

**FACULTY OF SCIENCE, ENGINEERING AND  
COMPUTING**

***School of Computer Science & Mathematics***

**BSc DEGREE**

**IN**

***Computer Science***

**PROJECT PROPOSAL**

Name: Stefan Tataru

ID Number: K1809106

Project Title: Agile Project Management Tool

Project Type: Build

Date: 30/04/2021

Supervisor: Graeme Jones

**Kingston University** London

Did you discuss and agree the viability of your project idea with your supervisor?	Yes
Did you submit a draft of your proposal to your supervisor?	Yes
Did you receive feedback from your supervisor on any submitted draft?	Yes

# Contents

<b>1 Introduction and Background .....</b>	<b>5</b>
<b>1.1 Problem.....</b>	<b>5</b>
<b>1.2 Aim .....</b>	<b>5</b>
<b>1.3 Objectives: .....</b>	<b>6</b>
<b>1.4 Solution .....</b>	<b>6</b>
<b>1.5 Method .....</b>	<b>6</b>
<b>1.6 Work Plan.....</b>	<b>6</b>
<b>2. Literature Review .....</b>	<b>8</b>
<b>2.1 Introduction.....</b>	<b>8</b>
<b>2.2 Agile.....</b>	<b>8</b>
<b>2.2.1 Scrum .....</b>	<b>8</b>
<b>2.2.2 Kanban.....</b>	<b>9</b>
<b>2.3 Waterfall vs Agile.....</b>	<b>9</b>
<b>2.4 Conclusion .....</b>	<b>10</b>
<b>3 Requirements.....</b>	<b>11</b>
<b>3.1 Introduction.....</b>	<b>11</b>
<b>3.2 SWOT Analysis.....</b>	<b>11</b>
<b>3.2.1 Introduction.....</b>	<b>11</b>
<b>3.2.2 Recommendation.....</b>	<b>12</b>
<b>3.2.3 Mitigating Threats and Weaknesses.....</b>	<b>Error! Bookmark not defined.</b>
<b>3.3 Stakeholders .....</b>	<b>12</b>
<b>3.4 User stories .....</b>	<b>13</b>
<b>3.5 Use Case Diagram .....</b>	<b>13</b>
<b>3.6 Use cases.....</b>	<b>14</b>
<b>3.7 Functional and Non-Functional Requirements .....</b>	<b>15</b>
<b>3.7.1 Functional Requirements .....</b>	<b>15</b>
<b>3.7.2 Non-Functional Requirements.....</b>	<b>16</b>
<b>3.8 Conclusion .....</b>	<b>16</b>
<b>4.Design Chapter.....</b>	<b>17</b>
<b>4.1 Introduction.....</b>	<b>17</b>
<b>4.2 Database Design .....</b>	<b>17</b>
<b>4.2.1 Introduction.....</b>	<b>17</b>
<b>4.2.2 ERD Entity Relationship Diagram .....</b>	<b>17</b>
<b>4.2.3 Data Dictionary .....</b>	<b>18</b>
<b>4.3 User Interface Design .....</b>	<b>19</b>
<b>4.3.1 Introduction.....</b>	<b>19</b>

4.3.1 Sketches.....	20
4.3.2 Activity Diagram.....	21
4.3.3 Navigation Diagram.....	22
4.3.4 Wireframes.....	22
4.4 Conclusion.....	24
5. Implementation Chapter.....	25
5.1 Introduction.....	25
5.2 Technologies.....	25
5.3 Architecture.....	26
5.4 Database implementation.....	26
5.4.1 Creating the Database.....	27
5.4.2 Populate the Database.....	27
5.4.3 Querying Data.....	28
5.5 Frontend.....	29
5.5.1 Modal Form.....	30
5.5.2 Cards.....	31
5.5.3 Progress bar.....	32
5.6 Backend.....	33
5.6.1 Database connection.....	33
5.6.2 Controller.....	34
5.7 Conclusion.....	35
6 Validation Chapter.....	36
6.1 Introduction.....	36
6.2 Validating the Project Requirements.....	36
6.3 Validating the Database Design.....	36
6.4 Usability testing.....	37
6.5 Code Testing Strategies.....	38
6.6 Conclusion.....	38
7 Critical Review.....	39
7.1 Introduction.....	39
7.2 Achievements.....	39
7.3 Problems and Limitation.....	39
7.4 Future Work.....	40
7.5 Legal, Ethical, Societal and Security Issues.....	40
7.6 Conclusion.....	40
References.....	42
Appendices.....	43

Appendix A - Deliverables .....	43
Appendix B - Wireframes .....	44
Appendix C - ERD Entity Relationship Diagram .....	45
Appendix D - Wireframes.....	46
Appendix E - Use Cases .....	51
Appendix F -Usability test questionnaire.....	53
Appendix G - Test Case Tables .....	54

# **1 Introduction and Background**

A successful project is strongly related to the chosen method of management. Agile methodologies have proved highly effective a high in managing projects by allowing the project's requirements to be changed in the process and quickly adapt to the new ones through an iterative approach of delivering the artefacts for a project. In many cases, it is the best approach because few projects can specify all the requirements before the start, and fewer do not require changing them during the production. Potential users can be teams that work on projects that do not have the resources to implement the entire stack of principles of the agile methods, students that want to use this approach for completing their projects. The main objective should remain the project itself, not learning how to apply complex project management technologies. This project will consist of a web application that will allow the users to manage their projects intuitively in an agile way.

The report contains six main chapters describing each stage through which the project has gone through. The Requirements chapter where are gathered all the functional and non-functional requirements and analysed the users. The second is the Design chapter, where the database design was created to store all the data needed for the product to meet all the client requirements. The third chapter will contain the implementation of all the project components, the database and the website connected to it. The Validation chapter contains the testing steps and how each artefact of the project was validated with the client. The last section of the report will contain information about the achievements of the project, the problems that were encountered with a critical summary of the entire project.

## **1.1 Problem**

Agile Project Management is a complex process that requires a significant amount of knowledge to be implemented and managed correctly for a project to be truly agile. In many cases, this position must be fulfilled by a specially dedicated person who will keep track of and manage the agile process. This requirement adds complexity to the projects, and it may be beneficial for big projects that require this approach because of multiple layers of monitoring outcomes. Still, it uses too many resources for small and possibly medium projects that do not need so much functionality, which distracts the developers from the actual goal of completing the project. In this case, it can be university, school or even small companies that need to manage small projects. Existing platforms can add too much complexity to the project and distract the students from the primary goal that is delivering the project.

## **1.2 Aim**

The project aims to offer users the ability to manage small-to-medium-sized projects easily. The idea is to create a tool that will contain only the essential functionality to manage a project and with a clean and simple design that will allow the manager and team to focus on deliverables rather than but not on how to set up and maintain the project managing tool.

### 1.3 Objectives:

- The platform should make agile project management intuitive and easy to understand.
- Design and implement a web UI (user interface) clean and straightforward, allowing users to execute the necessary operations as fast as possible.
- The application should have a responsive UI on the following platforms: Desktop, Phone, Tablet.

### 1.4 Solution

The solution would be a small system that contains only the basic functionality, allowing the users to implement the agile approach to a sufficiently good extent to achieve a successful project. The web application with a clean and simple design will allow the clients not to get distracted by too much functionality. If the users have some basic knowledge about agile practices, they will not require any further instructions in using the tool.

### 1.5 Proposed Project Management Methodology

For managing this project, it will be used as an agile approach. This method will allow delivering elements of the project incrementally and permit changing the deliverables, requirements, and goals during the development phase. The methodology will be Kanban, whose main concepts are *roles* that are split between each team member and *artefacts* the project deliverables. This framework balances the demand and the available capacity to fulfil the need, copes with change and allows the team to be in control of the project schedule and deliverables ('Kanban', 2021). The tool used to manage the project will be Trello, a free tool that allows creating and updating the status of the tasks and monitoring the process of delivering the artefacts.

### 1.6 Work Plan

Figure 1 represents a Gantt chart with the significant phases of the project and Table 1 with the list of the major deliverables and the dates when those artefacts must be ready. The complete list of deliverables can be found in Appendix A.

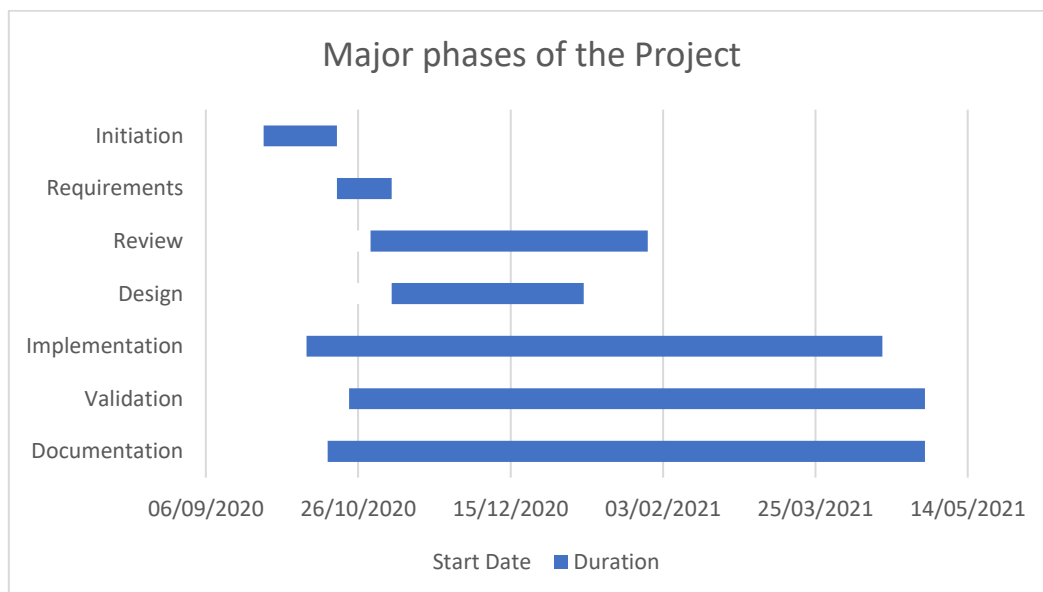


Figure 1.1 Major Phases of the Project

Phase	Milestone	
	Deliverable	Date
Initiation	Swot Analysis	09/10/2020
	Proposal	19/10/2020
Requirements	Stakeholders	16/10/2020
	Use Cases	23/10/2020
	Lis of requirements	06/11/2020
Review	Topic	30/10/2020
	First Sources	27/11/2020
	Mind Map	08/01/2020
Design	ERD	27/11/2020
	Sketches	11/12/20/20
Implementation	Database	30/10/2020
	Backend	15/01/2020
	Frontend	05/03/2020
Validation	Data generation	04/12/2020
	Data population	08/01/2021
Documentation	ToC Design	13/11/2020
	Requirements	27/11/2020
	Design	22/01/2021
	ToC implementation	05/02/2020
	Submission	30/04/2021

Table 1 -Milestones

## **2. Literature Review**

### **2.1 Introduction**

All projects require management, and very often, the type of management chosen depends on whether the project's outcome will be successful. Bloch (2012) researched a little more than 5000 IT projects. The findings from the research are that on average, 66% of projects had to go over budget, 33% of them had to go over schedule, and 17% of them had so much lousy impact on companies that, in some cases, they threaten the existence of the company. That is defined by the fact that most projects had to go over budget between 200% to 400% in the high end. The research shows that each project requires a type of management that will fit his needs. There is a transition from Waterfall to the Agile methodologies in software development described in this chapter. The transition takes place because in software development was found a clear pattern that the requirements will change over time and will need to be updated. Working on only one stage is less productive because it will give less interaction between stages and will not start the next phase until the team will not finish the current one. That leads to an innovative approach where the client will be involved in the development process from beginning to end and will be able to update the requirements that are allowed by the Agile Methodologies as Scrum or Kanban that will be discussed in this chapter.

### **2.2 Agile**

Agile practices describe the process where cross-functional teams work alongside the customer to gather the requirements and produce a viable product that will meet the latest needs of the clients. As stated in the Manifesto for Agile Software Development that can be found online (*Agile Manifesto*, 2001), the goal of the agile methodologies is to help manage projects that will have a successful outcome. The seventeen authors of this manifesto agree that a better way of managing projects is to put individuals and interactions over processes and tools, working software over documentation, customer collaboration over contract negotiation, and adapting the project quickly rather than stick to the plan. Information about the history of agile practices can be found on Wikipedia (*Agile software development*, 2020). The article states that the agile methodologies dates before the Agile Manifesto were published in 2001. Scrum dates from 1995, extreme programming 1996, feature-driven development 1997. Now those methodologies are referenced as agile methods.

#### **2.2.1 Scrum**

Scrum is one of the Agile Methodologies. It is used mainly for software development, but it also can be applied to different types of self-driven teams. The details of the workflow of a Scrum management method can be found online (Scrum, no date). First, all the team scrum master and product owner will have to participate in Sprint planning, a session where it is decided what task from the product backlog should be done during the sprint. This aspect is crucial because it should be realistic. After all, the time box is fixed and will not be modified during the sprint. The second part is the actual sprint, where the team will work to accomplish the task that was decided during sprint planning. During the sprint, the team will also have daily meetings called stand-ups where the team will share the work they are currently doing and, more importantly, flag any problems. At the end of each sprint, the team will have a sprint review where they will celebrate the accomplishments done during the sprint and return incomplete tasks to the backlog, and the process will start all over again.



The roles in Scrum that can be found online (*What is Scrum*, 2004) are as follow ScrumMaster, Product Owner, Team. The Scrum Master is responsible for teaching Scrum to all parties involved in the process to the required level, keeping the project's progress up to date and visible to all parties, and resolving any problems that may occur during development. The Product Owner is the person that will connect the development team and the client by gathering all the requirements in the Product Backlog and prioritising them. He also will decide the release of the features schedule and determine whether the product has the features and quality for release. The Team in Scrum is cross-functional and self-organising, which allows it to decide how it should do the work by splitting the task between the team members that should be on average from 5-10 people that will allow the best mix between productivity and communication.

### **2.2.2 Kanban**

Kanban is another Agile method that will help the team visualise the project's progress using the board. The Kanban board consist of cards that represent the task that must be done. Those cards are organised in columns that describe the current stages in which the task are currently. The columns usually are represented by labels like "To Do", "In progress", "Done", but it as well can be updated with the phases that best describe the workflow that the team uses to develop the project. Pavel (2019) speaks about the four core principles of Kanban that start with what you do now, agree to pursue incremental, evolutionary change, respect the current process, roles, and responsibilities, encourage acts of leadership at all levels. Those principles best describe a team that fully collaborates on work that must be done and removed the need for a specific leader but encourages all the members to act as a leader and manage the project collaboratively.

The difference between Kanban and Scrum is that Scrum is a more robust management practice. Even if it is an agile method, the sprints are fixed in length and will not allow changes during that iteration. Where the Kanban approach is that the team can work at the same time on a restricted number of things, and when the task was compleated, the member takes the next task assigned or replace it with a new one that is more urgent or if the current one suddenly become obsolete. Another difference that is advantageous to Scrum is the daily stand-ups and reviews that give the team more feedback on the work done and the work that needs to be done.

## **2.3 Waterfall vs Agile**

The Waterfall is a methodology that consists of 5 stages of requirements gathering and analysis, design, testing, project delivery, maintenance. It assumes that each phase must be completed in the exact order, and the team can not move to the next stage until the current stage is fully complete.

Comparing agile and Waterfall (Alexander, 2020), there are the following pros and cons. The agile methodology, because of short iterative sprints, the team will identify and fix defects faster, iterations allow quick changes to the project, development is flexible and rapid. Cons of using Agile can be that Agile is more complex than Waterfall, and in many cases, it requires an experienced person who will act as a Scrum master that will manage the team work neede to be done. The customer may not like to review so often the changes that were made to the project. The self-organising teams can be an issue for remote teams that will work in different time zones.

## **2.4 Conclusion**

This chapter was discussed the agile project management methods and a little comparison with the Waterfall and why it was decided that the agile approaches suit better software projects. The agile methodologies that were described are Scrum and Kanban. The difference between the two is that Scrum uses sprints time boxes where the team should work toward the tasks assigned to the sprint, wherein in Kanban, each member of the team will take one task at a time and work on it. Both Waterfall and Agile have pros and cons, and the teams should decide which one suit best their needs. Still, the project has a very high probability that it will fail if it will be managed with Waterfall management because the software development industry tends to change very fast. If the products do not meet the customers' needs, they become useless, and lots of resources are wasted if the requirements are not updated regularly.

## 3 Requirements

### 3.1 Introduction

Requirements gathering is a critical part of the development life cycle of a project. The list of requirements will describe the functionality that must be implemented in the system. Requirements will help during testing because they will provide a clear description of how the system should act and how the system behaviour will be considered successful. The elicitation method that was used is one to one interview. After that information, gathered was translated into user stories, which helped make a list of functional and non-functional requirements and design the use cases that must be followed to accomplish those tasks. After an analysis of the stakeholders was undertaken for this process, it was decided which to interview and SWOT analysis to determine whether the system is worth doing.

### 3.2 SWOT Analysis

The SWOT analysis technique is designed to take each layer of an object, product, or company and analyse the objectives, strategy, resources, and offers. Each element that was identified during the analysis process then is put in one of four categories:

- Strengths – advantages over similar competitors
- Weaknesses – disadvantages related to other competitors
- Opportunities – elements that may be used as an advantage for the company
- Threats – elements that will be a disadvantage

The decisions making after this process is a lot easier because it provides an overview of the object's current position, which was analysed.

#### 3.2.1 Introduction

A simple, intuitive, and informative web application will simplify keeping track of tasks, making project management more accessible and more straightforward. The project will be built with a responsive design that will allow users to use it on multiple platforms as desktop phones or tablets. It will enable users to create and maintain the plan to implement an idea. The application will contain only the essential functionality required to manage a project, excluding all the rest of the features, confusing the user. App users will be able to manage multiple projects.

#### Strengths:

- Simple and intuitive user interface
- The app will contain just the essential functionality for managing a project
- Responsive design, the app will be available on desktop and phones
- Keeping track of the project progress
- Having an editable members list which is composed of people that the current user worked on past projects or planning to work on the future ones

#### Weaknesses:

- Lack of complex functionality that the competitors provide
- Security problems are represented by threats to which the system may be vulnerable.

- GDPR constraints will add complexity to the system

### Opportunities:

- Making the agile methodologies more accessible

### Threats:

- Competition from the systems that are already on the market
- The timeframe assigned for the project can be not enough to make the project ready for production
- Incapacity of making the user aware of the product
- Technical problems that may occur

### 3.2.2 Recommendation

The project is worth doing because it will allow users to use an agile approach to manage projects. The application indeed will not be helpful for complex projects, which will require more functionality to be successfully implemented. Still, the application will simplify managing the project and allow quickly to interact with the members of the teams. With a clean design and only the essential functionality, the application will be perfect for small-medium projects.

### 3.3 Stakeholders

All Stakeholders have a high interest in completing a project that means that they all are interested in a good project management tool that will help them organise and deliver the required artefacts. In Figure 3.1, it can be found that the most critical stakeholders with high interest and high power are managers and teachers who will choose the tools that will be implemented. In this case, they must be managed closely so that the application will satisfy all their requirements. The second category will be the students and employees who will use the app and are also interested in the tool but have less power over it, and they will be less managed than the first group but keep informed regularly

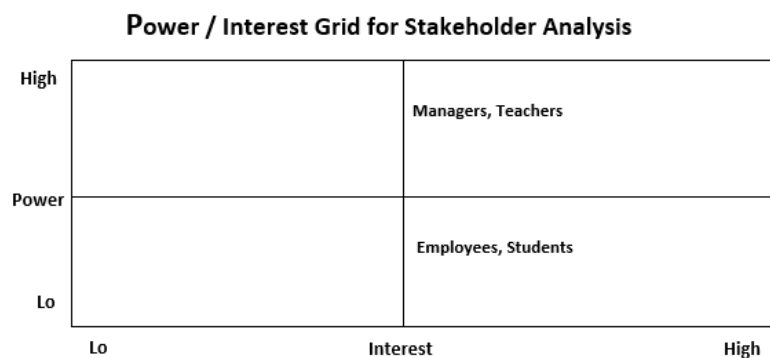


Figure 3.1 Power/Interest Grid

### 3.4 User stories

User stories are the requirements gathered from the stakeholders converted to a simple description containing information about a feature that is meant to be implemented. The format followed by user stories is as follows: As a “user type”, I want to “the action” so that “the goal to be achieved”. Any stakeholder can write those user stories at any point in time during the creation of the product. Usually, they are stored in the product backlog, and the product owner manages them.

Here is the list of the user stories for the project:

- As a user, I want to add tasks to the project so that I can take actions to solve them.
- As a user, I want to add tasks to the project so that I will be able to monitor the stages of development of the project.
- As a user, I want to visualise the progress of the project so that I will understand the metrics related to the successful completion of the project
- As a user, I want to be able to create a project so that I will be able to manage it
- As a user, I want to update the project’s vision to be up to date with the goals and requirements.
- As a user, I want to add/remove team members to the project so that the team will contain the required people for the successful completion of the project
- As a user, I want to add/remove team members to the project so that the team will contain the required people for the successful completion of the project
- As a user. I want to comment on the task so that the tasks will be more descriptive
- As a user, I want to see a list of tasks assigned to me to manage my time to get all planned tasks done.
- As a user, I want to be able to group projects in different categories so that I can access projects from the same category faster

### 3.5 Use Case Diagram

The use case diagram represents the elements of a new system. It is a visual representation of the behaviour that is expected from the system without including the exact steps of how it should implement the functionality. This diagram represents the system from a user point of View. It communicates the requirements that are expected from the system. Usually, the graph is simple and represents the user’s interaction with the system, the interaction between use cases inside the system. Figure 3.2 below illustrates the use case diagram with the elements that were found during the elicitation process.

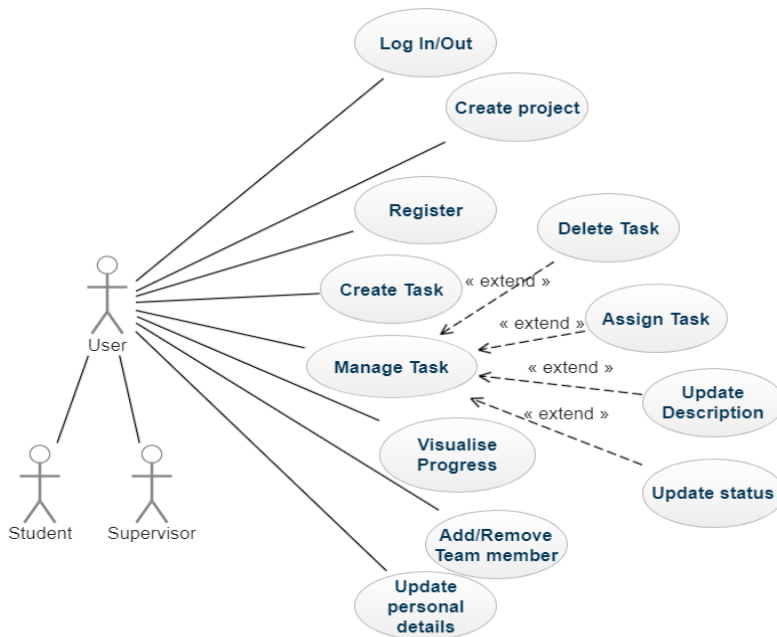


Figure 3.2 Use Case Diagram

### 3.6 Use cases

A use case consists of a list of steps that must be accomplished to achieve the desired outcome. They are based on functional requirements. The purpose of the use case is to explain actions that need to be taken by the user to achieve a goal. They will be used further by the developers to implement the functionality in the system. Use cases designed for this project contains the following sections:

- Precondition – the conditions that must be met by the system so that the user can perform the desired outcome
- Success End – When the task will be considered as completed
- Failed End – In which cases the task did not perform as expected
- Description – The steps that the user must undertake to accomplish the goal

Below is Use Case 3.3 that represents the steps needed to follow when the user wants to add a task to a specific project. The complete list of use cases can be found in Appendix E:

Use Case 1	Add Task	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select project
	3	Click on a new task
	4	Insert information about the task (Title, Assignee, Description)
	5	Click Add

### Use Case 3.3 Add Task

A comprehensive list of all use cases can be found in Appendix E. Table 3.3 lists each of the use cases that can be found there.

Use Case 1	Add task
Use Case 2	Manipulate task (delete, assign, update (content, status))
Use Case 3	Visualise Progress
Use Case 4	Create a Project
Use Case 5	Update Vision
Use Case 6	Add/remove team members
Use Case 7	Comment Task

Table 3.3 Full list of Use Cases

## 3.7 Functional and Non-Functional Requirements

The requirements of the project are the conditions under which the project will be categorised as complete. The requirements will be gathered from the stakeholders, and they will be prioritised so that the project will deliver iteratively, starting with the critical features. These features will give the user a pleasant experience by using the project. There are two types of requirements functional and non-functional. The functional requirements will contain the goals that are required to be offered by the system. Non-Functional Requirements will add a constraint on the system to increase the usability of it.

### 3.7.1 Functional Requirements

Functional requirements are the category that describes what the system will do, a statement that describes what the user wants to accomplish with the help of the system. The functional requirements will be prioritised with the help of MoSCoW. With this technique, the importance of each requirement will be decided and put the requirements in four different categories. Must have (M), the requirement is critical to the system, and it will fail if there will not be delivered even one of those requirements. Should have (S), the requirement is essential, but it can be left at the end. Could have (C), the requirement will improve the user experience but not critical, will be implemented if time and resources will be available. Will not have (W) these requirements are not necessary at that time for the system.

Below in Table 3.4 can be found a comprehensive list of functional requirements

ID	Functional Requirements	MoSCoW
1	Create Account	M
2	Log In/ Log Out	M
3	Edit personal details	S
4	Create a project	M
5	Update project	M
6	Delete project	S
7	Edit vision of the project	M
8	Add members to a project	M
9	Remove a member from the project	S
10	Create task	M
11	Comment task	S
12	Delete Task	S

13	Edit Task	M
14	Change the status of a task	M
15	Assign task	M
16	Visualize progress	S
17	List Friends	C
18	List my tasks	M
19	List Projects	M
20	List Sprints with tasks	C
21	Populate friend list	C
22	Chat through a private channel with friends	C
23	Assign agile roles	C
24	Group Projects	S
25	Search Projects	S
26	Filter Projects	S
27	Group Tasks	C
28	Create Status Labels	C
29	Search Task	S
30	Edit description of a task	S

Table 3.4 Functional Requirements

### 3.7.2 Non-Functional Requirements

Non-Functional Requirements describe how the system should act. This type of requirement will put constraints on the system to have a pleasant experience by using it. The conditions are defined in Table 3.4 that can be found below. The system will still perform as expected, but the usability will be affected by how easy the system will be used, which will negatively influence the user experience.

ID	Non-Functional Requirements
1	The system must have a responsive design for the following platforms desktop, mobile phones, and probable tablets
2	The system must be accessible at any time
3	The database must encrypt sensitive data
4	The application must not allow access to data for not authorised persons
5	Rendering of the web page must take not more than 2s
6	The user must get the response to the most complex request to the database in less than 4s
7	The design and user interface should be intuitive and ideally should not require instruction on how to use it

Table 3.5 Non-Functional Requirements

## 3.8 Conclusion

The requirements are needed to start designing the system were successfully gathered. The agile development methodology allows the requirements to change in the future, and the system will be adjusted to the updated needs. A list of Functional and Non-Functional requirements was created and prioritised in concordance with the stakeholders.



## **4.Design Chapter**

### **4.1 Introduction**

The design chapter aims to create a user interface that must satisfy the requirements gathered in the previous chapter. The first step is to decide how the information will be stored. The ERD will create a defined way of storing the data by choosing the relationship between entities and the required attributes for the project to run correctly. The second step in designing the database is creating the data dictionary that will have information about all the attributes in the database to cover all possible options of information and use the space assigned to the database properly. The second primary phase is to design the GUI (Graphical User Interface). For this project, it was decided to create the sketches, a low fidelity design approach to determine the pages required to fulfil all the requirements. The activity diagram will show the behaviour of the system. The Navigation diagram will provide an overview of how the pages will be linked together and what functionality they will host. The last step will be to design the wireframes that will look the closest to the final product and provide a layout of components on the page.

### **4.2 Database Design**

#### **4.2.1 Introduction**

The database will store the information needed for the project to run correctly. The database design aim is to provide access to accurate information. The data will be divided into tables following the Normalisation Rules where possible. The data in each field must be atomic and not transitive functional inside entities. During this process will be decided on the relationship between entities and determining the format of each field. On tables created will be applied some constraints to the attributes of each table where is possible.

#### **4.2.2 ERD Entity Relationship Diagram**

The ERD represents the structure of the database. The diagram will be composed of links describing the relationships between each entity and its attributes. This step is meant to create the data design to avoid redundancy and store it efficiently. The most complex part is to solve the cardinality between all the necessary entities so that the system will work properly. The tables in the database must also have primary keys and foreign keys. The keys are implemented after the structure of the database schema is decided, with cardinality in place. The primary key will identify the record in the table, and the foreign key will hold a link to those records.

In Figure 4.1, we can find the structure of the database. The main decisions are that a user will be linked to a project through an associative table called a team where a user will have an agile role and be related to a project. In this way, it is solved the relationship many too many between projects and users. The task will be directly linked to the project with a foreign key, and it will contain a foreign key to the user as well. In this way, it will be possible to see who is responsible for that task. In the database is present the sprint table and sprint-tasks it will allow us to choose how the project will be managed by using sprints or directly working on tasks.

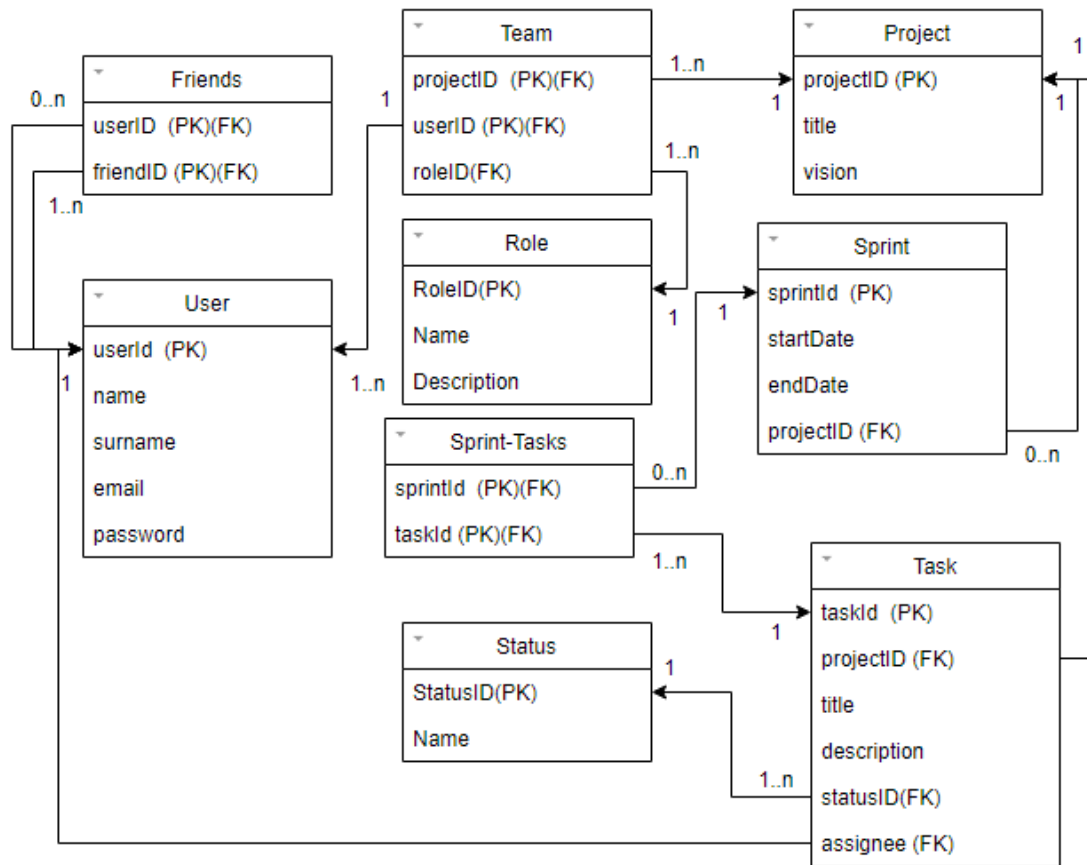


Figure 4.1 ERD

### 4.2.3 Data Dictionary

The data dictionary will be a guide for the database developer, who will implement the database. In this way, the person responsible for creating and maintaining the database will know the format in which the fields need to be stored and to use the storage available for the database efficiently. Table 4.2 below will contain the constraints that will be applied to each attribute in each database table. The regulations used are primary key, foreign key, not null, and unique.

The primary key constraint will check that the record inserted in the table will be unique. The foreign key rule will make sure that the record is linked to a valid primary key. The not null constraint will not allow introducing null values into the fields with this specific constraint. The last constraint used is unique that checks the data to be unique on the column that is applied.

Table	Column	Data	Constraint	
User	UserID	INT (7)	Primary Key	
	Name	VarChar2(20)	Not Null	
	Surname	VarChar2 (20)	Not Null	
	Email	VarChar2(50)	Not Null, Unique	
	Password	VarChar2(20)	Not Null	
Team	ProjectID	INT (7)	Primary Key	Foreign Key
	UserID	INT (7)		Foreign Key
	RoleID	INT(7)	Foreign Key	
Role	RoleID	INT(7)	Primary Key	
	Name	Varchar2(20)	Not Null	
	Description	Varchar2(400)	Not Null	
Project	ProjectID	INT (7)	Primary Key	
	Title	VarChar2(25)	Not Null	
	Vision	VatChar2(700)	Not Null	
Sprint	SprintID	INT (7)	Primary Key	
	startDate	Date	Not Null	
	endDate	Date	Not Null, >startDate	
	ProjectID	Number	Foreign Key	
Task	TaskID	INT (7)	Primary Key	
	ProjectID	INT (7)	Foreign Key	
	Title	VarChar2(25)	Not Null	
	Description	VarChar2(400)	Not Null	
	Status	INT (2)	Foreign Key	
	Assignee	INT (7)	Foreign Key	
Status	StatusID	INT(2)	Primary Key	
	Name	Varchar2(20)	Not null	
Sprint-Tasks	SprintID	INT (7)	Primary Key	Foreign Key
	TaskID	INT (7)		Foreign Key
Friends	UserID	INT (7)	Primary Key	Foreign Key
	FriendID	INT (7)		Foreign Key

Table 4.2 Data Dictionary

## 4.3 User Interface Design

### 4.3.1 Introduction

The user interface design phase's goal is to decide how the front end of the project will look and create the design and the links between web pages that will allow users to fulfil the requirements. To develop and provide a product that will give a good user experience. In the design phase will we will have to go through a couple of steps. The first step is drawing the low-fidelity screens that will represent the pages on the links between them. The second step is to design the activity diagram that will develop the workflows inside the system. The third step will be to create the navigation design that will show the client how to navigate between pages, the hierarchy and the functionality inside each page. The last step will be to

design the wireframes that will be converted from low fidelity sketches to high-fidelity screens representing the product's final look.

#### 4.3.1 Sketches

The sketches represent the first step in design and the user interface. They are helpful because of the low cost and the ability to quickly change them or start a new version without too many financial losses. At this step is decided the general layout of the pages, the links between pages and some functionality hosted on those pages. On figure 4.3 can be found the potential screens that will represent the front end of the project. We can see the login page, the registration page. After login, the user will land on the homepage to choose to see the projects to access all his tasks to see all the project's progres he is involved in and a list of co-workers. The sketches also represent how about the chat even if it will probably not be implemented. From the project page, the client can Edit the project and manage the tasks created under that specific project.

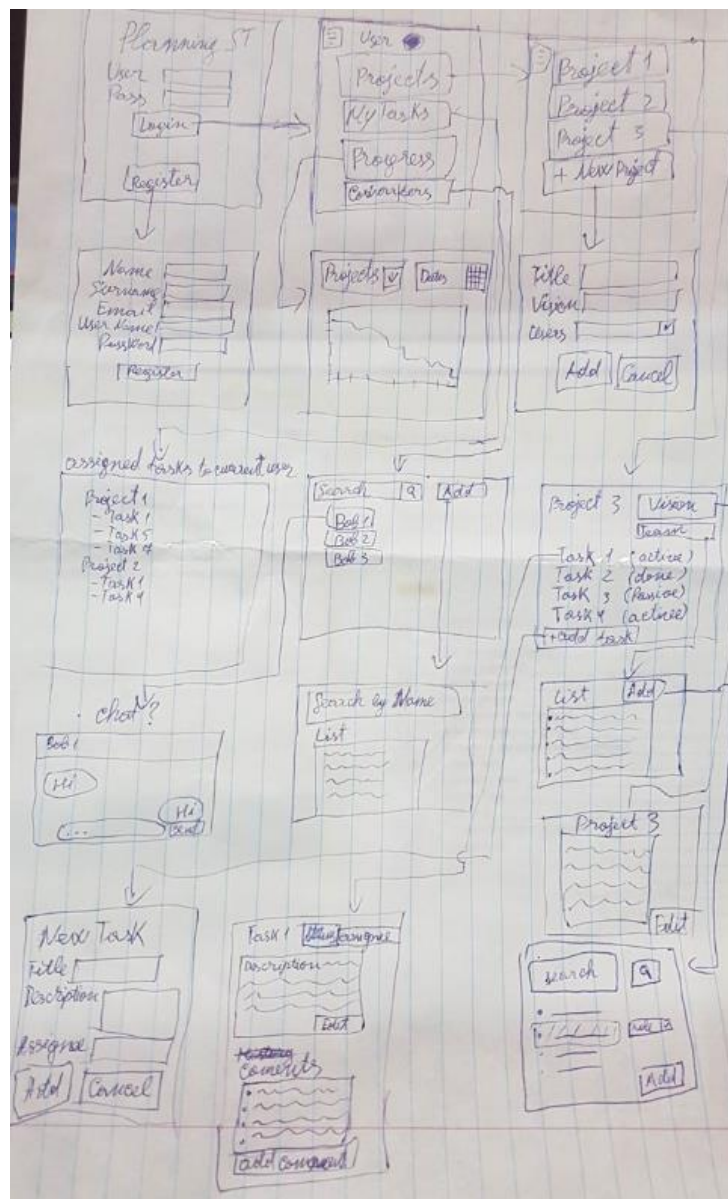


Figure 4.3 Sketches

### 4.3.2 Activity Diagram

The activity diagram using the functional requirements and the sketches designed will illustrate the workflow or the system. On figure 4.4 below, we can see the processes that will take place. The diagram will create the system's architecture where we can find what action must be executed for the client to achieve the goals that are requested from the system. We can see that by accessing the system, the client will have the option to log in or register. After login user will see the list of the projects, list his tasks, list his friends, and view the progress for all the projects. The user will select a project or create a new one on the page displaying the list of projects. After the user selects a project, the following functionality is made available to update the project, create new tasks and manage the team. After selecting a task, the functionality that allows managing the task will be available.

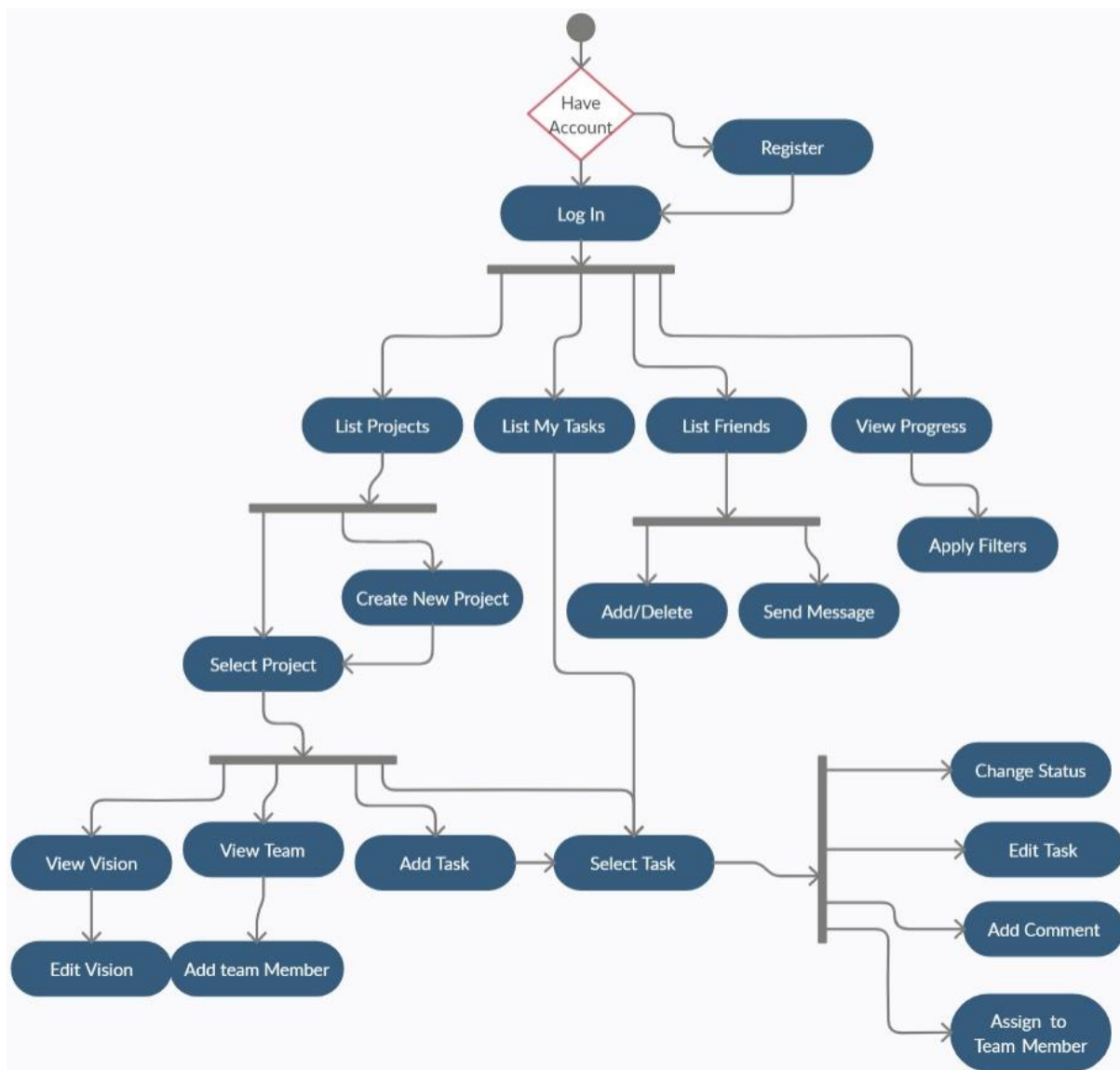


Figure 4.4 Activity Diagram

### 4.3.3 Navigation Diagram

A navigation diagram shows how the system is linked together and how the user can access the system's functionality. The list below represents the hierarchy of the current project. Each bullet point means how the pages are nested inside each other and how the user will navigate through the system to find the required functionality.

- Login
  - List Projects
    - Create a Project
    - Select Project
      - Edit Vision
      - Update Project
      - Delete Project
      - Add/remove members
      - Assign agile roles to members
      - List Tasks
      - Create Task
      - Select Task
        - Edit Task
        - Delete Task
        - Change Assignee
        - Change the Status
  - Progress
    - Filter projects
  - My tasks from all projects
  - Friends
    - List Tasks
    - Add friend
    - Remove friend
    - Chat
- Register

### 4.3.4 Wireframes

The wireframes are the last step in planning and designing the front end of the project. Here, the designers will gather all the information created during this phase. The sketches representing the first design of the web pages, the activity diagram represent the workflow of the system and the navigation design that links all the pages together and shows what functionality must be hosted on them. The wireframes will deliver a clean and simple design for the project and implement the same design rules for the entire project. The front end of the project that will be created after the wireframes can be not 100% identical to the wireframes. The web pages that would be made can be slightly different but follow the same general design rules as the layout of the content. The design can be changed somewhat, like the colour shades or text font and size. A couple of wireframes will be described below, but a comprehensive list of wireframes can be found in Appendix B.

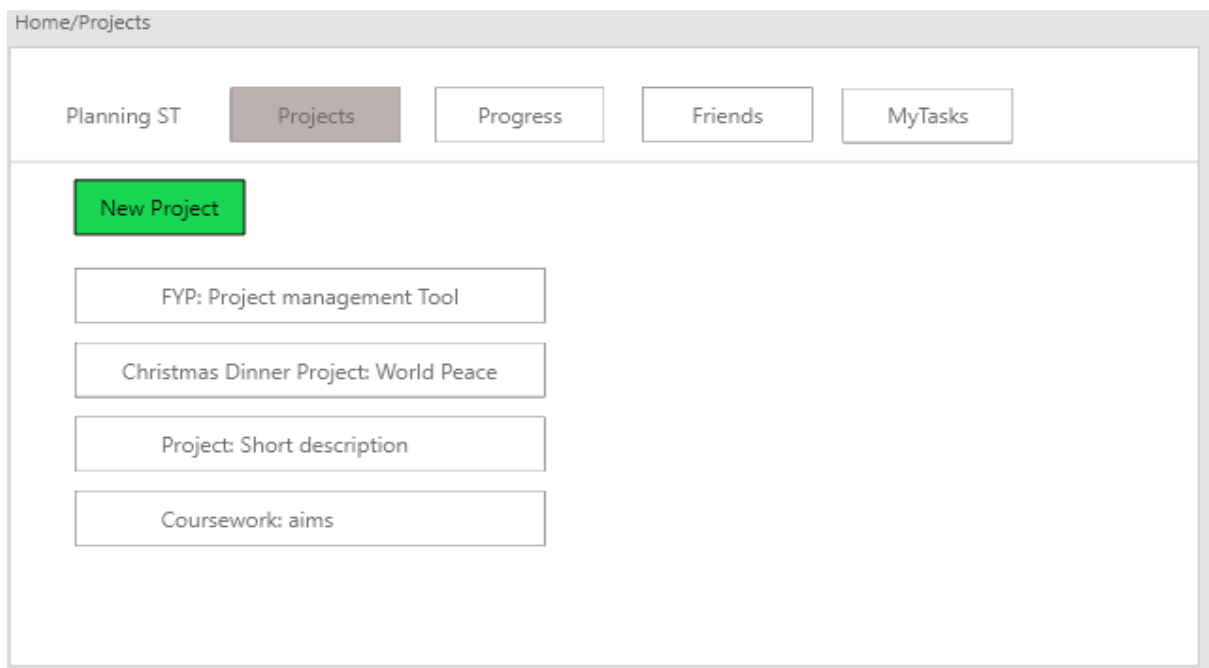
Figure 4.5 represents the page on which the user will land when he tries to access the system. The login page where he will be required to insert the account's credentials will register if the user does not have an account on this page.



The image shows a web browser window titled "Login". The main heading is "Planning ST". Below the heading, there are two input fields: "Username" with the text "Stefan" and "Password" with masked characters "\*\*\*\*\*". Below these fields are two green buttons: "Log In" and "Register".

Figure 4.5 Login page

After the user logs in, he will end on the homepage, represented by Figure 4.6. on this page, the user will have listed all the projects in which he is involved and will create a new project or select an existing one. On the homepage, the user will also be able to navigate to progress, friends, or my tasks.



The image shows a web browser window titled "Home/Projects". The main heading is "Planning ST". Below the heading, there are four buttons: "Projects" (highlighted), "Progress", "Friends", and "MyTasks". Below these buttons, there is a green button labeled "New Project". Below the "New Project" button, there are four input fields containing the following text: "FYP: Project management Tool", "Christmas Dinner Project: World Peace", "Project: Short description", and "Coursework: aims".

Figure 4.6 Projects page

The user will be able to see the progress of each project on the progress page represented by figure 4.7. This page will be beneficial for the supervisor because he can access the students' projects and see how they progress with the projects.

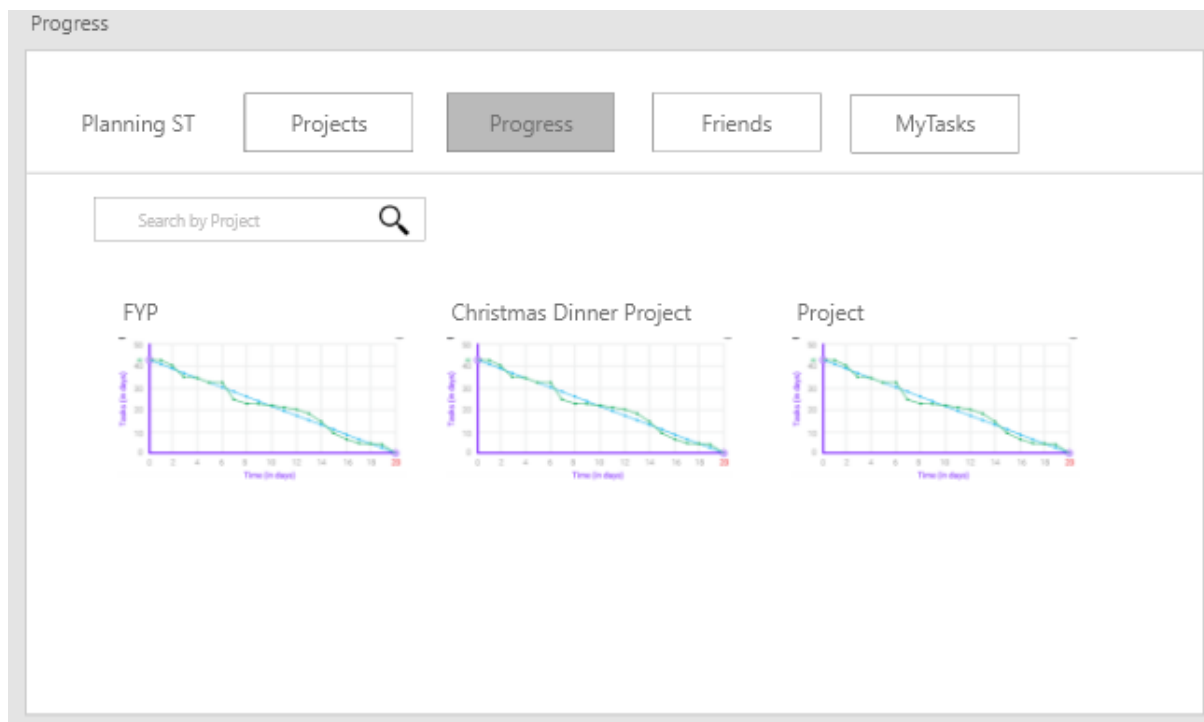


Figure 4.7 Progress

## 4.4 Conclusion

The design section aims to prepare all documentation for the project and prepare it for the implementation phase. All the steps are undertaken in the design phase to format and convert the information gathered during the requirement phase. Two main deliverables from this phase are the ERD and the wireframes. The ERD establish the database design the links between the tables and the attributes under all the tables. The project's design phase is a mandatory and crucial step because it will allow the system to work correctly, store the information provided by the client, and retrieve it correctly. The wireframes will gather the data from the sketches, the activity diagram, and then the navigation diagram. This information will create the front end of the system that will look very similar to the front end that will be implemented in production.

There is some difference between each step because of the agile approach chosen as a project management methodology. It allows changes because of the iteratively way of managing the project.



## 5. Implementation Chapter

### 5.1 Introduction

The implementation part of the project will have a goal to create a product that will fulfil the requirements that were decided to be needed with the client. The product will be implemented to combine the data gathered in the requirements chapter and the design chapter where the database ERD and the Graphical User Interface were created.

This chapter will have five major sections. Section 5.2 will describe the technologies that were used during the process of building the system. Section 5.3 will describe the MVC pattern and why it was chosen such an architecture for the system. The steps that were taken to implement the database will be explained in section 5.4.

### 5.2 Technologies

**Visual Studio Code**- the Integrated Development Environment (IDE) that was used to develop the project. The main reason it was chosen this exactly IDE over the rest of the options is that it supports all the programming languages used during building the project. The IDE offers a clean easy to use user interface. The IDE also has a terminal integrated that was very useful using GIT commands.

**KUNET** – is the webserver provided by the university that will allow hosting the website and the MySQL database.

**WinSCP** – this tool will allow the transfer of the files through the SFTP (Secure File Transfer Protocol) to the server. The tool will be used to transfer the files from the local computer to the KUNET, the webserver that will host the project's files.

**phpMyAdmin** – is a web application that will allow management of the MySQL database hosted on the university server. With the help of this tool, it will be possible to create the database and all the tables with the required attributes and relations between them. The tool also allows importing data from a CSV format that will be useful during the database population.

**Draw.io** – is a free diagram designer tool used to create the ERD (entity relationship diagram) for the database. The application has a clean and intuitive design that will not delay creating the ERD if the user will know how the ERD should look.

**Bootstrap** – a CSS framework that will help with the UI design (user interface). It will be used for fashionably displaying the data and arrange the elements in the desired layout. The framework also will remove the tedious work needed to create the design using CSS. It will give the ability to quickly create the responsive design of the project to be displayed in different formats on different platforms like mobile, desktop, or tablet.

**HTML** – Hypertext Markup Language will create the architecture of each web page sent from the server. HTML will design the frontend skeleton of the web page. The file will contain all the elements displayed by the web browser according to the user's requests and the application's functionality.

**PHP** – is a scripting language that will run on the server-side of the application. With the help of this language, all the CRUD (create, read, update, delete) operations will be executed on the database with the help of PDO that is the interface through which the

connection to the MySQL database will be made. PHP will also inject the data into the correct places in the HTML file.

**JavaScript** is the programming language that will create live interaction with the page, change the HTML and CSS on the client-side of the web, and contain many more APIs that will make the web pages interactive.

**GIT**– the version control tool will be used with GitHub to host and manage the project versions and track changes made to the project. It will also be used to go back at a previous version if there are too many errors to solve or changes were made and are no longer required in the last version of the project.

### 5.3 Architecture

Software architecture describes the structure of the system and how the components will interact with each other. Suppose the system follows a strict architecture that will allow changes and re-usage of code fundamental in agile management. The requirements will change over time, and good architectural decision at the beginning of the project offers the ability to adapt fast the system to the new functionality needed by the clients. Such a system will allow maintenance and debugging to be executed reasonably fast, offering a good usability experience. In case of an error or change of functionality, the person responsible for implementing the change will know what and where needs changes without knowing the entire system.

For this project, it was chosen the MVC (model view controller) pattern. This system architecture will split the project files into three categories that will have specific tasks to do. The first category will be the Controller that will execute the business logic. Here will be stored the files that will handle the user's requests and update the View. The Controller will process the requests then will call functions to manipulate data with the help of the Model. The second component will be the View, this part of the system will allow the user to interact with the system by sending the requests to the Controller and will also display the data received from the Controller to the user. The last component, the Model, will contain functionality that will manipulate data from the database and execute the CRUD (create, read, update, delete) functionality. The front end of the system is related to section 5.5, which shows how were implemented the containers used to display information about the modal forms and how the progress of each project will be shown. Section 5.6 will illustrate how the backend of the system work and how the controller model and View work together to create a reliable system for the client.

### 5.4 Database implementation

The database is one of the crucial components of the system because the client will expect that the system will store and will be able to provide access to the information introduced by the users. The database structure should allow the system to interact with the database and retrieve data required by the users as fast and secure as possible. MySQL was chosen, a popular relational database management system that will implement the ERD created during the system's design process. Kunet server will host the database, and phpMyAdmin will manage it, the web application that will create and manage it.

### 5.4.1 Creating the Database.

The web app phpMyAdmin offers the ability to create the database through the GUI (graphical user interface). It will not cover complex decisions like composite primary keys or declaring foreign keys during the creations of the tables. Those constraints will need additional steps to be applied to the tables when using the GUI. That why was decided to use SQL to create the database. It will allow learning the structure of the database more precisely, which will help create queries for the PDO (PHP Data Objects) that will connect and interact with the database. In figure 5.1 below can be found the final database structure that was implemented.

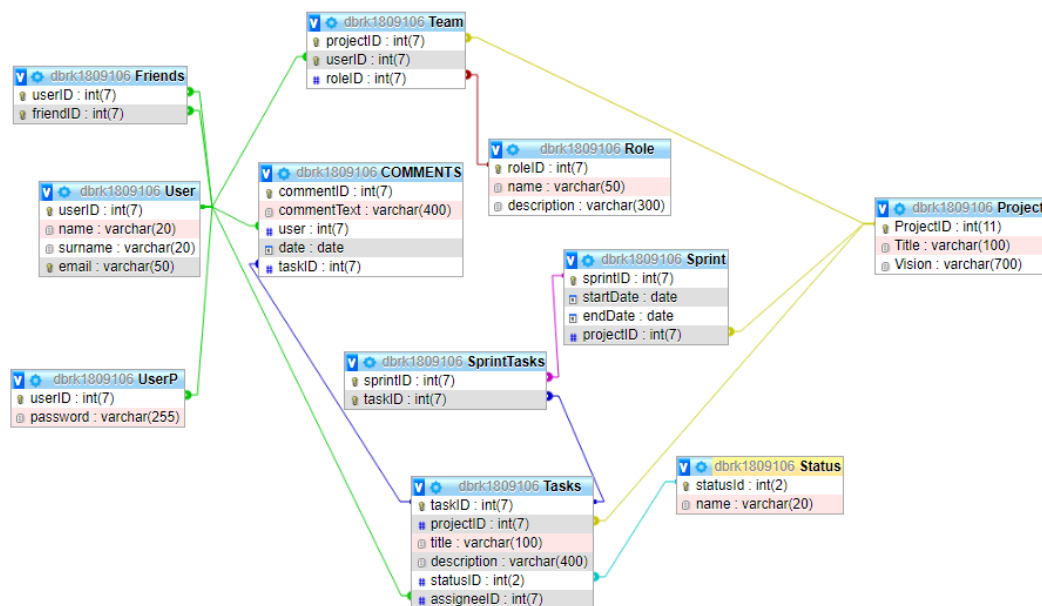


Figure 5.1 Database Schema

### 5.4.2 Populate the Database.

For testing purposes, the database will be populated with an estimative number of entries for each table. The purpose of estimating the number of entries is to predict how many entries the database will have when many users use the web application. Suppose the queries will be able to interact with the database at a decent level. In that case, that will affect the ability of the system to respond to user request and check that the response time continues to be at an acceptable level with big data.

To generate and populate the database it will be used CSV (comma separated values) files creating through excel. That will allow to easily insert the data because phpMyAdmin offers the ability to import data from a CSV file. Random names and titles for project tasks and users were generated with the help of mockaroo, a web application that will generate random data for testing purposes. That data already will be in a CSV format. Figure 5.2 below represents how the mockaroo will provide data to the user. The attributes from the database that required a more significant chunk of text, as are comments, tasks description, project visions, were generated through lorem ipsum generator because this data does not need to be too concise. Fields in the database, as are primary keys, will be autogenerated. The

foreign keys were generated inside excel with the help of the RANDBETWEEN() function that allowed the introduction of a range from which the excel will generate random numbers inserted in the foreign keys column. The foreign keys were populated so that they would have to give the users the necessary amount of data to show the ability of the system to display big data.

1	Oswald	Faye	ofaye0@google.it
2	Michail	Linthead	mlinthead1@nhs.uk
3	Kathy	Maffey	kmaffey2@businessinsider.com
4	Niel	Alebrooke	nalebrooke3@tuttocitta.it
5	Dania	Cunradi	dcunradi4@technorati.com
6	Kele	De Meyer	kdemeyer5@squidoo.com
7	Gleda	Frensche	gfrensche6@uiuc.edu
8	Augustine	Brucker	abrucker7@google.it
9	Rudy	Kolyagin	rkolyagin8@army.mil
10	Valerye	Rodgier	vrodgier9@squidoo.com
11	Ravid	Seeviour	rseevioura@example.com
12	Paton	Pickance	ppickanceb@livejournal.com
13	Denis	Bucklan	dbucklanc@epa.gov
14	Sara-ann	Plaide	splaided@icio.us
15	Roscoe	Buttle	rbuttle@sbwire.com
16	Celestyn	Gludor	cgludorf@sciencedaily.com
17	Debi	Masi	dmasig@omniture.com
18	Chet	Ikin	cikinh@theglobeandmail.com
19	Erv	Fathers	efathersi@purevolume.com
20	Gaelan	Buncher	gbuncherj@t.co
21	Perice	Cassie	pcassiek@tripadvisor.com
22	Flore	Richemont	frichemontl@archive.org
23	Johann	Gergolet	jgergoletm@hhs.gov
24	Wynn	Phinnessy	wphinnessyn@wired.com
25	Vinita	Costain	vcostaino@guardian.co.uk
26	Kittie	Fernando	kfernandop@ca.gov
27	Constanti	Curee	ccureeq@wikimedia.org
28	Ida	Hemms	ihemmsr@phoca.cz
29	Keefe	Yerbury	kyerburys@newsvine.com
30	Vinnie	Wegener	vwegenert@g.co
31	Gratia	Marians	gmariansu@hibu.com
32	Jackie	Jirasek	jjirasekv@apache.org
33	Mable	Leifer	mleiferw@omniture.com
34	Fremont	Jaukovic	fjaukovicx@godaddy.com
35	Ida	Fenn	ifenn@new.ly

Figure 5.2 User data sample

### 5.4.3 Querying Data

After populating the database, it is ready to test if the system can retrieve all the data needed to fulfil all the functional requirements created previously. To communicate with the

database will be SQL. This programming language allows interaction with a relational database management system to execute all the CRUD (create read update delete) functions.

An example of a query shown below in figure 5.3 is meant to get the number of tasks done, in progress, and tasks that need to be done. For this query, it was two types of JOINS, INNER and LEFT, to gather data from multiple tables. The query also contains three nested SELECT that are the same but repeated four times to collect the data for each task state. The query also includes functions like count() that will count and return the number of rows that were returned by the SELECT statement. The coalesce() function is used when there are no tasks under a specific status. In that case, the function will replace the value with 0 because the database will return a null, and it will create errors in the system when the data are needed to be displayed.

```
SELECT Tasks.projectID,Project.Title,
       COALESCE(toDo, 0) toDo,
       COALESCE(done, 0) done,
       COALESCE(inProgress, 0) inProgress
FROM Tasks
INNER JOIN Project on Tasks.projectID = Project.ProjectID
INNER JOIN Team on Team.projectID=Project.projectID
LEFT JOIN ( SELECT Tasks.projectID, count(name) done
            FROM Tasks
            INNER JOIN Status ON Tasks.statusID = Status.statusId
            INNER JOIN Project on Tasks.projectID = Project.ProjectID
            where name = 'done'
            group by Tasks.projectID, name
          ) DONE ON Tasks.projectID = DONE.projectID
LEFT JOIN ( SELECT Tasks.projectID, count(name) toDo
            FROM Tasks
            INNER JOIN Status ON Tasks.statusID = Status.statusId
            INNER JOIN Project on Tasks.projectID = Project.ProjectID
            where name = 'to do'
            group by Tasks.projectID, name
          ) TODO ON Tasks.projectID = TODO.projectID
LEFT JOIN ( SELECT Tasks.projectID, count(name) inProgress
            FROM Tasks
            INNER JOIN Status ON Tasks.statusID = Status.statusId
            INNER JOIN Project on Tasks.projectID = Project.ProjectID
            where name = 'in progress'
            group by Tasks.projectID, name
          ) INPROGRESS ON Tasks.projectID = INPROGRESS.projectID
WHERE Team.userID=?
group by projectID");
```

Figure 5.3 SQL Query Get Progress of a project

## 5.5 Frontend

The front end of the system is designed in a minimalistic way to provide a pleasant user experience to the client. This approach was chosen to increase productivity and not distract the users from managing the project efficiently. The client's functionality is also kept

clean and accessible without oversaturating the page with functions that the average user will not use.

The frontend will be implemented using PHP to dynamically add content and alter the web page structure, JavaScript for adding interaction to the UI, HTML through which the layout of the web pages will be designed, and Bootstrap to create the design.

### 5.5.1 Modal Form

Modal forms will be windows that will appear on top of the page and allow the user to interact with the database by providing new content that needs to be inserted or updated. When the user wants to act, they will click on the button that will be linked to the modal form and make that window pop up on top of the window and allow the user to insert or update the information.

The following example applies to all modal forms that have been created for this system adapted to the functionality that it represents. When the button is clicked, it will trigger the modal form, and it will appear on the top of the screen that is currently opened

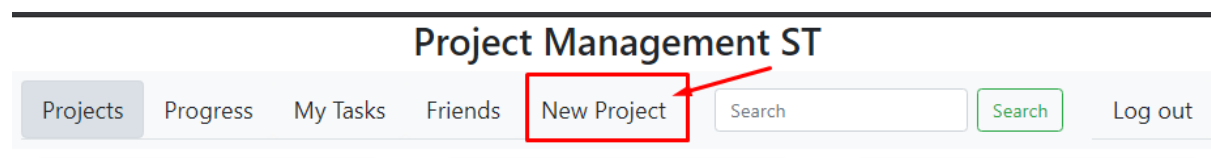


Figure 5.4 Button “New Project”

Figure 5.4 represents the button that will perform the action that will be executed when the button will be clicked. After the user clicks the button, the modal form will appear, which can be looked at in Figure 5.5 below.

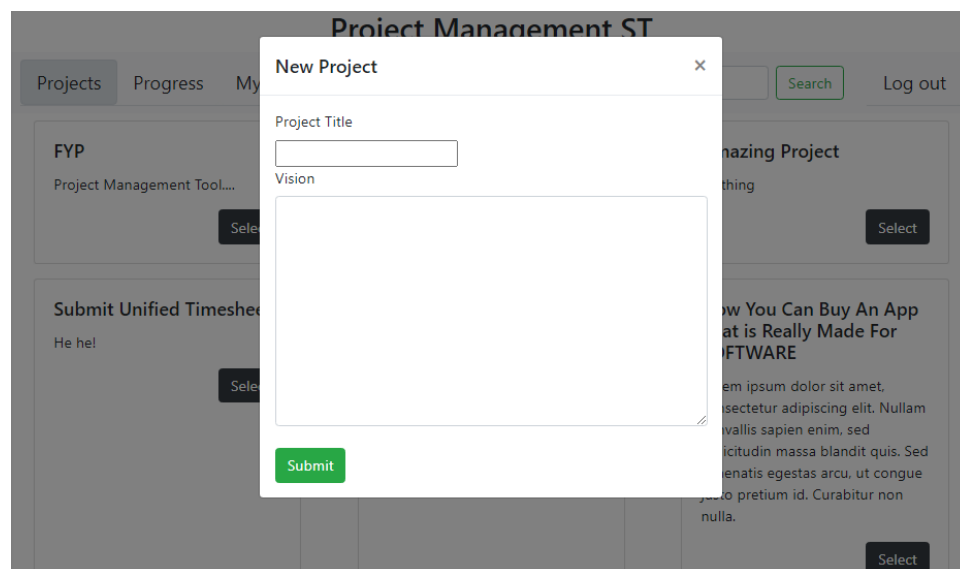


Figure 5.5 Modal Form “New Project”

The user will introduce the details needed to be stored on the modal form that appeared on the screen, as in Figure 5.6. When the user submits, the details will be sent to the database, and after submission, the page will reload, and the new project created will be displayed in the list of projects, as shown in Figure 5.7.

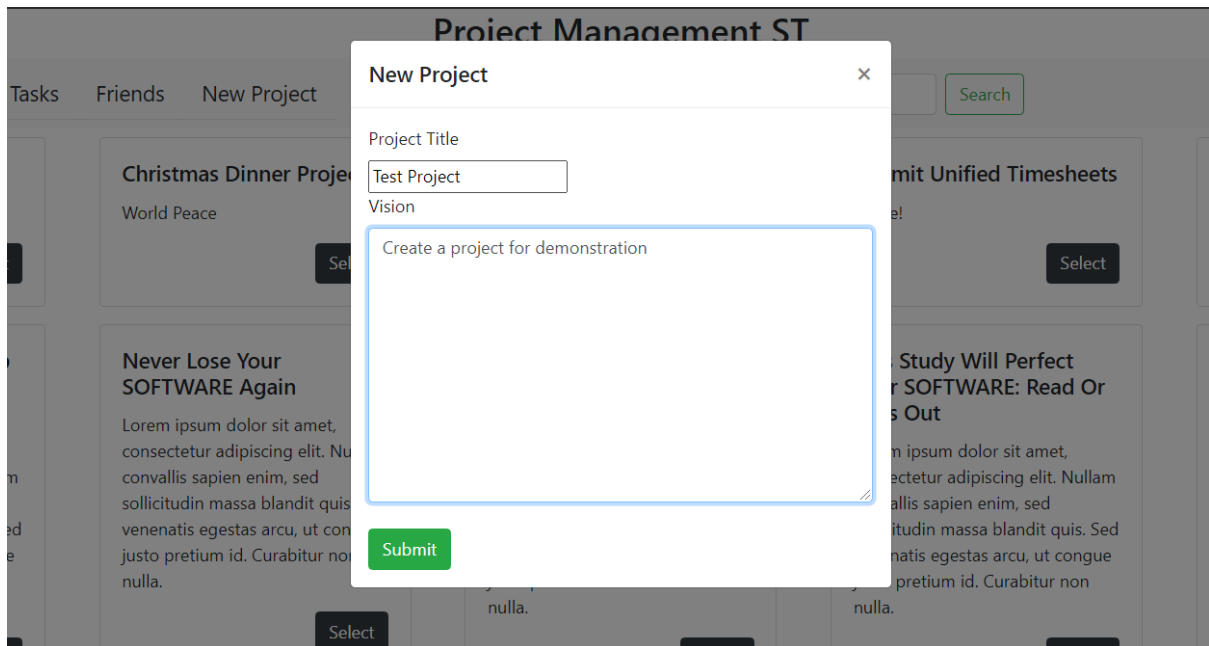


Figure 5.6 Fill the form with data.

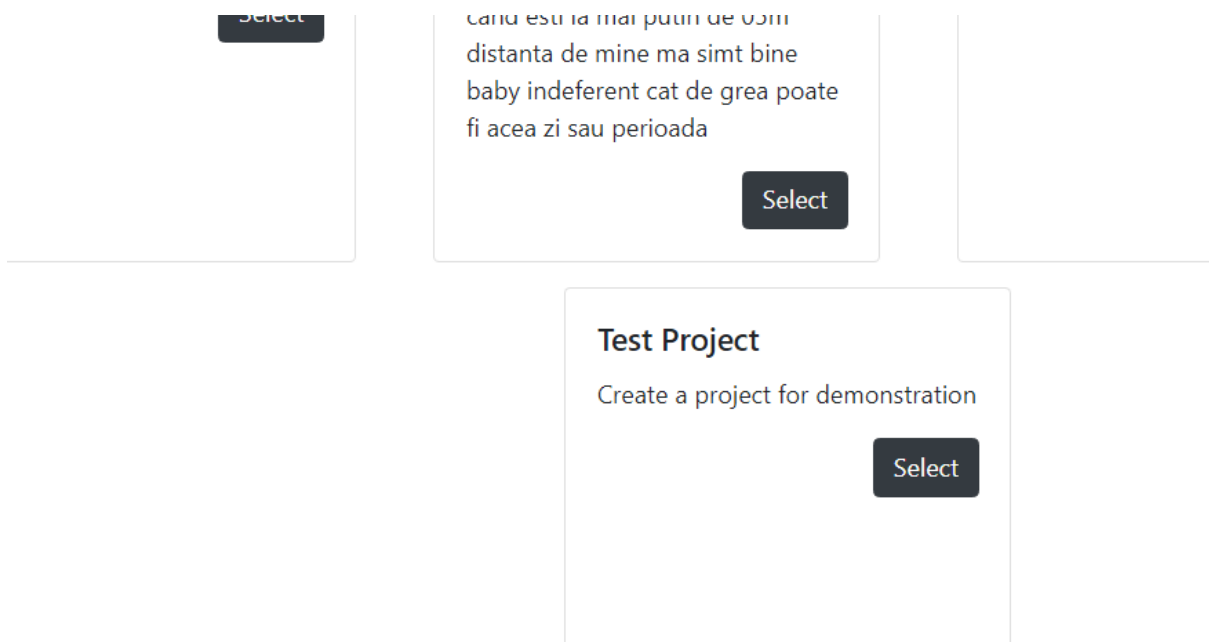


Figure 5.7 Project Created displayed in the list.

### 5.5.2 Cards

A card is a container that will display information about the projects, tasks, and progress. It offers the ability to customise the container with little style and markup. Bootstrap provides multiple options to customise the header, footer, content, and background of the card. This method is chosen to displays the content of the web pages because, in conjunction with PHP, it will allow the creation of

multiple similar elements, where each card will contain information about the project, task, or progress chart that will be placed inside them.

Below the Figure 5.7 represent how the card will be displayed and that the card will have a title, the description, and a button that, when clicked, will redirect the user to the next page. For example, if the user selects a project, the system will move to the page representing all the project's tasks.

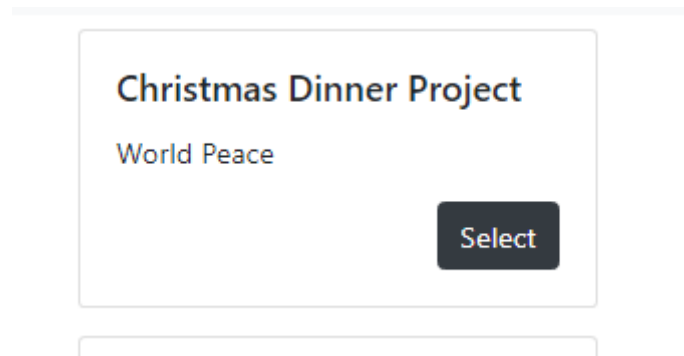


Figure 5.7 Card display in the browser.

When the page loads, the system will create cards for every element in the list that must be displayed. The following was achieved with the help of the for-each function in PHP. In Figure 5.8 below. The function will iterate through the elements that need to be displayed, create cards for each of them, and inject them in the HTML.

```
<!-- list of projects -->
<div class="d-flex flex-row flex-wrap justify-content-around">
  <?php foreach ($projects as $project) : ?>
    <div class="card m-2" style="width: 18rem;">
      <div class="card-body">
        <h5 class="card-title"><?= $project->title ?></h5>
        <p class="card-text"> <?= $project->vision ?></p>
        <form class="d-flex justify-content-end" method="post" action=" ../controller/tasksList.php">
          <input type="hidden" name="projectID" value="<?= $project->projectID ?>">
          <button type="submit" class="btn btn-dark card-link ">Select</button>
        </form>
      </div>
    </div>
  <?php endforeach ?>
</div>
```

Figure 5.8 Foreach code to display cards.

### 5.5.3 Progress bar

The progress of each project will be represented with the help of a bar that will aggregate all three states of a task: to do, in progress, done. This progress bar will allow us to visualise the work that was done toward the project and the work that needs to be done. In Figure 5.9 below can be seen the representations of the progress for all the projects. The page will also allow the user to search for a specific project or a group of projects that match the searching criteria.

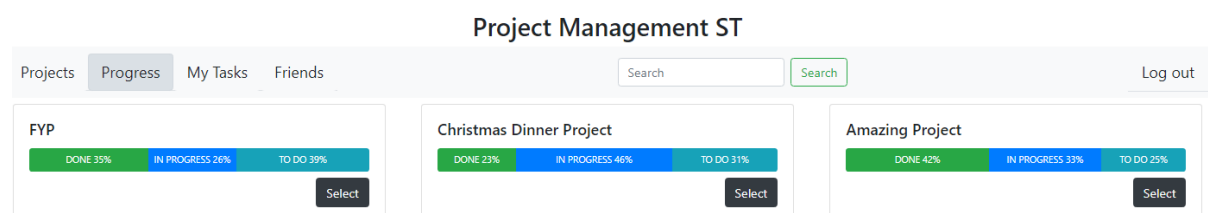




Figure 5.9 Progress bar

## 5.6 Backend

The backend of the system was built using PHP for the server-side code, SQL to interact and manage the database, and JavaScript to display messages to the users when they try to perform an action that is either not supported by the system or when the system denied the action.

### 5.6.1 Database connection

The system will interact with the MySQL database through the PDO (PHP Data Objects) and the PDO\_MYSQL driver. The PDO is a lightweight interface that will allow access and store data from the database. The interface itself will not connect and will not allow the system to perform functions on the database. That is why the system must use a specific driver to the database that stores all the information required for the project to work as the client expects. That driver will be PDO\_MYSQL because it will allow connecting and interacting with MySQL, the database that will provide the data from the system.

The Model responsible for the CRUD (create, read, update, delete) functionality will contain a file, "dataAccess.php", that will contain the connection to the database and the queries that need to be executed to achieve the expected requirements of the clients. In Figure 5.10 below can be seen how the connection to the MySQL database is made and a query that will extract all the projects for a specific user id then the response from the database will be fetched to a class "Project". The response of the function "getAllProjects" will be a list of "Project" objects returned to the part of the code that invoked the function.

```
$db_user = "k1809106";
$db_name = "k1809106";
$db_password = "k1809106";

$pdo = new PDO("mysql:host=kunet;dbname=$db_name", $db_user, $db_password);

function getAllProjects($userID)
{
    global $pdo;
    $statement = $pdo->prepare('SELECT Project.projectID, title, vision
                                FROM Project
                                INNER JOIN Team
                                ON Team.projectID= Project.projectID
                                WHERE Team.userID= ?');
    $statement->execute([$userID]);
    return $statement->fetchAll(PDO::FETCH_CLASS, 'Project');
}
```

Figure 5.10 Database connection

To fetch the data from the database there where created a couple of PHP classes that will store the values from the database, the classes do not need to match the tables in the database. However, it needs to match the queries that interrogate the database in this way the data that is returned by the query will be fetched to the correct fields of the class and will instantiate objects to represent the record. In this way, it will be easier to manage the data formatted as PHP objects inside the system. An example of how the queries will fetch the data from the database can be found in Figure 5.11, and the class that will store the results Figure 5.12.

```

function getAllProjects($userID)
{
    global $pdo;
    $statement = $pdo->prepare('SELECT Project.projectID, title, vision
                                FROM Project
                                INNER JOIN Team
                                ON Team.projectID= Project.projectID
                                WHERE Team.userID= ?');
    $statement->execute([$userID]);
    return $statement->fetchAll(PDO::FETCH_CLASS, 'Project');
}

```

Figure 5.11 FetchAll example

```

<?php
class Project {
    private $projectID;
    private $title;
    private $vision;
    private $tasks;

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

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

```

Figure 5.12 Project class

### 5.6.2 Controller

The controller part of the system will contain the business logic of the application. It will receive the user's requests and apply the specific functions that are meant to fulfil that request. The Controller will link the Model and View together by receiving input from the View, converting it to a command and executing it. If needed, it will pass the data to the Model, which will perform CRUD functions to the database. If the Model sends back data, the Controller will receive and prepare that data to be displayed in the View.

The Controller will start a session after the user will log in and require the files that are needed to display content to the user correctly and connect to the database to retrieve the data required by the user. Figure 5.13 can be found as an example of a controller responsible for the page that will display all the tasks assigned to the logged-in user. The Controller will start the session and check if the user is logged in. If that is not the case, then the system will redirect the user to log in. If the user

is logged in, then the Controller will require the files responsible for the Model, receive data, and prepare it to be passed to the View that will display the content to the client.

```
<?php
session_start();
if (!isset($_SESSION["loggedin"]) && !$_SESSION["loggedin"] === true) {
    header("location: loginRegistration.php");
    exit;
}

require_once("../model/project.php");
require_once("../model/task.php");
require_once("../model/dataAccess.php");

$projectsTemp = getAllProjects($_SESSION["userID"]);
$projects = array();
foreach ($projectsTemp as $project) {
    $project->tasks = getTasksByProjectIDAndUserID($project->projectID, $_SESSION["userID"]);
    if (count($project->tasks) > 0) {
        array_push($projects, $project);
    }
}

require_once("../view/myTasks_view.php");
```

Figure 5.13 Controller myTasks.

## 5.7 Conclusion

The implementation chapter had as goals to create the product that will meet the client's expectation. During the process, lots of new technologies were assimilated from creating the database and populating it with enough data to test the system under a high volume of data to check if it will still perform the same. Displaying the progress of each project so that the users will have a better understanding and have enough information to make decisions toward the projects that will be managed with the system that is created.

During the implementation phase of the project, it was implemented the majority of functionality. Following the MoSCoW prioritisation, all the must-haves were implemented and but there are still missing some could haves.

## 6 Validation Chapter

### 6.1 Introduction

The validation part of the system building is meant to test if the parts gathered so far meet the client's expectations and requirements. The testing of the solutions created for the project was done mainly internally, and the client was presented only with almost complete significant parts of the system.

The chapter is composed of 4 main subtopics. The requirements validation where it was agreed with the client what must be implemented in the system. The database validation was tested if the database that was designed can store and return data to the client in a reasonable time. The third topic is the usability test that was meant to present the front end created to the users and receive some feedback that will improve some bits of the system to make it more accessible and easy to use.

### 6.2 Validating the Project Requirements

Validating the project requirements is the process where it will be checked if the requirements gathered and prepared for future project building align with the needs of the customer. This process will consist of a walk-through of the list of requirements and a review of each requirement. This review of the requirements will be held with both the client and the developers when it was decided that the requirements list contains most of the project's requirements. At this meeting, all parties have gone through the entire list and checked if the requirements are valid if they are feasible if it is honest to implement them. Also, the parties can suggest new requirements to generate the system's full potential and truly meet the client expectations.

At the final of the meeting, the outcome must be a complete but not final list of requirements prioritised using the MoSCoW technic. The list will not be final because using the agile methodology to manage the project. It is possible to reiterate through each stage of the project multiple times to ensure that the final product will meet the updated requirements consistently.

### 6.3 Validating the Database Design

The database is crucial in this project because for almost every functionality the system will need to interact with the database connected to the system. Assessing the database was produced based on the following criteria: does the database store all the data needed by the client? Does the database respond with the required data in a reasonable amount of time? The database ERD was designed in concordance with the requirements in place and was tested and adjusted to host all the fields required to work properly. To test the ERD, it was created a table that will contain the requirements and the expected SQL Query that will execute that requirement. A part of this table can be seen in Figure 6.1 below. Another aspect of the database that needs to be tested was the data dictionary. It was optimised to cover most of the use cases in term of length of the attributes but the meantime so that it will be optimised in term of storage.

ID	Functional Requirements	SQL
1	Create Account	Insert Into User (name, surname, email, password) Values (\$name, \$surname, \$email, \$password)  Variables from HTML form
2	Log In	Select * From User Where name=\$name and password=\$password
3	Edit personal details	Update User Set columnName=\$detail Where userID=\$userID  \$detail info from form

		\$userID from php session
4	Create project	Insert Into Project (title, vision) Values (\$title, \$vision) SELECT LAST_INSERT_ID();not sure how it will work yet Insert Into Team (ProjectID, UserID) Values (The select value, \$UserID)  \$userID session variable
5	Update project	Update Project Set columnName=\$new Where ProjectID=\$projectID  \$projectID stored in a PHP page
6	Delete project	Delete from project where projectID=\$projectID  \$projectID stored in a PHP page
7	Edit vision of the project	Update Project Set vision=\$vision Where ProjectID=\$projectID  \$projectID stored in a PHP page \$vision from a form

Figure 6.1 SQL Queries

After the database design was tested and met all the gathered requirements, the next phase of database validation will be the response time that the database requires to respond to the system queries. After the database was deployed, it was populated with test data that is expected to be stored in the database when it will host users from a company. Then each query was run against the database, and the time need for the database to respond was recorded. If the time were reasonable, the database would not need improvements for that specific query or data layout. Each query took no more than 0.05 seconds to run, which means that the database was correctly designed and until there will not be any errors or will be added or changed any requirement that will involve the database it does not need any changes. The SQL query that is displayed in figure 5.3 run in 0.023 seconds. The proof of that can be found in figure 6.2 below.

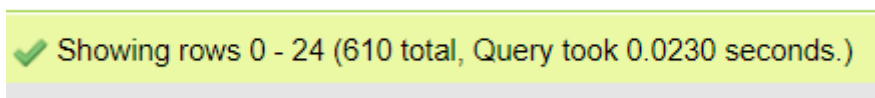


Figure 6.2 Time to select all the projects with numbers of task by category.

## 6.4 Usability testing

Before testing it with the client, the usability of the product has gone through the process called Guerrilla Testing or Hallway Usability Tests. This approach was chosen for its informal way to test the product. For this validation process, it is needed to have a couple of volunteers willing to spend a couple of minutes and try to solve a task by following a user journey. The participants were asked to complete a questionnaire while using the system. The questionnaire can be found in Table 6.3.

Task	Comment
Can you create a project	
Are you able to select it	

Can you create the task	
Can you add a comment to the task	
Change the status of a task	
Are you able to check the progress of the project	

Table 6.3 Usability test questionnaire

In Appendix F, two tables were completed by the volunteers who wanted to assess the product and give an opinion on it. It was chosen volunteers to test the system on different devices on a mobile phone and a laptop.

## 6.5 Code Testing Strategies

For testing the code of this system, it was chosen the black box technique. This method implies that the tester will check the system by not knowing how the system's components are working in the background or the code that represent them. This approach will focus on the requirements more than determining how the components and each feature in part need to be implemented.

The tester will elaborate test cases based on the requirements that were gathered previously. The goal is to check if the system will allow the user to execute the required functionality. The tester will try to predict and edge cases that will show how the system will act if the user introduces some erroneous data, conscious or unconscious. An example of the test tables created by the tester can be found in Table 6.4 below and a complete list in Appendix G.

Pass/Fail	Actual results	Expected results	Status	Description	Steps	Test Case	Functional requirements	Requirements	Process
Pass	Pop-up modal form with fields required for registration	Pop-up modal form with fields required for registration	done	Click on "Create One" on login page	1	Valid details	FR1		Register
			done	Insert valid details	2				
	Redirect to login page	Redirect to login page	done	Click on "Submit"	3				

Table 6.4 Test Case "Test register function."

## 6.6 Conclusion

The validation part of the project lifecycle is continuous, and it is not possible to evaluate it as complete. Only the database fully meets all the client's requirements, and the requirements are gathered fully according to the client. Together with the backend, the frontend is missing features that are not critical to the client but are needed to create a fully working product that will ease the project management for the client and future users.

## **7 Critical Review**

### **7.1 Introduction**

The final chapter of the report will summarise the work done during the entire life cycle of the project with reflections on artefacts that were created correctly and parts of the project that require improvements or changes to create the product desired by the client.

The chapter is composed of 5 sub-sections. Achievements where will be summarised the artefacts that have been gathered until now. Problems and limitation will be written on all the impediments of the project. Future work will contain information about how to avoid and remove all the limitation and problems encountered during the life cycle of the project. Legal, ethical, societal, and security issues have been researched and adapted the project accordingly. And the conclusion to the entire project.

### **7.2 Achievements**

Achievements of the project are the minimum vial product that can be presented to the client. That includes a website deployed and accessible over the internet by the client. It is securing the data with login functionality and hashing the user credentials that will secure the accounts of the users. The ability to create projects and tasks, form a team and track the status of each task and the progress of project.

The requirements artefacts are composed of the user stories created through elicitation of the client to create a list of functional and non-functional requirements. After gathering the list of requirements and prioritising the list. This list is the critical component of the entire project is based on this list.

The list of design artefacts is composed of two main parts the product and the database connected to the project. The database artefacts will contain the ERD (entity relation diagram) and the data dictionary to be able to implement the database. The artefacts meet the goal of storing precisely the data required by the project, and it is optimised to store the data efficiently. For the website, the artefact's creation started with sketches of the pages of the platform because this is the cheapest and fast option to decide with the client how the layout of the components will be. Based on sketches that were agreed with the client to move forward with the design, it was decided to create the prototype of the product with Adobe XD to create a clickable prototype to mimic the product functionality.

The final artefacts are the database that was built and deployed on the server. The database was populated with a large amount of data to test the time needed to access the data when the database is loaded. It meets all the requirements, and the time needed to extract data will not affect any way the user experience when using the app. The code that followed the MVC pattern to make it more maintainable and to be able to quickly add or update feature to meet the client requirements continuously. The front end is clean, and the functionality is quickly reachable. The backend uses the PDO to access data and will process the data and pass it to the views to be displayed. The website also was deployed to the server, and it is currently live. The product was also tested for the use cases that may create problems and adapted to handle them.

### **7.3 Problems and Limitation**

During the life cycle of this project, the biggest problem was the pandemic limitations that introduced some impediments on contacting the clients and the ability to negotiate online the design and the requirements that need to be done. Also, validating the artefacts online was a bit slower than usual. Even if online is possible to contact stakeholders at any time, because of inexperience in using the available tools to meet online, there were technical problems during the meetings, noise from the background, and even the ability to receive feedback was lower than at the face to face meetings. The

feedback from the clients is not only from the words that we receive but from the body language as well which was problematic to receive online.

Another problem was the responsibilities that were not related to the project, which did not allow the implementation of newer techniques, tools or frameworks to build the project to the most current standards. That leads to building the project with technologies that were already known to save time and provide a project that will meet the minimum required functionality.

A limitation related to the previous one was the low experience on building the entire project and gathering the documentation to cover all the aspects of the building process from the management of the project requirements gathering prototyping, building the project and testing it continuously keeping the client UpToDate.

## **7.4 Future Work**

The future work that needs to be done on the project is migrating the project to technologies that will allow the system to be split into two separate components. The backend to be represented by the REST APIs that will return the required data to the client. Furthermore, the frontend to be built with the help of a modern framework that will make API calls and display the data in a proper clean format. The frontend can be built with the help of the modern frameworks as a single page that will allow the page's design to be consistent and kept simple and even easier to navigate through the project.

After the project is migrated to the newer technologies, it will need to create the second user type. The second user type will be the supervisor who will manage the project and all the required details. The database was created to handle this type of requests.

Implement the missing requirements as the password recovery and account management page and the ability to create sprints and manage them. The current project can handle only project management in a Kanban approach.

The required additional functionality was postponed because of time constraints.

## **7.5 Legal, Ethical, Societal and Security Issues**

The project will follow the GDPR (General Data Protection Regulation) and ask for consent when it stores users' details. The application must also, upon request of the user, provide the ability to erase personal details. In case of any breach, it must report to users affected by that in 72h.

The main issue that may occur is the security levels that must be implemented. The application will store a minimum of the personal details of users. However, the main concern is the projects in which some of them can have confidential information that needs protection. It must provide a good level of security and encrypt essential data. Besides that, it must consider people with bad intentions who do not want to steal data but corrupt it. In this case, a couple of steps must be undertaken. Some of them are filtering requests that get to the database, clearly identify users before responding to the requests, backup the data.

When the platform will be published, it will inform the users that it may contain critical bugs that may not be found during the production phase and must be cautious with the information provided. The data may be unavailable even if the development team tries to solve the issues quickly.

## **7.6 Conclusion**

The project meets the minimum requirements that were negotiated with the client. It, of course, also needs additional functionality to make it stand out from the competitors, but that will be



achieved in the subsequent iterations. Hopefully, it will meet all the client expectations, and it will be possible to replace the current system. It was chosen a wrong approach to use the skills that already existed instead of spending the time learning the new ones that would possibly allow to build the project more professionally and create an easier to use project that would meet the requirements. The modern technologies would allow testing the product and making it more secure and preventing bugs early in the implementation stages. The newer frameworks as well would allow creating an easy to maintain project.

It will be discussed with the client who approaches to be taken for future works to continue the work and to deliver the project as agreed or to change technologies that are implemented and to deliver the project that will be easier to maintain and to add new functionality easier

## References

*Agile Manifesto* (2001) Available at: <https://agilemanifesto.org/> (Accessed: 04/01/2021)

*Agile software development* (2020) Available at:  
[https://en.wikipedia.org/wiki/Agile\\_software\\_development](https://en.wikipedia.org/wiki/Agile_software_development) (Accessed: 04/01/2021)

Alexander, Moira. "Agile vs. Waterfall: Project Methodologies Compared." *CIO* (2020): CIO, 2020-10-05. Web.

Atlassian (no date) *Scrum*. Available at: <https://www.atlassian.com/agile/scrum> (Accessed at 04/01/2021)

Bloch, M. (2012) *Delivering large-scale IT projects on time, on budget, and on value*  
Available at: <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value> (Accessed: 04/01/2021)

Pavel, N. (2019) *Different Agile Methodologies: Find Which One Fits Best Your Needs*  
Available at: <https://kanbanize.com/blog/right-agile-methodology-for-your-project/>  
(Accessed 04/01/2021)

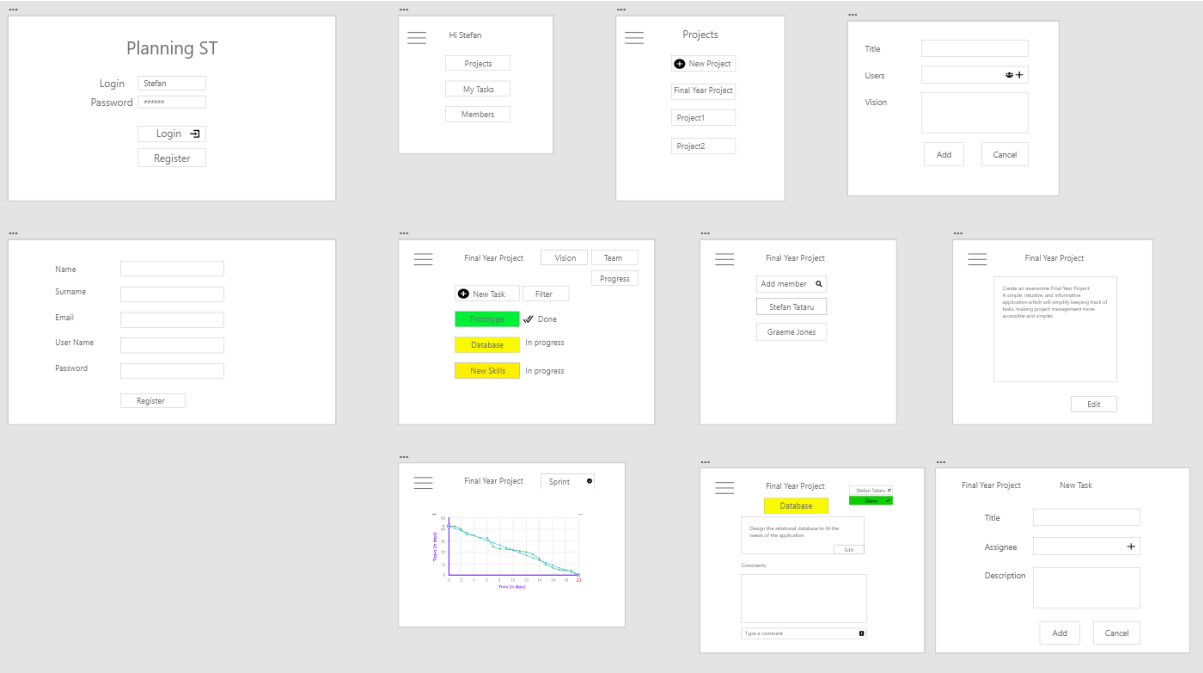
'Kanban' (2021) *Wikipedia*. Available at: <https://en.wikipedia.org/wiki/Kanban> (Accessed: 20 April 2021)

# Appendices

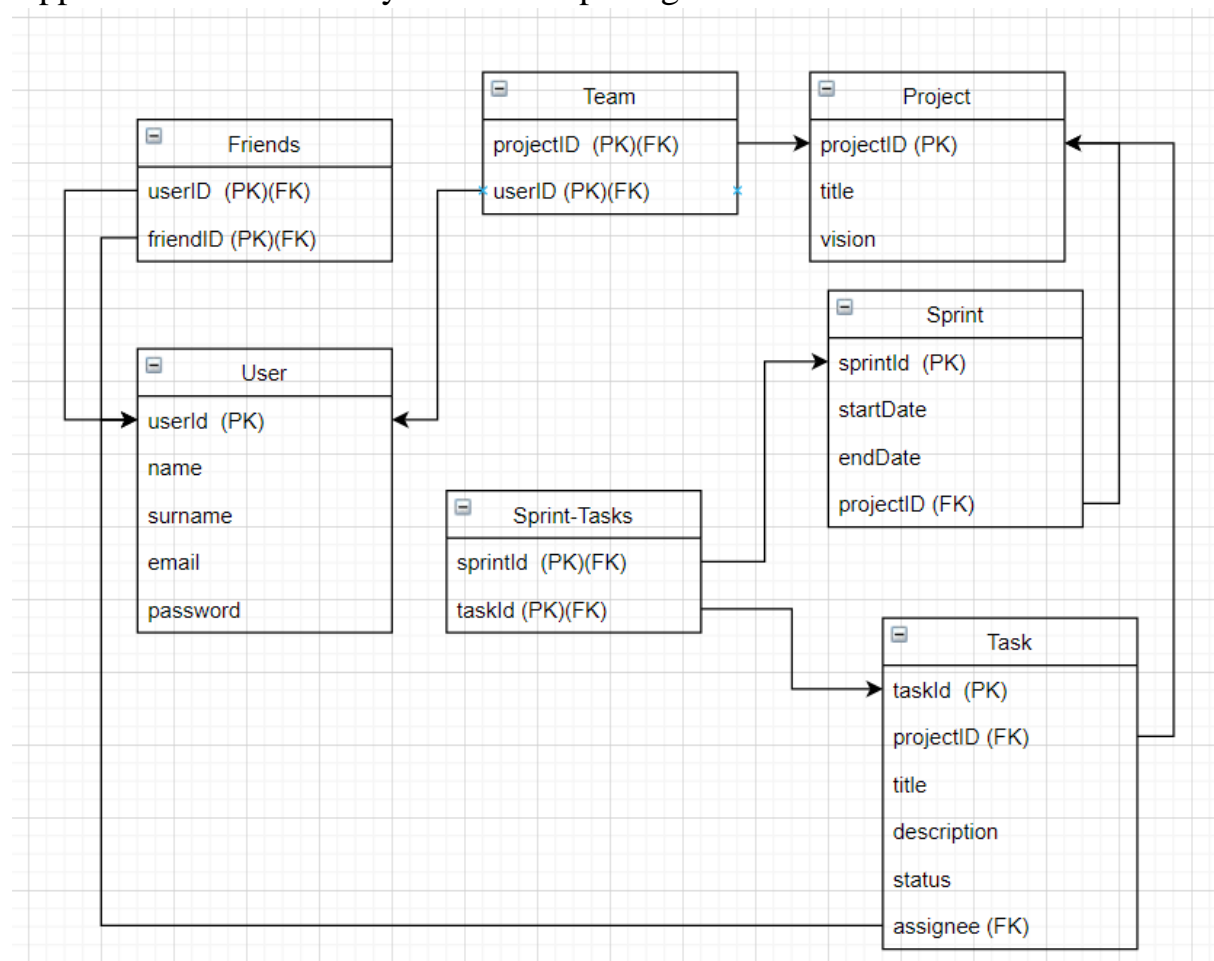
## Appendix A - Deliverables

25/09/2020	Documentation	Proposal Appendices	08/01/2021	Research	Key References and Mind Map
02/10/2020	Methodology	SWOT / Risk Analysis	08/01/2021	Implementation	Key report 2
02/10/2020	Documentation	Complete draft proposal	08/01/2021	Assessment	Early Prototype demonstration
02/10/2020	Research	Portfolio of Past Dissertations	15/01/2021	Research	Data generation plan
09/10/2020	Methodology	Initial ERD (Initial data structures)	15/01/2021	Implementation	Key form 2
09/10/2020	Implementation	Install development platform	15/01/2021	Documentation	Updated Design Chapter
09/10/2020	Assessment	Project Proposal	22/01/2021	Research	Identify most complex SQL
16/10/2020	Methodology	Stakeholder Analysis / User Types	22/01/2021	Research	Identify viva story
16/10/2020	Implementation	Establish PHP connection to database	22/01/2021	Implementation	Key form 3
16/10/2020	Documentation	TOC Requirements Chapter	22/01/2021	Implementation	Key report 3
23/10/2020	Research	Volume of testing data required	29/01/2021	Documentation	Draft Review Chapter
23/10/2020	Implementation	Simple SQL and PHP report	29/01/2021	Implementation	Key form 4
23/10/2020	Methodology	Use Cases and User Stories	29/01/2021	Implementation	Key report 4
30/10/2020	Research	Identify Review Topic	29/01/2021	Evidence	Populate database with required data
30/10/2020	Methodology	List of Requirements / MoSCoW	05/02/2021	Documentation	TOC Implementation Chapter
30/10/2020	Implementation	Simple Form with SQL insert	05/02/2021	Implementation	Key form 5
06/11/2020	Documentation	Draft Requirements Chapter	05/02/2021	Implementation	Key report 5
06/11/2020	Research	Identify SQL queries from requirements	05/02/2021	Documentation	Technologies and Architecture Section
06/11/2020	Implementation	Inner Join report	12/02/2021	Evidence	Test Tables
13/11/2020	Methodology	Sketches	12/02/2021	Implementation	Key form 6
13/11/2020	Research	Bootstrap/Mobile API Interface	12/02/2021	Implementation	Key report 6
13/11/2020	Documentation	TOC Design Chapter	12/02/2021	Documentation	Second Major Section
20/11/2020	Methodology	Activity and Sequence diagrams	19/02/2021	Research	Identify viva-oriented functionality
20/11/2020	Implementation	CSS Mockup of Report and Form	19/02/2021	Evidence	Simple reports utilising big data
20/11/2020	Research	Ten web and five iCat articles	19/02/2021	Documentation	Updated Review Chapter
27/11/2020	Implementation	Key report 1	26/02/2021	Documentation	Draft Implementation Chapter
27/11/2020	Methodology	Revised ERD and Data Dictionary	26/02/2021	Implementation	Login functionality
27/11/2020	Documentation	Updated Requirements Chapter	05/03/2021	Implementation	Dashboard 1
04/12/2020	Implementation	Key form 1	05/03/2021	TBC	TBC
04/12/2020	Research	Plan for prototype demonstration	12/03/2021	Documentation	Draft Validation Chapter
04/12/2020	Methodology	Test Strategy	12/03/2021	Evidence	Inner joins utilising big data
11/12/2020	Documentation	Draft Design Chapter	12/03/2021	Implementation	Dashboard 2
11/12/2020	Methodology	Wireframes and Navigation Diagram	19/03/2021	Documentation	Draft Introduction and Conclusion
			26/03/2021	Research	Revised viva-oriented functionality
			26/03/2021	Documentation	Appendices and references
			02/04/2021	Implementation	Dashboard 3
			02/04/2021	Presentation	Practice Vivas
			09/04/2021	Documentation	Complete draft report (and submitted)
			16/04/2021	Implementation	Last viva functionality implemented
			23/04/2021	Presentation	Practice Vivas
			30/04/2021	Documentation	Report Submission

# Appendix B - Wireframes



## Appendix C - ERD Entity Relationship Diagram



## Appendix D - Wireframes

Register

Username

Email

Password

Name

Surname

Register

Figure A.1

Status

Prototype

FYP

Assignee

Status

Design the relative needs of the application

Status

In Progress

Apply

Comments

Type a comment

Figure A.2

Assignee

Prototype

FYP

Assignee

Status

Design the relational database needs of the application

Assignee

Stefan

Done

Comments

Type a comment

Figure A.3

Add Project

Planning ST

Projects

MyTasks

Progress

Friends

New Project

FYP: Project management system

Christmas Dinner Project

Project: Short description

Coursework: aims and objectives

Insert Project

Project Title

Vision

Submit

Figure A.4

Add Task

FYP

Home Vision Progress Team New Task

Prototype

Database

ERD

Design Chap

New Task

Task Title

Description

Assignee

Submit

Figure A.5

Members

FYP

Home Vision Progress Team New Task

Prototype

Database

ERD

Design Chap

Members

Task Title

Description

Assignee

Submit

Figure A.6



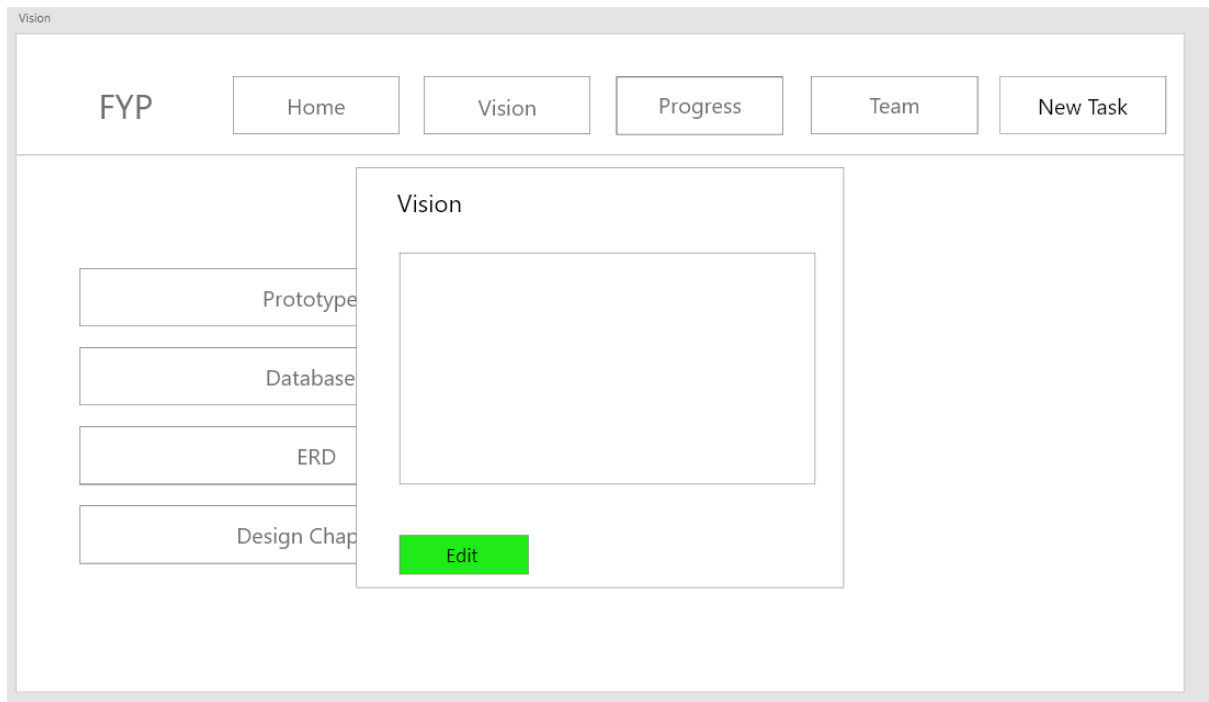


Figure A.7

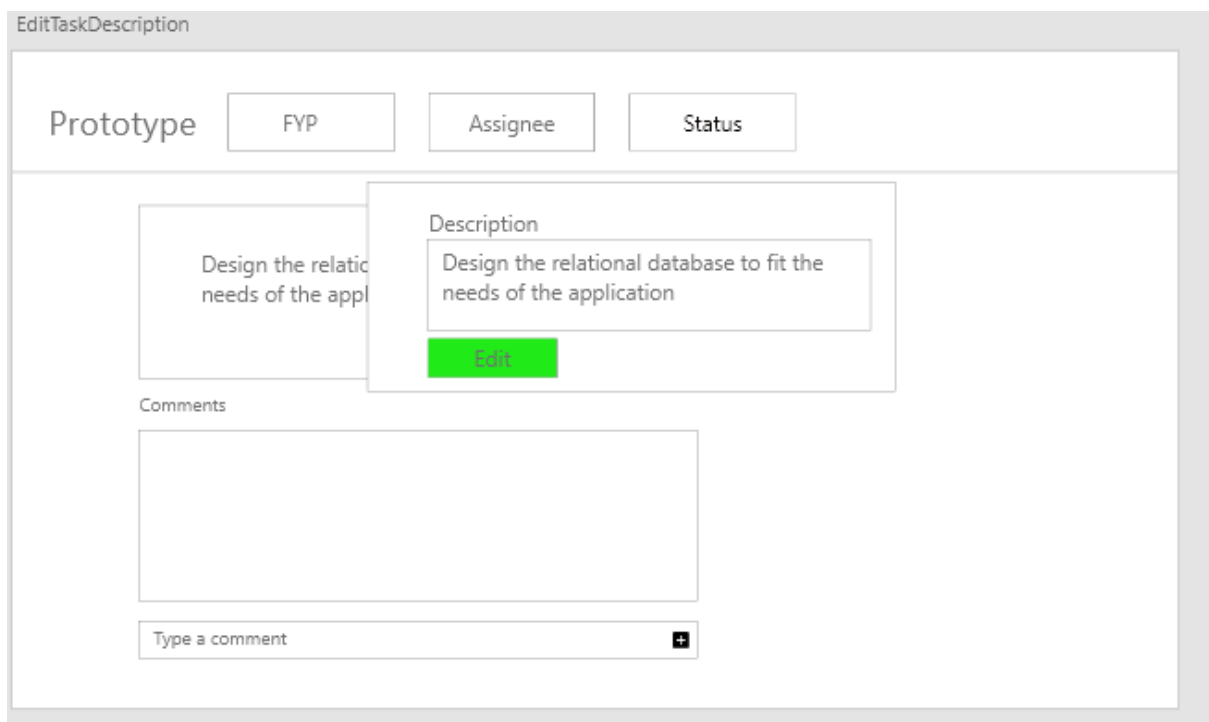


Figure A.8

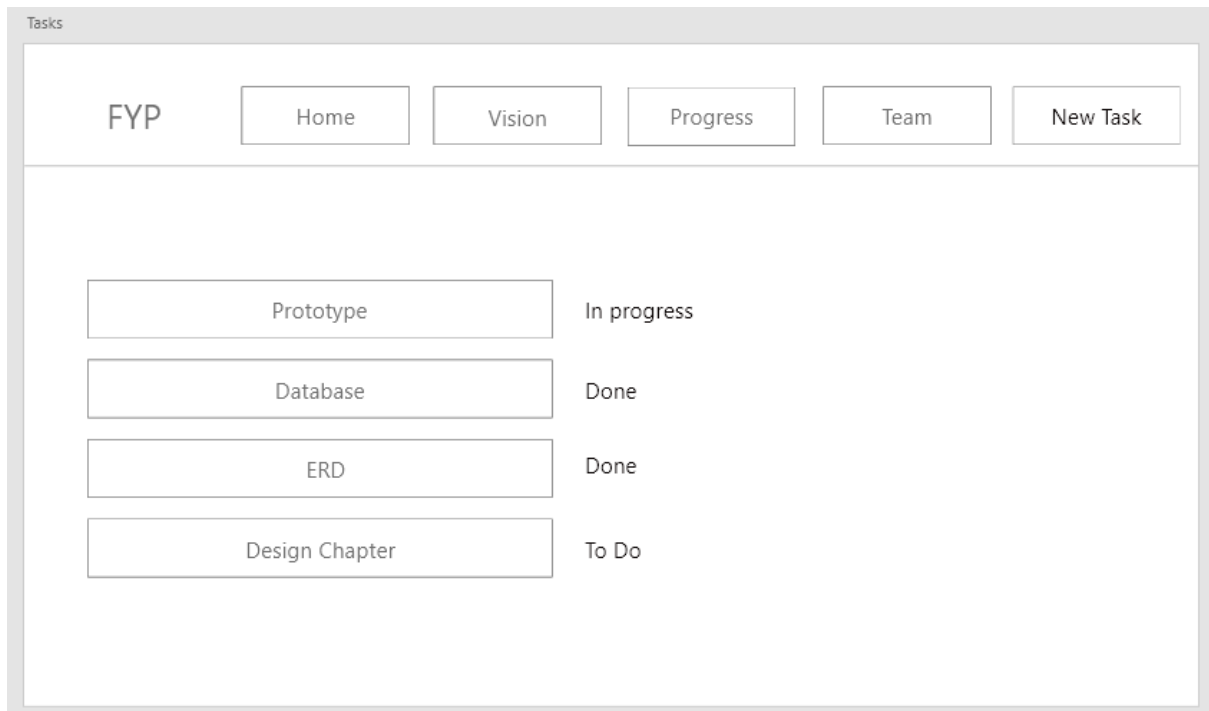


Figure A.9

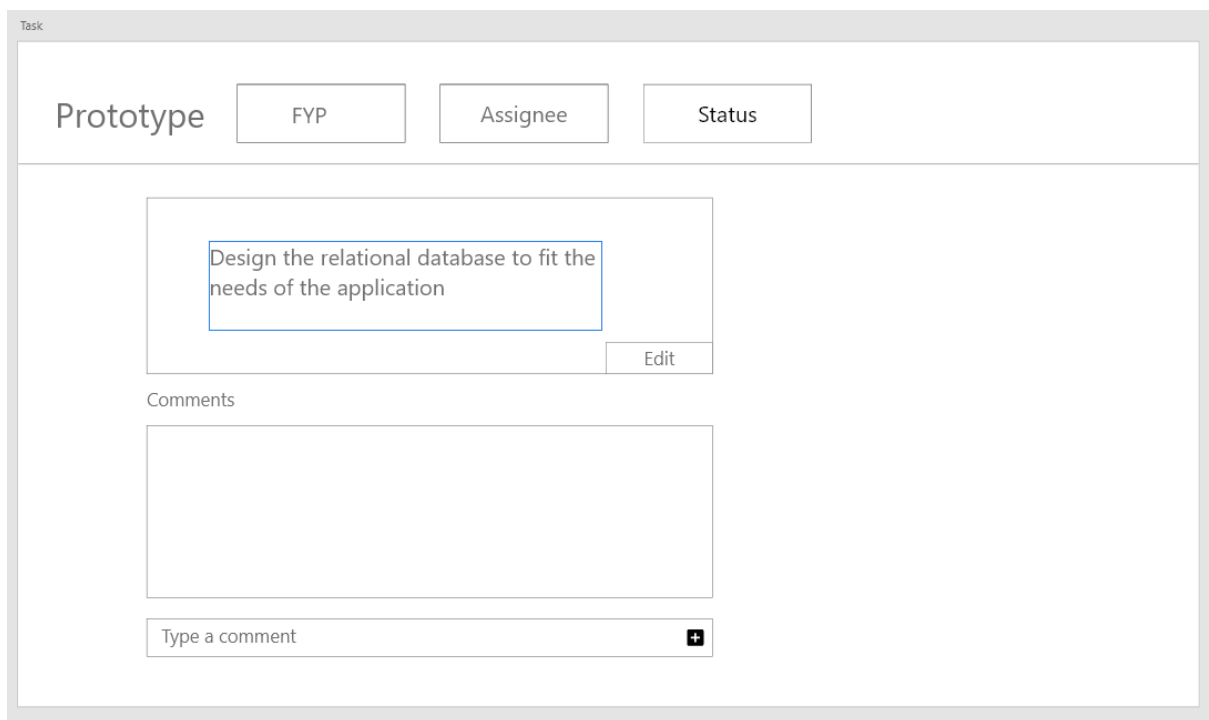


Figure A.10

## Appendix E - Use Cases

Use Case 1	Add Task	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select project
	3	Click on a new task
	4	Insert information about the task (Title, Assignee, Description)
	5	Click Add

Table A.1

Use Case 2	Manipulate task (delete, assign, update (content, status))	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select project
	3	Select a task
	4	Choose the desired action
	4.1	Edit
	4.2	Delete
	4.3	Assign
	5	Input information if required

Table A.2

Use Case 3	Visualise progress	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select project
	3	Click on progress

Table A.3

Use Case 4	Create a Project	
Precondition	Logged in	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Click on New Project
	3	Insert information about the project
	4	Click on add

Table A.4

Use Case 4	Update vision	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select project
	3	Click on vision
	4	Edit the text
	5	Click save

Table A.5

Use Case 5	Add/remove Team members	
Precondition	Logged in, existing project	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select Project
	3	Click on team
	4	Click on Add member/Remove person
	5	Find the person
	6	Click on Done

Table A.6

Use Case 6	Comment Task	
Precondition	Logged in, Existing project, existing task	
Success End	Store the changes to the database	
Failed End	Unable to perform the request	
Description	Step	Action
	1	Click on projects
	2	Select Project
	3	Click on Task
	4	Insert comment
	5	Click add

Table A.7

## Appendix F -Usability test questionnaire

The user that did the test on a mobile phone:

Task	Comment
Can you create a project	Yes, by clicking on the button that is on the navbar and completing the fields
Are you able to select it	Yes
Can you create a task	Yes, the process is very similar to creating a project
Can you add a comment to the task	It needed some time until I figured out that I have to navigate to task and scroll down to find the box for adding a comment
Can you change the status of a task	Yes, after selecting the task on the navigation bar
Are you able to check the progress of the project	Yes, by selecting the progress form the project page

The user that did the test on a laptop:

Task	Comment
Can you create a project	Yes, the button is visible
Are you able to select it	Yes
Can you create a task	Yes, after selecting the project
Can you add a comment to the task	Yes
Can you change the status of a task	Yes, after selecting the task on the navigation bar
Are you able to check the progress of the project	Yes, by selecting the progress on the main page then searching for the required project

## Appendix G - Test Case Tables

[illegible]



[illegible]



Project list	logged in,	FR4	add project	1	Click on New Project	done	pop-up form with fields required to create a new project	pop-up form with fields required to create a new project	Pass
				2	insert valid details	done			
				3	Click on Submit	done	redirect to project list	redirect to project list	
			add a project with empty fields	1	Click on New Project	done	pop-up form with fields required to create a new project	pop-up form with fields required to create a new project	Pass
				2	let the fields empty	done			
				3	Click on Submit	done	the message "Please fill out the fields."	the message "Please fill out the fields."	
		FR25	Search	1	Type full or partial name of the project	done			Pass





[illegible]

		FR1 6	see progr ess	1	Click on progres s		redirected to the progress page where is displayed the progress of the project	redirected to the progress page where is displayed the progress of the project	pass
		FR2 9	searc h	1	Type full or partial name of a task	done			Pass
					Click search	done	The list of tasks get updated with projects that match the criteria	The list of tasks get updated with projects that match the criteria	
Task page	Logged in,	FR3 0	edit descr iption	1	click on edit	done	pop-up form with fields required to update the description	pop-up form with fields required to update the description	Pass
					edit the text	done			
					Click submit	done	the description will be updated with the new text	the description will be updated with the new text	
		FR1 5	chan ge assig nee	1	click on assigne e	done	pop-up form with fields representing the current assignee and a list to select the new assignee	pop-up form with fields representing the current assignee and a list to select the new assignee	Pass

					select the new assignee from the team	done			
					click change	done	the assignee now is changed	the assignee now is changed	
		FR14	update status	1	click status	done	pop-up form with a field representing the status		Pass
				2	select a new status for the task	done			
				3	Click submit	done	The task status is now changed to the selected one	The task status is now changed to the selected one	
		FR11	Add comment	1	Type in the field a new comment	done			Pass
				2	click send	done	The comment will be appended to the bottom of the comments list	The comment will be appended to the bottom of the comments list	