**Interpretations of requirements:**

* There are a variety of user types which should be able to log in to the system. We have additionally implemented logging out of the system so a different user can log in without having to exit the system.
* The system has four different types of user: Admin, Registrar, Teacher, and Student. The brief states that an admin can ‘add and remove user accounts’, and that a registrar can ‘add and remove students’. We have interpreted this as registrars being able to add and remove student accounts whilst an admin is able to add all types of account.
* The raw mark out of 100 for resit grades is stored, and only capped when calculating the mean weighted mean grade.
* The beginning of a student email address consists of a concatenation of the initial letter of their forename, their surname and a unique distinguishing integer – this will also be their unique username used to log in. We decided to here employ a system of usernames in the interest of user convenience, and also to utilise an otherwise unused identifier.
* A 4-year integrated masters degree will be considered undergraduate whereas a 1-year masters programme will be considered postgraduate.
* A student will only be able to select optional credits from a module list suitable to their level and period of study.
* The system supports up to 26 periods of study but it is assumed that a student won’t reach or exceed that number as failing twice prevents them from proceeding.
* Each student is assigned a personal tutor, but this isn’t connected to a teacher user account as we considered this outside the scope of the brief.
* The brief states that modules have a default number of credits based on their level, we have however assumed the admin would know the system defaults and will input the correct credit values for each module.
* Any module added to a degree course will not affect previous students on that course; for example, adding a different module to the first year of a degree course would not affect the first DegreeLevel of students already on the course. Any module changes for a level of a degree course must therefore occur prior to the assignment of new students.
* *Links between a module and the Grades table are preserved when dropping a module from a degree course, which allows for both the continued existence of previous grades and for the module to remain on other degree courses. Deleting modules, degree courses, or departments from the system does however cascade to the Grades table.*
* The brief states that “if [MSc students] fail a 4-year degree at level 4, then they must graduate with the equivalent bachelor’s degree with credits already obtained. From this we have interpreted that any credits obtained from a failed level 4 wouldn’t count towards the degree classification, even if they have already been obtained.
* 1-year MSc courses count as a conceded pass if a module other than the 60 credit dissertation is marginally failed. The pass mark is equivalent to the 4th year of a 4-year undergraduate degree (50%), despite the level being 1.
* The dates in the system are pre-set and can’t be edited.
* The brief states that a conceded pass is when the student has marginally failed in one module with a score no more than 10% below the pass-mark. We have interpreted this as 30% when the pass mark is 40% rather than taking 10% off the pass mark.
* For a placement year, a placement module has to be manually created and added.

**Security features:**

* We have used PBKDF2 for hashing passwords which aims to reduce the vulnerability of encrypted keys to brute force attacks. A salt is added to the password which reduces the ability of rainbow tables and means multiple passwords cannot be tested simultaneously.
* A minimum password length has been implemented to reduce the effectiveness of a brute force attack as longer passwords have more possible values therefore increasing the time taken for the brute force attack. To extend the security of the system, the next step would have been to make the password include a mix of cases, digits and symbols but we didn’t implement these.
* A password field has been used when a user logs into the system which obscures the password, ensuring log in details cannot be obtained by watching someone log in.
* Where possible in data input, we have used drop-down lists so the user can only choose from what is already in the database. For example, when creating a new degree, the lead department can be chosen from a drop-down list of existing departments as the system would break if they tried to link a degree to a department that didn’t exist. In some cases, text fields had to be used, for example, adding the name of a new department. If these need to be in a specific format they are verified, for example, department code needs to be three capital letters. The database has maximum lengths on a number of string fields, but if the input was longer this would be caught in a try, catch statement so the database wouldn’t break.
* The system manages authorisation through Role-Based Access Control (RBAC), where the four different types of users are the different roles. Admins, Registrars and Teachers can read and write to certain tables of the database based on their tasks. Students can only read from the database, and the data must be related to them otherwise a breach of privacy could occur.
* We have used prepared statements to prevent SQL injections as this allows the code to distinguish between the code and the input regardless of the data inputted (so the user can’t change the intent of a query).
* Our current system has no controls to prevent concurrency conflicts (such as dirty and phantom reads), to extend the system further we could’ve looked into adding controls such as timestamping, phantom and intent locks and deadlock detection.

**Team Member Contributions:**

|  |  |
| --- | --- |
| **Madeleine Austen** | - Created the initial state machine diagram  - Updated the use case diagram and initial information model  - Wrote the initial interpretations introduction and security features discussion  - Wrote the initial version of Swing  - Assisted with testing of the system  - Collaborated on general planning and project organisation |
| **Matthew Prestwich** | - Created the initial use case diagram  - Wrote the initial version of AdminFunctions - Wrote the initial version of RegistrarFunctions  - Wrote the initial version of TeacherFunctions - Collaborated on later edits to the above  - Co-created early version of information model diagram - Assisted with database creation and modification  - Edited interpretations section of final report  - Collaborated on general planning and project organisation |
| **Byron Slater** |  |
| **That’s it** | Don’t need this space. |

**25% allocation to each team member, as agreed collectively.**