Academic Database Management System

Hamza Khan, Nikita Upadhyay, Cherith Boya. Mark Touzo, Alan Hiscott. Suhas Chenna

Group #11

University of Guelph

Engineering 1420

April 5th, 2025

Project Purpose:

The purpose of this University Management System is to streamline academic administration by providing a role-based platform for managing courses, students, faculty, and events. Built using JavaFX, the system ensures efficient operations and user-friendly interaction for administrators, faculty, and students. Key goals include enhancing data accuracy, enabling real-time updates, improving scheduling, preventing over-enrollment, and simplifying event and tuition management — all while maintaining secure, role-specific access to system features.

Project Timeline

The University Management System Program was developed by our team over the course of eight weeks. The project's objective was to provide an effective, role-based platform for overseeing academic and administrative activities in a academic setting, simulating a real universities platform.

To start the project, we divided into two subgroups, one of which worked on the front end (developing the JavaFX user interface) and the other on the back end (managing file storage, data structures, and logic). This made it possible for us to work quickly on both sides of the system simultaneously and made sure that the interface design complemented the functionality being created.

As the system design became progressively complicated, we decided to combine the teams and focus on the back end. We were able to collaborate on important features including file-based data integration, role-based access implementation, data consistency across modules, and authentication. It was easier to troubleshoot and test as a team when we worked as a single team, which also helped us maintain consistency amongst programs and effectively get the job done.

Below is a timeline of the project, including how our team organized the work and contributed to different parts of the system.

Week	***Date***	***Members Involved***	***Tasks***
1	(02/09 – 02/15)	**All Members**	We began by outlining the scope of the system, identifying the main modules and authentication. To structure our workflow, we split the team into two subgroups: • **Front-End Team (Nikita, Alan, Suhas):** Focused on designing the JavaFX-based GUI and layout. • **Back-End Team (Hamza, Cherith, Mark):** Responsible for setting up the core data models, file I/O handling, and role-based logic using the frames created. We agreed on key design principles, spent time figure out how to work JavaFX for the GUI, and created shared folders and documentation on One Drive to keep our code organized.

2	*Reading Week* (02/16 – 02/22)	**All Members**	The Front-End Team created the login screen, basic dashboard layouts, and navigation menus tailored to ADMIN and USER roles. Furthermore, the team got started on the chosen modules and their frames. • Nikita worked on the dashboards for both admin and user, and Suhas and Alan worked on starting the frames for modules student, subject and course. The Back-End Team began developing the authentication system, along with the data structure for students, faculty, and subjects. By the end of the week, we were able to log in with temporary credentials and navigate between pages, depending on the user's role. • Hamza worked on developing the data structures for the modules, and Cherith and Mark worked together on the authentication for both admin and user.
3	(02/23 – 03/01)	**Hamza, Nikita** **Cherith, Mark**	We made major progress on both the front-end and back-end for two core modules: Subject Management and Course Management. • Nikita completed the interface frames and layouts for these modules. These screens were styled to be consistent with the overall dashboard and menu structure. • Hamza started linking these front-end components with their respective functionalities. Admins could now add subjects and courses, and the system would validate. User roles were restricted to view-only access, in line with our role-based permissions setup. • Cherith and Mark completed the authentication for role-based login, ensuring those logging in were directed to the correct frame whether they be student, faculty or admin.
Phase 1 Deliverable: GUI Development, Core Functionality, and Authentication			

4	(03/02 – 03/08)	**All Members**	We started working on the remaining core areas: Student Management, Faculty Management, and Event Management.
			• The front-end team began designing the interface frames for these new modules. Nikita split up the work amongst all front-end team members. The frames followed the same structural and visual patterns as the earlier modules.
			• The back-end team was hard at work building the logic and functionality behind these modules, as well as finishing work from the previous modules.
			  o Hamza worked on completing subject and course modules to guarantee the system was interconnected between frames and functionalities.
			  o Cherith and Mark got started student management as this module was the largest, as well as integrating cross functionality between courses and student.

			We also began connecting modules together, so actions like assigning a course to a faculty member or enrolling a student would update files across the system.
5	(03/09 – 03/15)	**Hamza, Nikita, Cherith, Mark**	At this point the project was coming together, and we were continuing to polish and work on necessary modules. Basic functionalities of all modules were complete; however, specifications were still in progress. • Nikita completed all the front-end interface frames for the main modules. Each screen was in place and visually consistent. • Hamza started working on the basic functionality between faculty and course management, and began end-to-end testing for the flow between modules • Cherith and Mark continued working on student management, as basic functionalities were complete, and they got started on file handling complications and connecting the modules together.
Phase 2 Deliverable: Partial System Implementation with Role Management			
6	(03/16 – 03/22)	**Hamza, Nikita, Cherith, Mark**	The team now shifted focus to the backend. With all front-end interface frames complete, the four of us worked together on refining and finalizing the remaining backend logic across all modules. • We divided tasks dynamically based on priority and familiarity with different modules. Some members focused on improving the Student and Faculty Management logic, while others handled course registration, event signup logic, and data validation. • If any changes were needed on the front-end, such as adding a new field or button to match a backend feature, we made those updates collaboratively.
7	(03/23 – 03/29)	**Hamza, Nikita, Cherith, Mark**	The team continued working on backend development while also shifting focus to polishing the system and performing full end-to-end testing. • At this point, instead of having categorized work the team worked together on tasks and split work up regarding what needed prioritization in the moment. • During this time, we also began testing the entire system from the perspectives of different user roles, guaranteeing that correct frames were in place and file handling was working. • While the primary focus remained on backend development, we made several adjustments to the front-end interface based on feedback from testing.
8	*Presentation Week* (03/30 – 04/03)	**Hamza, Nikita, Cherith, Mark**	In the final week, we focused on cleaning up the system and making sure everything worked smoothly. We finalized file integration, confirmed that data was being saved correctly, and did a last round of testing across all modules. Minor bugs were fixed, and we made a few final changes to the user interface.
Phase 3 Deliverable: Full System Implementation			

Classes Breakdown and Identifying Classes Relationships/Key Methods.

Administrator Role – Managing the Schooling System

Admins are the core users of the system. They can add, view, edit, and remove courses, subjects, faculty, students, and events.

Creating new entries is done through dedicated screens. For example, the AdminAddCourseController, AdminAddStudentController, AdminAddFacultyController, and AdminAddSubjectController handle course, student, faculty, and subject creation. These classes collect form inputs, validate the data, and then save it to files like File_Courses.txt or loginTextFile.txt. Adding events (like Ceremonies or workshops) is handled using the AdminAddEventController.

If anything needs to be changed or removed, the admin uses controllers like AdminEditCourseController or AdminDeleteEventController. These allow existing entries to be edited or deleted directly from the interface, which avoids mistakes and keeps the system clean.

There are also list views such as AdminStudentListController, AdminFacultyListController, and AdminSubjectListController. These controllers display all entries in a scrollable view, support searching, and offer quick access to edit/delete options. The AdminCourseDescriptionController works similarly but for courses.

The admin's main screen (AdminMainMenuController) acts as a hub, letting them navigate to all parts of the system. While some features like tuition management (AdminTuitionManagementController) or enrollment control (AdminManageEnrollmentController).

To keep things consistent and secure, changes to files are handled through file-based functions like saving and updating .txt files, while user feedback is given through real-time GUI updates.

Faculty Role – Teaching Management

Faculty members log in and are taken to their own dashboard, managed by FacultyMainMenuController. They can view the courses they've been assigned to using the FacultyCourseManagementController, which reads data from personalized files like [facultyUsername]_teach.txt.

If faculty members need to update their details (such as email or department), they can do so through the FacultyEditProfileController. For simply viewing their data without editing, the system uses FacultyPersonalInformationController.

Each professor can also access student lists for their courses through the FacultyStudentListController. This helps them with tracking and communication.

Student Role – Course Enrollment and Self-Service

Students have their own dashboard (UserMainMenuController), which lets them access all key parts of their profile. They can view all available courses using UserCourseDescriptionController and enroll in courses with a single click. The data is saved to a file like [username]_courses.txt.

Once enrolled, students can use UserEnrolledClassesController to see their current courses. They can also drop courses if needed. Other screens let students view their profile (UserPersonalInformationController), update their information (UserEditProfileController), or check their grades through UserStudentGradesController.

A UserStudentTuitionController, is also there to show tuition balances and payment history. This will help students manage their academic finances directly from the platform.

Login and Access Control

The login process is handled by LoginController, which verifies credentials and routes the user to the right dashboard (Admin, Faculty, or Student). It checks against loginTextFile.txt, where all login data is stored.

In case a user forgets their password, the ForgotPassController allows password recovery by validating their identity and replacing the old password in the file.

To make things easier during onboarding, a utility class called codeEmailGenerator automatically creates strong passwords. This improves security and saves the admin time when registering new users.

Navigation between screens is made smooth by a helper class called SceneLoader, which loads different FXML scenes throughout the application without repeating code.

Old Classes & Testing Tools

During the early stages of development, some classes like HelloApplication, HelloController, and StudentLogin were used to test how the GUI worked and how login logic would function. These are no longer part of the main system but can still be used for basic testing or learning purposes.

System Flow Summary

- Admins manage everything: students, courses, faculty, and events.
- Faculty handle courses they teach and access student lists.
- Students enroll in courses, view grades, and manage profiles.
- All data is stored and updated using plain text files.
- Each screen (controller) handles a specific task and updates the GUI and files accordingly.
- Navigation and login are consistent and role-based.

Key Functions in the System (Core Logic)

To prevent repetitive explaining, only main key functions are shown below which are used in many classes repetitively

- addCourse() / addEvent() / addSubject() o Collect user input from forms o Validate data (e.g. no duplicates or empty fields) o Save new entries to text files (File_Courses.txt, etc.)
- saveChanges() o Used in edit screens to update old data o Replaces existing info in the file with new data o Ensures smooth editing without needing to re-create entries
- deleteCourseFromFile(String courseName) o Removes specific entries from the file based on name or ID o Used for courses, students, events, etc.
    - Keeps system data clean and accurate
- loadCourseList() / loadEventList() / loadStudents() o Reads text files and displays data in GUI format o Creates dynamic lists or panes (like scrollable views) o Automatically updates UI after changes

- handleEditButton() / handleDeleteButton() o Triggered when user clicks an "Edit" or "Delete" button o Opens related form or starts the deletion process

- initialize() o Runs automatically when a screen loads o Sets up buttons, inputs, and default settings o Makes sure the GUI is ready before user interaction
- navigateTo...() (e.g. navigateToCourses()) o Helps move between scenes or views
    - Uses the SceneLoader utility to load new FXML screens
- loadAssignedCourses() / loadEnrolledCourses() o Faculty: sees which courses they're teaching o Students: sees which courses they've enrolled in o Reads from personalized files like username_teach.txt
- generatePassword() o Creates a strong, random password for new accounts o Used during student and faculty registration o Improves security and speeds up onboarding
- handleLogin() o Verifies user email and password
    - Sends user to the correct dashboard (Admin, Faculty, Student)
- resetPassword() o Lets users reset forgotten passwords o Updates login file with the new secure password
- refreshView() / refreshEventView() o Reloads data on the screen after changes are made o Ensures up-to-date info is always shown without restarting

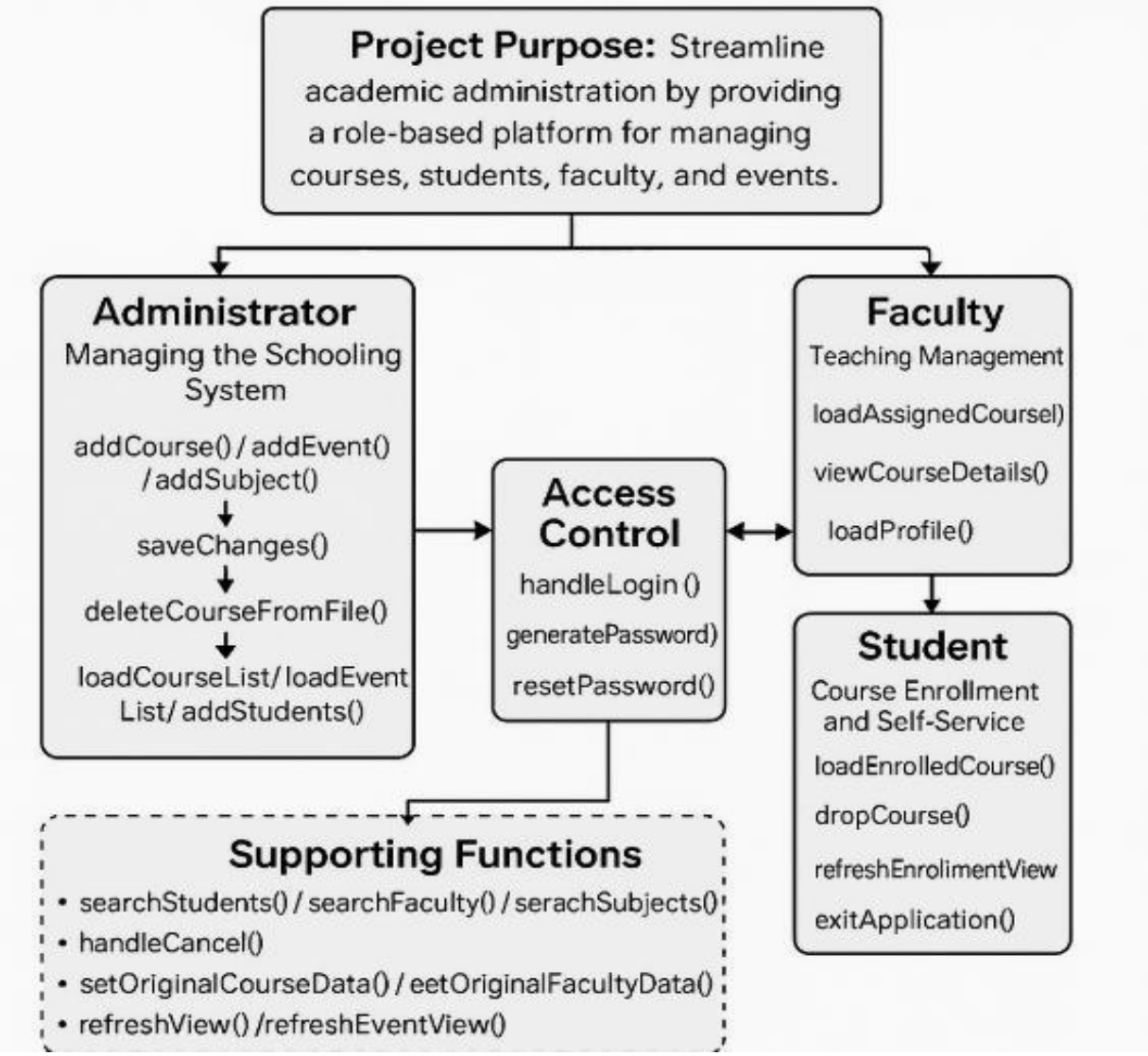
Supporting Functions (Usability & User Experience)

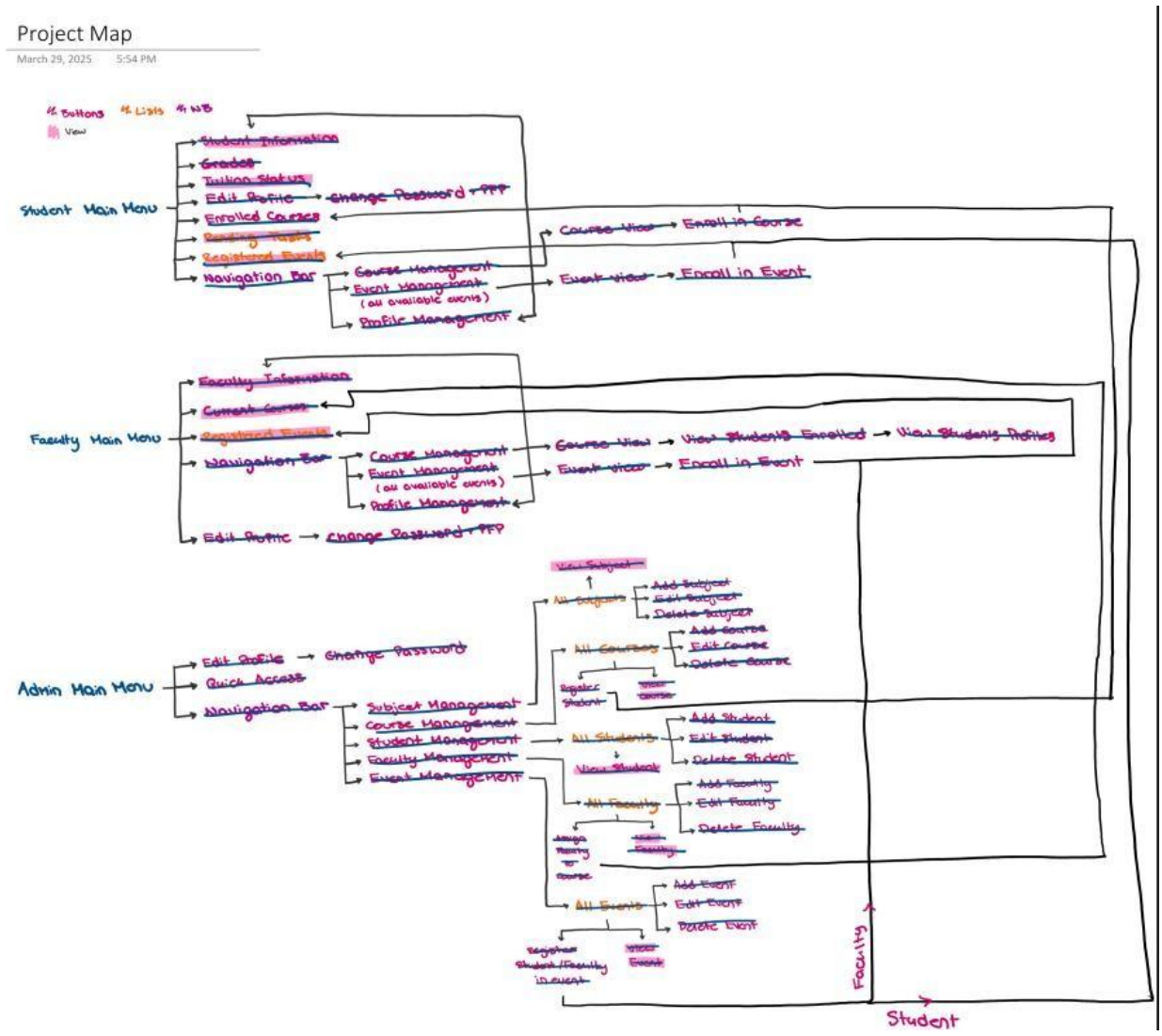
The following classes are repetitive classes through many of the controller classes as these classes below all the UI Experience to be smooth and user-friendly

- searchStudents(query) / searchFaculty(query) / searchSubjects(query) o Filters lists based on user input
    - o Helps quickly find people or subjects in large lists
- dropCourse(courseID) o Allows students to remove a course they're enrolled in o Updates their personal course file
- viewCourseDetails(courseID) / viewStudentProfile(email) o Lets faculty view full info about a course or student o Supports better academic tracking and communication
- viewPaymentHistory() (Planned) o Will show tuition records and payment details o Helps students stay on top of their fees

- handleCancel() o Returns to the previous screen without saving o Used in most forms and edit views
- setOriginalCourseData() / setOriginalFacultyData() o Loads existing info into a form before editing o Makes editing easier and prevents overwriting wrong data
- loadProfile() / loadPersonalInfo() o Displays current user info in read-only mode o Used in profile view screens
- refreshEnrollmentView() o Reloads the list of enrolled students or courses after updates
- exitApplication() o Cleanly shuts down the system from the login screen

VISUAL REPRESENTATIONS ~ (CLASSES RELATIONSHIPS + OOP CHECKLIST)




Project Map
March 29, 2025 5:54 PM
Buttons
Lists
DB
View
Student Main Menu
Student Information
Grades
Tuition Status
Edit Profile
Change Password + PFP
Enrolled Courses
Registered Events
Navigation Bar
Course Management
Event Management
(all available events)
Profile Management
Course View
Enroll in Course
Event View
Enroll in Event
Faculty Main Menu
Faculty Information
Current Courses
Registered Events
Navigation Bar
Course Management
Event Management
(all available events)
Profile Management
Course View
View Students Enrolled
View Students Profiles
Event View
Enroll in Event
Edit Profile
Change Password + PFP
Admin Main Menu
Edit Profile
Change Password
Quick Access
Navigation Bar
Subject Management
Course Management
Student Management
Faculty Management
Event Management
View Subjects
All Subjects
Add Subject
Edit Subject
Delete Subject
All Courses
Add Course
Edit Course
Delete Course
Register Student
View Course
All Students
Add Student
Edit Student
Delete Student
View Student
All Faculty
Add Faculty
Edit Faculty
Delete Faculty
Assign Faculty to Course
View Faculty
All Events
Add Event
Edit Event
Delete Event
Register Student/Faculty in event
View Event
Faculty
Student