University of Strathclyde Glasgow

Assignment of CS308 module

By

Group		30	
No.	**Student ID**	**Student Full Name**	**Signature (please insert here)**
1	202303592	Haroon Wahid	
2	202303075	Aljo Meledom	
3	202344514	Judin Alex	
4	202343827	Leo Ozturk	

5	202317631	Thomas Clarkson	

Declaration

We are the members, names are shown on the first page, of the group number (shown above) confirm that this report is entirely our own work unless stated and referenced otherwise.

Note: if you have taken or adopted certain piece of work or code from a given source or reference. Please mention this in the code as a comment together with the reference. Also, mention this in the section 7 of this report and list the reference in the reference list.

1. User requirements

Provide a list of the user requirements that explain the tasks associated with this software.

1.1. Students can sign up to the university portal.

1.2. Students can log in with their university credentials.

1.3. Students can update their passwords.

1.4. Students can view their enrolled course and modules together along with their marks.

1.5. Student can download a lecture note and a lab note for a given module in each week

1.6. Student can view their decision if it is available (Award, Resit, Withdraw)

1.7. Students can also view their marks progress in every module

1.8. Students can log out of their university portal dashboard

1.9. Lecturers can sign up to the university portal

1.10. Lecturer can log in with their university credentials

1.11. Lecturers can update their password

1.12. Lecturers can upload or update module materials each week including lecture notes and lab notes.

1.13. The lecturer can update the module information.

1.14. The lecturer can upload the teaching materials needed for the module.

1.15. The lecturer can view enrolled students in the modules they teach.

1.16. The lecturer can assign marks for every student's lab or exams.

1.17. Lecturers can log out of their university portal dashboard.

1.18. Managers can sign up for the university portal.

1.19. Managers can log in with university credentials.

1.20. Managers can update their passwords.

1.21. Managers can approve students and lecturers who have created an account.

1.22. Managers can manage other users' accounts (activate, deactivate, password reset).

1.23. Managers can add new courses or modules.

1.24. Managers can assign a module to a given lecturer.

1.25. Managers can enroll students on a given course.

1.26. **Managers can assign a module to a course so that this course will include this module.**

1.27. **Managers can Issue student decision (award, resit, withdraw).**

1.28. **Managers can update course information.**

1.29. **Managers can add business rules.**

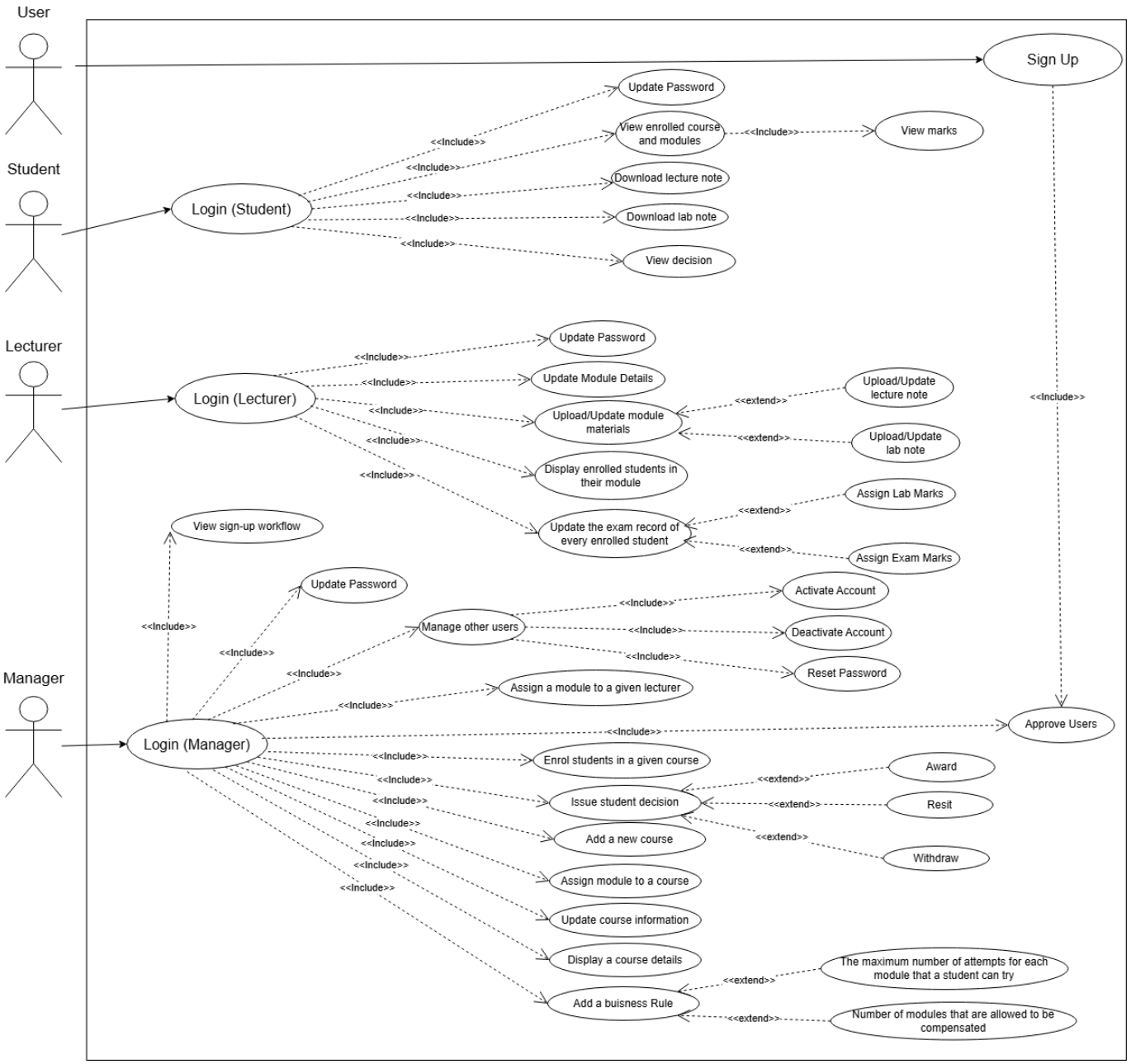
1.30. **Managers can display course details.**

1.31. **Managers can display module details.**

2. System Requirements

2.1. System use case diagram

Provide a use case diagram as a figure within the text and as a sperate image file called **SUCDrgm.tiff or any other format (such as pdf)** that can be zoomed in/out to display use cases and relationships.



2.2. System use case (SUC) details

Provide a tabular description (or narratives) to the following system use cases

- **Signup (for any user: lecture or student)**
- **Approve users**

- Validate login

- Enroll a student in a course

- Add a business rule

- Issue student decision

- Manage other user account

that explain these System Use Cases to developers. Each description has to describe one use case. For example, the following table describe a **login system use case:**

SUC name	**Login**
Actor	System user
Supporting actor	None
Pre-condition	User has an account
Trigger	User requests to login (e.g., by merely running the App)
Main Scenario	1. The system prompts the user to provide a username and password 2. The user inputs the required details and submits 3. The system invokes **validate login** use case and returns the result 4. If **validate login** returns True, system hides the login GUI and invokes **display Commands GUI** use case
Alternative scenario	**4.a validate login** use case returns (false) 4.a.1. System displays "invalid credentials" message, and increases attempt counter by 1 4.a.2. If the counter value <=3, system redisplays the login GUI. Otherwise, system displays "Too many login attempts" message and exits.
Post-condition	login timestamp is recorded

Table 1. The description of Login SUC

SUC name	Issue Student Decision
Actor	Manager
Supporting actor	None
Pre-condition	The student must be enrolled in this course, and the exam results must be available
Trigger	Manager selects the Issue Student Decision function
Main Scenario	1. The system displays a list of enrolled students and their exam results 2. The Manager selects a student to review 3. The system retrieves and displays the student's overall exam record 4. The Manager reviews the record and issues a decision (Award, Resit, Withdraw) 5. The system validates and saves the decision 6. The system notifies the student of the decision
Alternative scenario	**1.**Manager attempts to issue a decision without sufficient data on the exams. 2. The system displays a “Missing exam data” error message 3. The system prompts the Manager to review the exam marks before issuing a decision 4. Validation fails due to database error or invalid data 5. The system displays “Unable to save decision” error message and logs the error
Post-condition	The student’s decision (Award, Resit, Withdraw) is recorded in the database and a notification is sent to the student

Table 2. The description of Issue Student Decision SUC

SUC name	**Manage other user accoun**t
Actor	**Manager**
Supporting actor	**None**
Pre-condition	**The Manager is logged into the system with admin privileges**
Trigger	**Manager selects the "Manage Other Users" function from the admin dashboard**
Main Scenario	**1. The system displays the list of all registered users (students, lecturers, managers)** 2. The Manager selects a user account to manage 3. The system displays the user's account details and current status 4. The Manager chooses to activate, deactivate, or reset the user's password 5. The system executes the requested action and updates the account status accordingly 6. The system displays a confirmation message
Alternative scenario	1. The Manager tries to modify an account without the required permissions 2.The system displays "Access denied, insufficient privileges." 3. Account update fails due to a system error or connectivity issue 4. The system displays "Account update failed" and logs the error for review
Post-condition	The user's account status or credentials are updated and stored in the system. A log entry is created for auditing purposes

Table 3. The description of Manage other user account SUC

SUC name	Signup (for any user: Student or lecturer)
Actor	User (Student or Lecturer)
Supporting actor	System Database
Pre-condition	User has access to the signup page; system is operational and is linked to database
Trigger	User selects the 'Sign up" option from system's homepage or login page
Main Scenario	1. User clicks Sign up button 2. System displays registration page, and user enters the necessary details and submits the form 3. System validates the provided information and creates a new user account in the database 4. System sends confirmation email or success message and redirects user to login page
Alternative scenario	1. User clicks Sign up button 2. User enters invalid or incomplete details 3. System sends error message prompting corrections
Post-condition	User account is successfully created and stored in the database; user can log in using new details

Table 4. The description of Signup SUC

SUC name	Approve users
Actor	Manager
Supporting actor	System Database
Pre-condition	1. Student or lecture has completed the signup process and is awaiting approval 2. The manager is logged into the USMS with appropriate privileges 3. The user details are stored temporarily in the database with a 'pending' status
Trigger	The manager accesses the 'user management' or 'pending approvals' section in the USMS to review new user registration requests
Main Scenario	1. Manager logs into the USMS dashboard 2. Manager selects the "pending approvals' option 3. The system retrieves and displays a list of users awaiting approval including their role, department and course information 4. The manager selects the user and clicks 'Approve' and system updates user status to 'Approved' 5. System assigns appropriate permissions and sends automated email to the user confirming account approval
Alternative scenario	1. Manager rejects a user and system updates status to 'Rejected' 2. System error and displays an error message and logs event
Post-condition	The selected user status is changed to "approved", and students can now access their enrolled modules and view marks. Lecturers can access their teaching materials and manage module assessments. All approvals are logged for administrative tracking.

Table 5. The description of Approve users SUC

SUC name	Enroll a student in a course
Actor	Manager (primary actor who performs enrollments)
Supporting actor	Student (recipient of the outcome/notification)
Pre-condition	1 - Student has a created account and has been approved (managers approve sign-ups and manage accounts) 2 – Target course exists. 3 – Student is not already enrolled in a course. 4 – Enrollment period is open.
Trigger	Manager selects “Enroll students in a given course” from the manager at UI.
Main Scenario	1 – Manager opens an enrolment screen. 2 – Manager searches the student. 3 – Manager selects the course. 4 – System then validates student/account status and course availability. 5 – System checks that the student isn’t already enrolled. 6 – System creates the enrolment record linking student to the course. 7 – System associates the course module to the student and initializes the exam/mark records for those modules. 8 – System logs the action by manager. 9 – System confirms success and notifies the student.
Alternative scenario	1 – Student not found or not approved/active -> System shows error. Manager may redirect to approval flow. 2 – Course not found/inactive -> System shows error.
Post-condition	Students are enrolled in the selected course. Related modules are attached and ready for marks. Students will be able to view their enrolled course and marks in the system.

Table 6. The description of Enrol a student in a course SUC

SUC name	Add a business rule
Actor	Manager
Supporting actor	System Administrator
Pre-condition	The manager is logged in with permission to modify rules.
Trigger	Manager selects “Add Business Rule” from the system menu.
Main Scenario	1 – Manager opens the business rule interface. 2 – Manager enters rule details 3 – System validates the rule format. 4 – System saves the rules to the database. 5 – Confirmation message displayed.
Alternative scenario	-Invalid or conflicting rule -> system rejects input. -Missing data -> system prompts for completion. -Database error -> rule not saved.
Post-condition	New business rules are stored and become active for system validation and operations.

Table 7. The description of Add a business rule SUC

SUC name	Update Password
Actor	User (Student/lecturer/Manager)
Supporting actor	None
Pre-condition	User has an account
Trigger	User clicks the Update Password button
Main Scenario	1. System displays the Update Password GUI 2. User enters email, old password and then new password and confirms the new password. 3. User clicks Update Password 4. System checks that the old password is correct and the new passwords match 5. If both checks pass, system updates password and password updated successfully shown
Alternative scenario	4.1- Old password is wrong then system shows error message and then no change is made

	4.2- New passwords do not match then the system shows new passwords do not match and no change is made.
Post-condition	System returns user to the Login GUI

Table 8. The update Password SUC

3. Database Model

Provide a list of each entity in your E/R diagram and explain **very briefly** why this entity is required (reflect on the user requirements). For example,

3.1. **User**: is an entity to store users' details and accounts' credentials.

3.2. **Student:** extends user entity and stores information about students.

3.3. **Lecturer:** extends user entity and stores information about lecturers.

3.4. **Manager:** extends user entity and stores information about managers.

3.5. **Department:** required to store details of departments.

3.6. **Course:** required to store details of all the courses offered.

3.7. **Course_module:** links each course to its modules and their semesters

3.8. **Module:** stores details of individual modules.

3.9. **Module_attempt**: shows the attempt number of the student for a particular module.

3.10. **Module_assignment:** shows which lecturer teaches which modules.

3.11. **Semester:** required to organise the academic year and to distinguish between modules.

3.12. **Enrollment:** records which student is enrolled in which course.

3.13. **Module Attempt:** records each attempt a student has at a module and stores marks and attempt number.

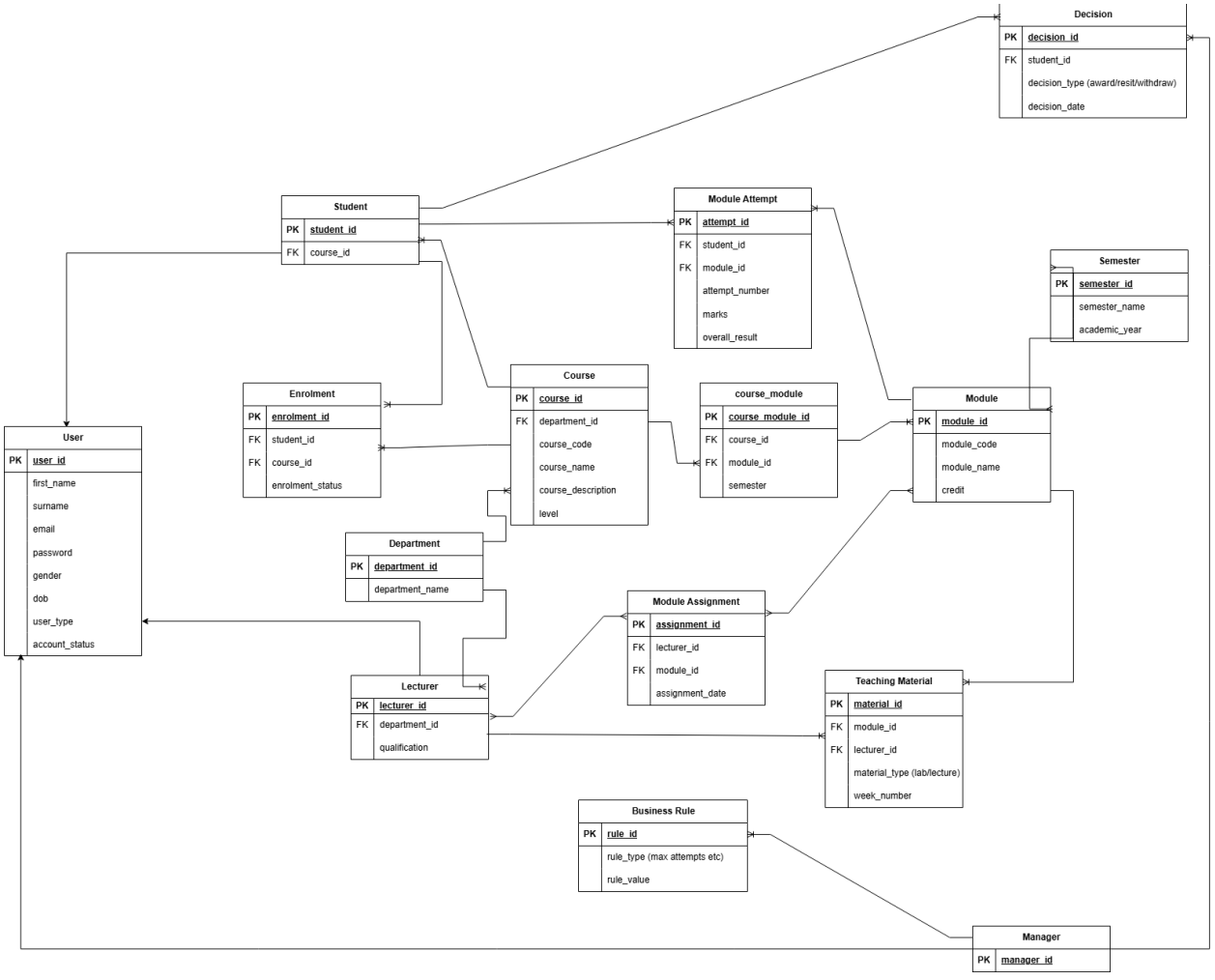
3.14. **Teaching Material:** required to store data about lecture and lab notes uploaded by lecturers for a particular module.

3.15. **Business Rule:** allows managers to make business rules.

3.16. **Decision:** handles the award/resit/withdraw decisions

3.17. **Module Assignment:** assigns a module to a given lecturer

Then provide the E/R diagram as a Figure within text and also as a separate image file called **ERD.tiff or any format (such as pdf)** that can be zoomed in/ out to view each field and its type.



When working in your database model, try to compose the SQL queries that you will use to achieve the user requirements stated in the first section. While you don't need to state your SQL queries in this section, these will be used in your implementation.

4. GUI Design

Provide figures as screen shots of all the GUIs that you have designed. Please provide a figure caption to explain what the figure shows. For example,

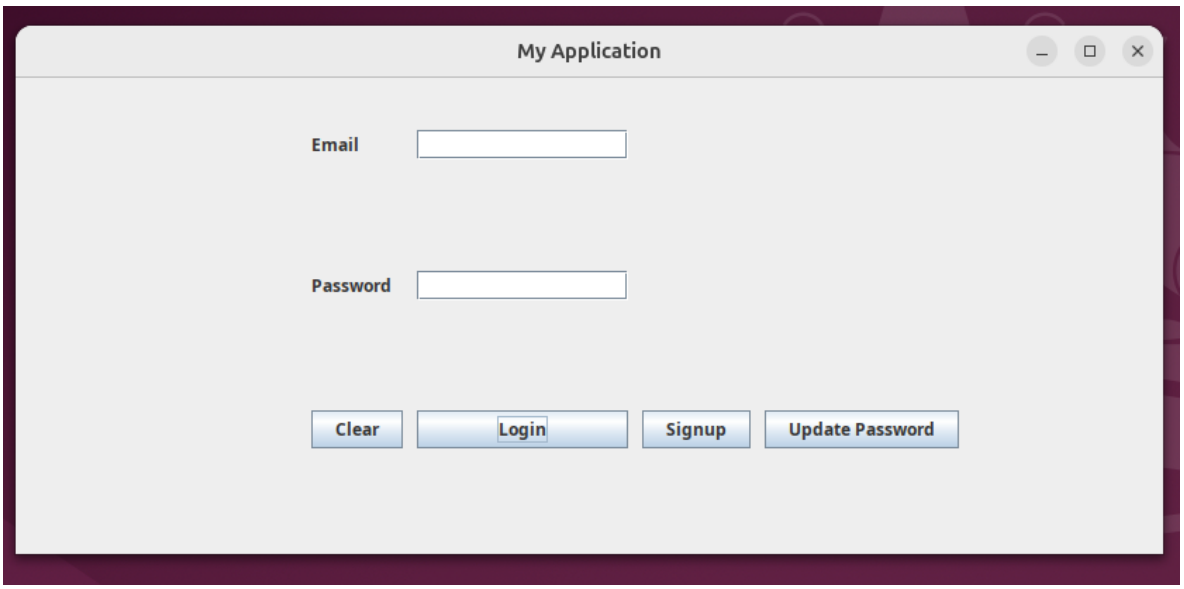


Figure 4.1 the Login GUI

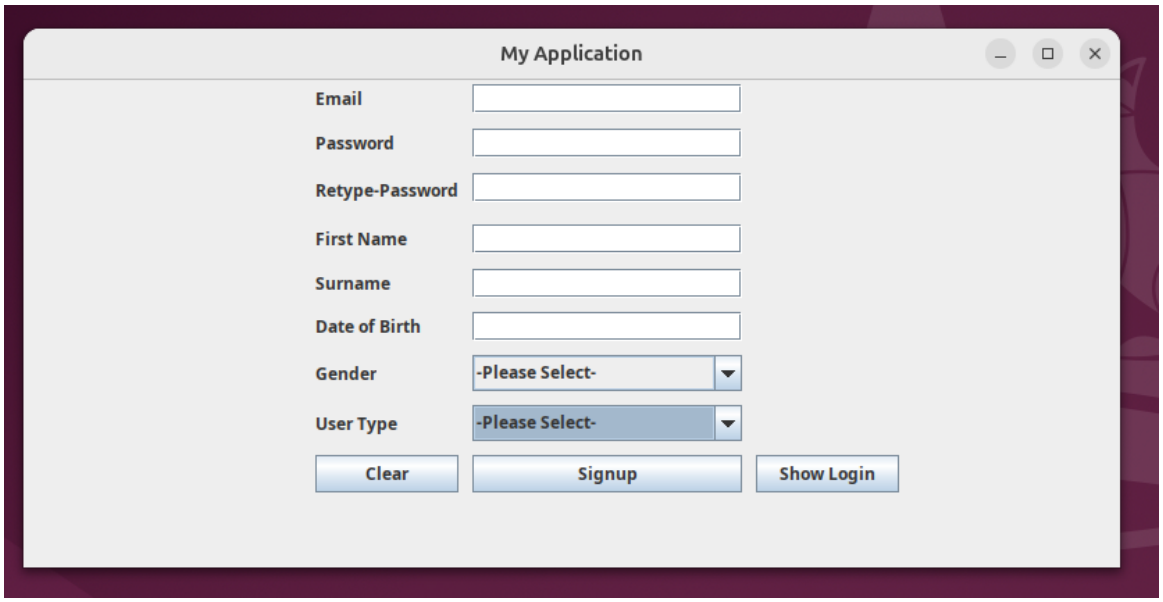


Figure 4.12 the Sign-up GUI

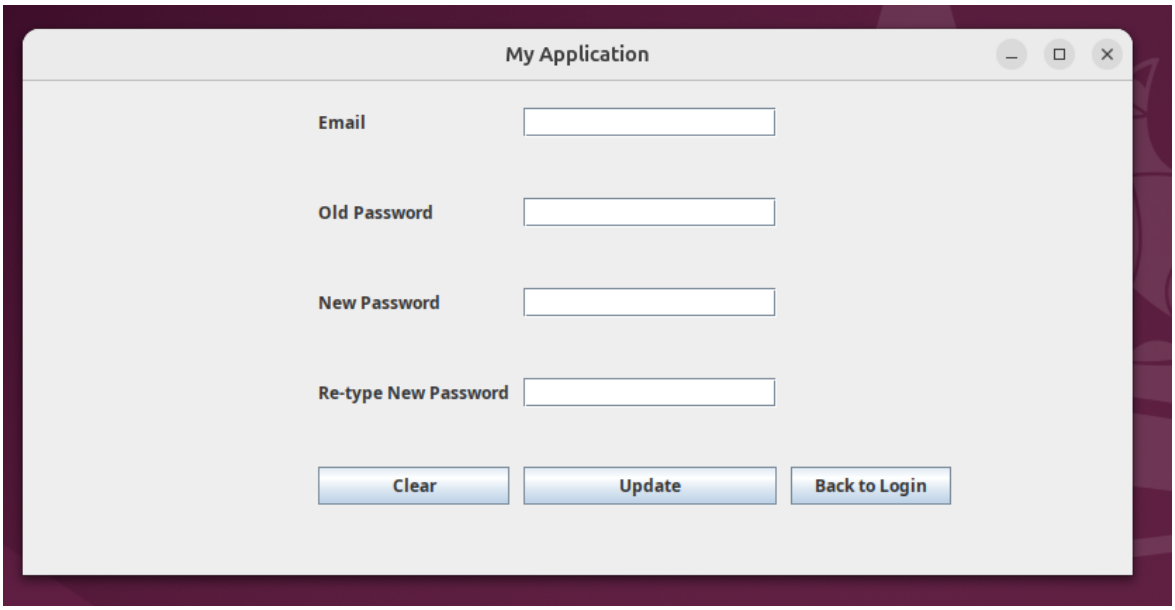


Figure 4.13 the Update Password GUI

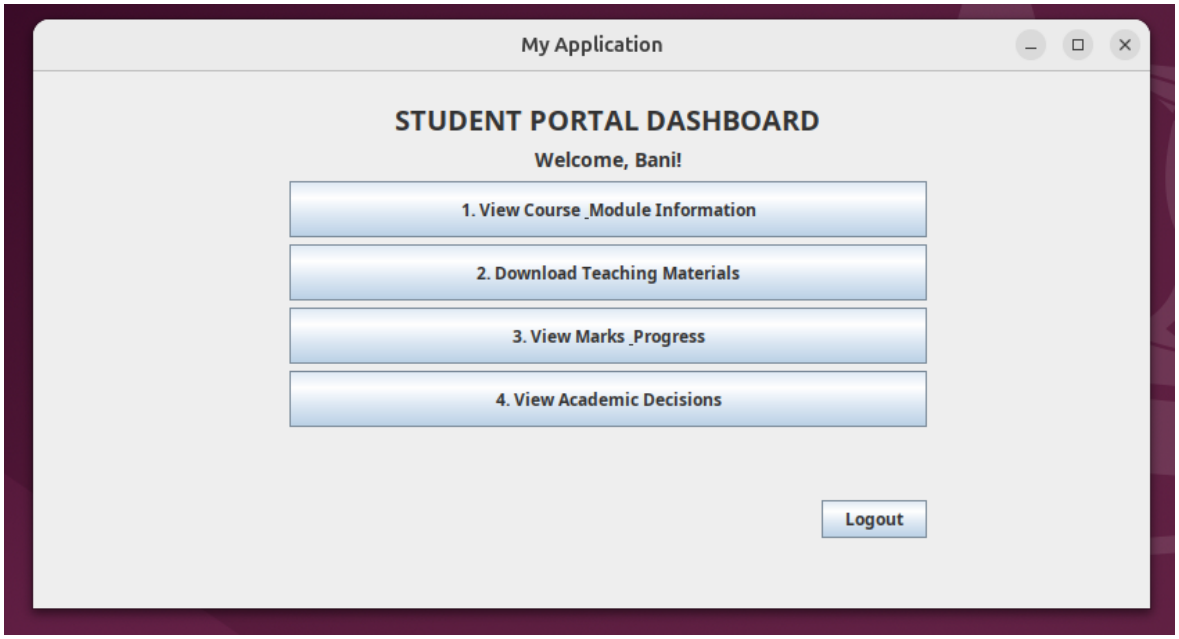


Figure 4.2 the Student Dashboard GUI

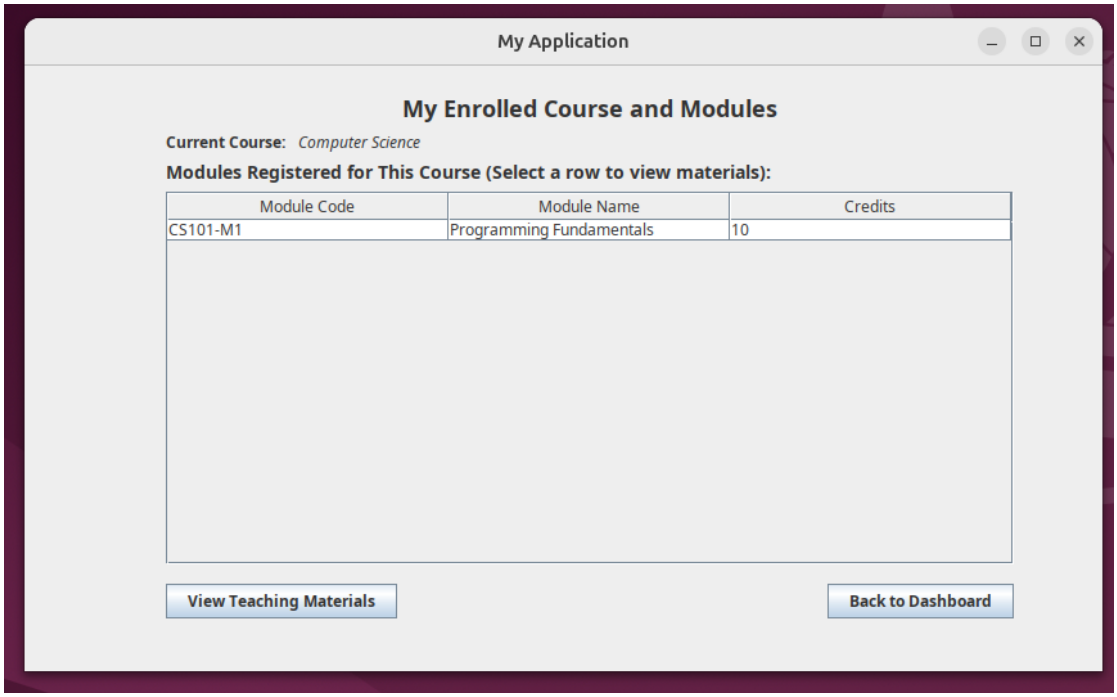


Figure 4.21 Student – View Course & Module Information GUI

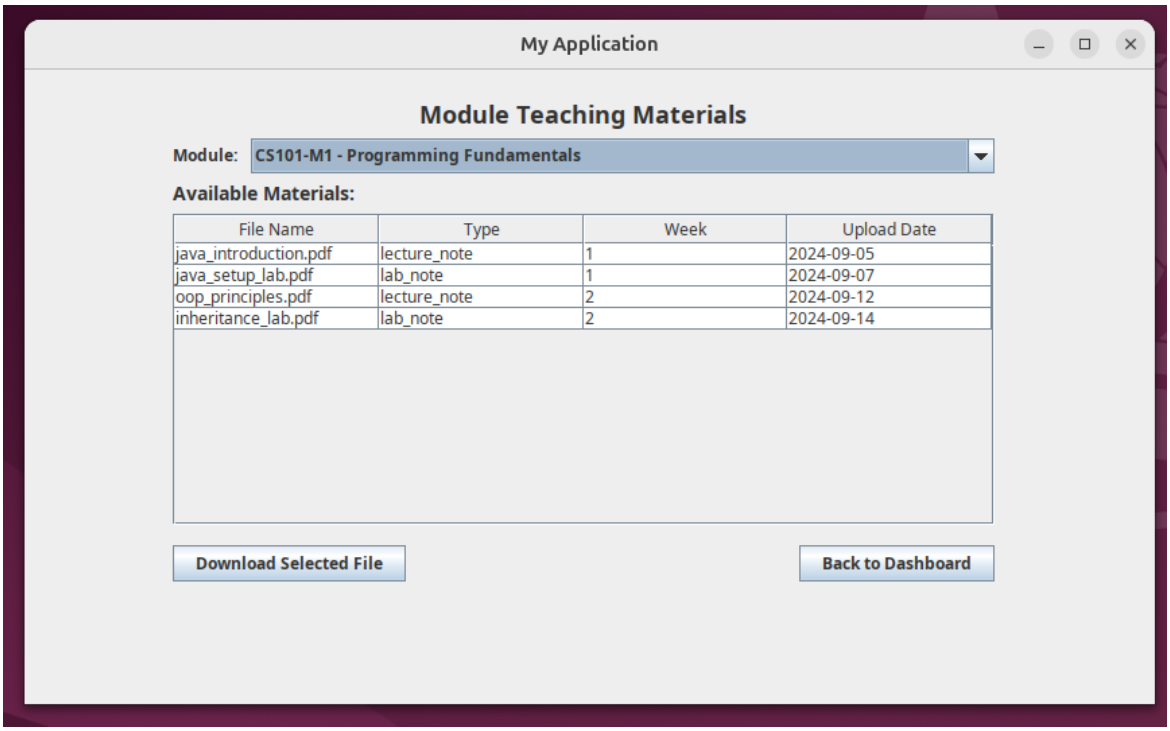


Figure 4.22 Student – Download teaching materials GUI

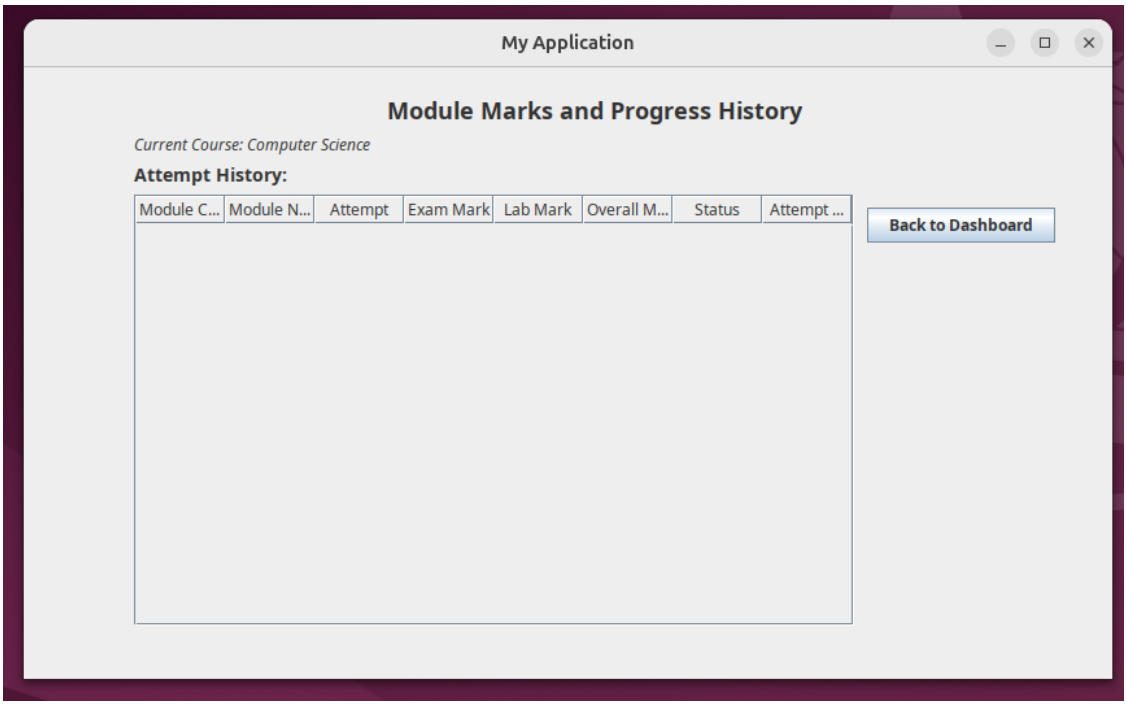


Figure 4.23 Student – View Marks GUI

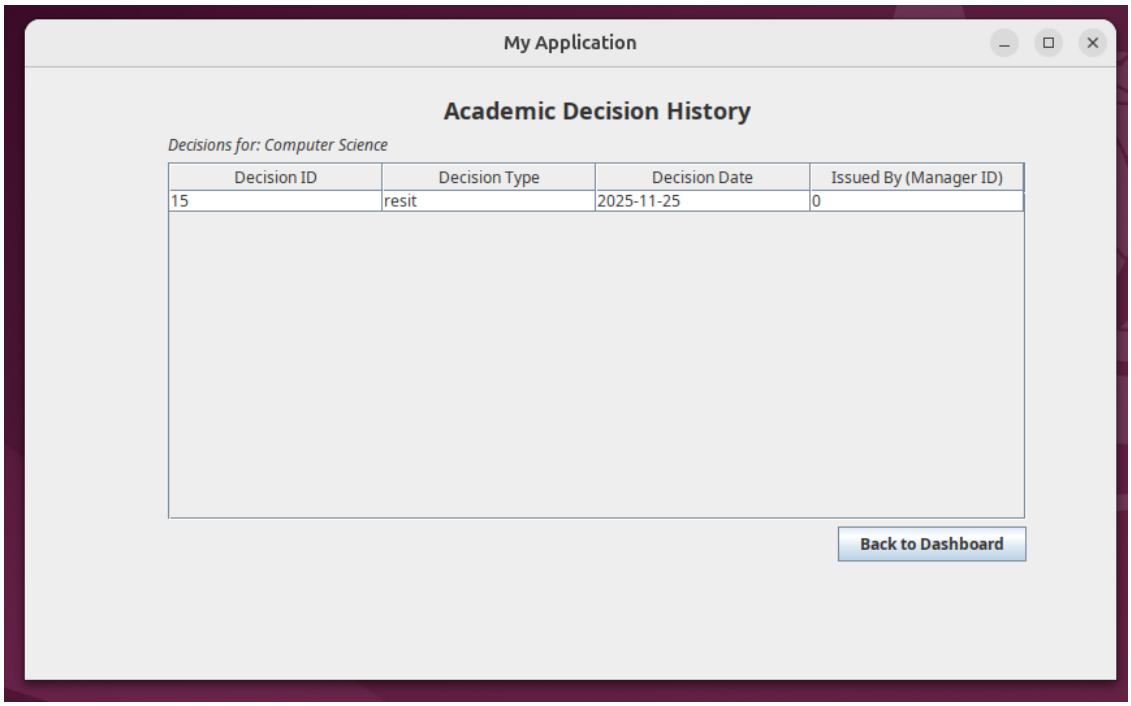


Figure 4.24 Student – View Academic Decision GUI

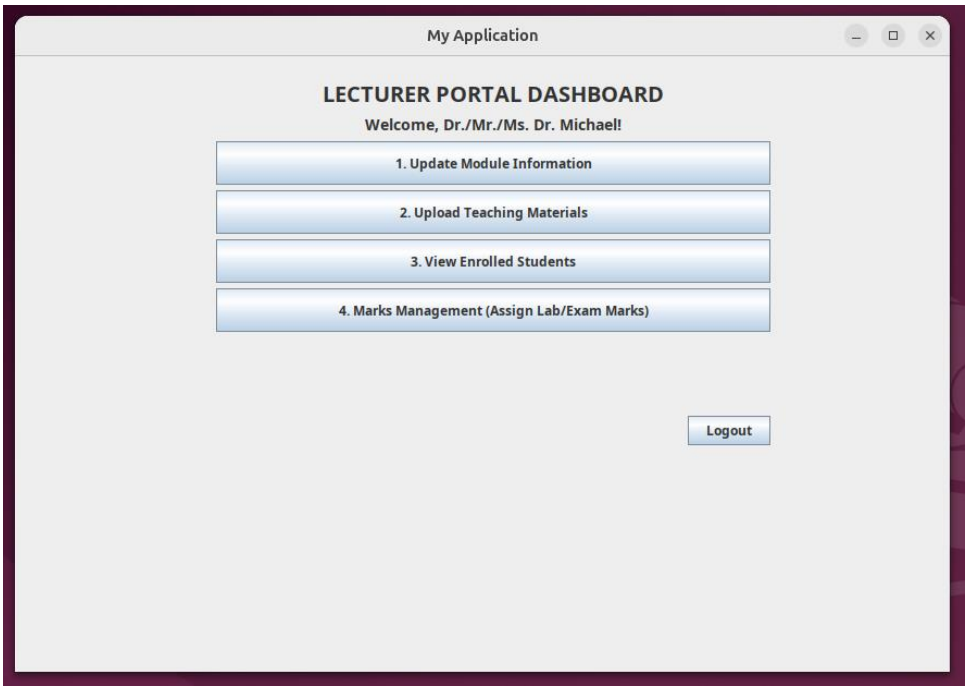


Figure 4.3 Lecturer Dashboard GUI

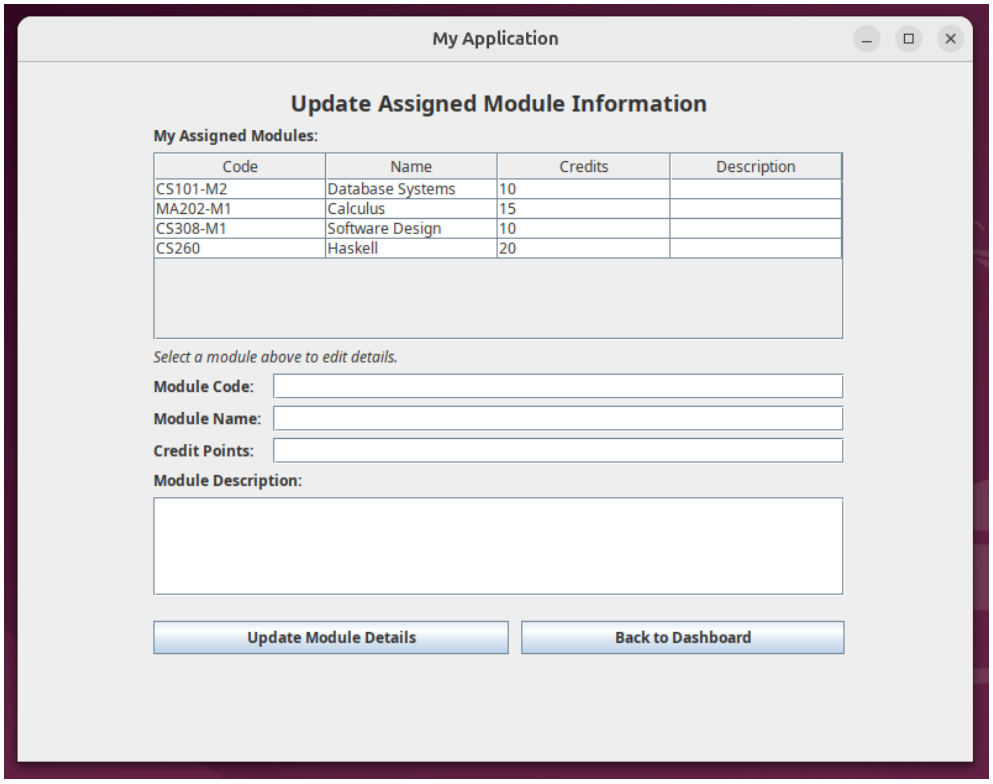


Figure 4.31 Lecturer – Update Module Information GUI

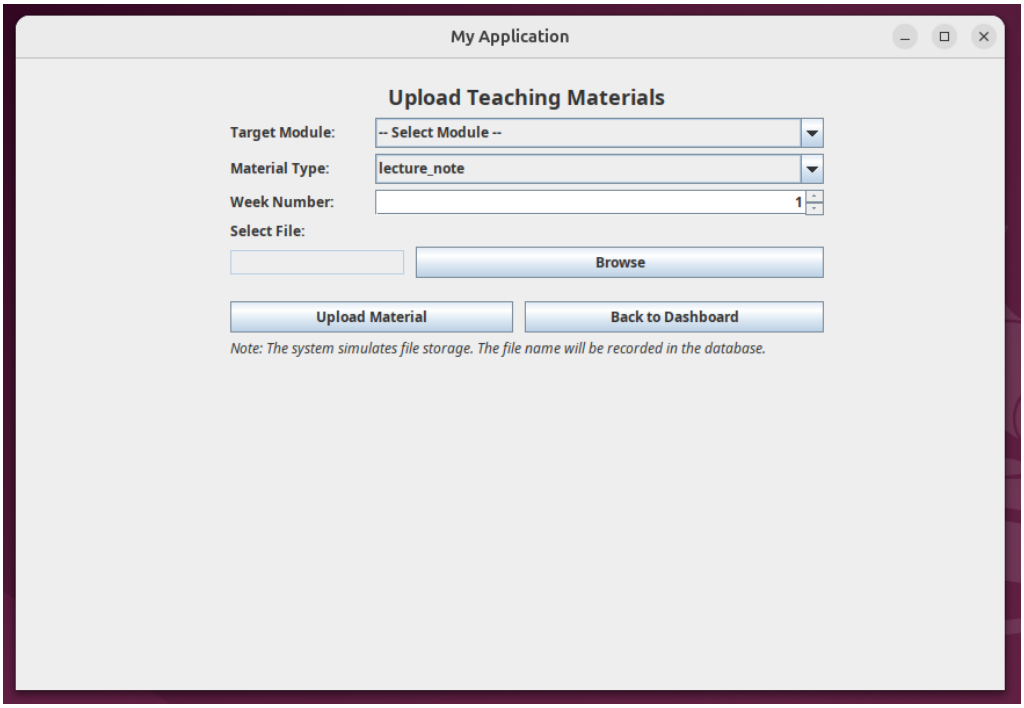


Figure 4.32 Lecturer – Upload Teaching Materials GUI

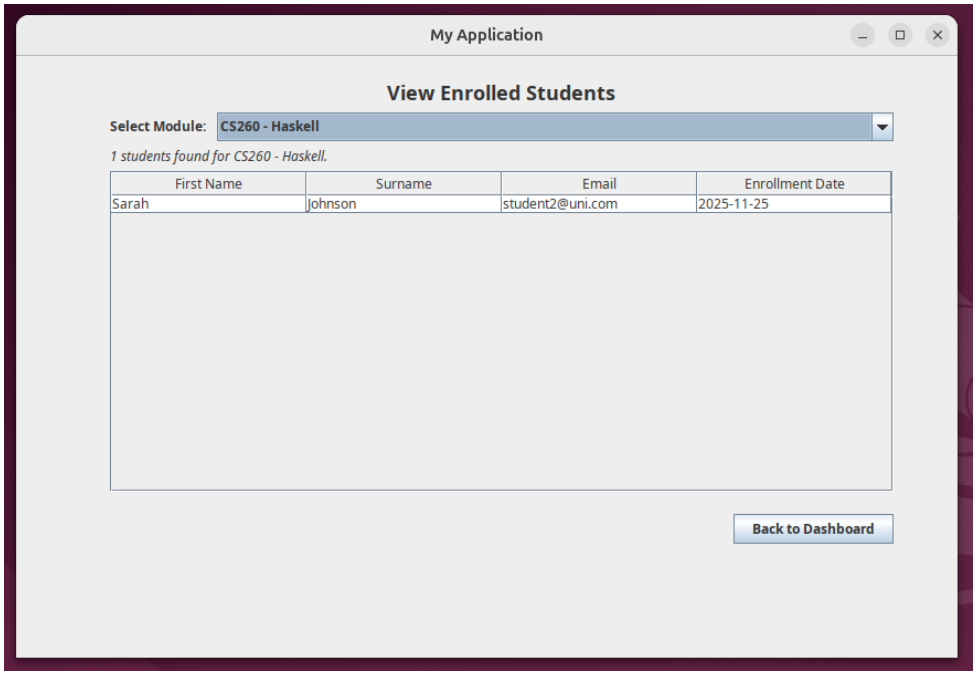


Figure 4.32 Lecturer – View Enrolled Students for lecturer GUI

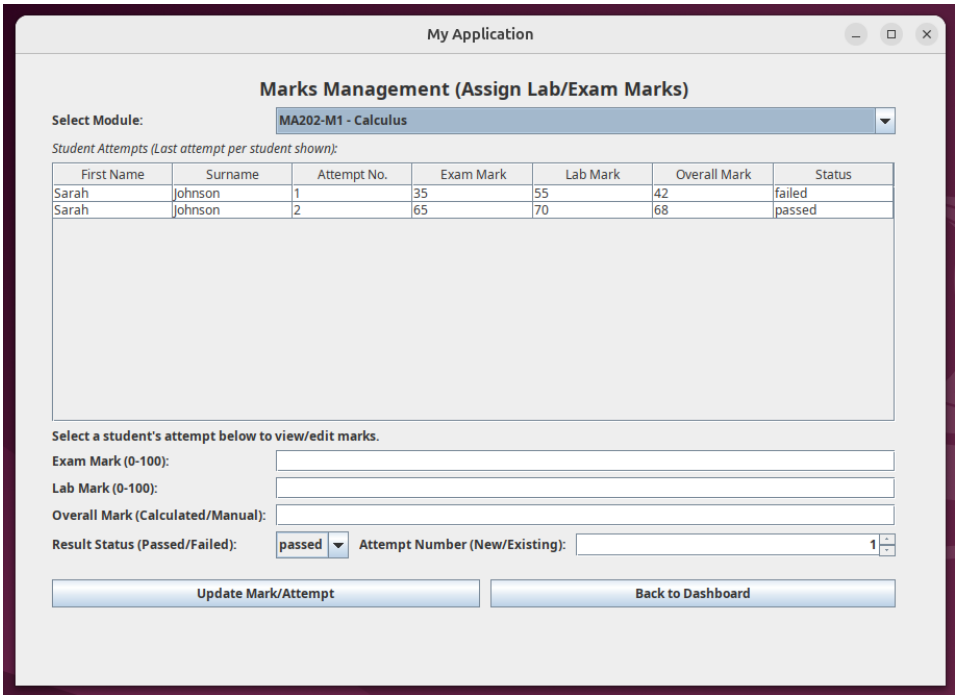


Figure 4.33 Lecturer – Marks Management GUI

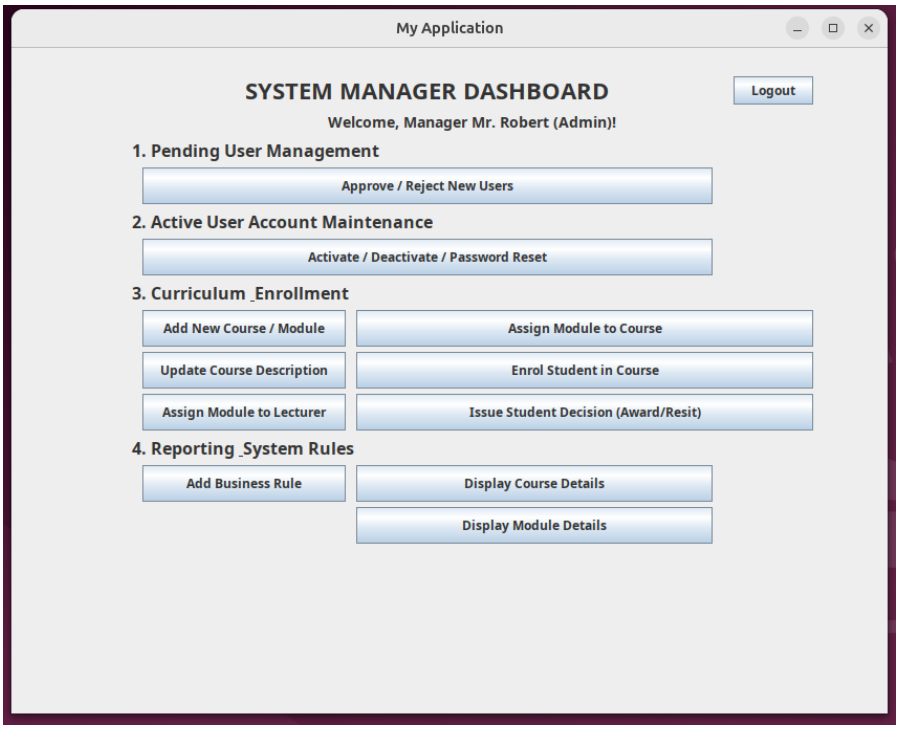


Figure 4.4 Manager Dashboard GUI

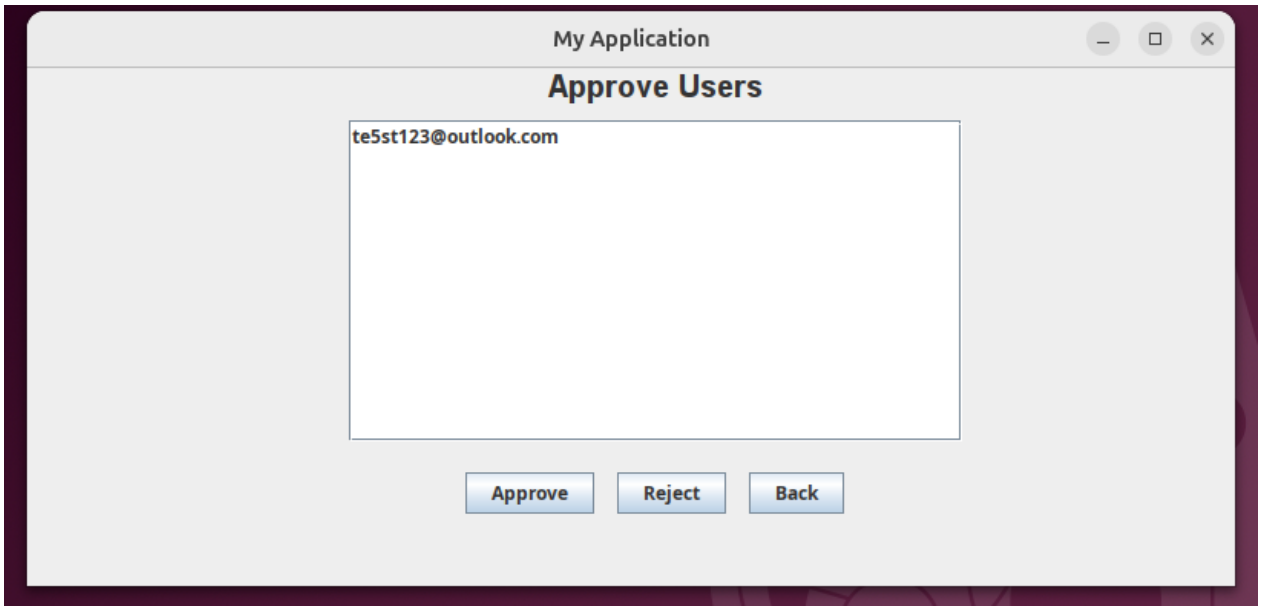


Figure 4.41 Manager – Approve & Reject Users GUI

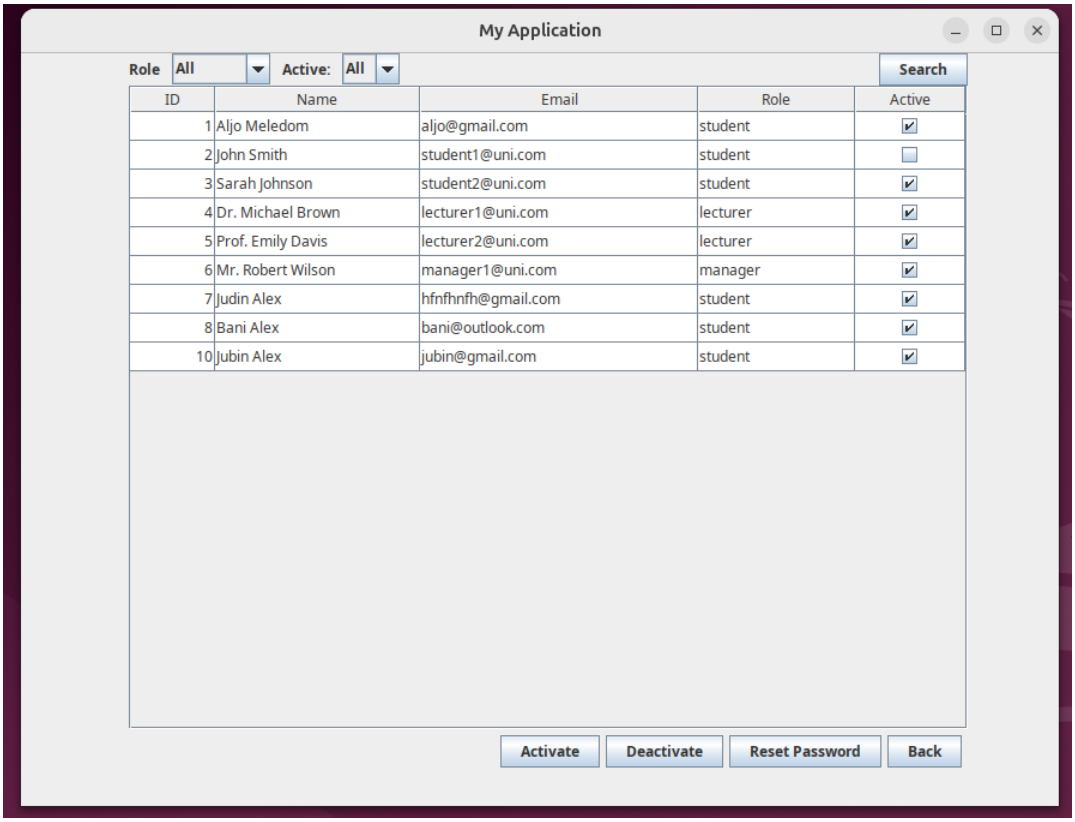


Figure 4.42 Manager – Approve/Deactivate Users & Password Reset GUI

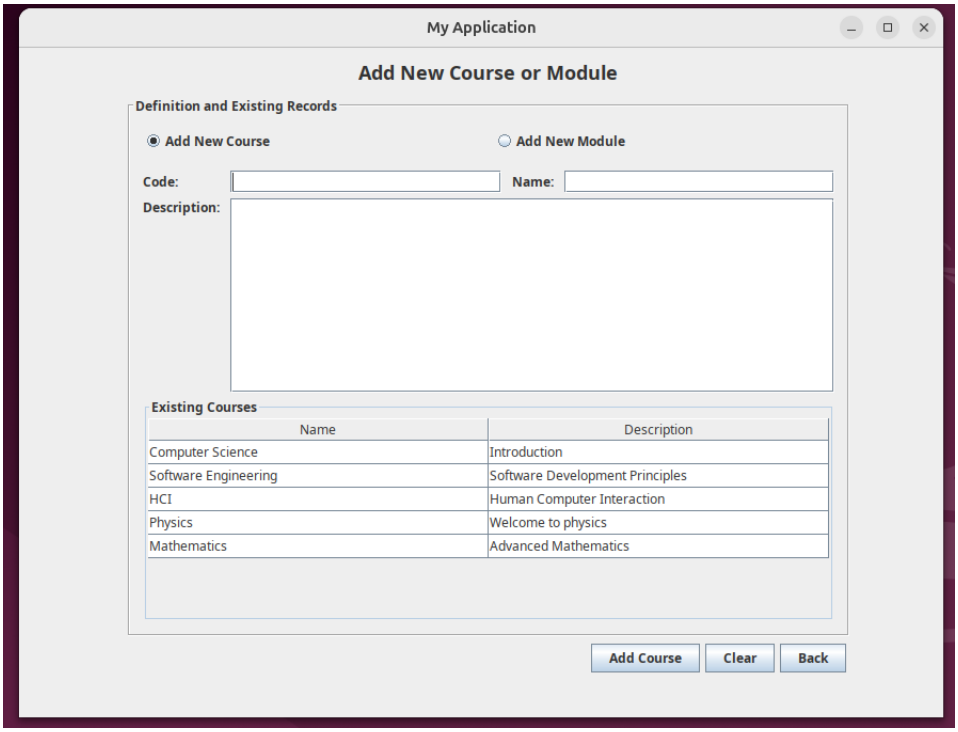


Figure 4.43 Manager – Add New Course GUI

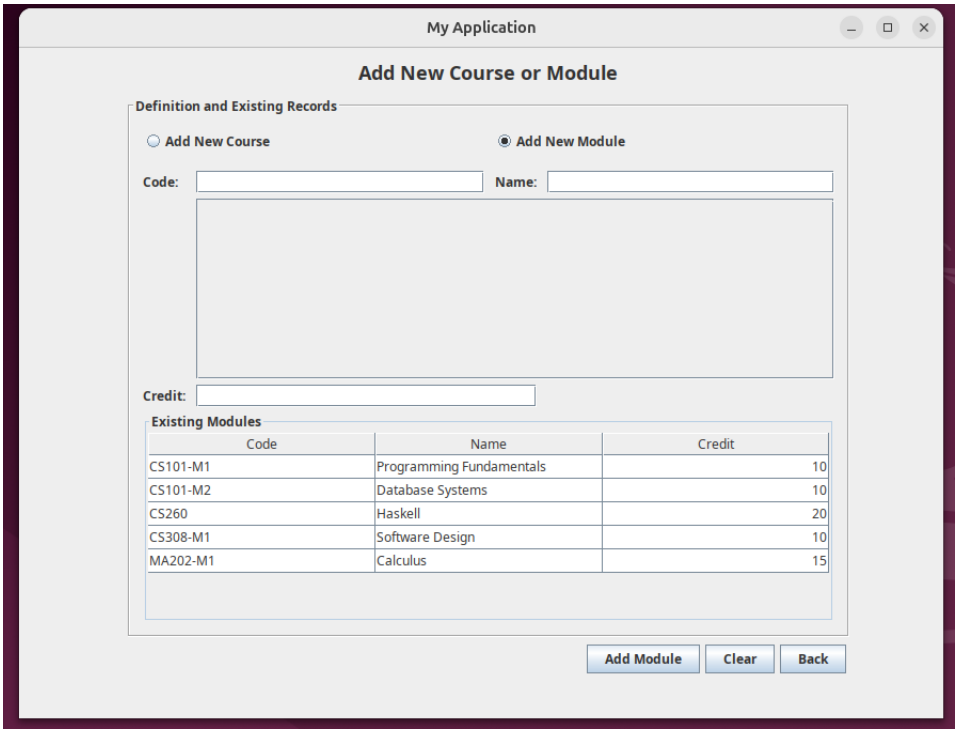


Figure 4.44 Manager – Add New Module GUI

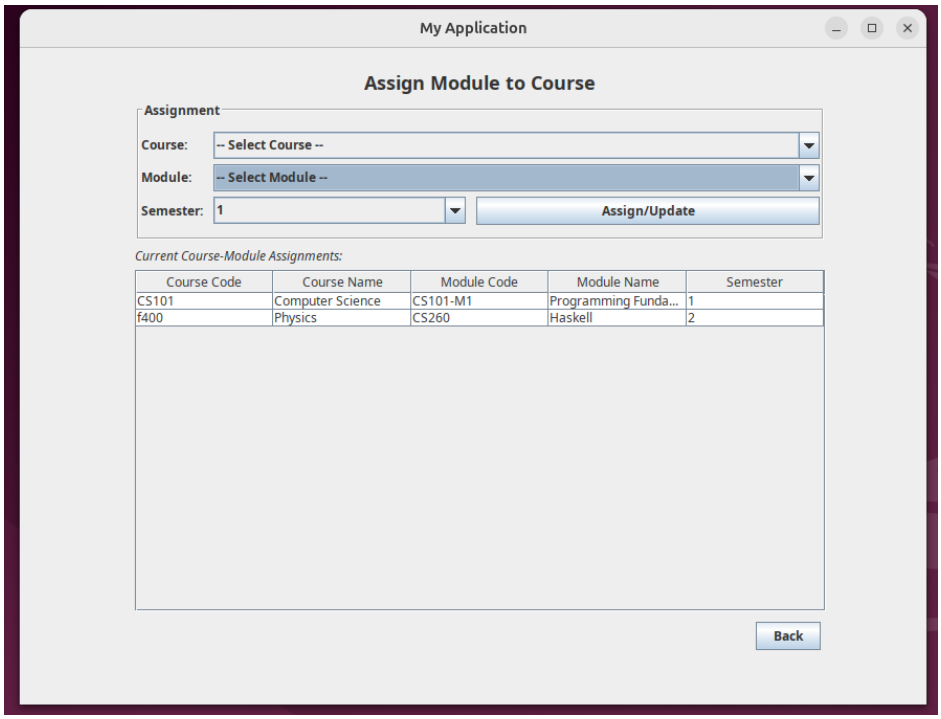


Figure 4.45 Manager – Assign Module to Course

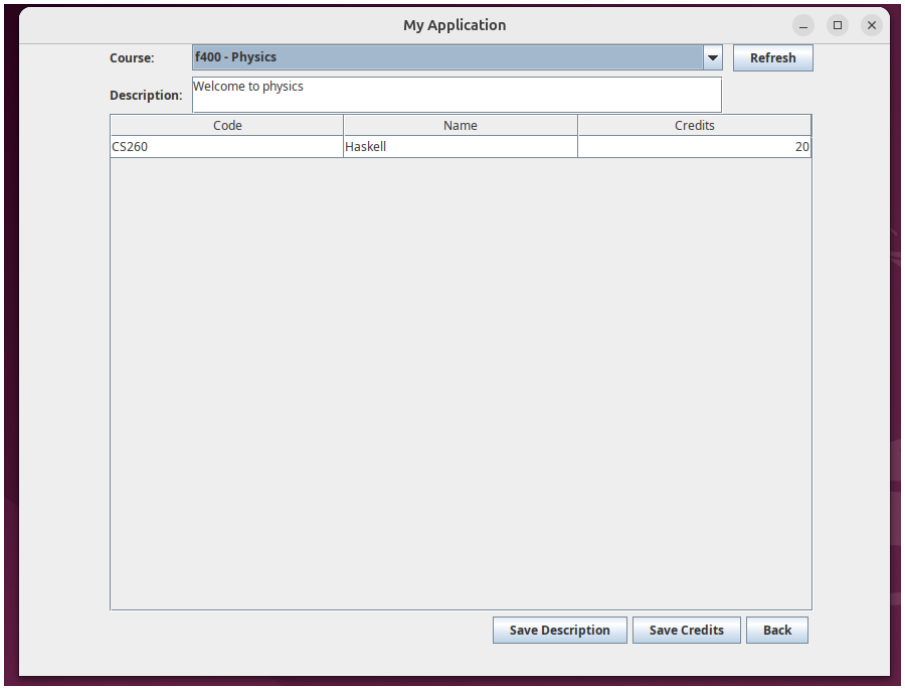


Figure 4.46 Manager – Update Course Description

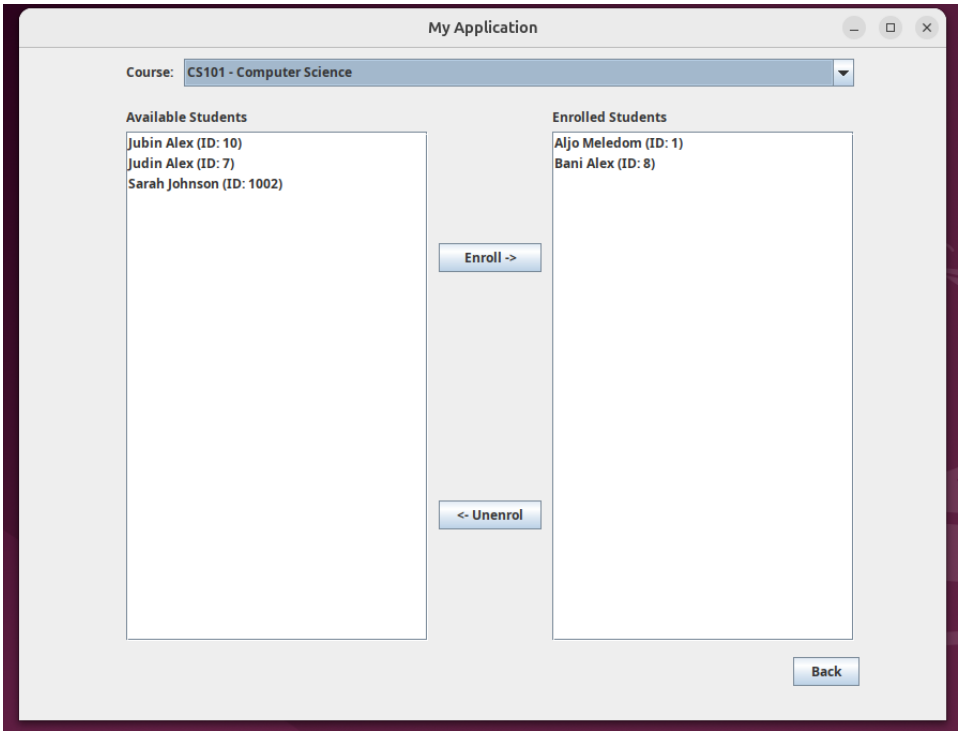


Figure 4.47 Manager – Enroll Student in a course GUI

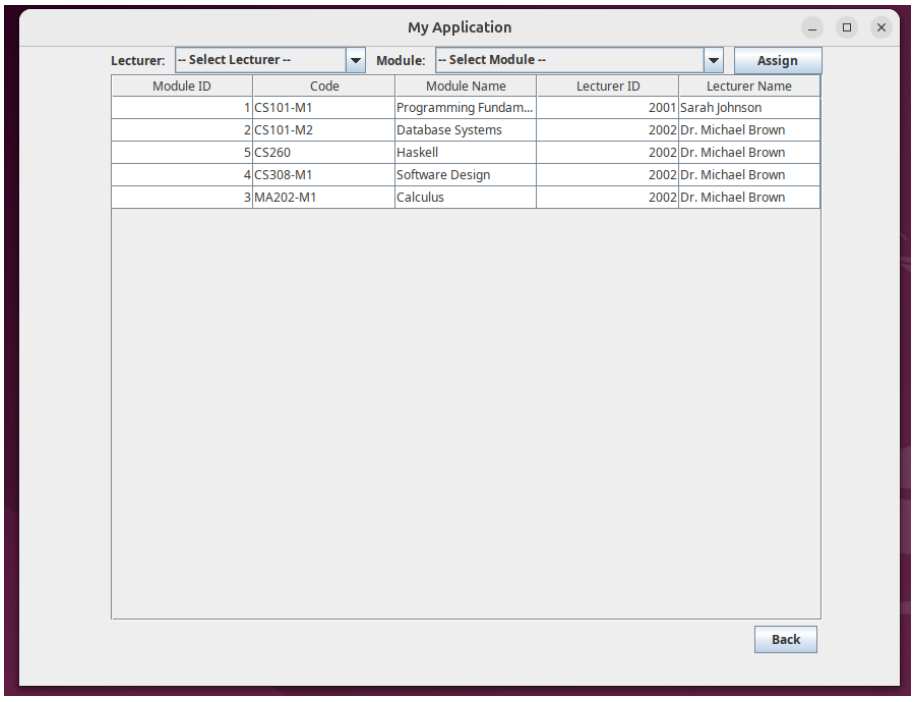


Figure 4.48 Manager – Assign Module to Lecturer GUI

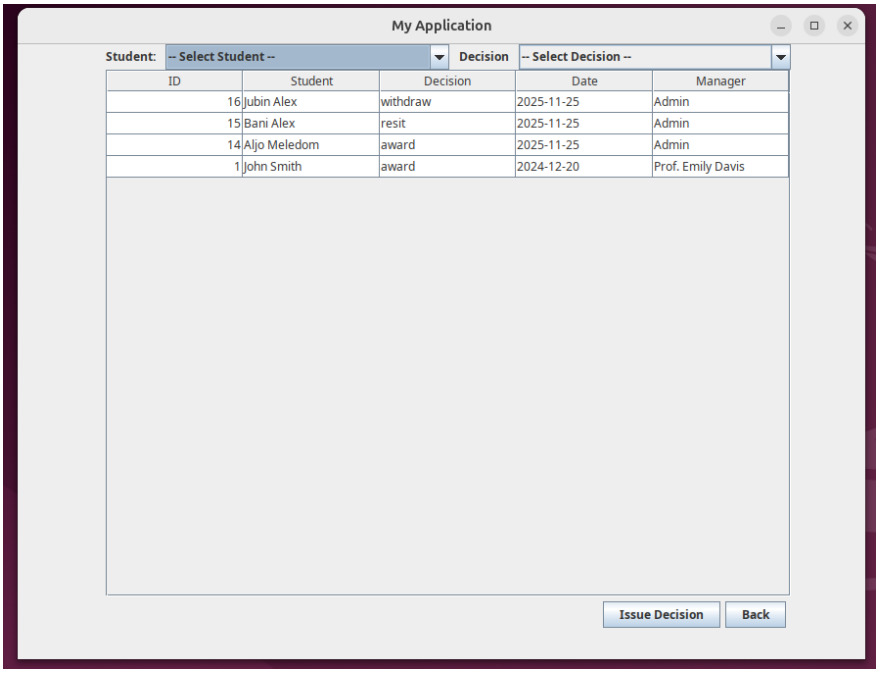


Figure 4.49 Manager – Issue Student Decision GUI

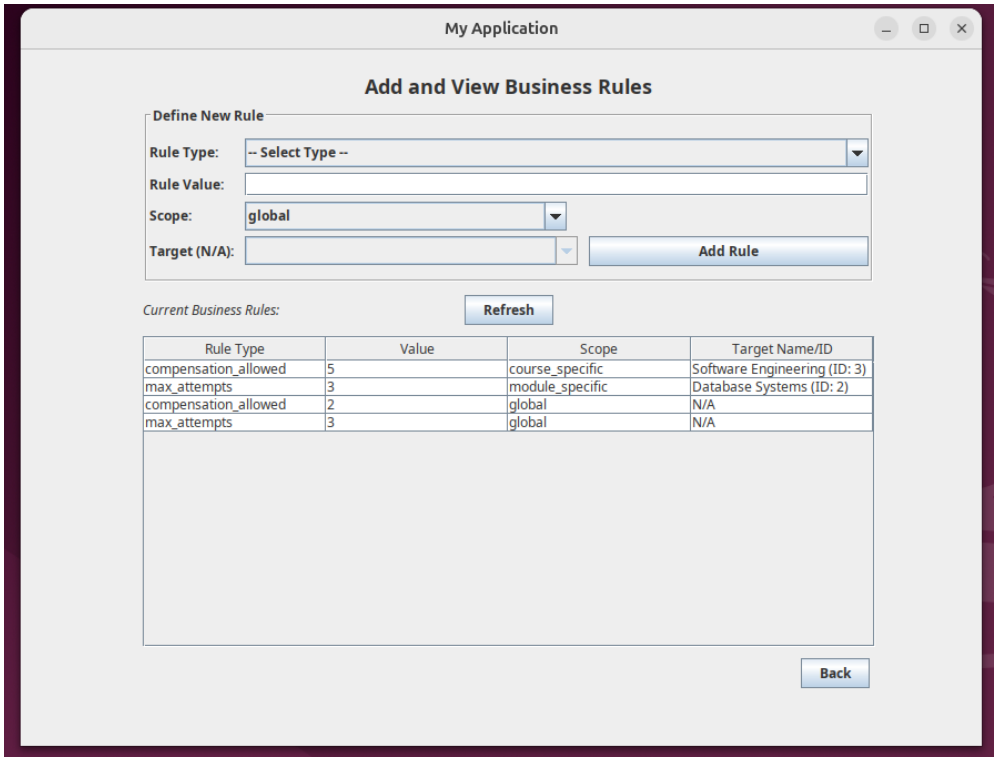


Figure 4.5 Manager – Add Business Rule GUI

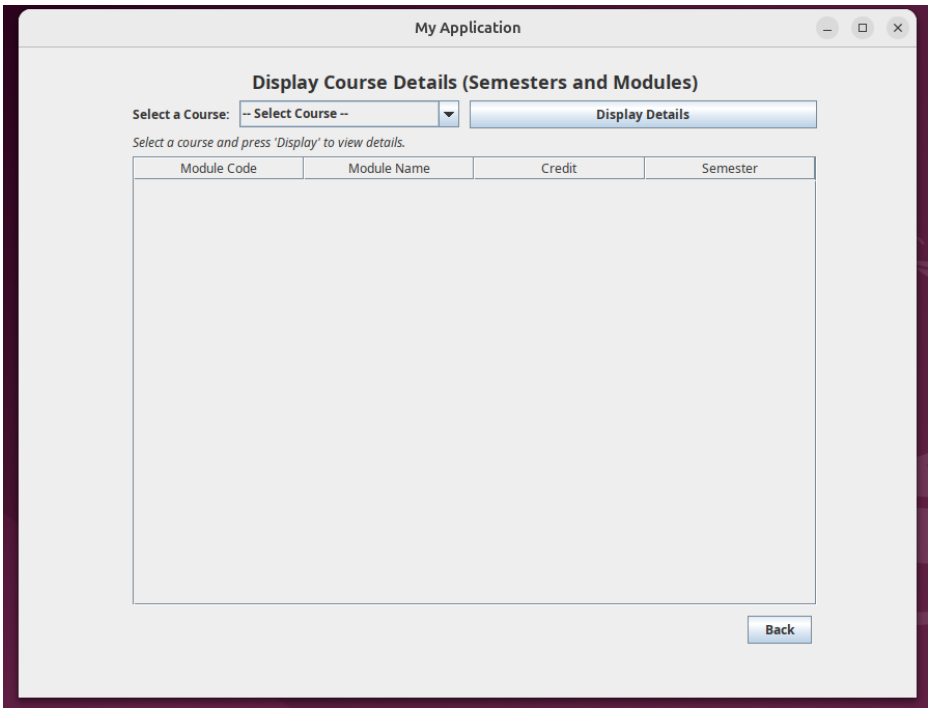


Figure 4.51 Manager – Display Course Details GUI

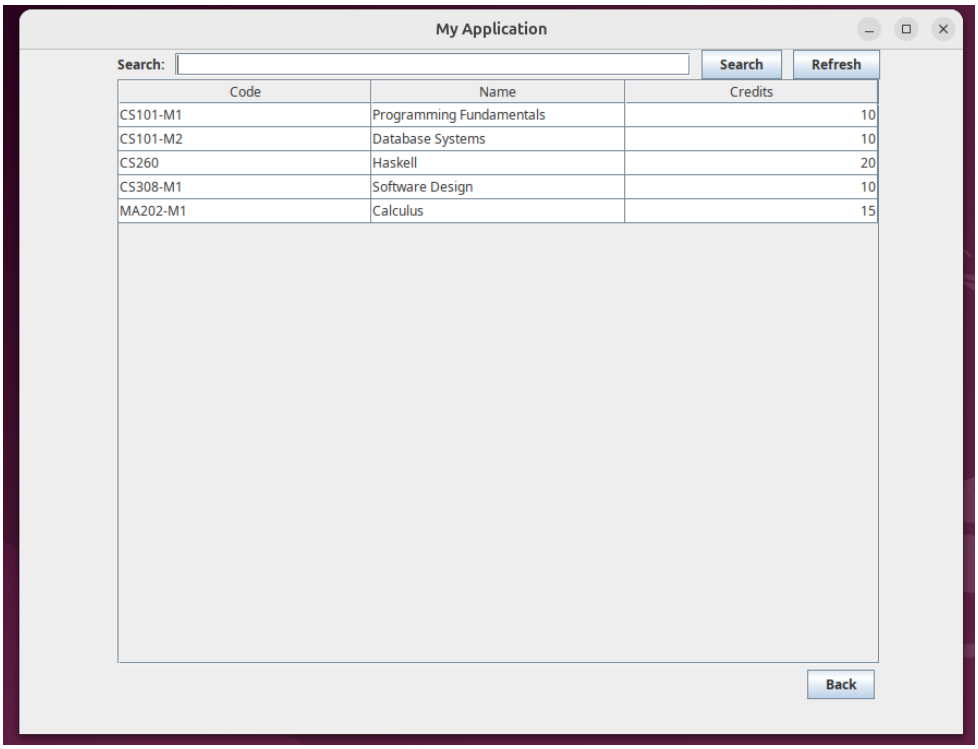


Figure 4.52 Manager – Display Module Details GUI

5. Class diagram

Provide a list of classes and explain very briefly why these classes are required (reflect on the system requirements):

5.1. Class ControllerUser: Handles user-related actions such as login, sign-up workflow, account activation/deactivation, password updates, etc. Required because all three actors (Student, Lecturer, Manager) interact with user accounts.

5.2. Class App: Entry point of the program; launches the system and coordinates initial setup

5.3. Class DAOUser: Handles database access for user data (saving, retrieving, activating, deactivating users). Required because the system needs persistent storage for accounts, enrolments, course data, etc.

5.4. Class IUserModel / ModelUser: Defines and implements the user data structure. Required because all user types need a consistent model for login, details, and roles.

5.5. Class IValidator / IValidatorImpl: Validates input. Required because sign-up, login, course creation, module updates all require validation for consistency and security.

5.6. Class MySQLConnect: Manages database connections. Required because the system stores users, courses, modules, enrolments, exam marks, decisions, etc.

5.7. Class ViewAddBusinessRule: Needed for managers use case, adds a business rule.

5.8. Class ViewAddCourse: Supports “Add a new course” requirement.

5.9. Class ViewApproveUsers: Supports “Approve Users” (activating accounts).

5.10. Class ViewAssignModuleToCourse: Assigns a module to a course

5.11. Class ViewAssignModuleToLecturer: Supports “Assign a module to a given lecturer”.

5.12. Class ViewCourseModuleInformation: Allows viewing course/module information. Required for lecturer and manager workflows.

5.13. Class ViewDisplayCourseDetails: Supports “Display a course details”.

5.14. Class ViewDisplayModuleDetails: Supports “Update Module Details” and “Display module details”.

5.15. Class ViewDownloadMaterials: Required because students can download lecture/lab notes.

5.16. Class ViewEnrolledStudents: Supports “Display enrolled students in their module” for lecturers. Also ties into “Update exam/lab marks”.

5.17. **Class ViewEnrolledStudents:** View the enrolled students in a module.

5.18. **Class ViewEnrolledStudentsCourse:** enrolls a student in a particular course.

5.19. **Class ViewIssueStudentDecision:** allows the manager to issue one of the 3 decisions (award/ resit/ withdraw)

5.20. **Class ViewLecturerDashboard:** The designated dashboard page for lecturer.

5.21. **Class ViewLogin:** Allows the user to login to his account dashboard.

5.22. **Class ViewManageAccounts:** Allows the manager to activate/deactivate all the accounts.

5.23. **Class ViewManagerDashboard:** The designated dashboard page for managers.

5.24. **Class ViewMarksManagement:** Allows the lecturer to assign marks to every student.

5.25. **Class ViewMarksProgress:** Allows the student to view their marks in their assigned modules.

5.26. **Class ViewSignup:** Allows the user to sign up to the university portal.

5.27. **Class ViewStudentDashboard:** The designated dashboard page for students.

5.28. **Class ViewStudentDecisions:** Allows the student to see their decisions assigned to them by the manager.

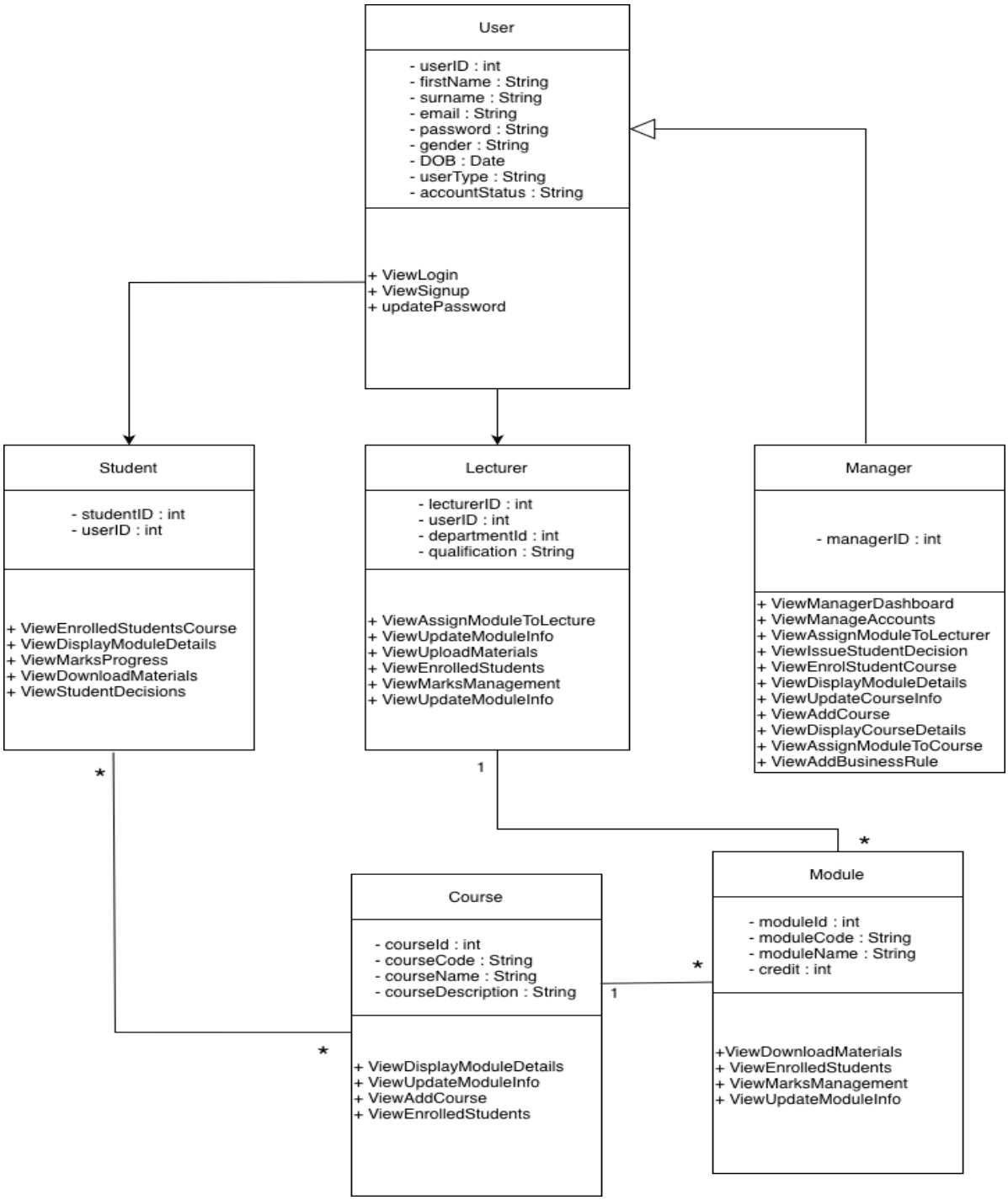
5.29. **Class ViewUpdateCourseInfo:** Allows the manager to update a particular course information.

5.30. **Class ViewUpdateModuleInfo:** Allows the manager to update the module's information.

5.31. **Class ViewUpdatePassword:** Allows the user to update their password.

5.32. **Class ViewUploadMaterials:** Allows the lecturer to upload teaching materials.

Then provide the class diagram as a separate image file called **classDgrm.tiff or any other format such as pdf** that can be zoomed in/ out to view each class and its methods.


User
- userID : int
- firstName : String
- surname : String
- email : String
- password : String
- gender : String
- DOB : Date
- userType : String
- accountStatus : String
+ ViewLogin
+ ViewSignup
+ updatePassword
Student
- studentID : int
- userID : int
+ ViewEnrolledStudentsCourse
+ ViewDisplayModuleDetails
+ ViewMarksProgress
+ ViewDownloadMaterials
+ ViewStudentDecisions
Lecturer
- lecturerID : int
- userID : int
- departmentId : int
- qualification : String
+ ViewAssignModuleToLecture
+ ViewUpdateModuleInfo
+ ViewUploadMaterials
+ ViewEnrolledStudents
+ ViewMarksManagement
+ ViewUpdateModuleInfo
Manager
- managerID : int
+ ViewManagerDashboard
+ ViewManageAccounts
+ ViewAssignModuleToLecturer
+ ViewIssueStudentDecision
+ ViewEnrolStudentCourse
+ ViewDisplayModuleDetails
+ ViewUpdateCourseInfo
+ ViewAddCourse
+ ViewDisplayCourseDetails
+ ViewAssignModuleToCourse
+ ViewAddBusinessRule
*
1
*
Course
- courseId : int
- courseCode : String
- courseName : String
- courseDescription : String
+ ViewDisplayModuleDetails
+ ViewUpdateModuleInfo
+ ViewAddCourse
+ ViewEnrolledStudents
*
1
Module
- moduleId : int
- moduleCode : String
- moduleName : String
- credit : int
+ViewDownloadMaterials
+ ViewEnrolledStudents
+ ViewMarksManagement
+ ViewUpdateModuleInfo

6. Test cases:

This document doesn't have any headings. To add headings to your Table of Contents, go to Home > Styles

Provide **system test cases** that can test the complete scenario of each **system use case** (SUC) that you have described in subsection 3.2. **Please note that these text cases are different from the test cases that you generated by using JUnit5** as the test cases generated by using Junit5 will be included in the **test folder** of your project that test individual units(classes). However, **system test cases** aim to test a complete scenario and not merely methods in a given class.

SUC Test Cases	Expected Output
Signup	System creates a new pending account if all inputs are valid and the email is not already registered.
Update Password	System verifies the old password, updates it to the new valid password, and confirms the change to the user
Validate Login	System checks the entered credentials and logs the user in if they are correct and the account is active.
Enroll Student in a course	System verifies eligibility and successfully adds the student to the selected course.
Add a business rule	System saves the new business rule and ensures it is applied wherever relevant in the system.
Issue student decision	System records the decision and makes it immediately visible to the student in their dashboard.
Manage other accounts	System updates the selected user account based on the manager's action (approve, deactivate, or reset) and reflects the changes immediately.

7. Adapted work from references

- This text is an example of reporting other works or pieces of code that you have used or adapted. The following methods in class A.java were taken from the source reference [1]. These methods are: **connect, disconnect, view and show**.
- The following methods in class B.java were taken from the source reference [2]. These methods are **rest, distribute, and cancelUpdate**.
- ..

References

[1]

[2]