Student Management System Plus

Final Project Report

Course: Object Oriented Programming Using Java

Student Name: Iddrissu Kamaldeen

Student ID: 01220191B

Department: Engineering Department

Submission Date: 26th February 2026

Table of Contents

1. Introduction and Problem Statement

Many academic departments still rely on spreadsheets to manage student records. This approach leads to serious issues such as duplicate student IDs, missing data, inconsistent reporting, and poor decision-making.

To solve this, I developed Student Management System Plus — a complete offline desktop application built with JavaFX, SQLite, and JDBC. The system provides a professional, user-friendly interface for adding, viewing, updating, deleting, searching, filtering, and reporting on student records. All data is stored locally in an SQLite database, ensuring full offline functionality on Windows machines.

2. Requirements Analysis

Functional Requirements

- Full CRUD operations on students
- Search by ID or name
- Advanced filtering (programme, level, status)
- Sorting by GPA and name
- Reports: Top Performers, At-Risk Students, GPA Distribution, Programme Summary
- Import from CSV with validation and error logging
- Export to CSV (full list + individual reports)
- Dashboard with live statistics

Data Validation Rules

- Student ID: 4–20 alphanumeric characters, unique
- Full Name: 2–60 characters, no digits
- Programme: required
- Level: 100, 200, 300, 400, 500, 600, 700
- GPA: 0.0 – 4.0
- Email: must contain @ and .
- Phone: 10–15 digits only

Technical Requirements

- Java + Maven build
- JavaFX GUI
- SQLite with JDBC (prepared statements only)
- Layered architecture (UI → Service → Repository)
- Unit testing with JUnit 5 + Maven

3. System Design and Architecture

The project follows a clean layered architecture:

UI Layer (JavaFX Controllers + FXML)
Service Layer (Business logic, validation, reporting)
Repository Layer (SQLite operations via JDBC)

Domain Layer (Student entity)
Util Layer (logging, file handling, ServiceLocator)

Database schema includes PRIMARY KEY on StudentID, NOT NULL, and CHECK constraints for Level and GPA.

4. Implementation Highlights

- MainController: Manages navigation between screens
- DashboardController: Real-time statistics
- StudentsController: Table, search, filters, add/edit form, delete confirmation
- ReportsController: Tabbed reports with filtering and export
- ImportExportController: CSV import with error report
- SettingsController: Configurable At-Risk GPA threshold
- Logging stored in data/app.log

5. Screenshots of the Application

Figure 1: Dashboard Screen

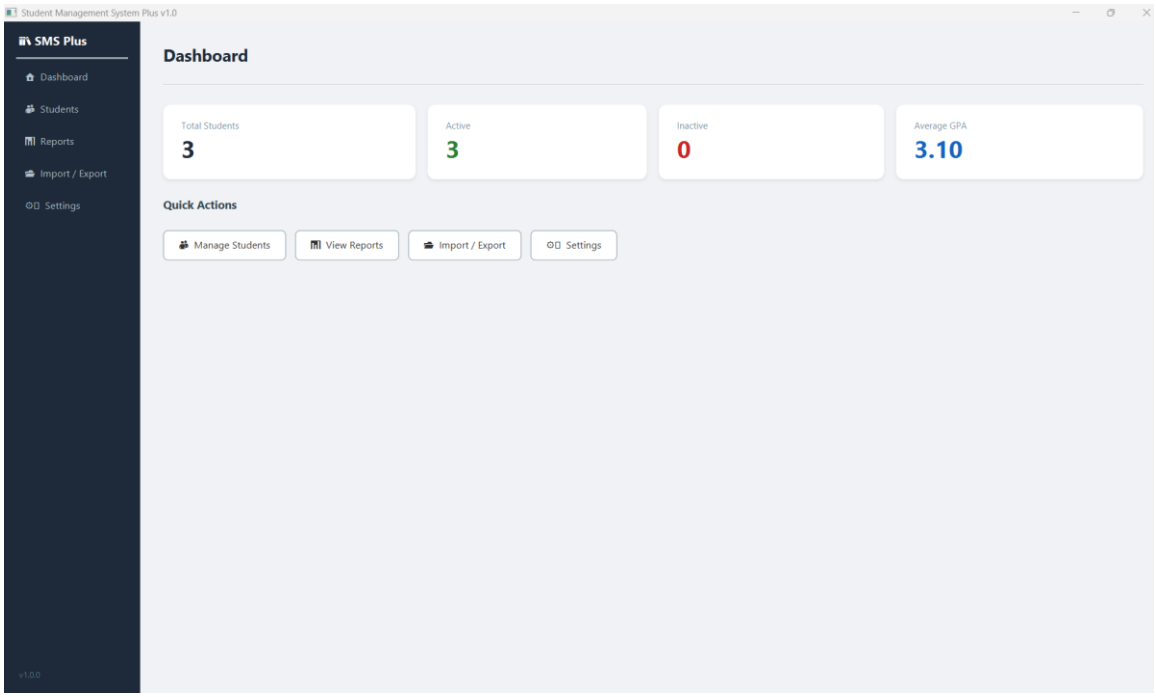


Figure 2: Students Screen

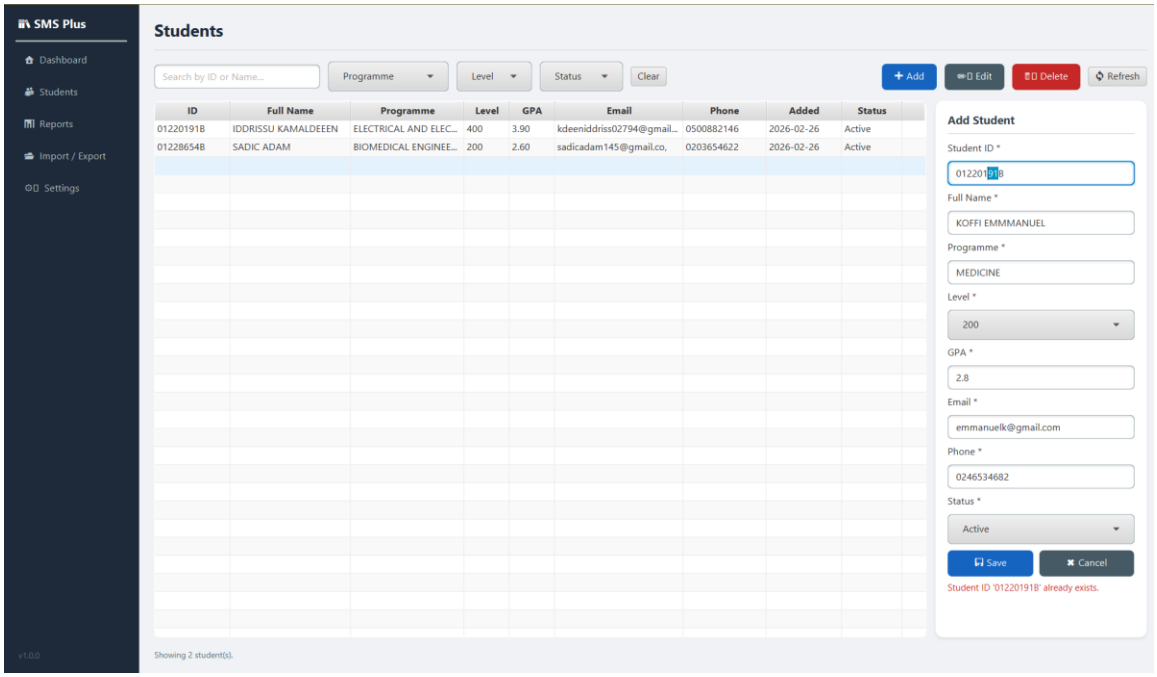


Figure 3: Reports Screen

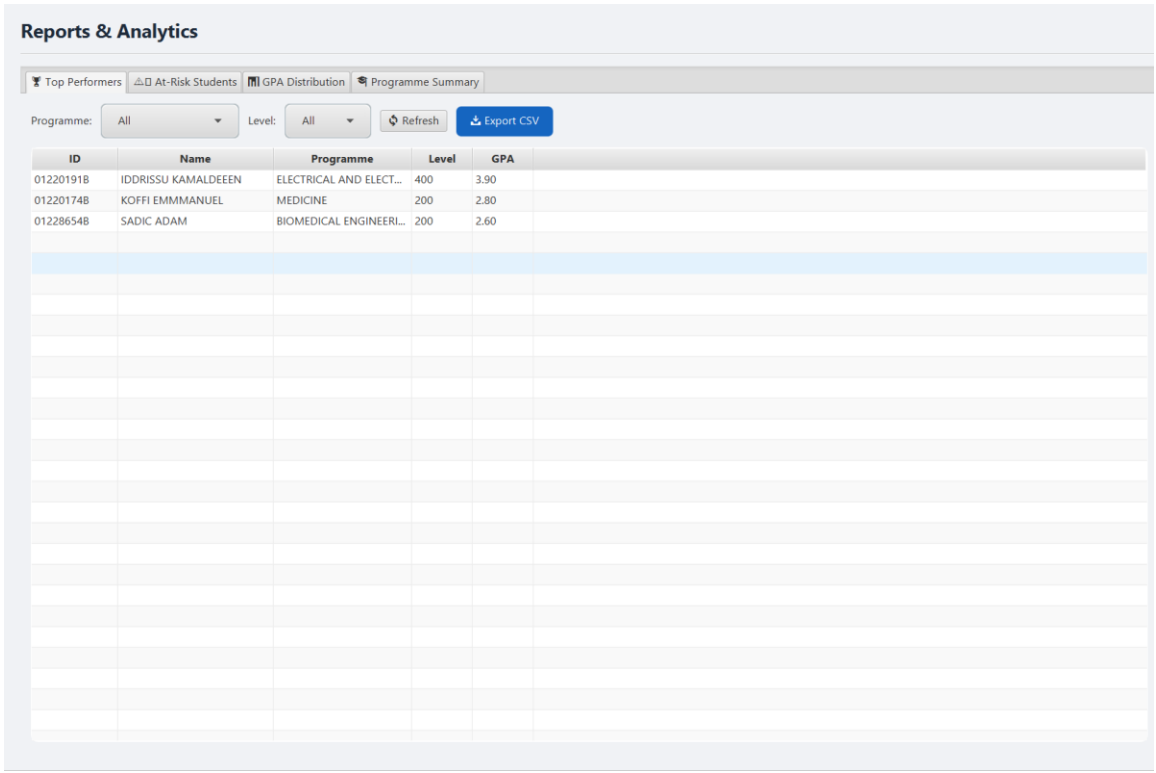


Figure 4: Import/Export Screen

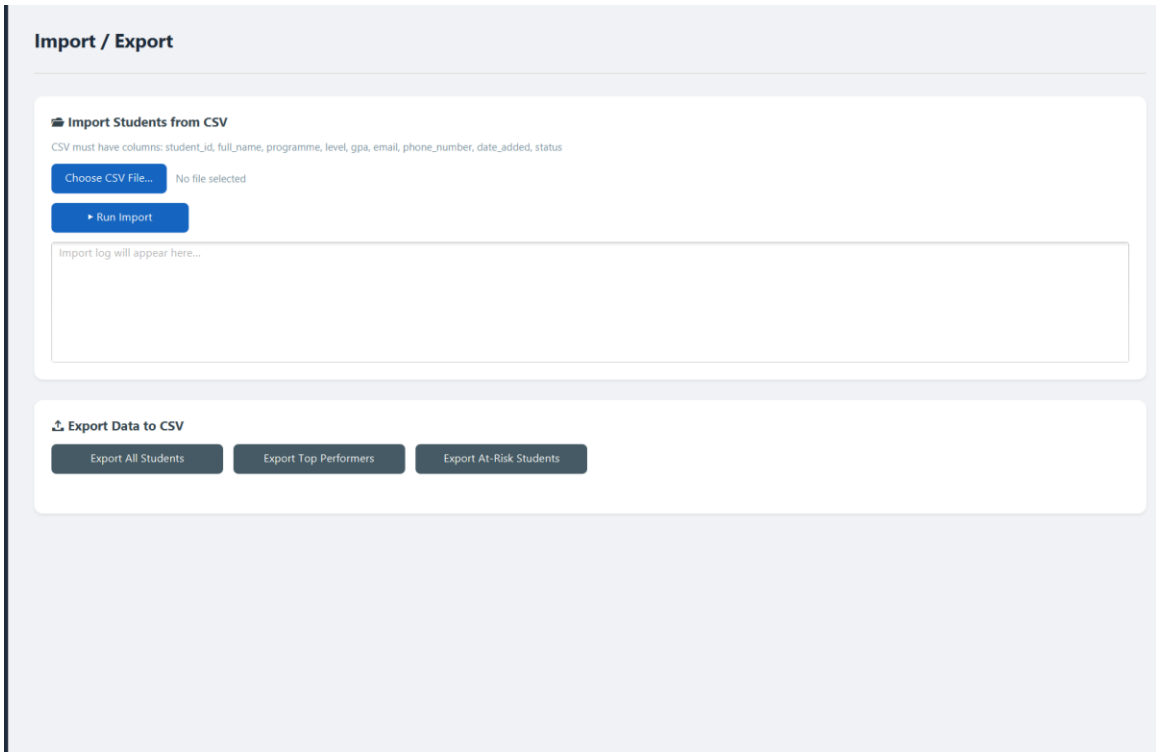
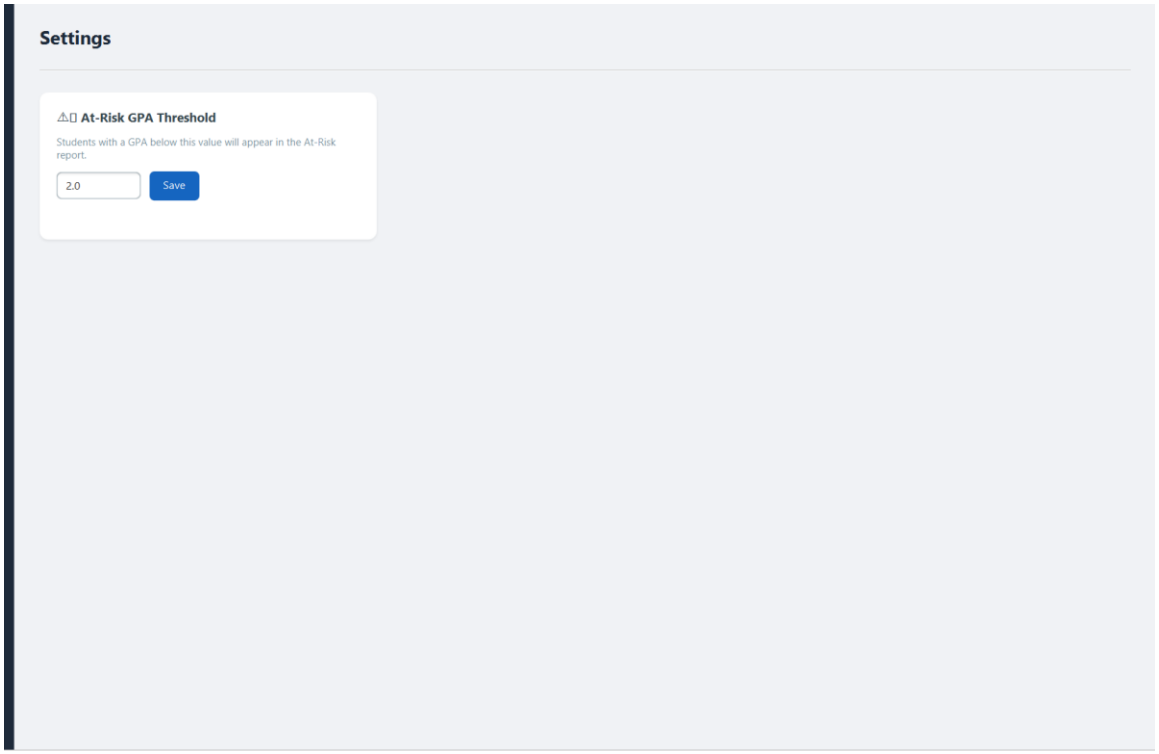


Figure 5: Settings Screen



(Actual screenshots to be inserted in final version)

6. Testing and Quality Assurance

- 12+ Unit Tests written and passing
- Student validation tests
- Service layer business rule tests
- Report calculation tests
- Repository CRUD tests
- Import validation tests

Command used: mvn test

7. Security and Data Integrity Measures

- All database operations use PreparedStatement
- CHECK constraints for level and GPA
- Input validation in UI and Service layers
- Duplicate Student ID prevention
- Confirmation dialogs for delete operations

8. Deployment and Run Instructions

For Windows users:

1. Extract the release zip
2. Double-click RUN.bat
3. If needed, use VM options:

--module-path "C:\Program Files\javafx\javafx-sdk-25.0.2\lib"
--add-modules javafx.controls,javafx.fxml,javafx.graphics,javafx.base

JDK Version Used: 21

9. HOW I RUN MY CODE

1. Using cmd prompt to run program

2. To run the command written is used:

C:\Users\tln\Downloads\StudentManagementSystemPlus\STUDENT-MANAGEMENT-SYSTEM>mvn javafx:run

10. Limitations and Future Improvements

Current Limitations

- No user authentication
- No photo upload
- No PDF export

Future Enhancements

- Add login system

- Dark mode support
- PDF report generation
- Backup and restore feature

11. Conclusion

This project demonstrates Object-Oriented Programming principles, layered architecture, JavaFX GUI development, secure database handling with SQLite, and professional software engineering practices.

The system is robust, user-friendly, and meets all assignment requirements.

12. References and AI Disclosure

References

1. OpenJFX Documentation
2. SQLite JDBC Driver Documentation
3. Maven Official Documentation

Date: 25 February 2026