Student Mark Management Portal

Zakiya Safi 1319070

Ahmed Ali Karani 1036074

Zubair Ahmed Bulbulia 1249593

Kyle Morris 1112649

5 October 2018

Contents

A. Pair Responsibilites   60

1. Introduction

The following report document contains the software information of a Student mark management system, affectionately named Gr(A+)der. Within this report, all necessary system design is covered, namely: project model, all system framework information, project scope and use cases. Furthermore, the system implementation includes testing, task-responsibility breakdown, as well as all functional system layers.

1.1. Project Definition

It is required to develop and implement a student mark system, which allows a convenient, user-friendly way for staff to record marks, and students to view and query marks.

1.2. Problem Statement

It is required that all available marks for enrolled courses corresponding to a specific student number should be visible by that student.
Student marks must only be viewable by users who have access to them.
Marks must only be able to be uploaded, and edited by administrative personnel.

Thus, one of the most important tasks is the restriction of access to certain users. Therefore access control will be implemented in the form of username/password combinations that achieve the above during the login process.

1.3. Project Objective

The portal needs to produce an efficient, user-friendly system which ensures the following:

⋆ A student can view their available marks

⋆ A student can query their marks

⋆ A student can only view their own marks

★ Administrative personnel can upload and modify marks.

1.4. Project Stakeholders

Below is a list of stakeholders:

★ Student

★ Course Coordinator

★ School Administrator

★ Database Administrator

★ System Administrator

2. Requirement Specifications

2.1. Project Scope

The task requirement is to develop a web-based system with the capabilities of being able to record marks of various assessments, for a multitude of students. It will thus function as a database for all student marks, across various courses. On the front-end, it is required to implement separate functionality and limitations for three different categories of user - student, course coordinator and school administrator. There must exist the capability for Course Coordinator and School Administrator to be able to access, edit and update marks on the database, as well as being able to view student marks. From the student side, they need to be able to view their marks from the 'portal'. They must also have the ability to query a certain assignment mark if they believe there to be an error with it. Students must have the limitation of not having access to other students' marks, and not being able to make modifications to their own marks. There are two back-end roles that need to be fulfilled: The database administrator, who is responsible for ensuring the correct format of marks is recorded, that there are no erroneous values stored, and that backups exist and are frequently maintained, and the Systems administrator who

is tasked with ensuring the correct system configuration is established, as well as the security perspectives of the portal.

2.2. Requirements Gathering

It is important to understand what is required from the system in terms of its stakeholders in order to deliver a valid solution. This can be done by correctly extracting specifications from the problem at hand. In developing a system that verifies these specifications, a valid solution can be delivered.

The approach taken was extracting information about users and their activities in the system from the problem statement. The activities were then translated into user requirements and further translated into system requirements. Due to the relatively short time frame provided to the team, the iterative prototyping method was used in order to aid in finding additional requirements while progressing in terms of delivering a minimum viable product (MVP).

Since the system (Gr(A+)der) is meant to join the suite of other systems offered by the university (such as Self-Service and Sakai), research was conducted on how these portals work. The results allowed the team to understand the standards expected from the new system - in terms of performance, look and feel, functionality, required input on login, flow between pages, and overall quality. The research provided requirements that will allow the system to seamlessly integrate into the existing suite.

Research was conducted on essential functionality that a system should contain in the present and future to gather further requirements that the team may have overlooked.

The requirements gathering process was essential in starting the prototyping phase correctly.

2.3. Product Description

The project requirement that the group is required to fulfil is that of the Student Mark Management System. The end result will be a fully-functioning web portal where students are able to view their marks for a number of courses.

2.3.1. User Characteristics

Below describes each user of the system, and how they interact with the system:

Student: The system will create and display all available marks for a student, in a user-friendly way. The system will allow the student to query marks as well as provide the student with marks that are required in future assessments to obtain a pass in the course.

Course Coordinator: The Course Coordinator will be able to add courses to the system, assessments pertaining to courses, and weightings of assessments. The Course Coordinator will also enrol students in courses and enter their marks per assessment. These functionalities will be provided by the system.

School Administrator: The system will provide the School Administrator with the same functionality as that of the Course Coordinator as well as additional functionality. The additional functionality includes generating statistics regarding the performance of courses the School Administrator is involved in as well as viewing information on plagiarism offences.

2.4. Formal Requirements Definition

2.4.1. Requirements

The below tables will list and describe the requirements as understood by our Gr(A+)der system. Each table will list the requirements, divided into relevant categories, followed by columns that classify the requirement as either:

F: Functional (core functionality that is integral to the system) or,

NF: Non-functional (the methods in which the core functionality will be delivered).

The further classification is then split up into:

M: Mandatory (requirements key to project success), or

O: Optional (requirements that are not necessarily key to a successful project but may improve the final product).

	Performance Levels	F	NF	M	O
1.	The system must handle all traffic levels without severely impacting the user experience.		X		X
2.	The system must work efficiently on all devices.	X		X	
3.	The system should have quick response times in completing functions.		X	X	

	Scalability	F	NF	M	O
1.	The system should be able to manage, and adapt to a growing database size, without causing major system disruption.		X		X
2.	The system should allow for addition in performance hardware to deal with increasing traffic in the future.		X		X

	Interface	F	NF	M	O
1.	The system should provide an interface that is usability focussed allowing access core system functionality.	X		X	
2.	The system should provide a user-friendly interface that is intuitive for all system users.		X		X
3.	The system should provide an aesthetically pleasing interface.		X		X
4.	The system should provide an interface with a minimal learning curve.		X	X	

	Functional Components	F	NF	M	O
1.	Allow the Course Coordinator to add a course.	X		X	
2.	Allow the Course Coordinator to enrol students.	X		X	
3.	Allow the Course Coordinator to add assessments.	X		X	
4.	Allow the Course Coordinator to add or change a student mark in an assessment.	X		X	
5.	Generate statistics for the School Administrator.	X		X	
6.	Allow the School Administrator to view offences.	X		X	
7.	Provide students with results pertaining to assessments.	X		X	
8.	Provide students with percentages required (in future assessments) to pass a course.	X		X	

	Reliability	F	NF	M	O
1.	The system should always be available for the users to interact with.		X	X	
2.	The system must provide all available marks to the student.	X		X	
3.	The system should consistently deliver accurate output to the user.		X	X	

	Security	F	NF	M	O
1.	A student should only be able to view his/her marks.		X	X	
2.	An authentication system (involving a username and password) should be implemented in order to securely access personal information and ensure that lower level users cannot access higher level functionality.		X	X	
3.	The Course Coordinator should be restricted from School Administrator specific priorities.		X	X	
4.	The Student should be restricted from tasks that are specific to the Course Coordinator and the School Administrator.		X	X	

	Robustness	F	NF	M	O
1.	In case of system failure, the system should be able to recover quickly.		X	X	
2.	The system should respond to erroneous input through defensive programming.		X	X	
3.	The system should be respond to a diverse set of devices.		X	X	

	Quality	F	NF	M	O
1.	Provide a well polished system overall.		X	X	
2.	Provide a system with minimal bugs ensuring tasks are carried out correctly and consistently.		X	X	

2.4.2. Assumptions

Below are defined the assumptions for each user type. In constructing the system, we assume the following:

2.4.2.1. Student:

⋆ Student marks will be available from the portal.

⋆ Students feel comfortable logging into and using the web portal.

⋆ Students have an understanding of a web-based application and so understand possible system delays.

⋆ Students make effective use of the system.

2.4.2.2. Course Coordinator:

⋆ CC will be responsible for uploading marks onto the database.

⋆ CC is comfortable with dealing with and making changes to the system.

⋆ CC understands how to apply a certain weighting to a course.

2.4.2.3. School Administrator:

⋆ SA knows how to navigate between different courses.

⋆ SA is comfortable with creating comparative statistics on different courses.

⋆ SA is able to identify students with offenses and can remove them from the course (unenrol).

2.4.2.4. Database Administrator:

* The database is correctly created with the appropriate relationships between entities being implemented.

* The data types of respective fields are appropriately defined.

* Any erroneous data that appears during testing will be rectified.

* Backups will be done regularly, and stored appropriately.

2.4.2.5. Systems Administrator:

* The security of the system will ensure no ungranted access is provided to respective users.

* The system will communicate between the front and back-end without error.

2.4.3. Dependencies

The below describes vital information and system capabilities which are necessary for the system to run optimally:

* **Access to information:** In order for the system to fully function, we require access to certain university property. This is required as part of the initial system implementation, and is in the form of student, course, and assessment information.

* **System resources:** We require the front and back ends of the system to effectively interact and communicate, in order for the system to correctly function.

* **Internet connectivity:** It is required to have an internet connection in order for the system to request and access information off the remote server, and also for students to view their marks on the portal page.

* **Database:** For the system to be operational it is necessary to have an integrated database with the correct student, course, and assessment information.

Software Dependencies:

- ⋆ **Maths/Stats/Graphs:**Access to the Desmos/Plotly API or R for use for reporting purposes, to create graphs of course averages, etc.

2.4.4. Constraints

The system will be constrained in terms of:

- ⋆ Time provided (semester to design, create and implement project).

- ⋆ Resources provided: this includes storage space and performance of servers from which we will access and store data.

2.4.5. Operating Environment

2.4.5.1. Operating System

Any Operating System containing a web browsing application.

2.4.5.2. Architecture

Web-based application; client-server web app. Having a client-server architecture provides many useful advantages; namely having a centralised database management system located server-side provides an added layer of data access control and data security. It also allows for concurrent access and centralised processing.

2.4.5.3. Database

Centralised, relational SQL database.

2.4.5.4. Database Type

Centralised (Database records will be on a main server).

2.4.5.5. Platform

ASP.NET 2.0.

2.4.5.6. Object Relational Mapper

Entity Framework 6.

2.4.5.7. Languages

Back-end: C# and LINQ for database querying.
Front-end: HTML, CSS, and JavaScript.

2.4.6. Data Management

This concerns the storage of the input and output within the system. All data being used in the system will be stored in a database on a server provided by the University. The data will then be accessed by the logical layer and displayed through the interface layer of the 'Gr(A+)der' system. At this point the data will be informative.

2.4.7. Portability

The system should have the capabilities to run on any device that has a web browsing application without the need for additional browser plug-ins.

2.5. Use Cases

2.5.1. Use Case Set

Below is a list of project Use Cases that were identified as being necessary for the system. They are organised according to the feature they relate do. The Use Cases have been structured in such a way to allow for more efficient administrative design to be implemented.

2.5.1.1. Core Use Cases

These use cases represent the core functionality of our system.

Use Case	Actor
View Marks	Student
Query Marks	Student
Upload Marks	Course Coordinator/ School Administrator
Add Course	Course Coordinator/ School Administrator
Add Assessment	Course Coordinator/ School Administrator
Enrol Student	Course Coordinator/ School Administrator
View Offences	School Administrator

2.5.1.2. Reporting Use Cases

These use cases are related to the integrated reporting capabilities of our system.

Use Case	Actor
View Comparative Reports	School Administrator
View Course Reports	Course Coordinator

2.5.1.3. Maintenance Use Cases

These use cases deal with the creation, deletion, viewing and updating of all other entities not included in the core use cases, but still required for the functionality of the system.

Use Case	Actor
Maintain Student	Course Coordinator/ School Administrator
Maintain Course Coordinator	Course Coordinator/ School Administrator
Maintain School Administrator	School Administrator

2.5.2. Use Case Description

2.5.2.1. Core Use Cases

View Marks

A student will select the View Marks option on their home page, accessible only after

they have logged in. On doing so they will be presented with a list of all the courses for which they are enrolled. They will select a particular course, and will be able to see all the assessments for that particular course, and on request their marks for those assessments. They will also be able to view the mark(s) that need to be achieved on their remaining assessment(s) to achieve a pass for that particular course.

Query Mark

While a student is viewing their marks, if they believe that their mark for an assessment is incorrect for any reason, they will select the Query Mark button associated with that particular assessment. They will then be asked to fill in a form explaining their reason for querying the mark. This query will then be sent to the respective Course Coordinator. A student is only allowed 3 legitimate queries per year, and the system will ascribe an offence to a student if they submit a query which turns out to have no error. I.e if a student has 3 queries remaining, they query a mark which has no error, they then lose a query (i.e. they then have 2), but if they query a mark which is an error, they keep that query.

Upload Marks

A CC or SA will select the Upload Marks option on their home page, accessible only after they have logged in and selected a particular course and assessment. On doing so they will be presented with a list of all students enrolled in that course, and will then enter a mark for every student. When they click the Save button, these changes will be saved to the database.

Add Course

A CC or SA will select the Add Course option on their respective homepage They will

then be presented with a form where they will have to fill in the details of the course, including the course name, course code, and course coordinator. If the CC is adding a course, they will automatically be set as the Course Coordinator. If the SA is adding a course they must select a course coordinator from a list of all the CCs on the system. Upon completing this form they will select the Save button and the new course and its details will be saved to the database.

Add Assessment

From the homepage of the CC or SA, they will be able to view courses. They will then be presented with a list of all their courses. When they select a particular course, they will be able to view all the assessments for that particular course, and add new assessments. When they select to add an assessment, they will be presented with a form where they must fill in the details for that particular assessment, including the assessment name, total mark as well as the course percentage (the total percentage that this assessment contributes to the entire course). Upon completing this form they will select the Save button and the new course and its details will be saved to the database.

Enrol Student

From the homepage of the CC or SA, they will select to view courses. They will then be presented with a list of all their courses. When they select a particular course, they will be able to view all the students enrolled in that particular course, and enrol new students in that course. When they select to enrol a student they will be presented with a form where they may enter the student IDs of the students to be enrolled. The system will cater for the user to enter a list of student IDs of many students who need to be enrolled, for the ease of the user. Once they select Save, these students will then be enrolled in the course and the changes will be saved to the database.

View Offences

A SA will select the View Offences option on their homepage. They will then be presented with a list of all students who have committed an offence, as well as the details of the student offence, including the course, date and assessment.

2.5.2.2. Reporting Use Cases

View Course Reports

A CC will select the View Course Reports option on their homepage. On doing so they will be presented with a list of all their courses. When they select a particular course, the report for that particular course will be displayed.

View Comparative Reports

A SA will select the View Course Reports option on their homepage. On doing so they will be presented with a list of reports. When they select a particular report, that report will be displayed.

2.5.2.3. Maintenance Use Cases

Maintain Student

Includes the use cases Add Student, View Student, Update Student and Delete Student.

Maintain Course Coordinator

Includes the use cases Add Course Coordinator, View Course Coordinator, Update Course Coordinator and Delete Course Coordinator.

Maintain School Administrator

Includes the use cases Add School Administrator, View School Administrator, Update School Administrator and Delete School Administrator.

Maintain Assessment

Includes the use cases View Assessment, Update Assessment and Delete Assessment. (Not Add Assessment as this is already included and detailed in the core use case set).

3. System Design

A 3-tier approach was used in designing this system as it allowed the team to break the problem statement into three main areas requiring attention and also allowed for splitting of tasks amongst team members. The three tiers consist of the presentation layer, business logic layer, and data layer. The three tier approach was also chosen as it is in accordance with the MVC (model, view, controller) architecture being used in development of the project. This allows the system design to be directly implemented in the same way it was designed (separate tiers).

3.1. Design Goals

Since the users of the system are Students of the university, Course Coordinators, and School Administrators, the system will be subjected to high traffic volumes.

The aim of the system is to cater to academic tasks and therefore should serve its purpose in a professional and efficient manner.

By analysing the requirements, the system has been designed to favour speed and usability over other areas.

3.2. Domain Model

3.2.1. Informal Description

It is required to design and create a web-based system, which serves the primary purpose of allowing students to view university course marks. The system must be easy to use

and navigate.

The following functionality must exist:

- ⋆ Students can login, view their uploaded marks, and query a mark.

- ⋆ Course coordinator can add/remove students to/from courses, upload marks, modify marks and allocate plagiarism and other offences against a student.

- ⋆ School administrator can perform the same tasks as course coordinator (from an access point of view). Their main roles however concern comparative course mark results, and student offences.

3.3. System Use Cases

Throughout the system, various scenarios are present, and exist. System use cases represent these scenarios.

3.3.1. Final Use Case

Following on from the initial Software Requirement Specification (SRS V2.1), the final project Use Case Set is defined below. Each case is separated by business feature, and further subdivided into feature.

Core Use Cases:

1. View Marks

2. Query Marks

3. Upload Marks

4. Add Course

5. Add Assessment

6. Enrol Student

7. View Offences

Reporting Use Cases

1. View Comparative Reports

2. View Course Reports

Maintenance Use Cases

1. Maintain Student

2. Maintain Course Coordinator

3. Maintain School Administrator

The list of Actors which will ink to the Use Case Set are as follows:

⋆ Student

⋆ Course administrator

⋆ School administrator

3.3.2. Use Case Diagram
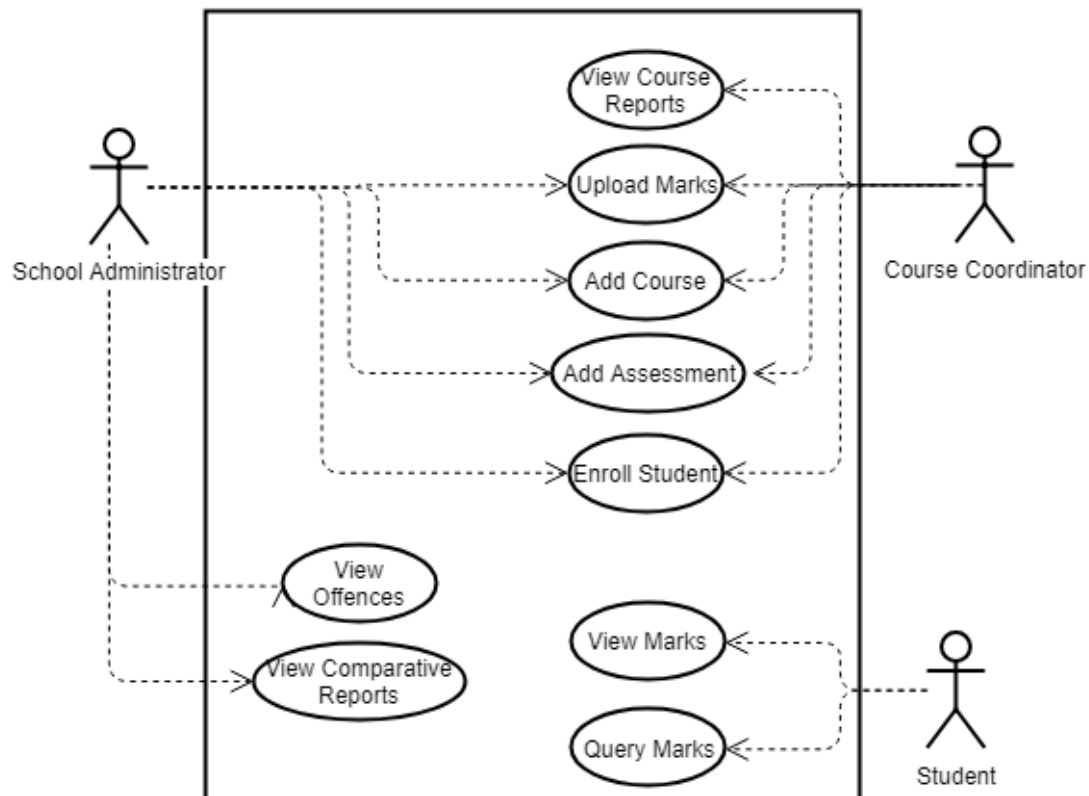
Use case diagram explanation:

Our system is comprised of 3 primary actors, as all actors are able to (in one way or another) make changes or directly interact with the system. The primary actors are student, course coordinator, and course administrator.

The student can invoke the view marks and query mark use case, respectively. This allows them to view the marks for all courses associated with their student number.

The course coordinator can invoke the View course report, Upload marks, Add course, Add assessment, and Enrol student use cases. This covers all administrative tasks involved with mark recording and student enrolment.

School administrator is able to invoke the same use cases as the course coordinator, as well as being able to invoke View offences, and View comparative report use case. Their primary role is concerned with student offences, however, due to administrative privileges they are able to invoke same use cases as course coordinator.

3.4. System Framework

The system has been developed using an ASP.NET 2.0 core, which is a server-side web-development platform. We have chosen to write the ASP.NET application in C#, and to use an MVC architecture, discussed later in the report. In the implementation of this ASP.NET application, we have made use of Entity Framework, an Object Relational Mapper which coordinates all interaction between the system and its database. We have adapted a code first approach, based on EF, which allows us to code first and then have EF create the database based on the code. Since we have used C#, we also used LINQ (Language Integrated Query) to save a retrieve data from the SQL database.

3.4.1. System Architecture Overview

Based on the requirement specification, group composition, time frame, and SCRUM Agile methodology chosen, the team has chosen the ASP.net Model View Controller (MVC) framework as the system architecture.

MVC allows for:

* Separation of a system into three smaller components being the Data Layer (model), Presentation Layer (UI or view), and Business Logic Layer (controller).

* Decoupling of modules enabling parallel development and separation of concerns.

* Simpler integration in terms of implementation and deployment.

* Usage of different languages (HTML,CSS,JavaScript,C#).

⋆ Test Driven Development.
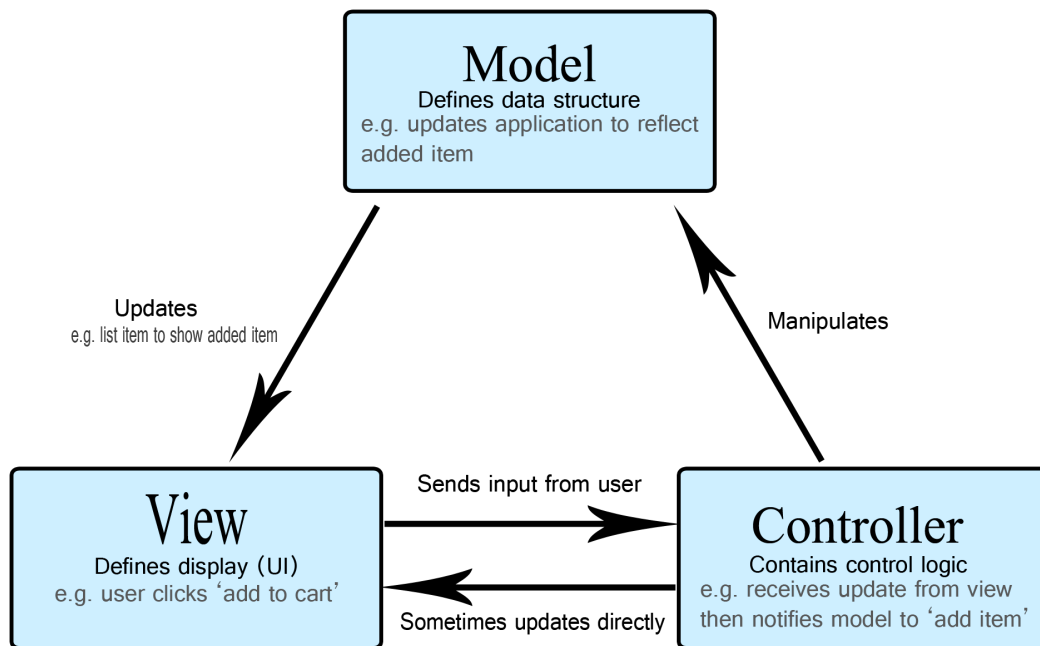
3.4.2. Detailed Architecture and Design

3.4.2.1. Presentation Layer

The presentation layer involves the interface of the system and forms part of the front end of the system. The main design goal is to design a usability focused, responsive, user friendly interface. The interface should be lightweight in order to meet performance requirements, however, it should still be aesthetically pleasing. The system should also contain a logical flow of events allowing for easy navigation.
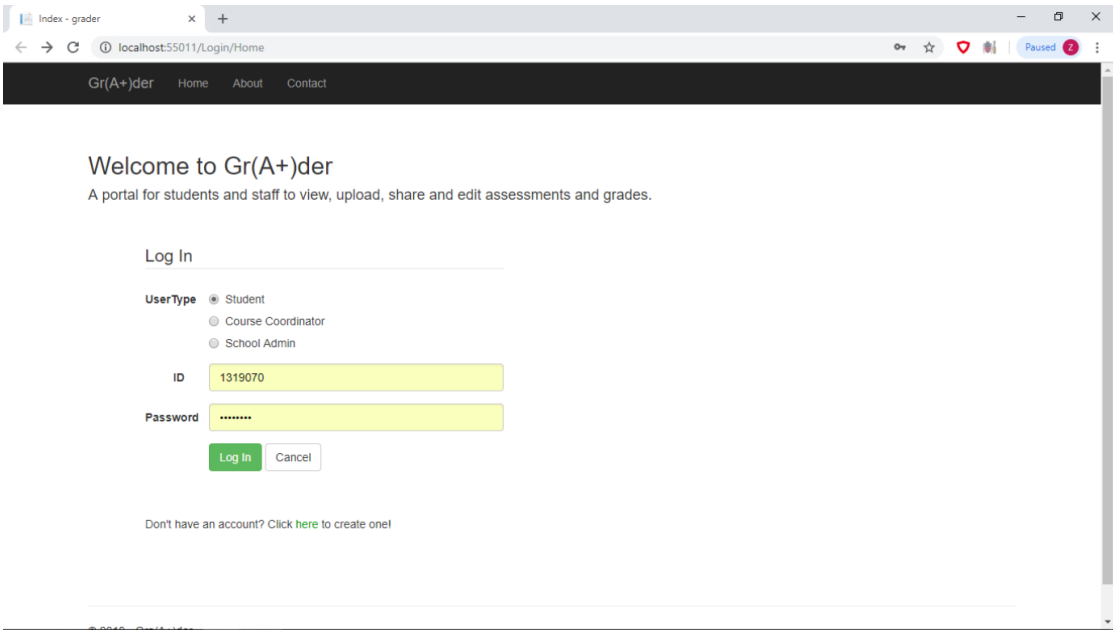
The interface has been designed for reuse as certain pages are used by both the Course Coordinator and School Administrator. This reuse allows for issues to be resolved faster and improves the speed in the addition of functionality in the future.

The team has used paper prototyping in order to achieve the design goals of the interface. The designs were then translated into view pages using HTML and CSS. The outcome of the presentation layer is depicted below.

1.The Login Screen

On this screen, access is only granted when a user enters their student number (ID login for administrator), and password associated with their account.

Then, depending on the type of login, once the system has verified the login credentials, the user is directed to the appropriate home page.

2.Home Course Page

The content displayed on this page depends on what user logged in. It is separated into:

- ⋆ Student
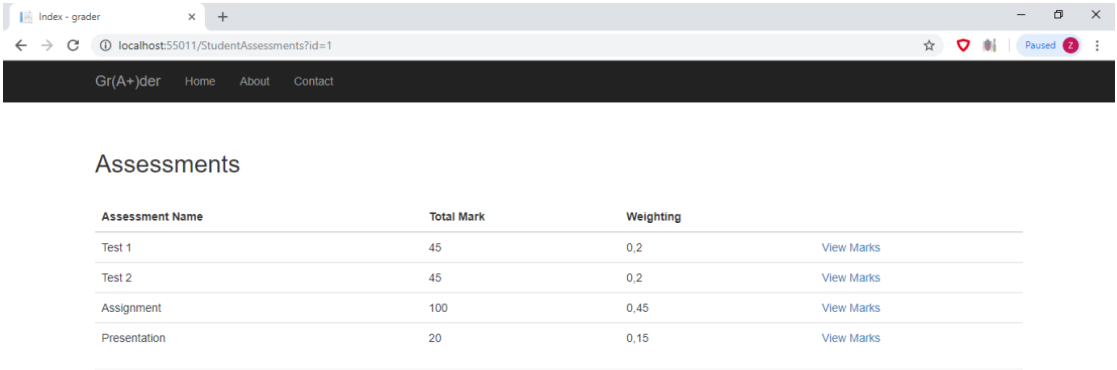
- ⋆ Course Administrator

- ⋆ School Administrator

Each 'home page' has features and components available specifically to the type of user accessing the system.

Student:

When the student is directed to the course home page, they are presented with content displaying the courses they are currently enrolled in (and thus able to access, view, and query).
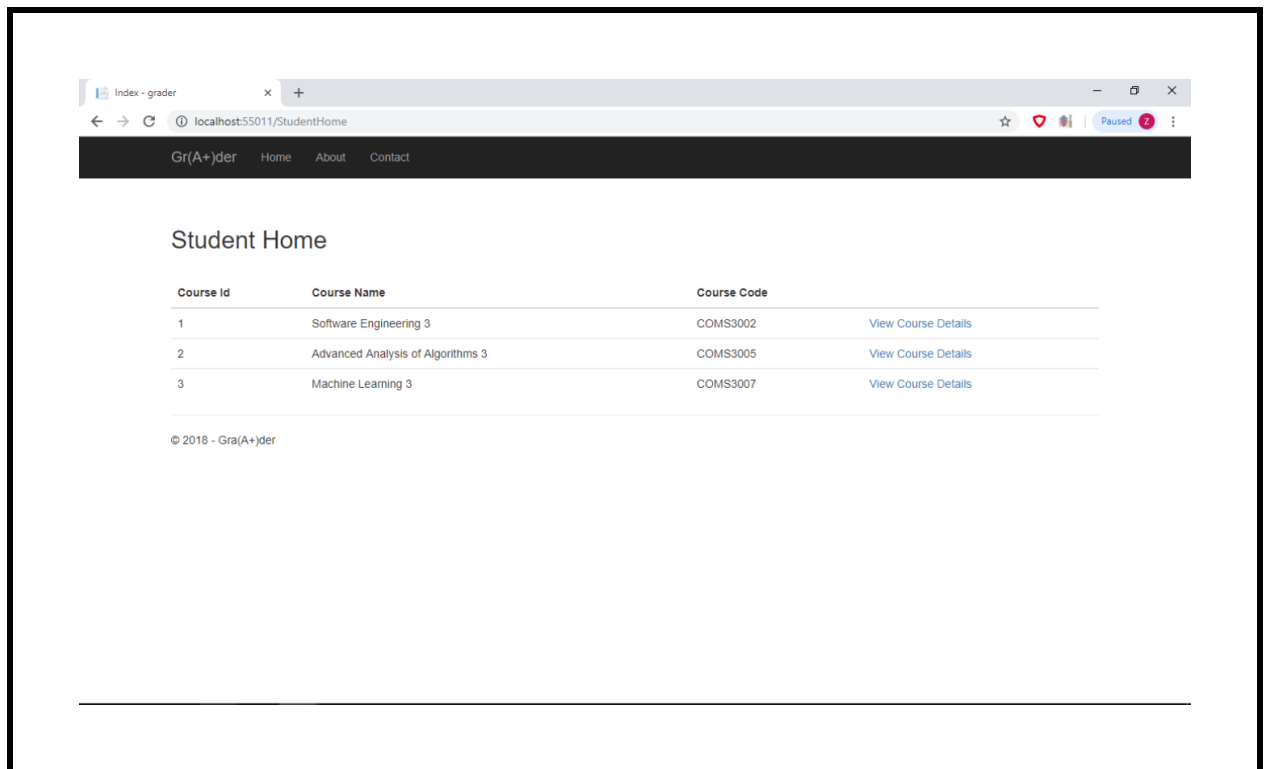
When the students selects "View Course Details", they are directed to an overview page, which shows a breakdown of all their assessments for that course, the mark the test was out of, and the weighting of the test (i.e. The contribution of that test to their overall mark. e.g. a weighting of '0.2' means the test counts 20% out of the available 100% for that course).
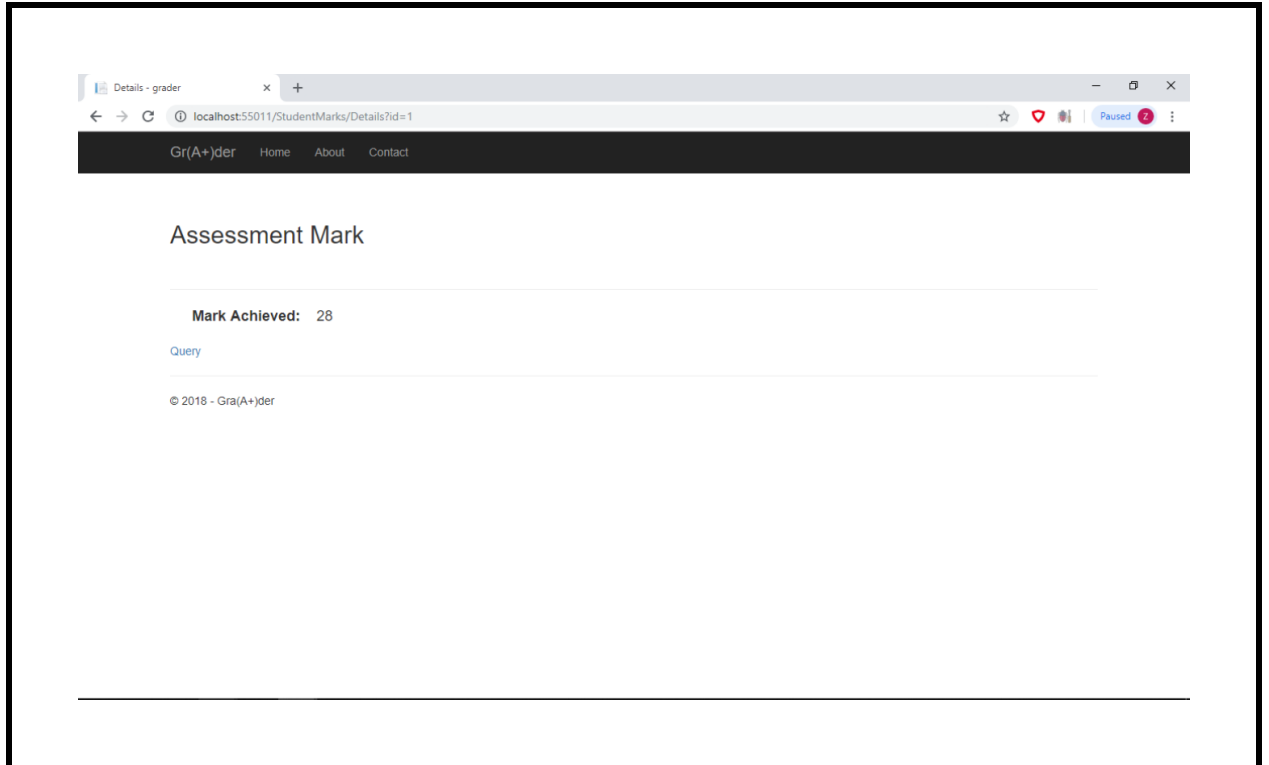
When the user (Student) selects "View Marks", they are then taken to a page which shows the mark they obtained for that particular assessment. On this page, they are able to make use of the "Query" function, discussed in Section 4.1.3.

Course Coordinator:

Once admin is granted access, their home page shows a list of current courses. On this page, they can either gain access to an existing course,by selecting "details", or add a new course, which they are then responsible for, or to view reports for their current courses. If they choose to add a new course, the page may need to be refreshed, before the new course is shown on the My Courses list.
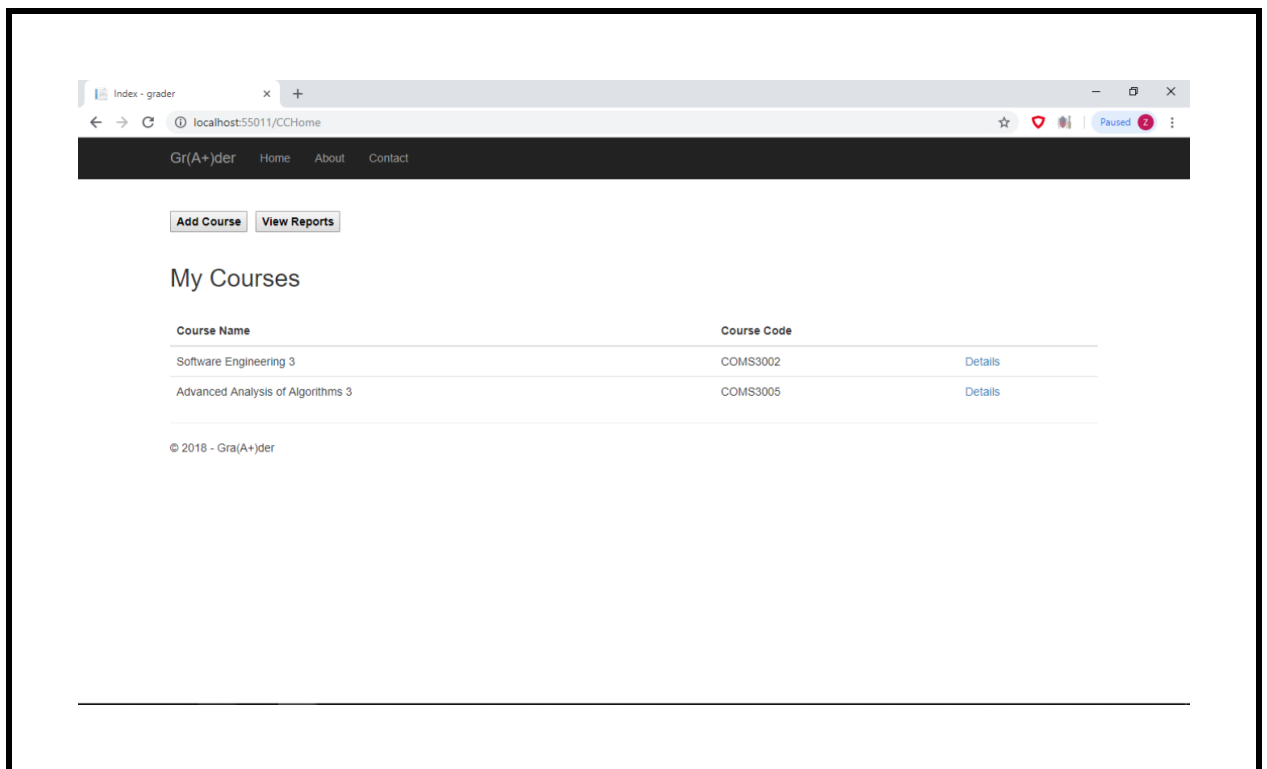
When the user selects "Add Course", they are then directed to the below page where they enter the details pertaining to the specific course.

From the My Courses" page, if the user selects "Details" on a specific course, they are brought to an "Assessments" overview page.

Here, they are able to:

⋆ Add an assessment

⋆ View Reports

⋆ Manage marks of a specific assessment
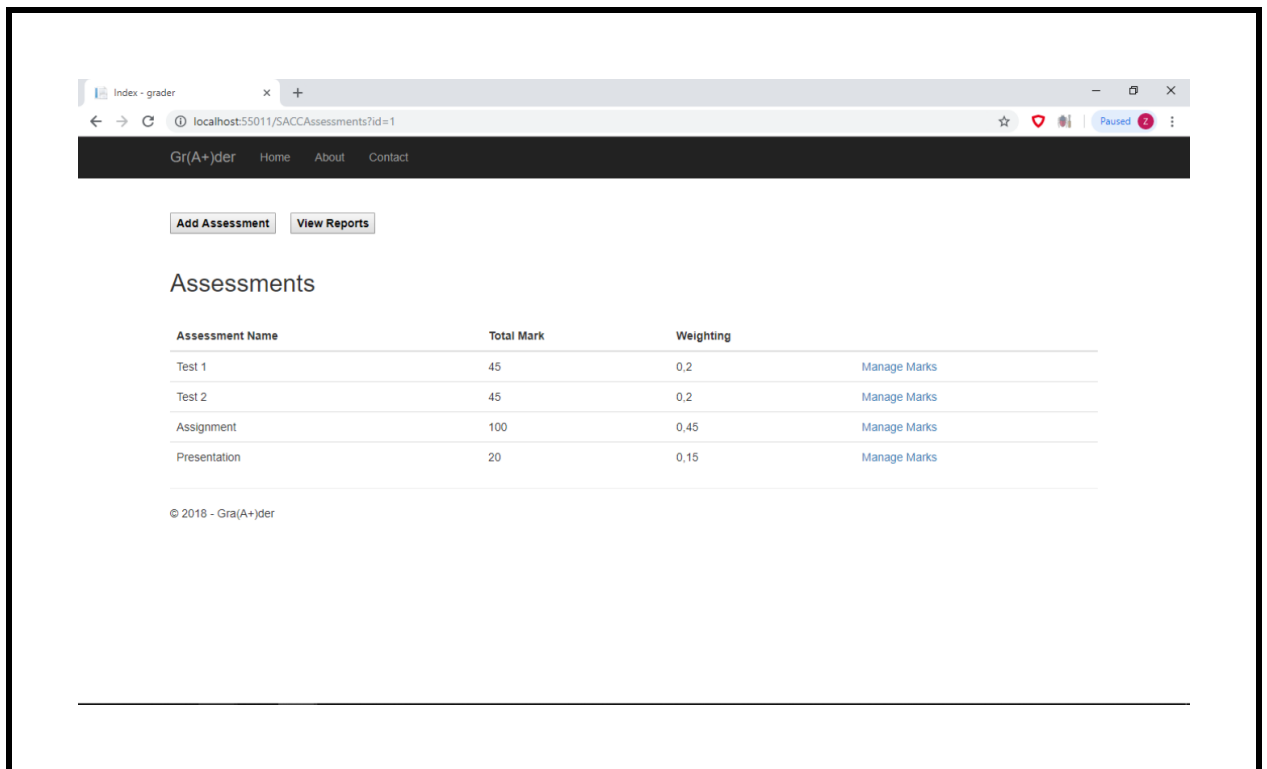
When selecting "View Reports", the user is directed to a page which displays a bar graph, showing the mean mark for each assessment recorded on the system, for the course they are currently viewing. This gives an easy to interpret, overview of the existing assessments.
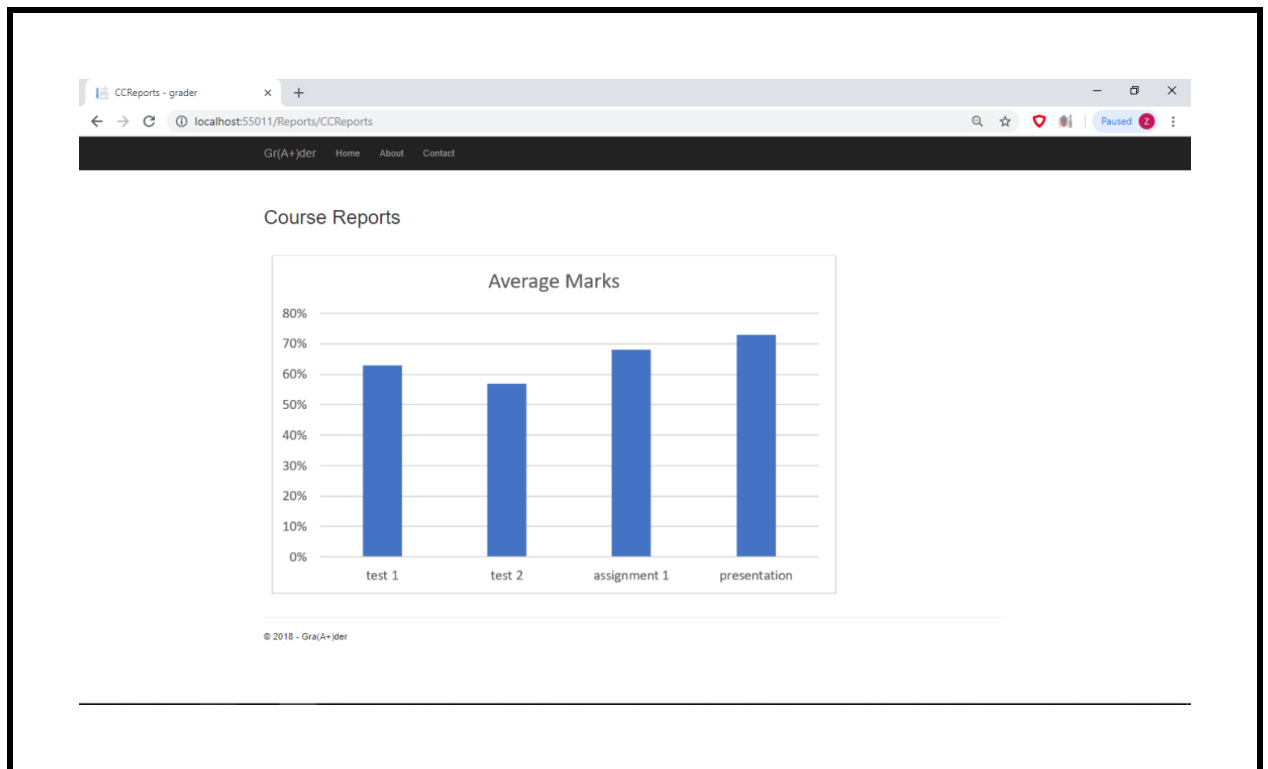
When selecting "Manage marks" from the "Assessments" page, the user is taken to a page where they are able to enter the mark achieved by all students enrolled in that course. The system will automatically generate the 'Assessment ID', which is the ID associated to that specific assessment.

When selecting "Save Marks", the entries/updates that have just been made are then saved to the database and will now be viewable to student users, as well as being included in the data used to generate the report graphs.

School Administrator:

On login,the School admin is taken to a page very similar to that of Course Coordinators' home page. The exceptions being that there is the option to "View Offences" (not available to Course Coordinator), and "View Reports" (Generating different data to that of Course Coordinator).
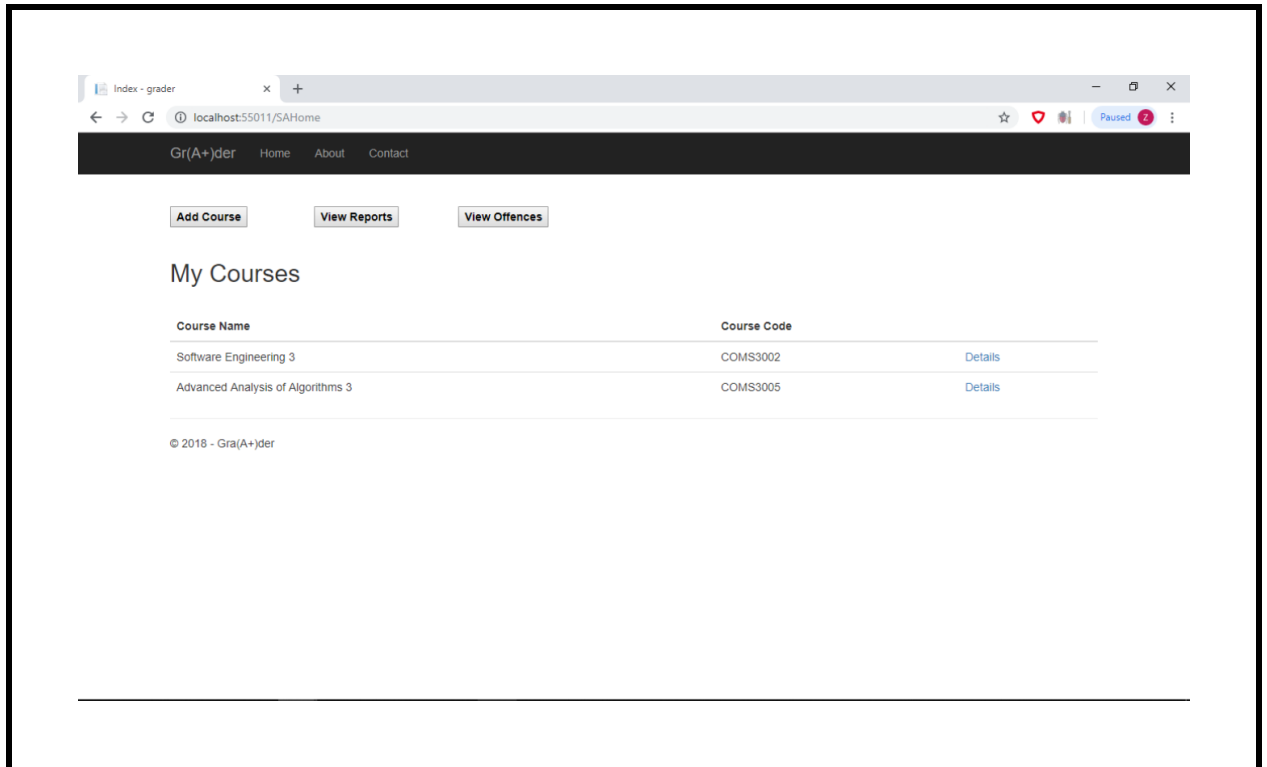
When "View Offences" is selected, the user is taken to a page which shows all recorded offences. It includes the reason for the offence, as well as options to remove (delete), edit, or view details associated with this offence.
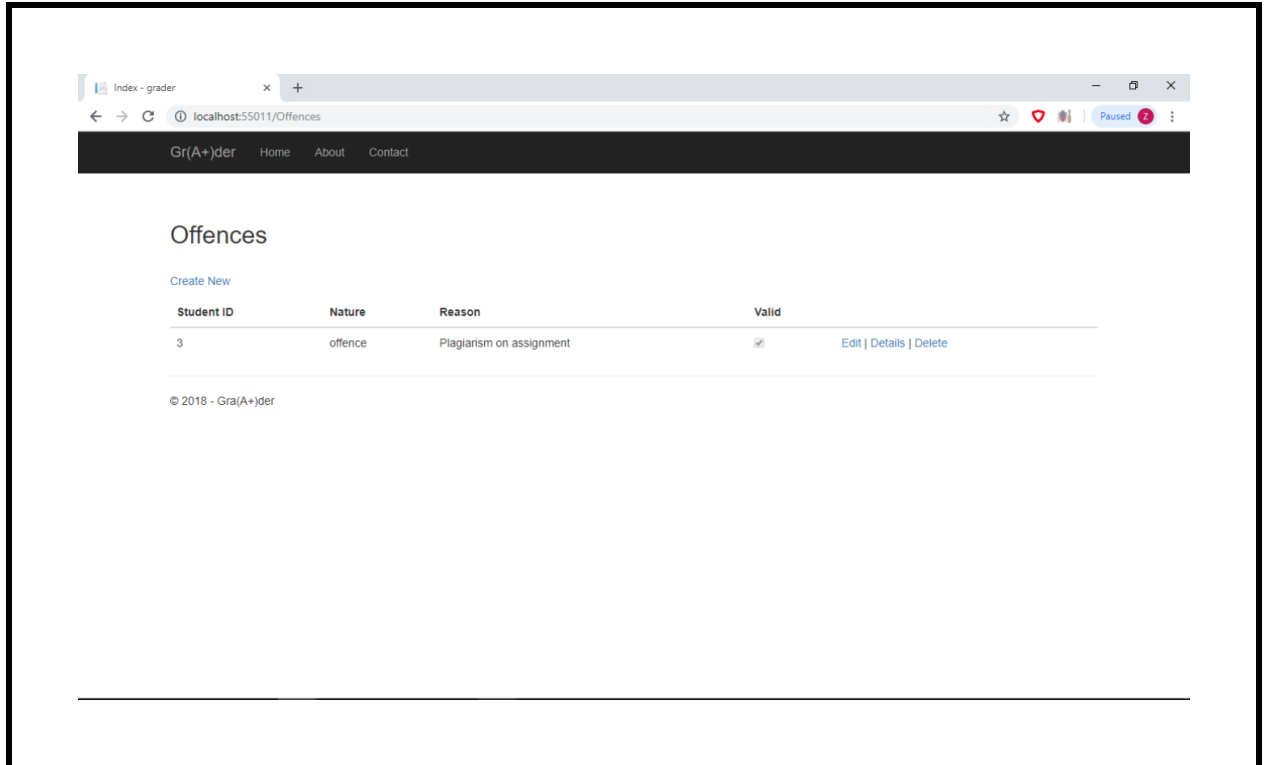
When school administrator selects "View Reports", they are shown a cumulative overview of all courses which they have access to. This means a line graph is shown, depicting the rise or fall in marks in the respective courses.

The aim of presenting the data in this manner, is to allow the SA to see how the course has progressed, as assessments were completed.
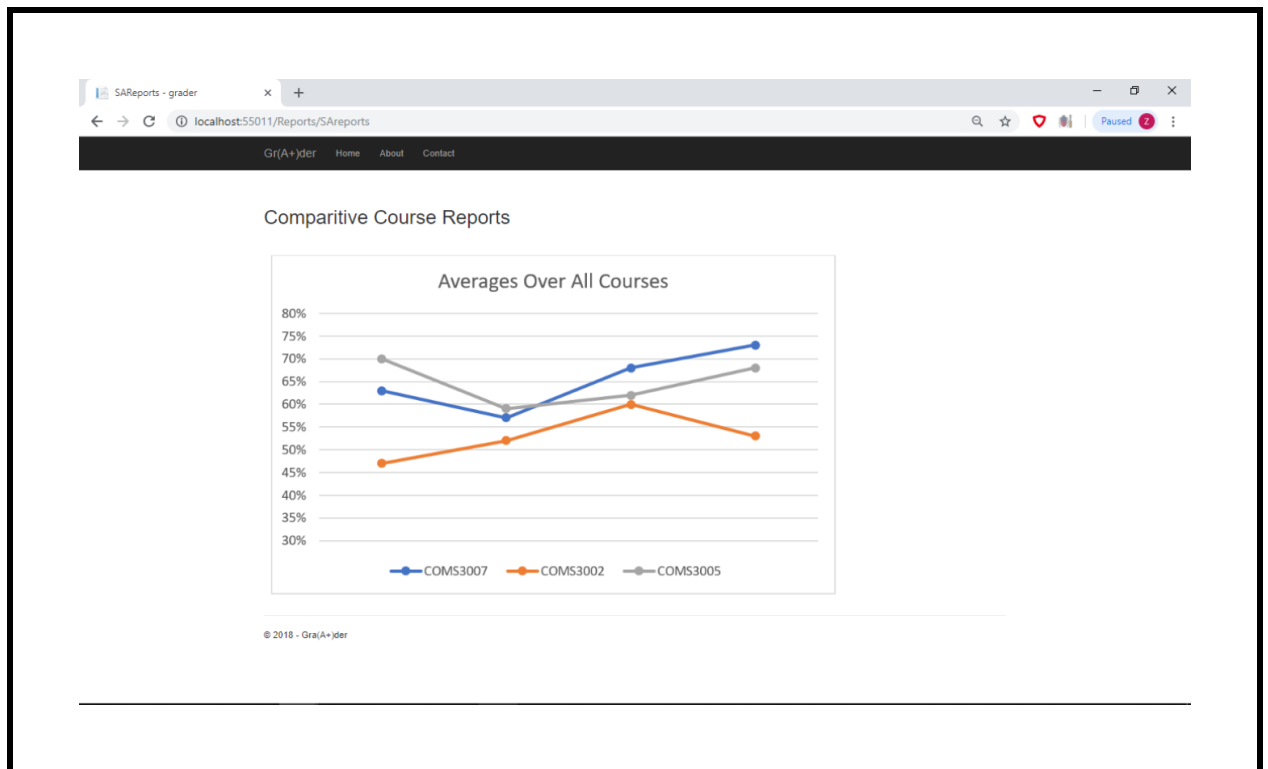
3.4.2.2. Business Logic Layer

The business logic layer is where the core system functionality is implemented. It serves as a go between for the other two layers (i.e. input is received from the presentation layer and is passed to the data layer, and the requested information is relayed back to the presentation layer).

The business logic layer has to be in exact accordance with the core use case set in order to deliver a valid system (a system that is in accordance with the problem statement).

The business logic layer mainly includes developing functions and procedures in the chosen programming language which is C# and uses Entity Framework for database interaction. Each methods purpose is based on the Use Cases defined in Section 3.2.

The business logic layer connects to the data layer by means of the Models connection with the Controllers which is used whenever communication with the data layer is required.

3.4.2.3. Data Layer

This Data Layer is the lowest level of the system and serves as the basis of the operations carried out and displayed in other layers. It is essential for core functionality and therefore has to be designed in a way where minimal errors can occur.

The design has to ensure data integrity, persistence, security, and manipulation. This will be achieved through the use of a relational data model using Structured Query Language (SQL) in order to manipulate data and define relationships (primary and foreign key relationships).

The database will be hosted on the university servers. This will allow the system to be deployed on the premises and will allow for easy transfer of necessary data from

other systems on campus. The data layer can also be scaled as the university increases capacity in the future.

The data layer interacts with the business logic layer using the Entity Framework to retrieve and manipulate any necessary data in the database. The design of the data layer, achieved through a Code First approach, allows this data to be pulled and used in an efficient manner.

After requirements gathering, the necessary attributes were extracted as well as their data types. The attributes, entities, functions/procedures, and relationships, pertaining to them are detailed in the Class Diagram below.

3.4.3. Class Diagram

The class diagram is an accurate description of the code that will be produced. The class diagram aids in identifying the relationships between the different classes which we implement in the system.
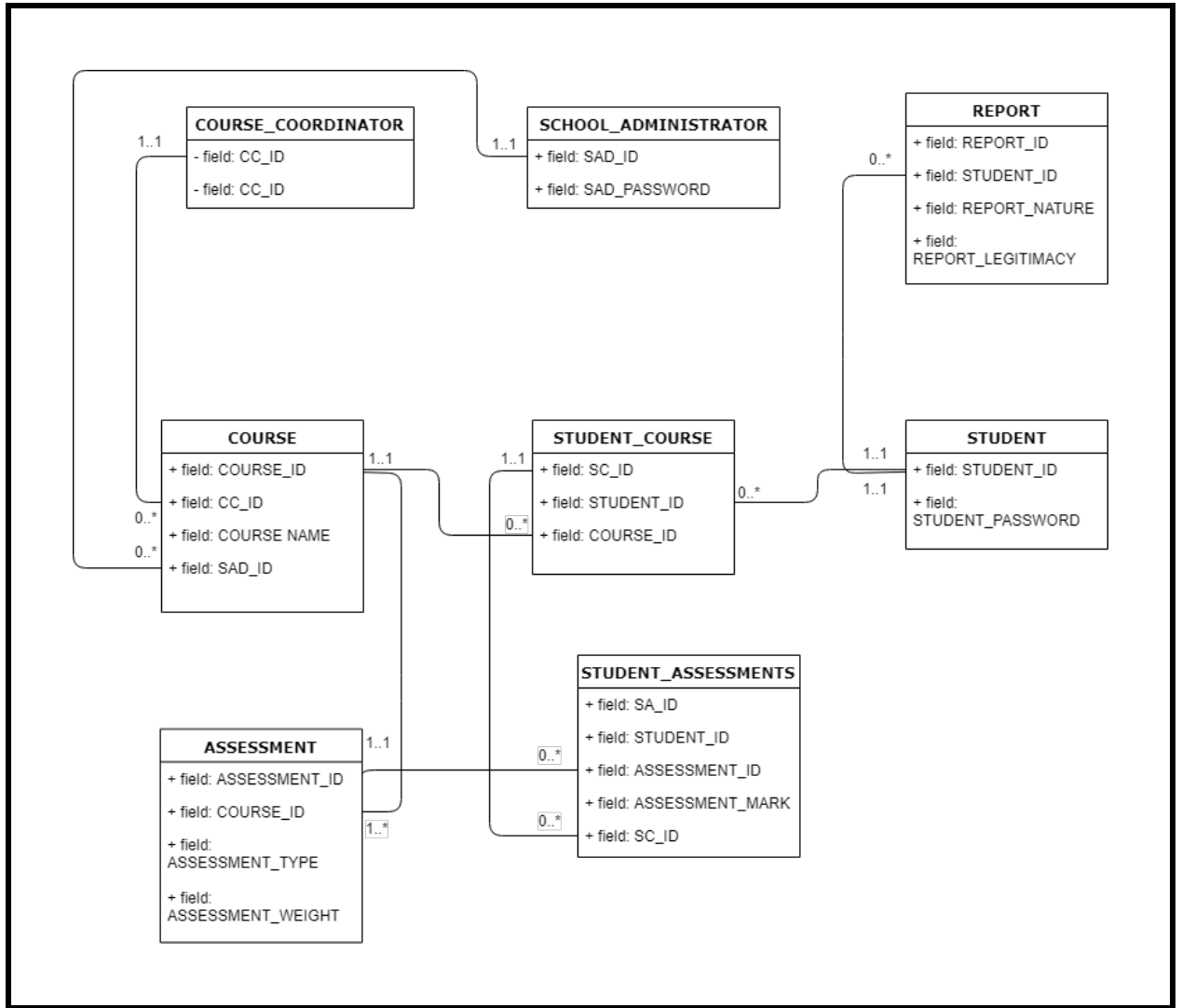
3.5. Pair 1: Front End Design

The front end pair are responsible for the design and implementation of the presentation layer. The focus of the front end is delivering an interface that provides a good user experience while also providing an aesthetically pleasing interface. This involves creating wireframes of the relevant screens. These screens are then turned into an actual interface design. Once the design is complete, it is checked to ensure that key system logic can be implemented so as to satisfy the functional and non-functional requirements relating to it. After the front end is completed, it is then integrated with the back end, described

below.

3.6. Pair 2: Back End Design

The back end pair are in charge of the business logic layer as well as the data layer. The use cases of the system must be translated into a class diagram. This will allow the relevant functions and procedures to be designed accordingly. Based on the class diagram the data layer can be designed by deciding what is required to be stored in the database as entities as well understanding the relationships between these entities (primary and foreign key relationships).

4. System Implementation

4.1. Presentation Layer

The presentation layer is implemented in a simplistic manner to ensure that each page is clear in function. This allows the user clarity on where he/she is and how to carry out the necessary functions (adding, viewing, changing and removing). The interface is implemented to maximise usability thus improving the user experience by allowing quick completion of tasks. While user experience is the focus, an aesthetically pleasing and minimalistic approach is used to create a good looking user interface. These components of the interface make the system seamless to use for all users. In order to achieve the above mentioned, it is required to add the necessary components (buttons, tables, graphs, scroll bars, etc) where they are required, but we must ensure we dont overwhelm the user, thus maintaining the minimalistic approach. As the system will be new, and users will not have experienced it before, we will take a build up approach with regards to the presentation layer functions. By this, we mean that in the initial system, we will ensure the core components are included, to allow basic functionality to be carried out, without any confusion. As we improve on the system, rectify any flaws, and update the system, we can then begin implementing new features in the system. By doing this approach, we allow the user to be comfortable with using the system, as well as allowing them to

identify what key features they believe would best benefit the system. By allowing the actual users of the system to be involved in the developmental process - by means of giving input towards system design and functionality, we allow the system to have the most relevant features, achieving a better user experience.

The front end developers followed the design of the mockups from the design phase and implemented them using HTML and CSS. The Bootstrap library along with Scaffolding (contained in the MVC framework) was used in depth in order to minimise the implementation.

The flow between pages has been done in a logical manner to make the system quick to learn for all users as soon as deployment occurs. This will allow us to achieve the above mentioned, so as to allow users to be confident when navigating and using the system. The interface implementation is further explained in sections below, divided into the different interface pages.

4.1.1. Login Page

The login page comprises of a simple form requiring an id, password and type of user (being either Student, Course Coordinator, or School Administrator). Once correct credentials are entered, the student will be directed to his/her course home page.

4.1.2. Courses (Student)

The course page for students will be their home page. On this form, they will be able to see an overview of all courses they are enrolled in, and will be provided the option to select an individual course. This will be achieved by having a link on each course allowing them to be selected. which will then take them to the next page, the assessment page. The user will also be able to log-out from this page.

4.1.3. Assessments (Student)

The assessment page will show all marks for all types of tests (tutorial tests, class semester tests, exams, lab exams, etc) that are associated to this user, for the course which they selected on the previous page. There will not be the option to select a test, as this serves no purpose. From this page, the only options provided to the user are to return to the course home page, or to select a specific assessment mark and be able to query it. This is made possible by having a Query button associated to each assessment, where the user will simply click on the button to submit query to the course coordinator.

The user will also be able to log-out from this page.

4.1.4. Courses (Course Coordinator/School Administrator)

The course coordinator and school administrator will both be redirected to the same home screen, upon successful login. This has been done both for simplicity, and due to the fact they have a very similar set of administrative privileges, as defined in the use case diagram.

The easiest way to separate and restrict access to certain functions, specifically with regard to the student offences, is to simply disable the functionality of the respective buttons, for that user.

On their home page, there will be buttons and/or drop down bars to allow the user to:

⋆ Select an existing course

⋆ Add a new course

⋆ Access Plagiarism offences

If they create a new course, they will need to enter the course details, including: course name, description, etc. Once this is done, the user may need to refresh the page in order for the new course to be added to the list of existing courses.

For the School Administrator, they will have the above option, but this is not their primary role or function, and so the 2 primary functions that are available to them will be:

1. To select a course and be able to view statistics on it. I.e Be able to view average course marks, student comparative marks for a particular assessment, and other useful statistical results which we deem are necessary to present to the user. (This data and results shown can be improved/changed on the next iteration of the system, if it is deemed necessary by the School administrator).

2. View offences against students. The most user-friendly way to do this, is to have a list of all students who have a/an offence(s) against their name, and then be able to view that particular student and see the nature, and number of offence(s) against them.

4.1.5. Assessments (Course Coordinator/School Administrator)

The user will be guided to this page after selecting a specific course from the above mentioned courses page.

When the user selects an existing course, they will then be directed to a page which shows the overview of that course, including: course description, number of students enrolled. There will be a number of options on this page, namely:

⋆ Add/remove students to/from the course.

⋆ Create a new assessment.

⋆ Upload marks for a specific assessment.

⋆ Edit existing assessment marks.

⋆ View Course report.

$\star$ View student mark queries (specific to School administrator).

The user can then select one of the above options to implement an update to the selected, existing course data.

The user is also able to logout from this page.

4.1.6. Reports (Course Coordinator/School Administrator)

The user is directed to this page when selecting the view course report option on the assessment page (described above).
This page shows read-only information, as it is a direct output of the data (marks) currently stored in the database, and applying to that specific course.
The type of statistical information that will be shown here includes: The user is directed to this page when selecting the view course report option on the assessment page (described above).
This page shows read-only information, as it is a direct output of the data (marks) currently stored in the database, and applying to that specific course.

The type of statistical information that will be shown here includes:

$\star$ Number of students enrolled in the course.

$\star$ Number (percentage) of students passing the course - obtaining over 50%

$\star$ Individual assessment information (including average mark for the assessment, % of students that failed, % of students who achieved distinctions).

In order to give a good, summarised overview of a particular course or assessment, graphing will allow an easily interpreted, overview of that. Thus, it will be available as part of the report page.

4.1.7. Offenses & Reports (School Administrator Specific)

This page is accessible from the course home page, when logged in as an administrator. The purpose of this page is to be able to view problematic students. The page will provide an overview of all students who have plagiarism offenses recorded against them. This is a most useful approach rather than selecting students with offenses from within a specific course, as the focus here is the student and offenses, not the course. As described in the core use case section, the 3 available queries apply to the student, not course specific.

4.2. Business Logic Layer (BLL)

The business logic layer functions as the middle-man, connecting and allowing data to flow between the presentation layer and the data layer. From the presentation layer, when a user selects a specific course, that information is sent via the Business Logic Layer, to the data layer, requesting the data pertaining to that course to then be sent back to the presentation layer, where it is manipulated and used to display useful information. Similar requests and processes exist for all the functions that need to be executed in the presentation layer.

The Business Logic Layer concerns the automation of processes which need to execute between the presentation and data layer.

In the implementation phase of this layer the System Use Cases are transferred into C# functions and procedures.

4.3. Data Layer

In its most simplistic form, the data layer concerns how the database is connected to the business logic layer. For our system, we created a database using Entity Framework with a Code First Approach.

4.4. Dependencies

For the purpose of efficiency and accuracy, wherever possible, we will make use of already existing libraries in order to design, create, and implement our system. The dependencies include:

* ⋆ ASP.NET 2.0.

* ⋆ Bootstrap.

* ⋆ Entity Framework .

4.5. Testing

In order to ensure our system worked and functioned correctly, we implemented a variety of test case situations. Most of these test cases were developed and tried once the system was completed. The reason it was done this way, was that we identified improvements to implement from our design phase to the implementation phase, and so in order to make sure we had successfully achieved this, we used the test cases.

The test cases allowed us to see that the correct output was displayed to the user for a number of given inputs.

The below describes some of the test cases we made use of to ensure the system had optimal functionality:

Student:

- ⋆ When selecting a course from the home page, the user is redirected to a page displaying all marks for that specific course.

- ⋆ When viewing a specific mark, the user is able to click query, and this sends a notification to the course admin indicating a potential error on this specific mark.

Admin (Course & School):

- ⋆ On completion of adding students to a course, once refreshed, that course shows the updated student enrolment list.

General:

- ⋆ When logging in, and selecting type of user, we ensured that access was only granted when there was a triple-authentication match. This meant that username and password matched, and then that these login credentials matched the user-type that was indicated on the login page (student/course-coordinator/school-administrator).

Black box and white box testing is used to test back end and front end features of the system respectively.

4.5.1. Server-Side

Verification was done to check that various components of the system worked correctly such as the database connectivity, and data retrieval of courses and assessment marks. Unit tests were conducted to ensure that, given certain parameter values, various methods return the correct data. These were first traced by hand and then the unit tests were used to assert whether these and the output of the functions do match.
The output of the relevant courses/marks was also monitored by similarly. Both of the above tests successfully completed and verified that the server-side works correctly.

4.5.2. Interface

Unit tests are used to test each of the interface views. This is done by entering known, test-specific login credentials for each type of user on the login screen and observing the resulting interface transition. This ensures a specific interface view and related components are created based on the type of user so that the user is given only the system features that they require access to. This is then tested by a user, who attempts to use the interface to navigate between various screens, and ensures no unanticipated behaviour occurs The above tests are completed successfully and determine that the interface works correctly according to the design.

5. SCRUM Milestones

5.1. Project Sprint Planning

5.1.1. Team

The team was split up into the following: Project owner, SCRUM master, and the developmental team, which was further divided into front and back-end roles.

Project Owner: As this task forms part of an overall course assessment, the lecturer - Professor Ekow Otoo, is assumed to be the project owner. Instruction and guidance as to the correct design, implementation, and report of the project was given, and additional support was provided by means of the project brief, consultations and emails.

Scrum Master: This contributor was simultaneously our Team Lead. The role was fulfilled by Zakiya Safi. This role included ensuring sprints were completed on time.

Development Team: The project was taken on by four members in total. Thus, the team was equally divided into 2 front-end developers, namely: Zakiya and Ali. The back-end was developed by Zubair and Kyle.

5.1.2. Sprints

It was decided to break the project down into 4 distinct sprint runs. The aim was to allocate an equal time to each sprint run but this could vary according to product complexity, risk assessment, and degree of oversight desired within each sprint run. Following the completion date of each sprint, a short period of time is allocated to a sprint break. During this time, a review of the completed sprint will take place. This involves examining the success of the sprint run, and improving any areas of concern, before moving on to the next sprint run.

Sprint 1: Scheduled to begin on August 6th, and be completed by the 20th of August.
Sprint 1 rest: 21st of August - 22nd of August.

Sprint 2: Scheduled to begin on August the 23rd, and be completed by the 6th of September.
Sprint 2 rest: 7th of September - 8th of September.

Sprint 3: Scheduled to begin on September 9th, and be completed by the 23rd of

September.

Sprint 3 rest: 24th of September - 25th of September.

Sprint 4: Scheduled to begin on September 26th, and be completed by the 4th of October.

Sprint 4 rest: 5th of October - 6th of October.

5.1.3. Stand Ups

Stand up meetings were held for 15 minutes on the first day of every sprint and every Wednesday. Due to timetable constraints and subject to availability, meetings on other days were done through messages and emails to keep everyone updated on the project.

5.1.4. Detailed Sprint Execution

5.1.4.1. Sprint 1

This was our initial sprint. During it, we focused on tasks that would allow the best setup of the project. The aim here was to ensure that we considered all requirement aspects of the project, allowing us to prioritise the order in which we created the system, and went about the development.

This included: This was our initial sprint. During it, we focused on tasks that would allow the best set-up of the project. The aim here was to ensure that we considered all requirement aspects of the project, allowing us to prioritise the order in which we created the system, and went about the development.

This included:

- ⋆ The initial design definition: This process allowed us to identify who was going to be responsible for what parts of the system development (workload allocation), according to our areas of expertise and knowledge.

- ⋆ We identified the group lead.

- ⋆ We set in place when we would have meetings (in order to ensure we kept on track, and that all members were up-to-date with the system development progress).

- ⋆ Defining our input and output requirements of the project (based on our SRS document).

- ⋆ Identified what the key features we needed to incorporate into the system were.

- ⋆ Formulated the project name.

- ⋆ Set-up our resource network, including our GitHub repository, and creating a Google Drive folder, enabling us to collaborate easily, and efficiently.

5.1.4.2. Sprint 2

The second sprint mainly concerned requirement aspects of the system. This involved further developing our ideas defined in Sprint 1, and then to formally define developmental aspects of the project.

The process involved:

- ⋆ Developing the software requirement definition.

- ⋆ Formalising our project description. This allowed us to have a detailed plan of what our system needed to do, so that we had a model on which we would define and create our system.

- ⋆ Define the project assumptions. Doing this enabled us to know what requirements we needed to consider, which would be assumed (i.e. users roles and ability to use the system), and what features we would omit (based on requirements). We incorporated prior knowledge, as well as research to make informed decisions.

- ⋆ Defining the Project Scope.

★ Formulating Use Cases. This allowed us to identify how the users (actors) would interact with the system.

★ Defining the Database: This included designing the database, identifying necessary tables, developing keys, describing rules for database implementation (NULLS, removing entries, etc.).

5.1.4.3. Sprint 3

The penultimate sprint focused on the split between front-end and back-end tasks. This sprint thus required the team to be split into the front and back-end roles that were defined in Sprint 1, and to work according to this.

In this sprint, we took what had been previously defined and introduced in the last Sprint, and further developed these system components. This meant we took the theoretical development, and created practical results from it during this sprint.

Front-end:

★ **Interface design:** by this point we had already defined how the system should work and flow, and so we not created the actual interface based on this.

Back-end:

★ **Database:** We had, by this point, identified the relational entities that needed to be defined, as well as how the various tables and structures needed to interact with each other. We therefore created the actual database system at this point, adhering to the design we had earlier constructed.

5.1.4.4. Sprint 4

The final sprint involved integrating the now existing front and back-ends of the system, as well as finalising and completing the report documentation.

System: As mentioned, at this point, we had completed the initial front-end and back-end system components, and so we combined them to achieve the first instance of our working system. This required incorporating the requirements we had previously specified and making sure that the system adhered to this, as well as the features we were required to include, were in the system, and worked.

Reporting: Throughout the design and development of the system, we were basing the system on our documentation. However, as requirements, time deadlines, and ability dictated, we updated and altered the report to incorporate the system which was actually developed. We kept track of changes during the process, and included these in the final documentation.

5.2. Project Sprint Retrospective

Sprint #	Start Date	End Date	Areas Of Success	Areas Of Concern	Improvements To Be Made
Sprint 1	6 Aug	20 Aug	Project definition done properly, roles allocated, team in agreement about approach. Repository set-up and methods in place to collaborate effectively.	Team worried about being able to achieve full working system in time constraint.	Adhere to the process that is defined. When concerned, approach group and resolve issues to maintain efficiency. This maintained cohesion.
Sprint 2	23 Aug	6 Sept	Expanding on project requirements done comprehensively. Project report updated to reflect current position in project.	Due to other commitments, sprint meetings could not be carried out, in person, every Wednesday as defined.	Allow the approach to be flexible to incorporate changes. This included using online meetings as legitimate, informative meetings.
Sprint 3	9 Sept	23 Sept	Front-end interface was designed as per specification. Back-end database was correctly defined. Communication worked well, despite not always meeting in person.	Concerns over whether the system would be integrated correctly, specifically concerning combining front and back-end systems.	Instead of conforming to front/back end split, if possible to help on an end requiring more work, do so.
Sprint 4	26 Sept	4 Oct	System integration worked, project report was comprehensive and completed.	System is not ready to be published. A few features still to be added before releasing.	Define a simpler model in SRS in order to achieve successful development. Rather add features as time permits.

6. Conclusion

The details regarding the design and implementation of a web application for a student marks system are documented above. The functional and non-functional requirements of the system have been determined and a design has then been formed. This design is expanded upon using UML models and is separated into three layers. It is implemented as a web application which, when tested, meets the necessary specification requirements. Lastly, a description is given of how the Agile SCRUM process has been followed by means of sprint planning and a sprint retrospective.

Appendices

A. Pair Responsibilites

A.1. Pair 1: Front End

The front end developers served the primary role of developing the visual aspects of the system. This involved all features and tasks in the presentation layer, and included coding, design and documenting their workings. An important role here was to create a visually appealing, yet simple to use and navigate system, to give the best user experience.

A.2. Pair 2: Back End

Our back-end development team had the task of developing the business logic layer, and ensuring that it fetched the correct data from the data layer, when requested. This included correctly designing the database system to prevent errors when data is requested. Tasks included designing, coding and reporting all development in this aspect.

A.3. General

Although tasks were allocated, and roles defined from the start, when it was required, we were able to be flexible and help each other wherever necessary. This meant that we were not limited to only front end/ back end roles. A positive from making use of this approach was that all members had a very good understanding of the overall system, and so when troubleshooting, we were able to more easily identify and rectify errors.