Team Report

Interpretations of the brief:

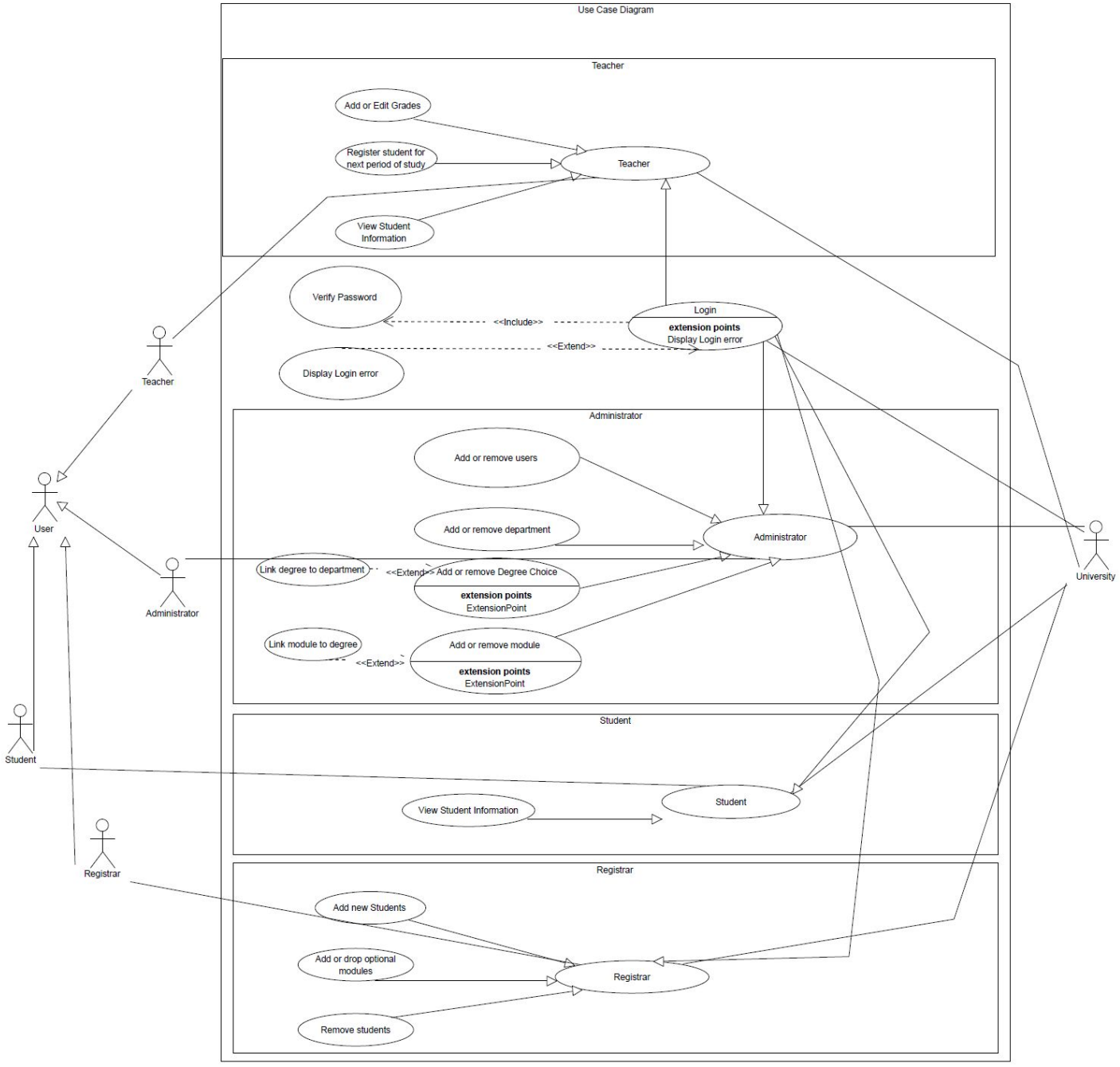
Assumptions made about the registration system include the creation of new students accounts by registrar. That includes input of both the information general for all users and specific to students only. We also assumed that there is no need for personalized screen for each registrar account. As follows all registrars open the same type of interface without their personal data visible. Third assumption is that registrar is not concerned about students who have failed for the second time or graduated. In summary registrar does the following: creates new students accounts, registers existing unregistered students by checking their module credits add up and changes their registration status to registered.

For the teacher system, we assumed that in the event of a student failing the year that they should automatically be registered to retake the year if they had not already failed that year previously. Furthermore, we assumed that a teacher should be able to edit the grades for any student taking any module as many times as they would like to, and also that the weighted mean grade should be calculated automatically. We also assumed that to avoid unnecessary confusion that the teacher should only be able to add grades for modules that a given student is currently registered. In regards to failing module we have assumed that in the event of a student failing one module badly (i.e. they do not achieve a conceded pass) that we should add their overall average to their record and not add increase their level or period of study until a resit mark has been added for them and they have passed the year overall. For placement years we have assumed that a student will be assigned a single module worth one-hundred and twenty credits which the teacher can then enter a single grade to signify whether or not the have passed their placement year and may progress to their final level of study.

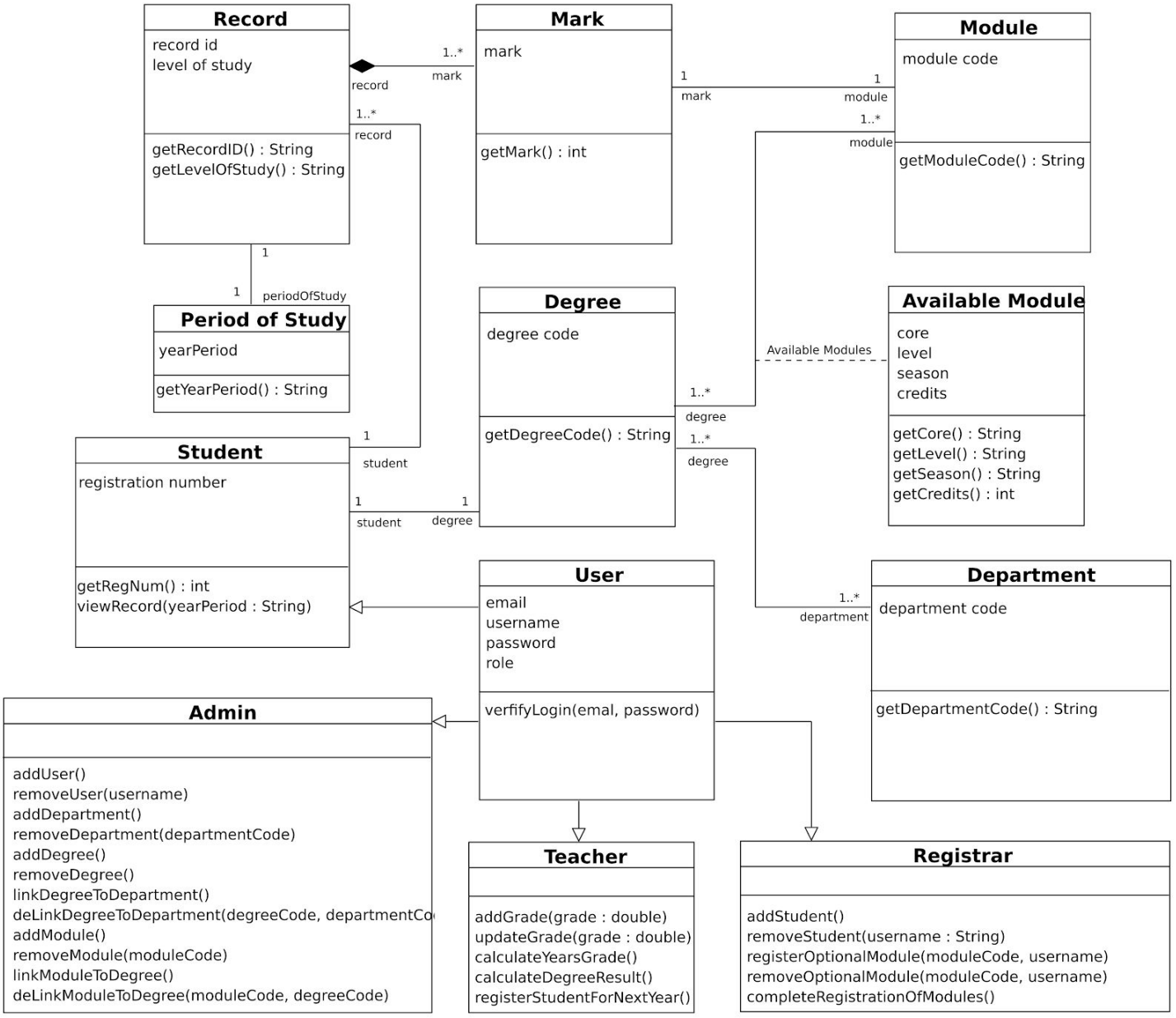
For the admin system we made the assumption that a year in industry is graded and thus treated as a module worth 120 credits, meaning a student can pass or fail their year in work. We also assume that a degree is linked with its lead department, so when a degree is created a entry in the department-degree linking table is also added with the lead department.

For the system as a whole we assumed that there needs to be a log off button on every page of the user interface as it is reasonable to assume that once you have logged in that you need to be able to log out.

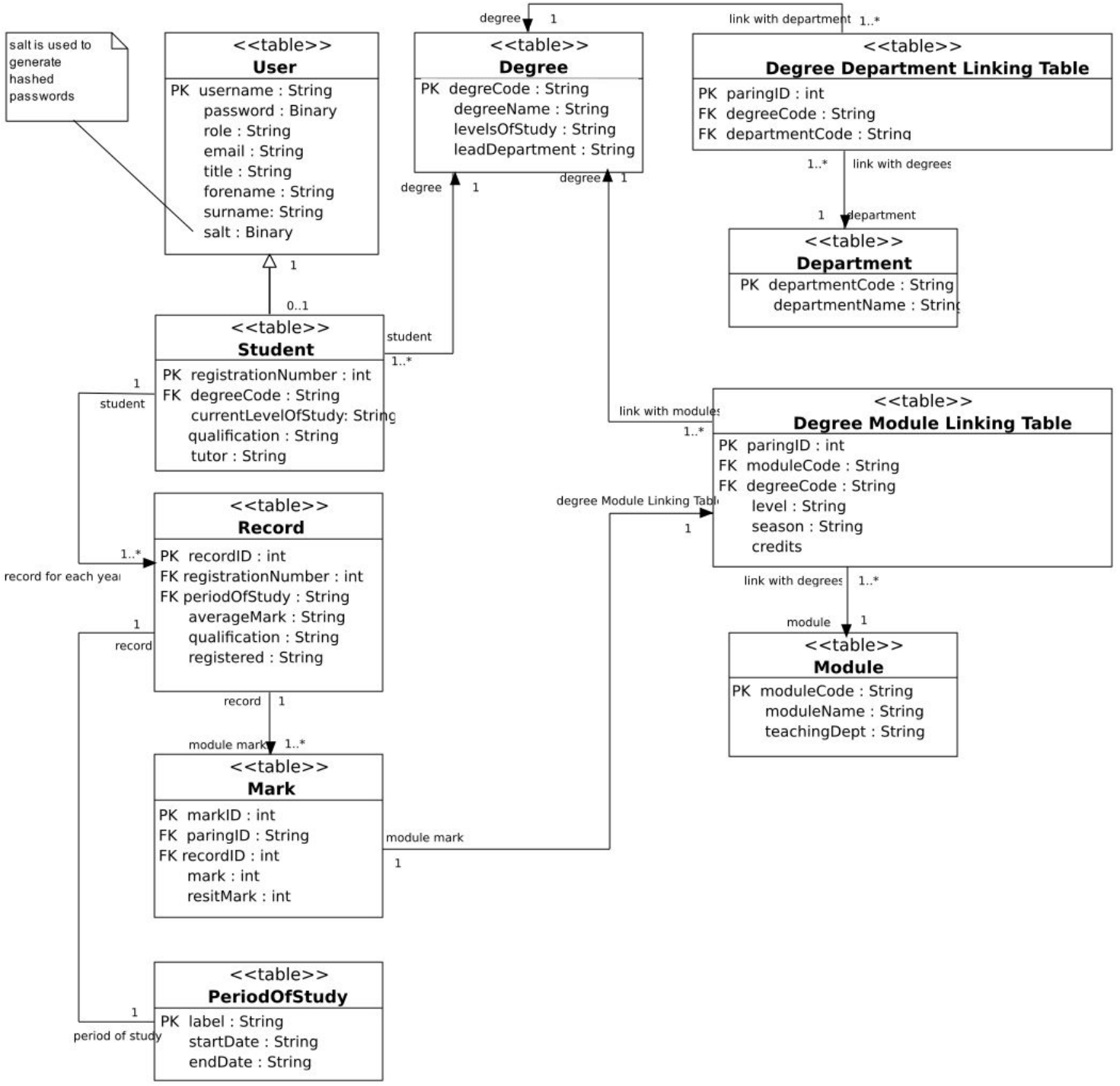
UML Use Case Diagrams:



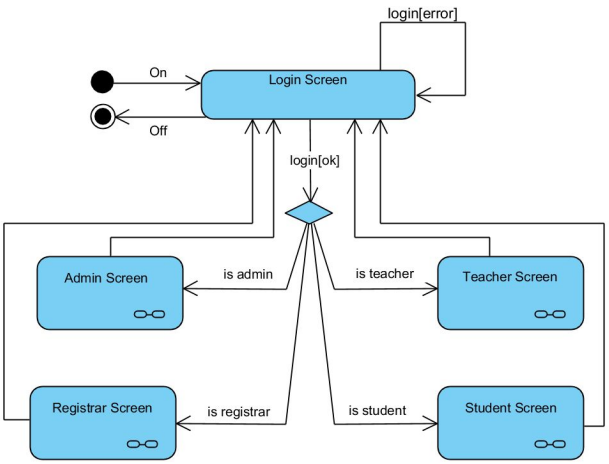
UML Class Diagram of Initial Information Model:

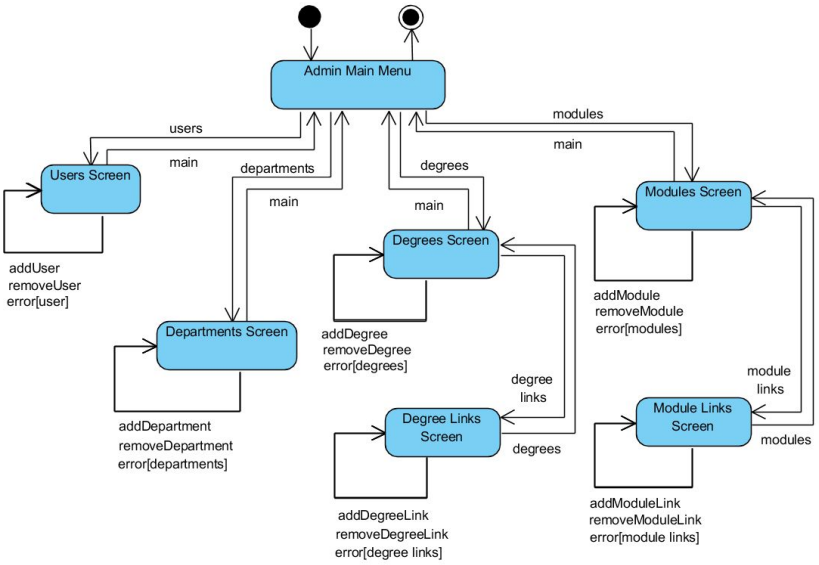


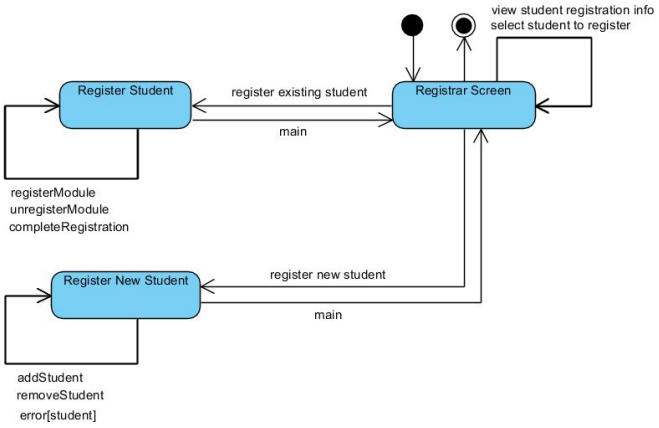
UML Class Diagram of Normalised Database Model:

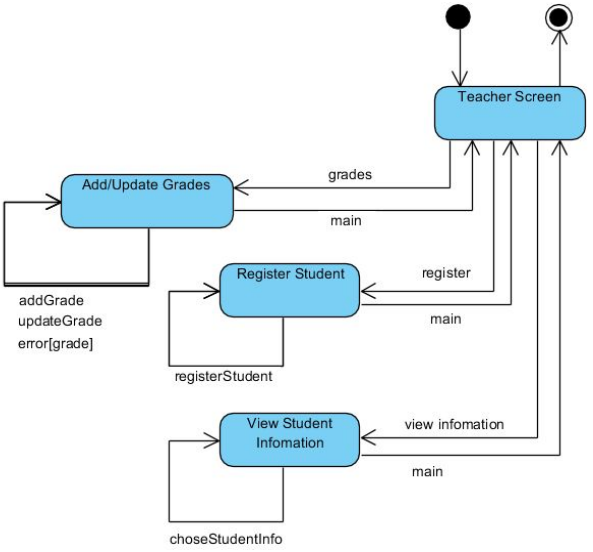


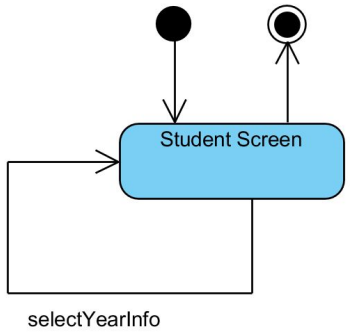
UML State Machine Diagram of the User Interface:



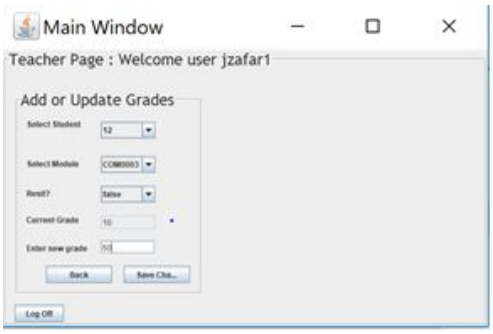


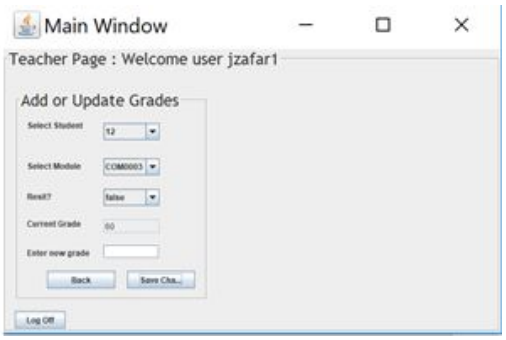




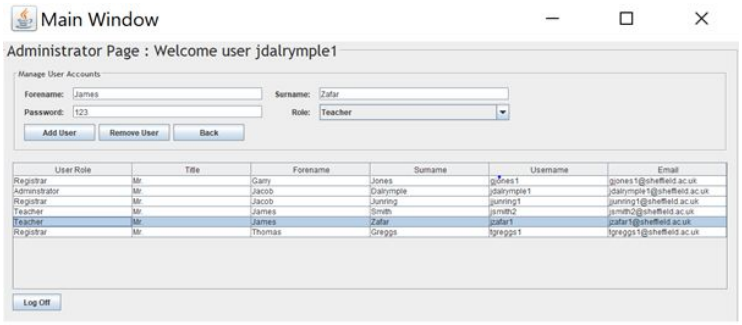


Screenshot of some System features:

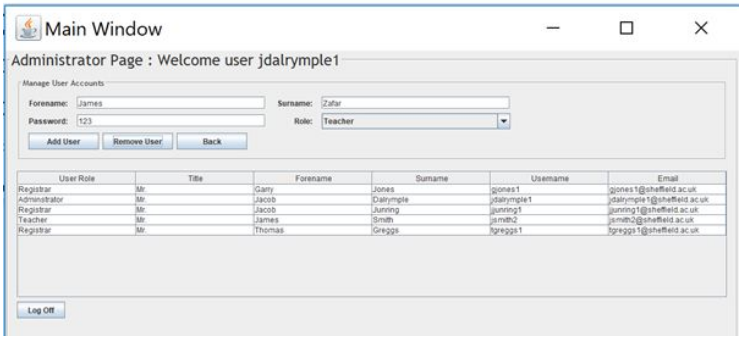




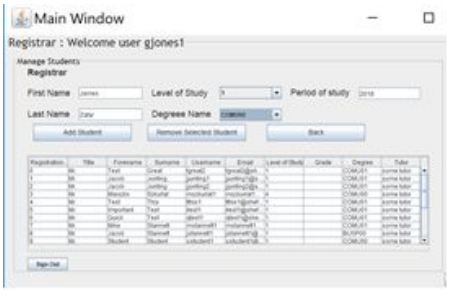
A screenshot showing a teacher adding/updating grades

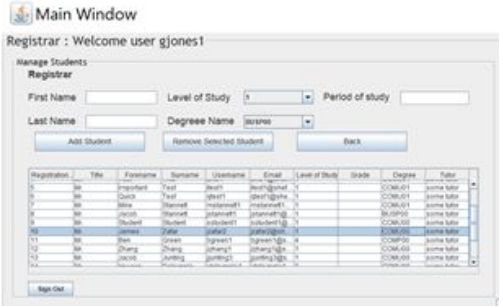


A screenshot showing an admin adding a new teacher with the role teacher

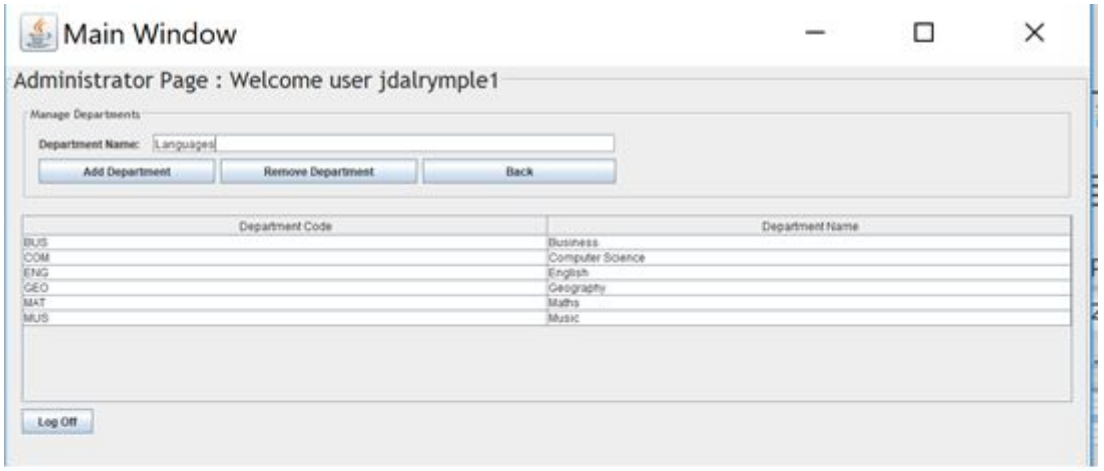


A screenshot showing an admin adding a new module

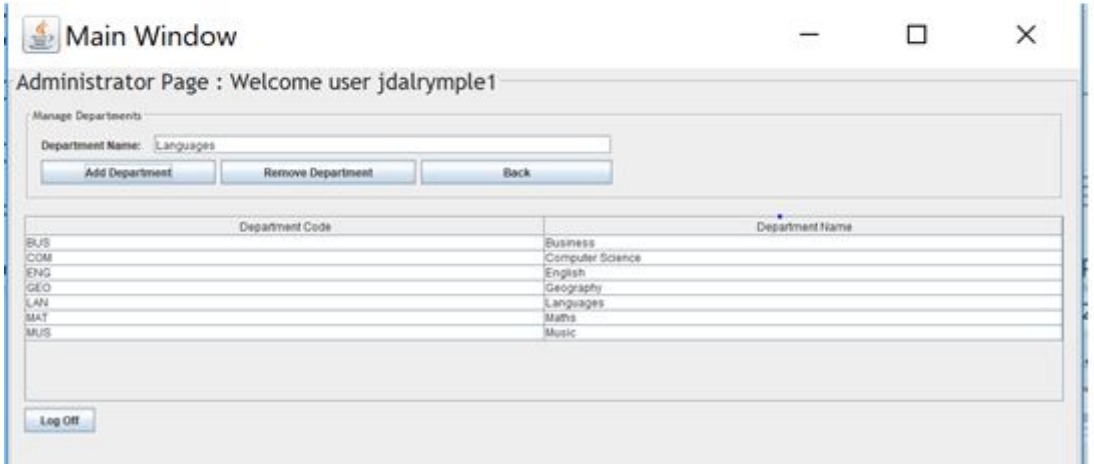


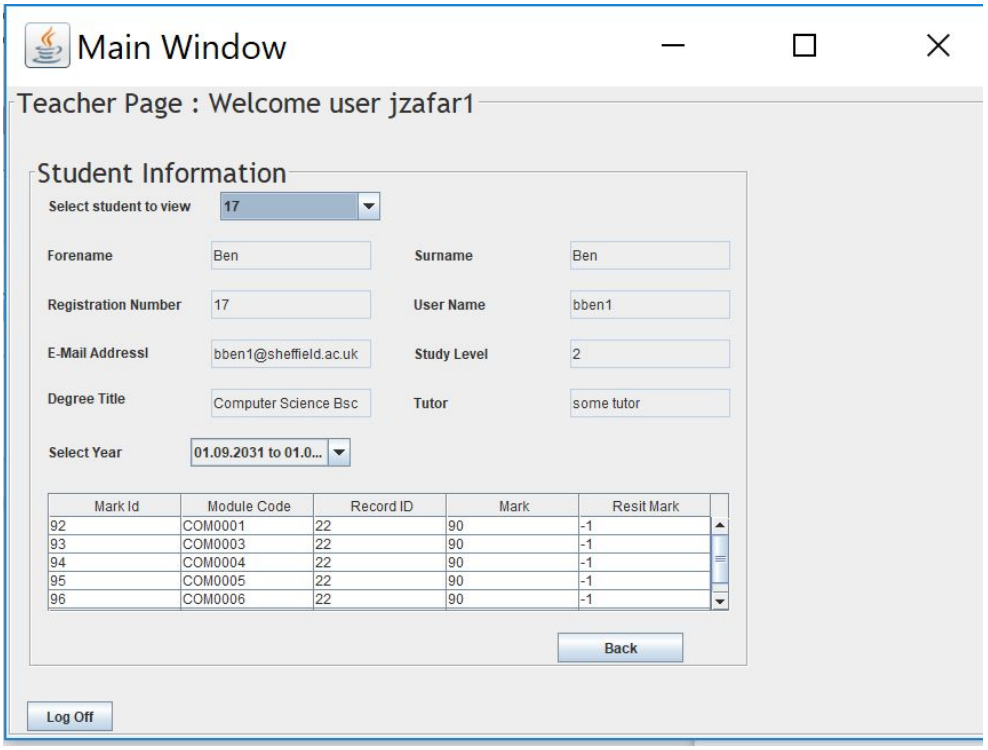


A screenshot showing a registrar adding a new student



A screenshot showing an admin adding a new department





A screenshot showing a teacher viewing the information of a given student

Discussion of Security Features:

We are using hashed passwords in the database. When a user enters their password, the system hashes their password with the supposed user's salt; if the hashed password is equal to the one in the database, the user is granted access. Salt is a unique byte array which is allocated to each user, by hashing their password with this unique array it prevents any dictionary attacks on our login system. Java security library was used to create methods responsible for hashing. The code for it can be found in PasswordHasher.java in src.sql.controller package.

There are three privileged roles in our system: admin, registrar and teacher. Of those three only admin can determine the role of other users. In order to prevent privilege escalation admin interface is password protected and not accessible to other users. In fact all user roles have their own unique interfaces with capabilities limited to taking actions specific for their roles.

To prevent SQL injection we are using input whitelisting based on ASCII values of characters within input Strings. There shouldn't be vulnerability to unicode characters based attacks but given the complexity we assumed this is out of scope of the project and didn't analyze it fully. Whitelisting of input is recommended by OWASP as one of main lines of defence against SQL Injection. The other one being proper structuring of SQL queries. We did structure our SQL statements either by using prepared statements or surrounding user input with single quotes and using as little user input as possible. Almost all statements will act only on tables directly specified in our code. The input whitelist consists of: capital letters, lowercase letters, digits, space, full stop. Methods used for input whitelisting can be found in SQLValidation.java in src.sql.controller package.

Distribution of Work:

Name	Description of Work Done	Effort Points out of a 100
Jacob Dalrymple	I have completed the Admin system with the controllers and SQL operations along with the GUI accompanying it. Completed the database operations for login along with hashing the passwords. Also linked the UI together with the login system. Also completed the backend for the student UI, with the controllers and related SQL operations along with some GUI elements. And implemented the opening of connections/statements/results in the database model. Implemented the displaying of the database in the UI through the databaseView class as well. Completed the UML class diagram for the normalised database.	30
James Zafar	I assisted with designing the teacher and registrar UI as well as creating the core controller and functionality for the teacher UI. I added the necessary validation for input fields within teacher. For the team report I assisted in writing the assumptions made for the general parts of the system and teacher, as well as creating the screenshots and the UML case diagram.	30
Mieszko Szkurlat	Registrar UI sql controller and logic. Parts of Admin UI sql controller and logic. Validation for registrar UI. Anti - SQL Injection input whitelisting reusable class. Whole database. Half of registrar and teacher interaction synching. Parts of admin database model file and database model file as well as whole registrar database model file. Security features description. Description of assumptions made for registrar.	30
Junting Zhang	Login UI and registrar UI interface ,UML class diagrams	9
Ben Greenhill	Parts of the student UI interface.	1