

FACULTY Exact sciences

"Computer Science Department"



PROJECT REPORT

University Academic Platform

STUDENT CONTRIBUTORS

- DIFALLAH AKRAM
- SELLAMI MOHAMED ODAY

Academic Platform

Introduction

Brief Overview

This project presents a comprehensive and scalable **Academic Platform** tailored for modern university environments. Designed to streamline both teaching and learning processes, it integrates key functionalities such as course management, assessments, and secure user profiling. Drawing inspiration from established systems like Moodle, this platform introduces an intuitive interface and supports role-based access for students and instructors alike.

Among its primary features are the ability to manage quizzes, assignments, practical work submissions, and collaborative tools, all within a unified interface. Users are empowered to securely view and update their profiles, ensuring data privacy and personalization.

Security and data integrity are ensured through hashed password authentication and a role-verification mechanism based on a pre-approved matricule list. At its core, the system is backed by a relational database, enabling efficient storage and retrieval of academic records, user profiles, and course content.

The result is a functional, extensible, and user-focused academic environment that promotes engagement, consistency, and streamlined workflows between all educational stakeholders.

Outline of the Report

This report provides a detailed overview of the development process, design considerations, and technical solutions implemented. It is structured as follows:

- **Analysis and Specifications:** Defines the core functional and non-functional requirements, complemented by a comprehensive Use Case Diagram that models system interactions and user flows.
- **Software Design:** Explores the overall software architecture through a Package Diagram and supporting UML models including Class, Sequence, and State Machine diagrams. This section emphasizes system modularity and scalability.
- **Implementation:** Documents the chosen technology stack, including programming languages, backend frameworks, libraries, and APIs. It also highlights implementation strategies and development best practices.
- **User Guide:** Offers an illustrated walkthrough of the platform's features with annotated screenshots to ensure ease of adoption for end users, focusing on usability and accessibility.
- **Conclusion and Perspectives:** Summarizes the project outcomes, discusses encountered limitations, and proposes future directions to enhance platform functionality, scalability, and user satisfaction.

Analysis and Specifications

Functional Requirements

Role	Action	Description
Student	Register	Create an account using valid university credentials.
Student	Login	Authenticate using matricule and password.
Student	View	Access available courses, quizzes, exercises, and practical works.
Student	Submit	Submit completed quizzes and practical works.
Student	Search	Search by course, quiz, exercise, or practical work.
Student	Profile View/Edit	View and modify personal profile information.
Student	Logout	Safely exit the platform.
Teacher	Register	Sign up using verified institutional credentials.
Teacher	Login	Access teaching dashboard via login.
Teacher	Manage Courses	Add, edit, or delete course materials.
Teacher	Manage Exercises	Add, edit, or delete exercises.
Teacher	Manage Quizzes	Add, edit, or delete quizzes.
Teacher	Manage Practical Works	Add practical works and view student submissions.
Teacher	Profile View/Edit	View and update personal profile.
Teacher	Logout	Securely log out of the system.

Table 1: Functional Requirements by User Role

Use Case Diagram:

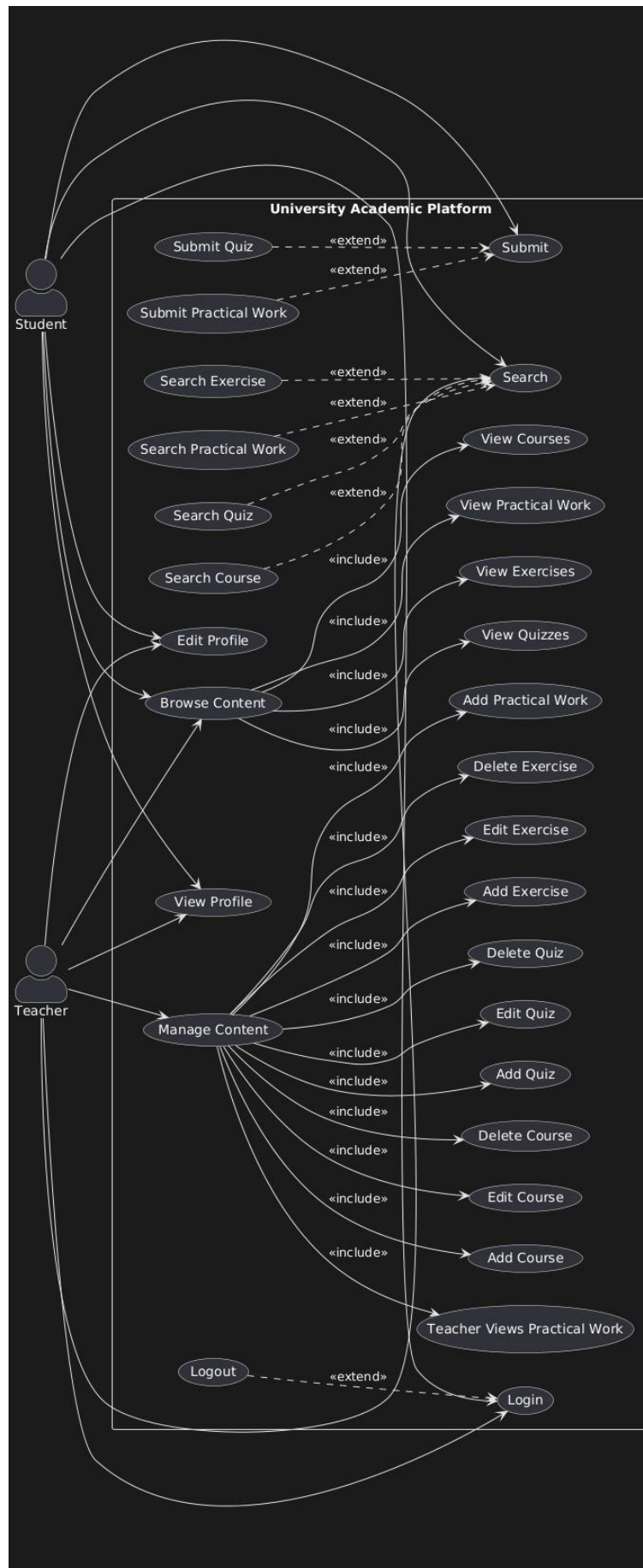


Figure 1: Use case diagram ***Non-Functional Requirements***

In addition to the core functionalities, the Academic Platform adheres to several non-functional requirements that ensure its performance, reliability, and user satisfaction. These requirements define how the system operates rather than what it does.

Performance

The platform is optimized to support multiple concurrent users (teachers and students) with no noticeable lag or delay. Database operations, including content loading and submission handling, are optimized for quick execution (under 2 seconds in normal conditions).

Security

All user passwords are securely hashed using a strong hashing algorithm before being stored. Authentication and authorization mechanisms are implemented to restrict access to valid users only. Role-based access ensures that users can only perform actions specific to their permissions (e.g., students cannot add content). The system uses parameterized SQL queries (PreparedStatement) to protect against SQL injection attacks.

Usability

The user interface is designed to be intuitive and user-friendly, supporting smooth navigation and interaction. Features are logically categorized (e.g., learning content, profile management, submissions), enhancing the overall user experience.

Maintainability

The code is modular and follows a clean structure, using layered architecture (models, services, utilities). Java naming conventions and consistent formatting practices are followed for better readability and collaboration. Exception handling and logging are in place for easier debugging and future maintenance.

Scalability

The database schema is normalized and extensible, allowing easy addition of new features (e.g., new user roles or content types). The application structure allows scaling both horizontally (adding more instances) and vertically (enhancing features).

Reliability

The platform is designed for stability and consistency, ensuring that user data is not lost or corrupted. Input validation and exception management contribute to predictable and stable system behavior.

Portability

Developed in Java, the application can be deployed on any environment that supports a Java Runtime Environment (JRE). The system does not rely on platform-specific features, enhancing its adaptability across different operating systems.

Data Integrity

Registration is restricted to users listed in a pre-validated database (ValidID), ensuring authorized access. All user actions are validated and verified against system constraints to preserve data consistency.

Accessibility

The interface design follows accessibility best practices, including readable fonts, clear layout structures, and keyboard navigability.

Logging and Debugging

System actions such as login attempts, registration, and database operations are logged for monitoring and debugging. Error messages are descriptive and help guide users in resolving issues where applicable.

Backup and Recovery

The system uses a persistent relational database backend that can be periodically exported and backed up. Recovery mechanisms can be established through standard database restoration practices.

Software Design

This section illustrates the global architecture and dynamic behavior of the system using standard UML diagrams. It includes the Package Diagram for high-level structure, and Class, Sequence, and State Machine Diagrams for detailed component interactions.

Global Architecture: Package Diagram

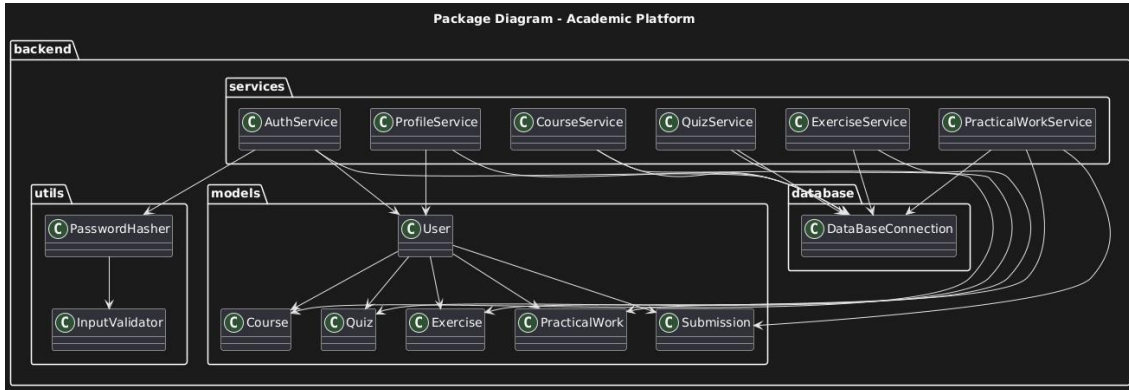


Figure 2: Global Software Architecture – Package Diagram

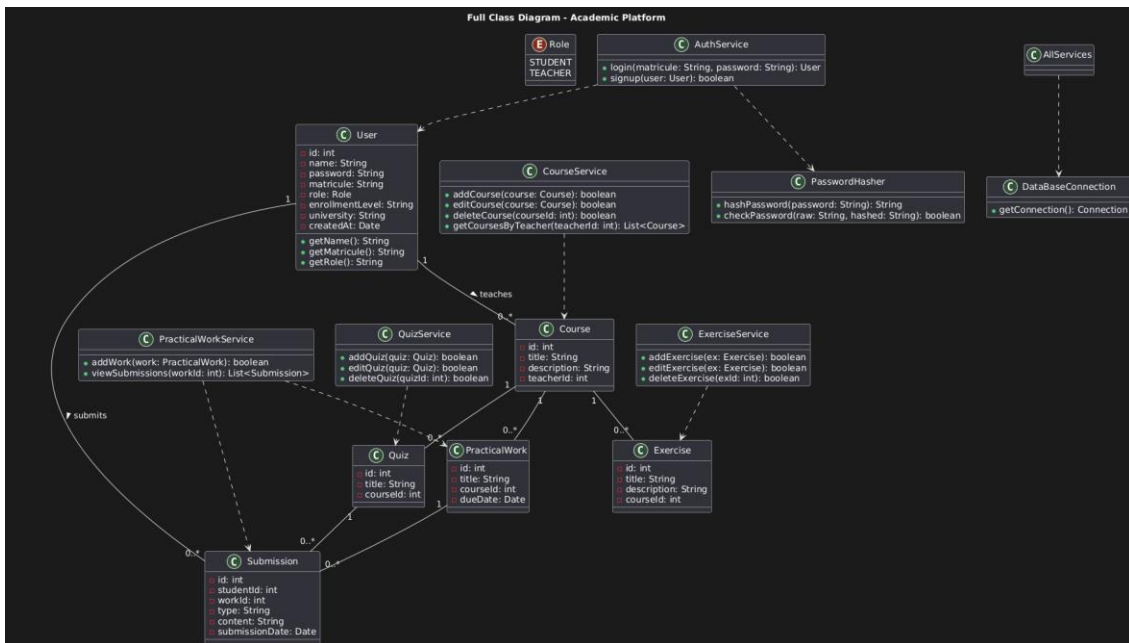


Figure 3: UML Class Diagram

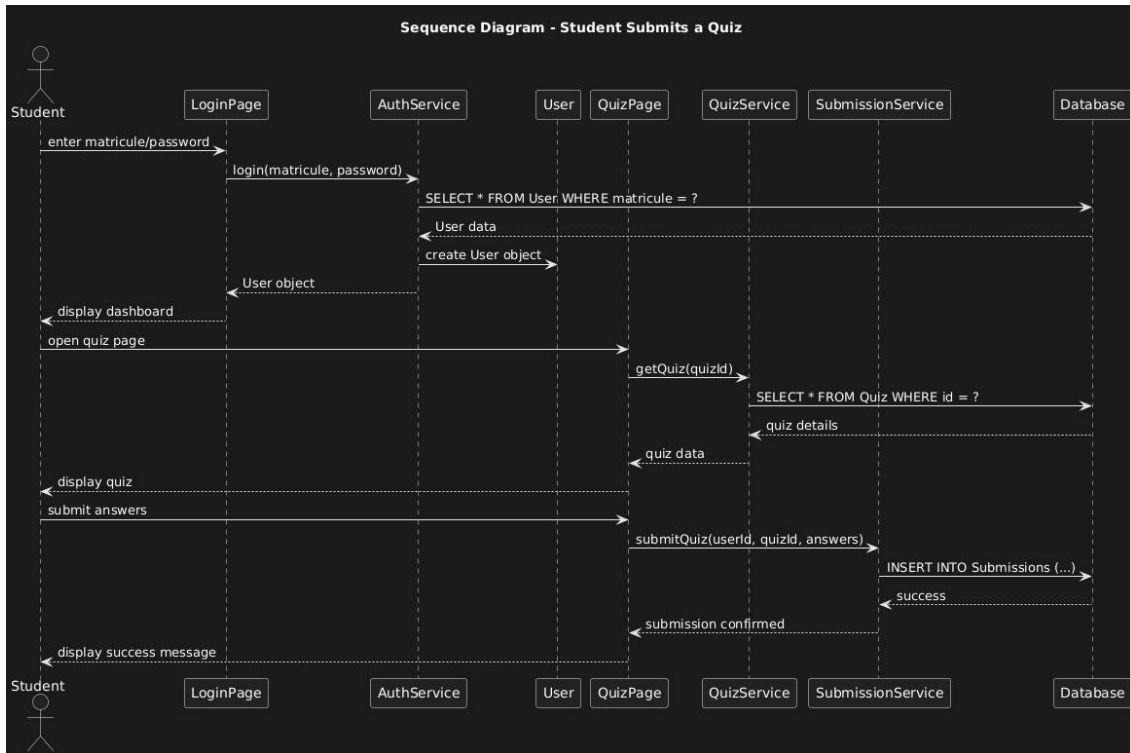


Figure 4: UML Sequence Diagram

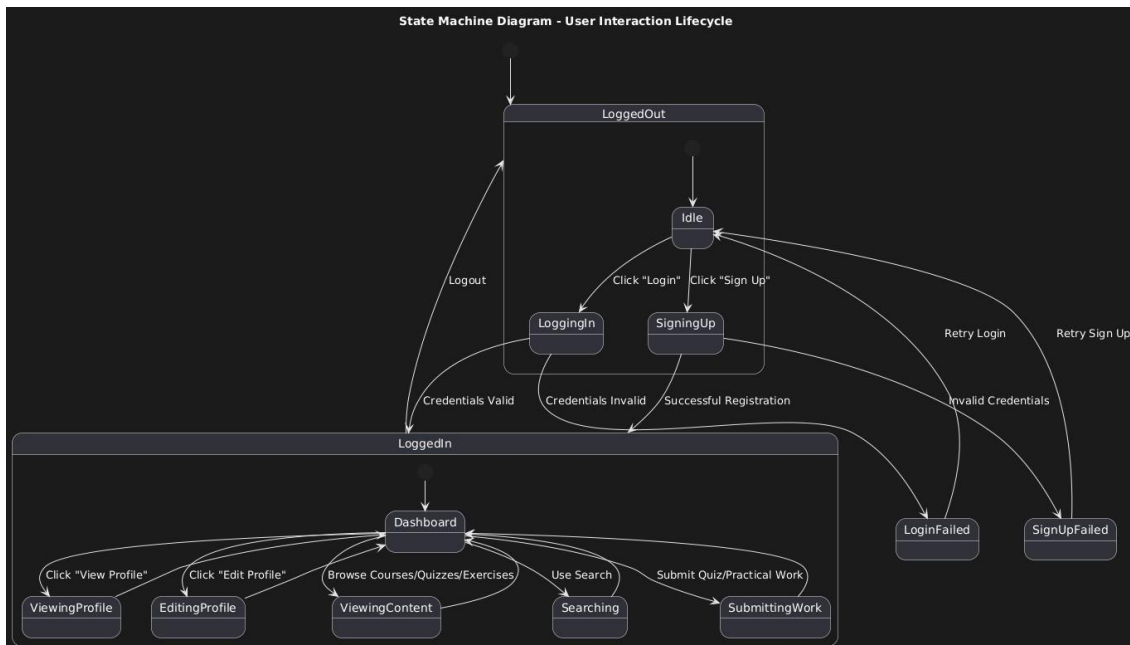


Figure 5: UML State Machine Diagram

5. Implementation

Technologies Used

The development of the Academic Platform was carried out using a combination of modern technologies and tools, ensuring modularity, scalability, and maintainability. Below is a summary of the technologies employed across different layers of the application:

Layer	Technology / Tool	Description
Frontend	JavaFX (FXML)	Used for building the graphical user interface. FXML provided a clean separation between design and logic.
Frontend	SceneBuilder	Assisted in designing and previewing the JavaFX interface visually.
Backend	Java (JDK 17+)	Core programming language used for logic implementation.
Backend	JDBC	Used to connect and interact with the relational database (SQL).
Backend	Custom Authentication Service	Developed for handling login, signup, and password hashing using BCrypt.
Models	POJOs	Represented key entities such as User, Course, Quiz, Exercise, and Practical Work.
Utilities	PasswordHasher Utility (BCrypt)	Ensured secure password storage and verification.
Database	MySQL / SQLite	Relational database used to store users, roles, courses, quizzes, etc.
Version Control	Git	Used for source code management and collaboration.
Build Tool	Maven / Manual Compilation	Managed project structure and dependencies (optional depending on setup).

Table 2: Summary of Technologies Used in the Academic Platform

Architecture Highlights

The system adopts an MVC-like structure:

- **Model:** Java classes representing application data.
- **View:** FXML files rendered by JavaFX.
- **Controller:** Service classes such as AuthService, UserService, etc.

Secure Authentication: Passwords are hashed before storage, and user roles are verified against a valid matricule list.

Modular Codebase: Structured using packages such as models, services, utils, and database to ensure a clean separation of concerns.

6. User Guide

This section provides visual guidance on the key functionalities of the Academic Platform. The screenshots below illustrate various user interactions, including login, dashboard access, course management, and submission handling.

screenshot_login.png

Figure 6: Login Interface – Secure access via matricule and password

screenshot_dashboard.png

Figure 7: Student Dashboard – Overview of enrolled courses and upcoming tasks

screenshot_course_management.png

Figure 8: Teacher Panel – Course creation and content upload interface

screenshot_submission.png

Figure 9: Assignment Submission –Student view for uploading answers and tracking status

7. Conclusion and Perspectives

Project Review

This Academic Platform project successfully delivered a comprehensive and secure university-level learning management system. The platform integrates core functionalities such as user authentication, course management, quizzes, exercises, practical work submissions, and profile management for both students and teachers. Throughout development, emphasis was placed on security (including password hashing and role validation), usability, and modular design. The system facilitates efficient communication and collaboration between students and teachers, supporting a broad range of academic activities. The platform's intuitive interface built with JavaFX and robust backend services implemented in Java have demonstrated reliability and ease of use.

Limitations

Despite the successful implementation of many key features, the project has some limitations. Currently, the system lacks a dedicated grading module, which limits the ability of teachers to evaluate student performance within the platform itself. Additionally, while core search and profile management functionalities are present, advanced analytics and reporting capabilities are not yet integrated. The platform also assumes a static list of valid matricules and roles, which might require manual updating in a real-world environment. Furthermore, the user interface, though functional, could benefit from enhanced responsiveness and accessibility features.

Future Improvements

To further enhance the platform's usefulness and completeness, future work will focus on adding grade management capabilities. This includes allowing teachers to submit grades, view individual student grades, and calculate average grades for courses and assessments. Additional improvements could incorporate more sophisticated user role management, real-time notifications, and integration with external academic systems. Enhancing the UI/UX to support mobile devices and providing richer analytics and reporting tools will also be priorities to improve user engagement and administrative oversight.