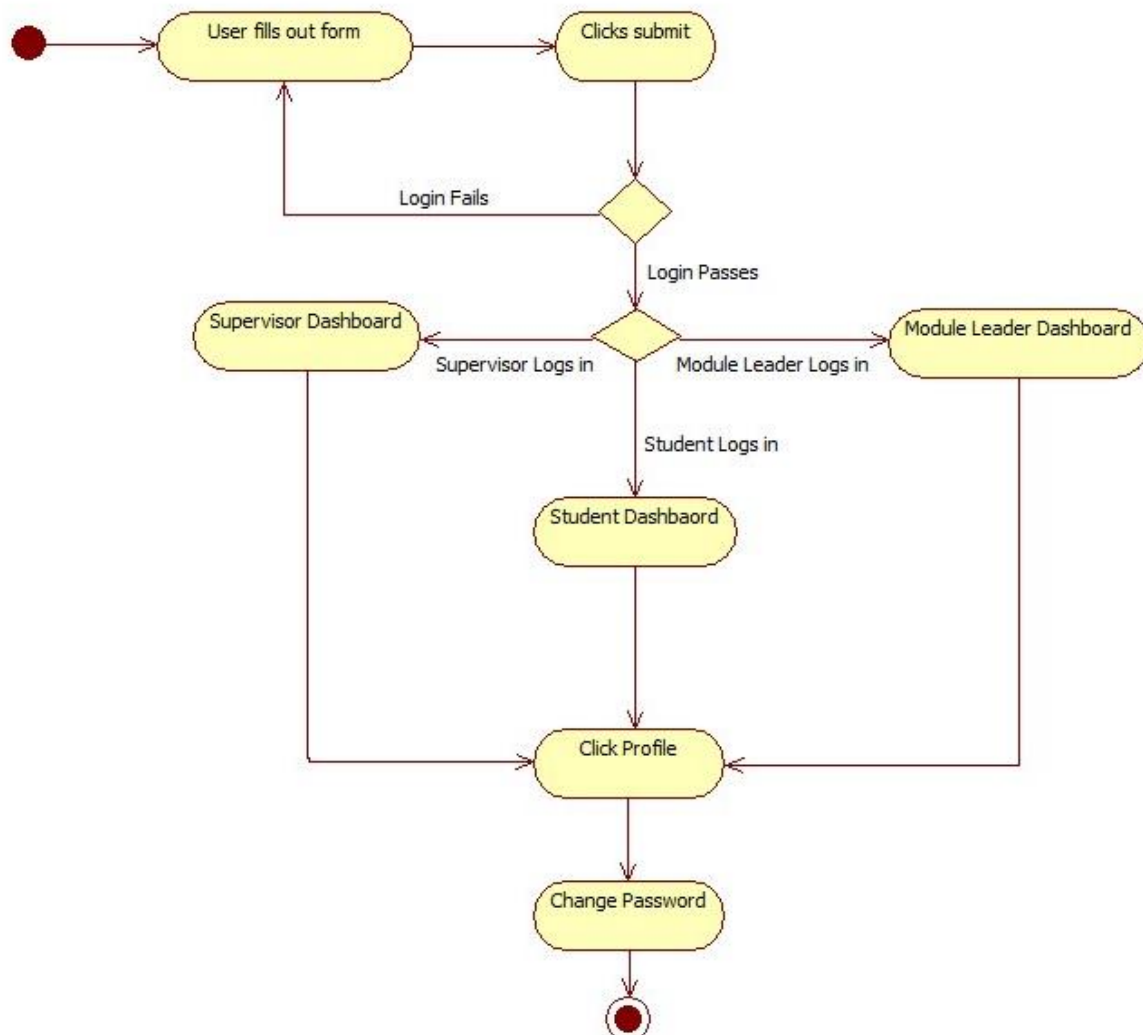
Edit screens will enter a pop up so they always go back to the last screen they was on.



Activity Diagrams

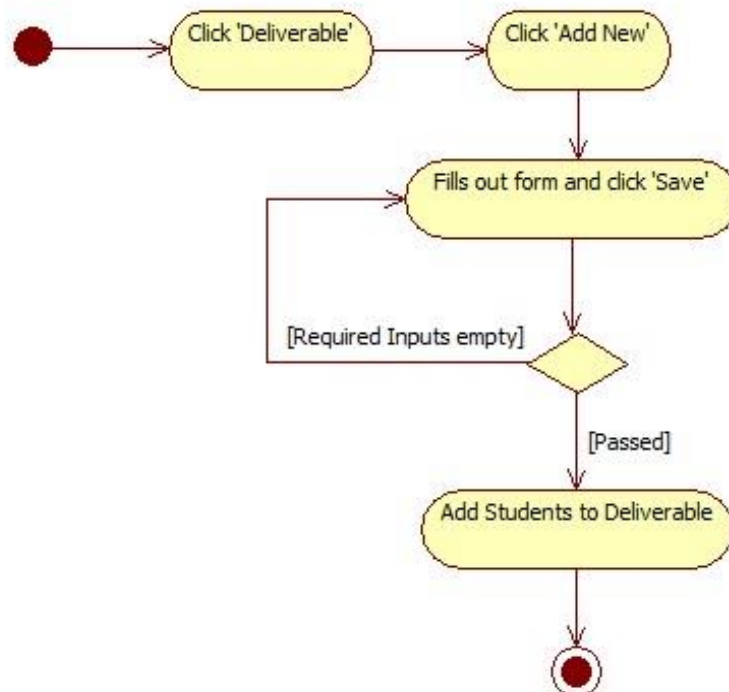
Log in, User Permission and Change Password

This diagram shows the progress off logging in by filling out the log in form. Once they have logged in successfully, the system then checks which user type they are and directs them to the correct dashboard. From this the user, will be able to go ahead and change their password.



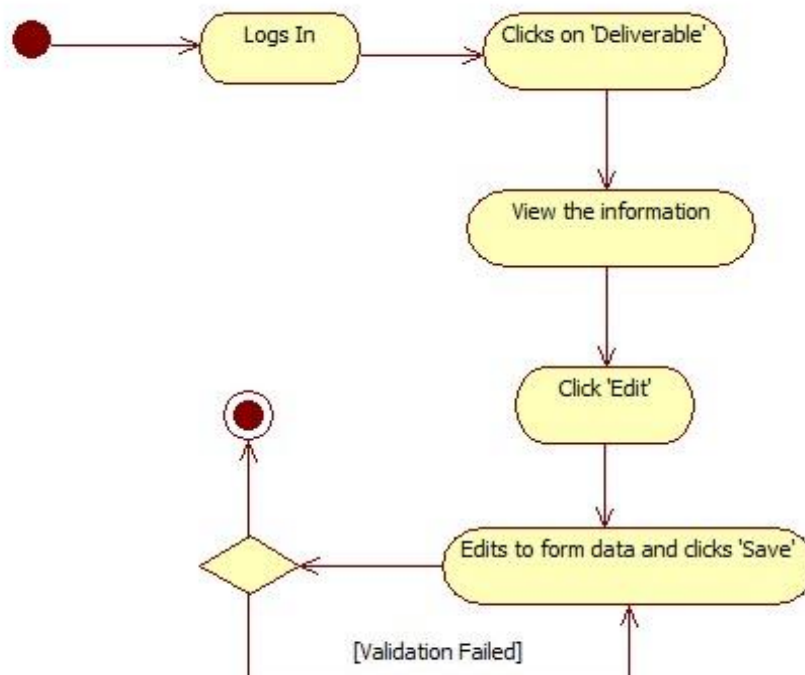
Add New Deliverable

Here the user is creating a new deliverable and then selecting the students which the deliverable applies too.



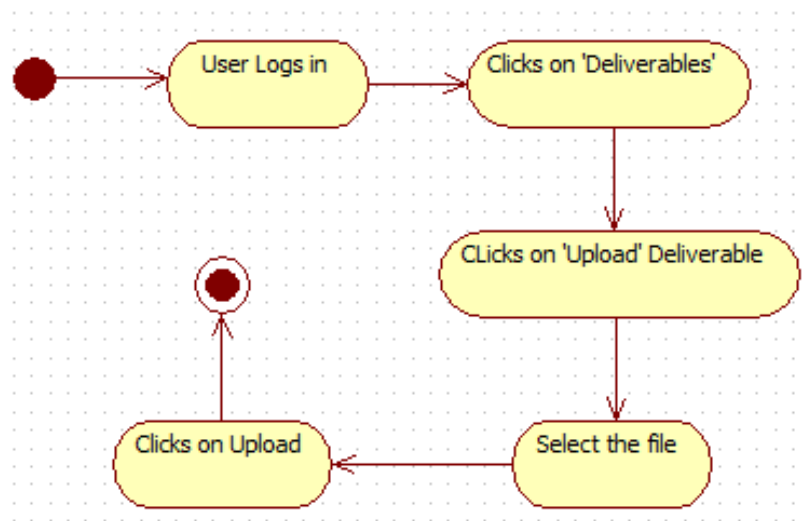
View and Edit Deliverable

The user will be able to view the/ their deliverables and edit them if required. Validation plays an important part for when the deliverable is being edited.



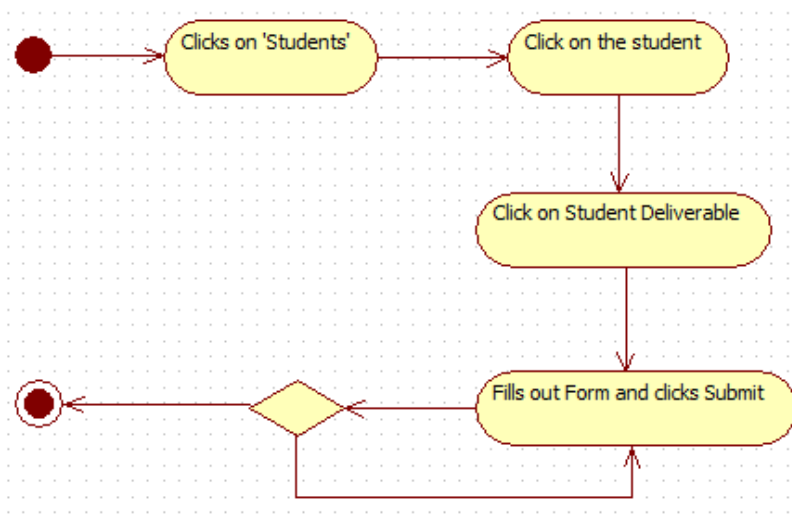
Upload Deliverable

The student user will need to upload deliverables they have completed. The process can be seen below.



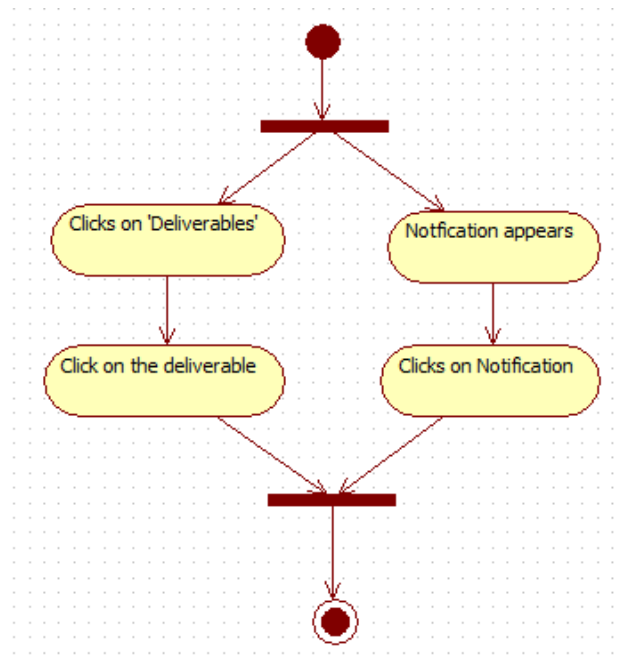
Create Feedback

Once students have submitted a deliverable, the supervisor will be able to give feedback. This is an important functionality which will support the student.



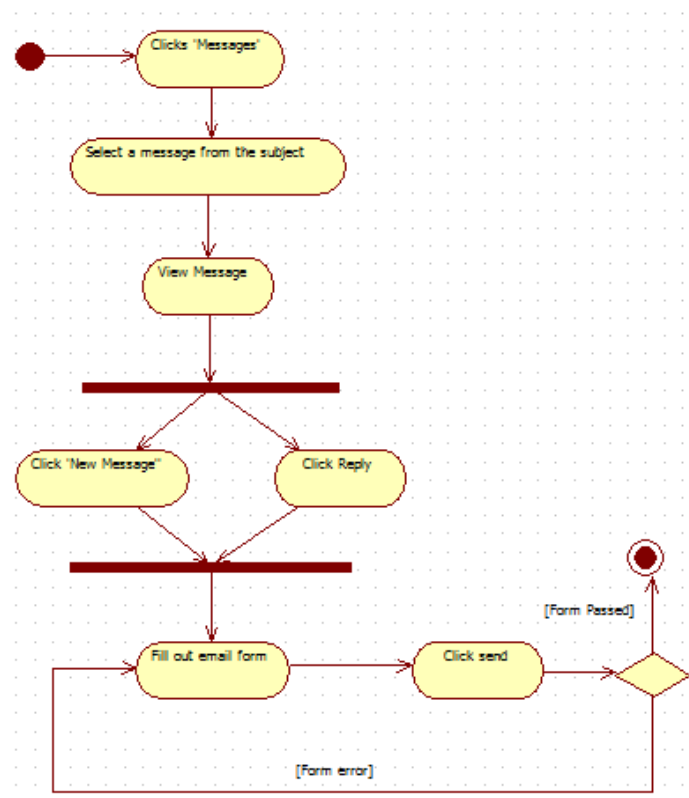
View Feedback

The users can view feedback which has been submitted to them.



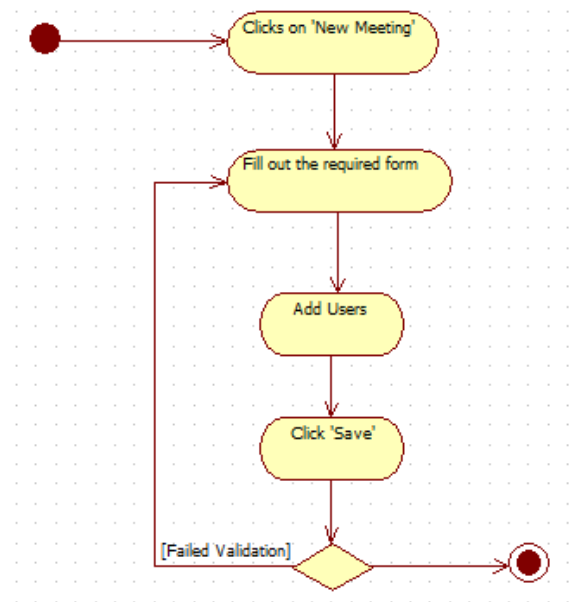
View and Send Message

The system will allow users to send and receive messages between each other. This is more of a complex diagram as there is an 'OR'. This is because the user can click on either button to continue their activity to reach their goal.



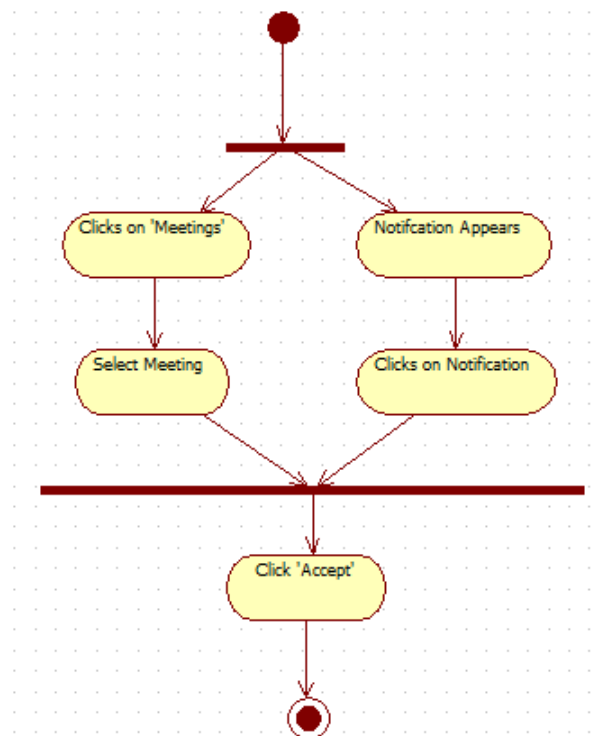
Create Meeting and Invite

Users will be able to create and invite other users to a meeting. This is a straight forward process to reach the goal, but the system will validate the data.



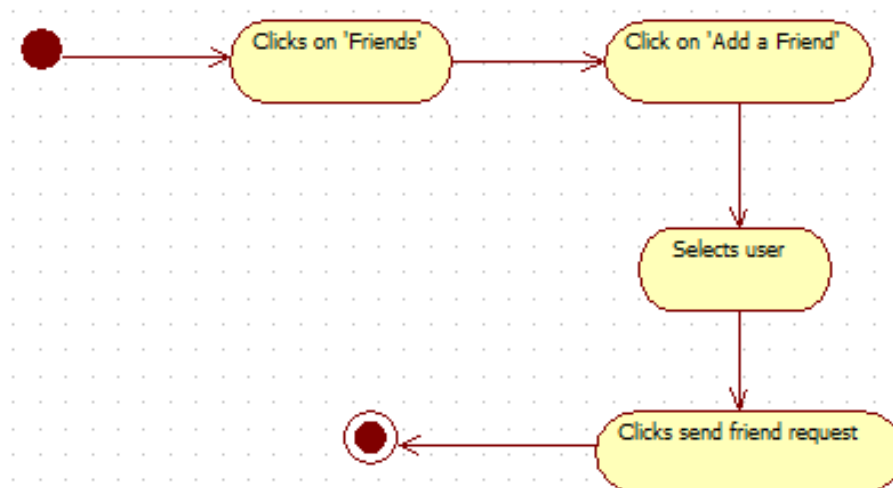
Accept Meeting

Once a user has been invited to a meeting, the user can select if they 'Accept' the meeting. This is couple of ways in which this can be achieved as is seen below.



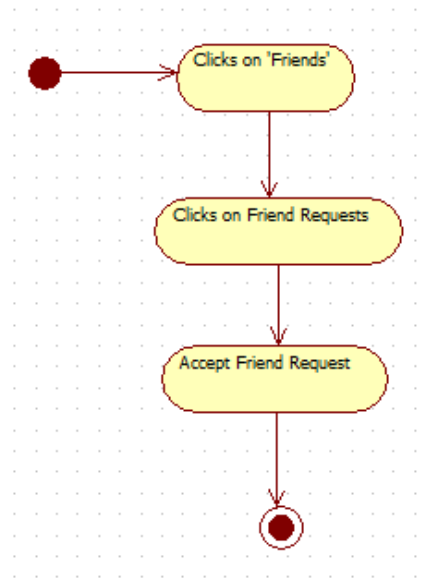
Add Friend

Students will be able to add their friends to view their progress of their final year project.



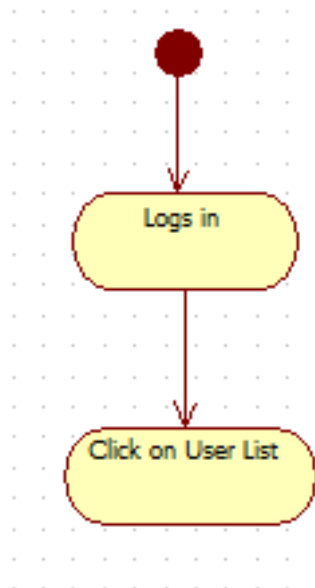
Accept Friend Request

User type Student will be able to accept the friends which can view their progress.



View Users

Module leaders will want to be able to view all the users which they are responsible for.



Data Dictionary

Relation Name	Attribute Name	Data Type	Length	Key Type	NOT NULL	Other Constraints
<u>Student Projects</u>	Project_id	CHAR	10	PK	NOT NULL	
	Student_id	CHAR	10	FK	NOT NULL	
	Date_started	Date			NOT NULL	
	Description	VARCHAR	255			
	Status	VARCHAR	255		NOT NULL	
	Title	VARCHAR	255		NOT NULL	
	year	Date			NOT NULL	
<u>Student</u>	Student_id	CHAR	10	PK	NOT NULL	
	User_id	CHAR	10	FK	NOT NULL	
	dob	DATE			NOT NULL	
	Supervisor_id	CHAR	10		NOT NULL	
<u>Friend Link</u>	Link_id	CHAR	10	PK	NOT NULL	
	Student_id_link_1	CHAR	10	FK	NOT NULL	
	Student_id_link_2	CHAR	10	FK	NOT NULL	
	status	VARCHAR	100		NOT NULL	
<u>Student Meetings</u>	Student_meeting_id	CHAR	10	PK	NOT NULL	
	Meeting_id	CHAR	10	FK	NOT NULL	
	Student_id	CHAR	10	FK	NOT NULL	
	Attendance_status	TINY INT	1		NOT NULL	
<u>Meetings</u>	Meeting_id	CHAR	10	PK	NOT NULL	
	Date_time	DATE TIME			NOT NULL	
	duration	CHAR	10		NOT NULL	
	Building_id	CHAR	10		NOT NULL	
	Description	VARCHAR	10		NOT NULL	

	Room	VARCHAR	10		NOT NULL	
	Title	VARCHAR	10		NOT NULL	
SupervisorMeetings	Supervisor_meeting_id	CHAR	10	PK	NOT NULL	
	Meeting_id	CHAR	10	FK	NOT NULL	
	Supervisor_id	CHAR	10	FK	NOT NULL	
Supervisor	Supervisor_id	CHAR	10	PK	NOT NULL	
	User_id	CHAR	10	FK	NOT NULL	
	department	VARCHAR	255			
User	User_id	CHAR	10	PK	NOT NULL	
	Date_cretared	DATE			NOT NULL	
	Last_logged_in	DATE				
	firstname	VARCHAR	255		NOT NULL	
	surname	VARCHAR	255		NOT NULL	
	password	VARCHAR	255		NOT NULL	
Messages	Message_id	CHAR	10	PK	NOT NULL	
	subject	VARCHAR	100		NOT NULL	
	Date_time	DATETIME			NOT NULL	
	message	VARCHAR	255		NOT NULL	
	Student_id	CHAR	10	FK	NOT NULL	
	Supervisor_id	CHAR	10	FK	NOT NULL	
	Sent_by	CHAR	10	FK	NOT NULL	
Department	Department_id	CHAR	10	PK	NOT NULL	
	department	VARCHAR	255		NOT NULL	
Deliverable	Deliverable_id	CHAR	10	PK	NOT NULL	
	Supervisor_id	CHAR	10	FK	NOT NULL	

	Deliverable_edit_id	CHAR	10	FK	NOT NULL	
	title	VARCHAR	255		NOT NULL	
	type	VARCHAR	255		NOT NULL	
	comments	VARCHAR	255		NOT NULL	
	date	DATE			NOT NULL	
	Date_due	DATE			NOT NULL	
	status	VARCHAR			NOT NULL	CHECK('Complete' 'Pending')
	Date_uploaded	DATE				
	Last_uploaded	TIMESTAMP				

Deliverable_edits	Deliverable_edit_id	CHAR	10	PK	NOT NULL	
	Deliverable_id	CHAR	10	FK	NOT NULL	
	Edited_by	CHAR	10	FK	NOT NULL	
	Approved_by	CHAR	10	FK	NOT NULL	
	Date_edited	DATE			NOT NULL	
	title	VARCHAR	255		NOT NULL	
	type	VARCHAR	255		NOT NULL	
	comments	VARCHAR	255		NOT NULL	
	date	DATE			NOT NULL	

Student_deliverable_feedback	Student_deliverable_feedback_id	CHAR	10	PK	NOT NULL	
	Student_deliverable_id	CHAR	10	FK	NOT NULL	
	Supervisor_id	CHAR	10		NOT NULL	
	feedback	VARCHAR	255		NOT NULL	
	Score	CHAR	10			

Student_deliverable	Student_deliverable_id	CHAR	10	PK	NOT NULL	
	Deliverable_id	CHAR	10	FK	NOT NULL	
	Student_id	CHAR	10	FK	NOT NULL	
	Project_id	CHAR	10	FK	NOT NULL	

Building	Building_id	CHAR	10	PK	NOT NULL	
	Location_id	CHAR	10	FK	NOT NULL	
	name	VARCHAR	255		NOT NULL	

Location	Location_id	CHAR	10	PK	NOT NULL	
	county	VARCHAR	255		NOT NULL	
	Line1	VARCHAR	255		NOT NULL	
	Line2	VARCHAR	255		NOT NULL	
	Campus	VARCHAR	255		NOT NULL	
	Postcode	VARCHAR	15		NOT NULL	
	town	VARCHAR	255		NOT NULL	

Supervisor Studnets Interface

PDMS

Dashboard

Students

Deliverables

Meetings

Messages

Reports ▾

Search

Students

+

Name

View Information

George Blue

⌵

Test Bob

⌵

Matt Clem

⌴

Project Name:

Testing

DOB:

1994-09-02

Gender:

Male

Uni ID:

K0000000

Matt Cole

⌵

Jenny Edwards

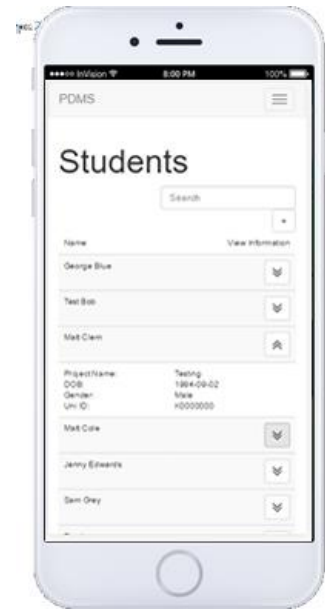
⌵

Sam Grey

⌵

Test 1

⌵



Supervisor Deliverables Interface

PDMS

Dashboard

Students

Deliverables

Meetings

Messages

Reports ▾

Search

Deliverables

+

Title

Action

Wireframes

⌵

Create Table of Contents for Design Chapter

⌵

Last Updated:

06/12/2016

Date Uploaded:

06/12/2016

Type:

documentation

Description:

Research and create TOC for the design chapter

Date Due:


2016-12-11


Status:

Pending

No of students:

8



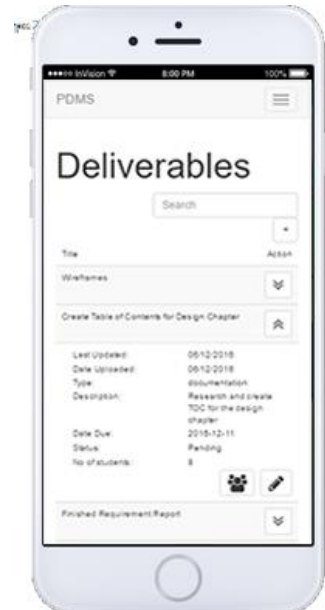


Finished Requirement Report

⌵

Insert and Update Statement

⌵



Add a new deliverable

The desktop view shows a modal window titled "Update Deliverable" with the following fields:

- Title:** Create Table of Contents for Design Chapter
- Due Date:** 2016-12-11
- Type:** documentation
- Description:** Research and create TOC for the design chapter

At the bottom of the modal are two buttons: "Cancel" and "Update Deliverable".

The mobile view shows the same "Update Deliverable" modal, adapted for a smaller screen. The fields and buttons are clearly visible and touch-friendly.

Add students to deliverable

The desktop view shows a modal window titled "Add Students to Deliverable" with a table of students and their assignment status:

Student Name (Project Title)	Action
George Blue	✓
Matt Clem	✓
Matt Cole	✓
Jenny Edwards	✓
Sam Grey	✓
Zoo Lu	✓
James McLean	✓
Peter Parker	+
Bob Test	+
Paul Work	+

The mobile view shows the "Add Students to Deliverable" modal, with the student list and action icons (checkmarks and plus signs) adapted for a smaller screen.

Supervisor Meetings Interface

uniPDMS Students Deliverables Meetings Messages Reports ▾ Graeme ▾

Search

Meetings

+

Title (Date)

Weekly Meetingg
16 May 2017
12:05

⌵

Date:

16 May 2017 12:05

Duration:

120 minutes

Room No:

109

Building:

John Galsworthy(Penrhyn Road)

Description:

Make sure you have information about your demos

Student Invited:

Matt Clem, Matt Cole, Tom McLean, James McLean, Jenny Edwards, Peter Parker, Sam Grey, George Blue, Zoo Lu, Paul Work, Tyler Watts, Sarah McLean

⌵

⌵

Weekly Meetingg
16 May 2017
12:05

⌵

Progress Reportt
10 May 2017
10:05

⌵

Weekly Meeting
10 May 2017
10:05

⌵

uniPDMS

Search

+

Title (Date)

Weekly Meetingg
16 May 2017
12:05

⌵

Date:

16 May 2017 12:05

Duration:

120 minutes

Room No:

109

Building:

John Galsworthy(Penrhyn Road)

Description:

Make sure you have information about your demos

Student Invited:

Matt Clem, Matt Cole, Tom McLean, James McLean, Jenny Edwards, Peter Parker, Sam Grey, George Blue, Zoo Lu, Paul Work, Tyler Watts, Sarah McLean

Student Deliverables Interface

uniPDMS Project Deliverables Meetings Messages ▾ Friends Tom ▾

Deliverables

Title

Create Table of Contents for Design Chaptera ✓

⌵

Finished Requirement Report ✓

⌵

Insert and Update Statement

⌵

Navigation/Sequence diagrams

⌵

Type:

1

Description:

Please complete and return by given date

Date Due:

2016-12-16

Status:

Pending

Uploaded:

No

⌵

Implement mockups of key webpages

⌵

Wireframes

⌵

uniPDMS

Deliverables

Title

Create Table of Contents for Design Chaptera ✓

⌵

Finished Requirement Report ✓

⌵

Insert and Update Statement

⌵

Navigation/Sequence diagrams

⌵

Type:

1

Description:

Please complete and return by given date

Date Due:

2016-12-16

Status:

Pending

Uploaded:

No

⌵

Implement mockups of key webpages

⌵

Student Meetings Interface

uniPDMS

Meetings

Title (Date)

Weekly Meeting
30 May 2017 13:00

⌵

Weekly Meetingg
16 May 2017 12:00

⌵

Date:

16 May 2017 12:00

Duration:

120 minutes

Location:

John Galsworthy (109)
Penrhyn Road
Kingston University
Penrhyn Road
Kingston upon Thames

Comments:

Make sure you have information about your demos

Accept Meeting?

✗

✓

Weekly Meeting
07 May 2017 13:00

⌵

Meetings

Title (Date)	Action
Weekly Meeting 30 May 2017 13:00	⌵
Weekly Meetingg 16 May 2017 12:00	⌵
<div> <div>Date:</div> <div>16 May 2017 12:00</div> </div> <div> <div>Duration:</div> <div>120 minutes</div> </div> <div> <div>Location:</div> <div> John Galsworthy (109) Penrhyn Road Kingston University Penrhyn Road Kingston upon Thames </div> </div> <div> <div>Comments:</div> <div>Make sure you have information about your demos</div> </div> <div> <div>Accept Meeting?</div> <div> <div>✖</div> <div>✔</div> </div> </div>	
Weekly Meeting 07 May 2017 13:00	⌵
Catch up 2 21 April 2017 14:00	⌵

Student Inbox Messages Interface

Messages - Inbox

Subject (Date)	Action
Feedback Graeme Jones 05 April 2017 09:00	⌵
Good Work Graeme Jones 01 April 2017 10:27	⌵



Student Inbox Messages Interface

uniPDMS Project Deliverables Meetings Messages Friends Tom		
Messages - Outbox		
Subject (Date)	Action	
Response to feedback Tom McLean 09 April 2017 13:00	✉	
test Tom McLean 18 April 2017 13:30	✉	
Meeting today? Tom McLean 18 April 2017 13:46	✉	
Hi Is the meeting still on today? Tom		
Gello Tom McLean 18 April 2017 15:18	✉	



Student Create Messages Interface

uniPDMS Project Deliverables Meetings Messages Friends Tom		
Messages - Outbox		
Subject (Date)	Action	
Response to feedback Tom McLean 09 April 2017 13:00	✉	
test Tom McLean 18 April 2017 13:30	✉	
Meeting today? Tom McLean 18 April 2017 13:46	✉	
Hi Is the meeting still on today? Tom		
Gello Tom McLean 18 April 2017 15:18	✉	

Create Message to Supervisor

Subject

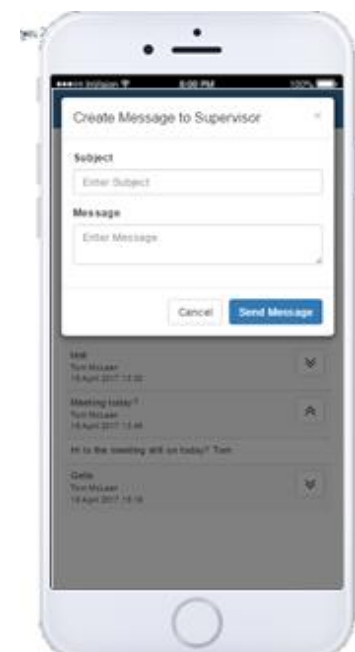
Enter Subject

Message

Enter Message

Cancel

Send Message

































Example data in tables

As there is lots of data in each table, only a small of data will be shown in each print screen.

Users

user_id	gender	user_name	user_type	date_created	last_logged_in	firstname	surname	password	email
35	Male	graemejones1	1	2016-12-06	2016-12-06 06:59:49	Graeme	Jones	0f1fe6e61caff48eb9e4e4739091472132343f91	gjones@kingston.ac.uk
36	Male	smithjohn1	1	2016-12-06	2016-12-06 07:11:01	John	Smith	0f1fe6e61caff48eb9e4e4739091472132343f91	jsmith@kingston.ac.uk
39	Male	paulneve1	1	2016-12-06	2016-12-06 07:19:11	Paul	Neve	0f1fe6e61caff48eb9e4e4739091472132343f91	pneve@kingston.ac.uk
41	Male	timbloggs1	1	2016-12-06	2016-12-06 07:22:06	Tim	Bloggs	0f1fe6e61caff48eb9e4e4739091472132343f91	tbloggs@kingston.ac.uk
42	Male	benfrost1	1	2016-12-06	2016-12-06 07:23:00	Ben	Frost	0f1fe6e61caff48eb9e4e4739091472132343f91	bfrost@kingston.ac.uk
43	Male	rockywatts1	1	2016-12-06	2016-12-06 07:32:25	Rocky	Watts	0f1fe6e61caff48eb9e4e4739091472132343f91	rockywatts@kingston.ac.uk
44	Male	billgates1	1	2016-12-06	2016-12-06 07:32:53	Bill	Gates	0f1fe6e61caff48eb9e4e4739091472132343f91	billgates123@kingston.ac.uk
45	Male	bootstrap1	1	2016-12-06	2016-12-06 07:33:29	Boot	Strap	0f1fe6e61caff48eb9e4e4739091472132343f91	booty@kingston.ac.uk
47	Male	franklampard1	1	2016-12-06	2016-12-06 07:34:05	Frank	Lampard	0f1fe6e61caff48eb9e4e4739091472132343f91	lampard8@kingston.ac.uk
48	Male	terryblue1	1	2016-12-06	2016-12-06 07:37:45	Terry	Blue	0f1fe6e61caff48eb9e4e4739091472132343f91	terry26@kingston.ac.uk
49	Male	luke123	1	2016-12-06	2016-12-06 07:40:11	Luke	Hull	0f1fe6e61caff48eb9e4e4739091472132343f91	luke123hull@kingston.ac.uk

Building Table

		building_id	name	location_id
<input type="checkbox"/>	 Edit  Copy  Delete	0	John Galsworthy	0
<input type="checkbox"/>	 Edit  Copy  Delete	1	Sopwith	0
<input type="checkbox"/>	 Edit  Copy  Delete	2	Eadweard Muybridge	0
<input type="checkbox"/>	 Edit  Copy  Delete	3	Learning resources centre	0
<input type="checkbox"/>	 Edit  Copy  Delete	4	Business School	1
<input type="checkbox"/>	 Edit  Copy  Delete	5	Lawley lecture theatre	1
<input type="checkbox"/>	 Edit  Copy  Delete	6	Frank Lampl	1
<input type="checkbox"/>	 Edit  Copy  Delete	7	Hawker Wing	2
<input type="checkbox"/>	 Edit  Copy  Delete	8	Studios	3
<input type="checkbox"/>	 Edit  Copy  Delete	9	Nurse Class Room	4

Deliverables Table

deliverable_id	supervisor_id	title	type	comments	date_due	status	date_uploaded	last_updated
1	22	Wireframes	3	Please create a few wireframes to show the flow of...	2017-05-05	Pending	2016-12-06 19:00:00	2017-04-20 00:36:29
2	22	Create Table of Contents for Design Chapter	1	Research and create TOC for the design chapter	2016-12-11	Pending	2016-12-06 22:00:00	2016-12-06 22:00:00
3	22	Finished Requirement Report	1	Please submit by turn it in by this date. Want an...	2016-12-16	Pending	2016-12-08 10:00:00	2016-12-08 10:00:00
4	22	Insert and Update Statement	6	Produce the code to insert and update deliverables...	2016-12-22	Pending	2016-12-07 13:31:31	2016-12-08 12:38:30
5	22	Navigation/Sequence diagrams	1	Please complete and return by given date	2016-12-16	Pending	2016-12-08 06:19:24	2016-12-08 06:19:24
6	22	Identify SQL queries from requirements	1	Produce 5 SQL queries from requirements	2016-12-23	Pending	2016-12-08 13:00:00	2016-12-08 16:00:00
7	22	Implement mockups of key webpages	5	Produce Mock ups of key webpages	2016-12-08	Pending	2016-12-22 12:00:00	2016-12-22 12:00:00
8	22	Testing	3	Testing 123 456 gnitset	2017-01-28	Pending	2017-01-28 00:00:00	2017-01-29 00:00:00
9	25	Design Chapter	1	Design chapter complete	2017-04-24	Pending	2017-04-12 00:00:00	2017-04-12 00:00:00

Deliverable_Type Table

deliverable_type_id	deliverable_type_name
1	Documentation
2	Methodology
3	Submission
4	Technology
5	Design
6	Implementation

Departments

department_id	department
The department the supervisor works for	
1	Computer Science
2	Information Systems

Location

location_id	campus	line1	line2	town	county	postcode
Click the drop-down arrow to toggle column's visibility		Kingston University	Penrhyn Road	Kingston upon Thames	Surrey	KT1 2EE
2	Roehampton Vale	Kingston University	Friars Avenue	Roehampton	London	SW15 3DW
3	Knights Park	Kingston University	Grange Road	Kingston Upon Thames	Surrey	KT1 2QJ
1	Kingston Hill	Kingston University	Kingston Hill	Kingston Upon Thames	Surrey	KT2 7LB
4	St Georges	St George's University of London	Cranmer Terrace	London	London	SW17 0RE

Meetings

meeting_id	date_time	Title	duration	description	building_id	room
1	2017-01-10 12:30:00	Weekly Meeting	90	Weekly meetings to see how you are progressing in ...	0	209
2	2017-01-04 13:30:00	Weekly Meeting	60	Weekly meetings to see how you are progressing in ...	0	122
3	2017-01-10 13:00:00	Weekly Meeting	120	Weekly meetings to see how you are progressing in ...	0	2
4	2017-03-10 13:00:00	Weekly Meeting	90	Weekly meetings to see how you are progressing in ...	0	211
5	2017-03-10 13:00:00	Weekly Meeting	90	Weekly meetings to see how you are progressing in ...	0	65
6	2017-01-17 13:00:00	Weekly Meeting	90	Weekly meetings to see how you are progressing in ...	0	32
7	2017-01-24 12:00:00	Weekly Meeting	60	Weekly meetings to see how you are progressing in ...	0	233
8	2017-02-07 13:00:00	Weekly Meeting	60	Weekly meetings to see how you are progressing in ...	0	541
9	2017-05-16 12:00:00	Weekly Meetingg	120	Make sure you have information about your demos	0	109

Messages

message_id pk	subject	date_time	message	user_from fk	user_to fk	flagged true or false
1	Feedback	2017-04-05 09:00:00	please come along to the meeting today to get some...	35	54	0
2	Response to feedback	2017-04-09 13:00:00	Okay. I will be a the meeting later.	54	35	0
3	Good Work	2017-04-01 10:27:25	Your final year project is coming along well. Keep...	35	54	0
4	Testing	2017-04-16 15:13:52	Testing this works	5	22	0
5	Testing 2	2017-04-16 15:15:15	testing	5	22	0
6	Online Sweet Shop	2017-04-18 13:11:26	Please let me know if you like my idea I have had ...	9	22	0
7	test	2017-04-18 13:15:37	test	9	22	0
8	test	2017-04-18 13:30:56	test	54	35	0
9	Pick and Mix Online Shop	2017-04-18 13:45:27	Just wondering if you liked my project idea? I hav...	58	35	0
10	Meeting	2017-04-18 13:46:54	Hi	54	35	0

Student

← T →	student_id	user_id	dob	supervisor_id	uni_id University Unique Id
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	5	54	1992-02-01	22	K1038383
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	6	55	1994-06-08	22	K1038282
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	7	56	1993-01-02	22	K0000000
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	8	57	1994-11-29	22	K1013223
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	9	58	1994-06-06	22	K2013066
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	10	61	1991-12-11	22	K1029933
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	11	62	1991-12-11	22	K0432212
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	12	63	1990-10-02	22	K1029382
<input type="checkbox"/> Edit <input type="copy"/> Copy <input type="delete"/> Delete	13	65	1990-10-02	22	K0000000

Student_Deliverable Table

student_deliverable_id	deliverable_id	project_id	Complete <small>true or false</small>	student_complete	dir
13	2	1	1	0	../../../../deliverables_submitted/54/Thomas_McLean_d...
14	2	2	0	0	
15	2	4	0	0	
16	2	5	0	0	
17	2	7	0	0	
18	2	11	0	0	
19	3	1	1	0	../../../../deliverables_submitted/54/testfile.txt
20	3	2	0	0	

Student_Meetings

student_meeting_id	meetings_id	student_id	attendance_status
1	1	15	0
2	1	34	0
3	1	14	0
4	1	12	0
5	1	9	0
6	1	11	2
7	1	37	1
8	1	21	1
9	1	7	1
10	1	20	1

Supervisor

supervisor_id	user_id <small>FK to user table</small>	department_id
22	35	1
23	36	1
25	39	1
26	41	1
27	42	1
28	43	1

Student_Projects

project_id	student_id	date_started	status	title <small>Title of student FYP</small>	year	description
1	5	2016-09-01	Pending	PDMS Project	2016	Building a Project Deliverable Monitoring System
2	6	2016-08-01	Pending	Testing	2016	No description set
3	7	2016-09-05	Pending	My project	2016	No description set
4	8	2016-10-04	Pending		2016	Sutton Website
5	9	2016-10-11	Pending	Pick and Mix Online Shop	2016	Imagine Wilko's 'pick and Mix' but online! That's ...
6	10	2016-12-01	Pending	Testing	2016	No

Supervisor Meetings Table

supervisor_meetings_id	meeting_id	supervisor_id <small>fk</small>
1	1	22
2	2	22
3	3	22
4	4	22
5	5	22
6	6	22
7	8	22
8	9	22
9	9	22
10	10	22

Testing

Login Functional testing

Test Case ID	Functional Requirement	Steps (Description)	Expected Results	Pass/Fail	Comments
1	Student Log in	1. Enter URL 2. Enter login details for student 3. Click login	Application will take the user to the student area	Pass	
1.2	Supervisor log in	1. Enter URL 2. Enter login details for supervisor 3. Click login	Application will take the user to the supervisor area	Pass	
1.3	Module Leaders log in	1. Enter URL 2. Enter login details for module leader 3. Click login	Application will take the user to the module leader area	Pass	
1.4	Restrict unauthorised login	1. Log in as a student user. 2. Type in supervisor URL	To be redirected to the student dashboard	Pass	
1.5	Log out	1. Click log out in the navbar	Session data to be killed and user to be back at the login screen	Pass	

Supervisor Functional Testing

2	View student data	1. Click student on the nav bar 2. Click on a student row to view information	Information to be able to be viewed	Pass	
2.1	Search students list	1. Click on student in the navbar 2. Type in the search box	The correct students to show to the search	Pass	
2.2	Add deliverable	1. Click on deliverables in the navbar 2. Click on the + button 3. Add new deliverable information 4. Click add new deliverable	New deliverable to be inserted into the database and to show	Pass	
2.3	View deliverable information	1. Click on deliverable on the navbar 2. Click on one of the deliverables	Information to be shown on click.	Fail	Correct information did not show
2.3.1 Retest	View deliverable information	1. Click on deliverable on the navbar 2. Click on one of the deliverables	Information to be shown on click.	Pass	Correct information showed correctly.
2.4	Add student to deliverable	1. Click on deliverable on the navbar 2. Click on a deliverable 3. Click on the icon to add students to a	Number of students to update within the deliverable information.	Pass	

		deliverable 4. Selected students that wish to be added 5. Click 'Add Students'			
2.5	Edit deliverable	1. Click on deliverable on the navbar. 2. Click Edit pen icon) 3. Edit deliverable and click update	Deliverable to display with the latest version	Fail	Did not update correctly
2.5.1 Retest	Edit deliverable	1. Click on deliverable on the navbar. 2. Click Edit pen icon) 3. Edit deliverable and click update	Deliverable to display with the latest version	Pass	Updated correctly
2.6	View meeting data	1. Click on meetings on the navbar. 2. Click on a meeting in the table	Information of the meeting to appear	Pass	
2.7	Add new meeting	1. Click on meetings on the navbar. 2. Click the + button 3. Fill in form 4. Click Add new meeting	New meeting to appear	Pass	
2.8	Edit meeting	1. Click on meetings on the navbar. 2. Click on a meeting from the table 3. Click Edit meeting 4. Update meeting 5. Click update meeting	New meeting to appear	Pass	
2.9	Add students to meeting	1. Click on meetings on the navbar. 2. Click on a meeting from the table 3. Click add new students to meeting button 4. Select students 5. Click update	Meeting to be update in table and student to receive a notification	Pass	
2.10	View Inbox Message	1. Click on messages then inbox. 2. Click on button to view message	The system to show the message	Pass	
2.11	Send Message	1. Click on message inbox 2. Click on + button 3. Fill in form 4. Click send message	Message sends to student	Pass	
2.12	View Outbox message	1. Click on messages then outbox. 2. Click on button to	The system to show the message	Pass	

		view message			
2.13	View Deliverable Report	1. Click on reports 2. Click on Deliverable Report	System shows the report	Pass	
2.14	View Uploaded Deliverables	1. Click on Uploaded Deliverables	System shows list of uploaded deliverables	Pass	

Non-Functional Test Cases

User with no programming experience were invited to test the system. A table has been created to show which goals they were asked to test with a success of fail.

Student

Only students were invited to take part in the student application test.

User one (Male, 20, Student)		
Goal	Success / Fail	Comment
Edit the project description	Success	User Edited the project description quickly and easily
Upload a deliverable	Success	Deliverable was uploaded successfully
Accept the meeting 'Weekly meeting 2'	Success	Meeting was found quickly and the correct description was read back.
Accept a friend request from Tom	Success	Friend request accepted
Read message with on the date of 11/12/2017	Success	Message read successfully
Send message	Success	Message sent

User two (Male, 23, Student)		
Goal	Success / Fail	Comment
Edit the project description	Success	Successfully done
Upload a deliverable	Success	Deliverable was uploaded successfully
Accept the meeting 'Weekly meeting 2'	Success	Meeting was found quickly and the correct description was read back.
Accept a friend request from Tom	Success	Friend request accepted
Read message with on the date of 11/12/2017	Success	Message read successfully
Send message	Success	Message sent

User three (Female, 22, Student)		
Goal	Success / Fail	Comment
Edit the project description	Success	User Edited the project description quickly and easily
Upload a deliverable	Success	Deliverable was uploaded successfully
Accept the meeting 'Weekly meeting 2'	Success	Meeting was found quickly and the correct description was read back.
Accept a friend request from Tom	Success	Friend request accepted
Read message with on the date of 11/12/2017	Success	Message read successfully

Send message	Success	Message sent
--------------	---------	--------------

Supervisor

Only supervisor was invited to take part in the student application test.

User one (Male, 55, Student)

Goal	Success / Fail	Comment
Create a new deliverable and add it the students	Success	Completed successfully
Create a new meeting and add students	Success	Completed successfully. Meeting test created
Look at your inbox and read the message from jenny	Success	Completed successfully
How many students are being supervised	Success	Answered correctly
View the deliverable report and read how many people completed the wireframe deliverables	Success	Struggled at first but then came back with the correct answer
Send message	Success	Successfully completed

User two (Male, 38, Student)

Goal	Success / Fail	Comment
Create a new deliverable and add it the students	Success	Completed successfully
Create a new meeting and add students	Success	Completed successfully. Meeting 'meeting' created
Look at your inbox and read the message from jenny	Success	Completed successfully
How many students are being supervised	Success	Answered correctly
View the deliverable report and read how many people completed the wireframe deliverables	Success	Successfully completed
Send message	Success	Successfully completed