UNIVERSITETI METROPOLITAN TIRANA

FACULTY OF COMPUTER SCIENCES AND IT

Software Engineering Program

Object Oriented Programming

Final Project Documentation

Student Management System

Worked by: Krisalda Mihali

1. Introduction

The **Student Management System (SMS)** is a software application designed to help educational institutions efficiently manage student data, courses, and academic performance. The system integrates user management features such as secure login and registration, along with comprehensive management functionalities for student records, course enrollments, and grades. Built with **JavaFX** for the graphical user interface (GUI) and **MySQL** for database, it supports essential operations like adding, updating, and deleting student and course information. The system is designed using key object-oriented programming (OOP) principles such as **abstraction**, **encapsulation**, **polymorphism**, and **inheritance** to promote maintainability and scalability.

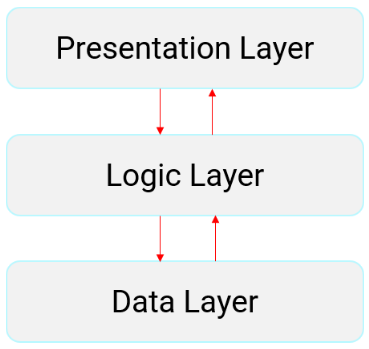

Fig.1. Key Architecture

2. Key Features

The system includes several core functionalities that serve the needs of administrators:

- **User Authentication (Login/Register):** Users can securely log in or register within the system, ensuring that only authorized users can access and manage data.
- **Dashboard:** The dashboard provides an overview and easy access to various system functionalities, including student management, course management, and grade management. Additionally, the dashboard displays student statistics retrieved directly from the database. These statistics are visualized using various types of charts.
- **Student Management:** Administrators can manage student records, including adding, editing, and deleting students. The system also supports viewing student details, such as name, gender, enrollment date, and contact information.
- **Course Management:** The system allows administrators to manage courses by adding new courses, updating course details (e.g., course name, department, instructor), and deleting courses. It also provides an overview of available courses.
- **Grade Management:** The grade management feature allows administrators to input, update or delete grades for students in specific courses, providing a means of tracking academic progress.

- **Graphical Data Visualizations:** The system includes charts and tables to visualize key data points, such as the total number of student enrollments, gender distribution, student data, courses and grades tables.
- **JavaFX Interface:** The application leverages **JavaFX** for creating a responsive and user-friendly interface, using components such as **TableView** for displaying tables of students and courses, **TextField** for input forms, and **Buttons** for executing various operations.

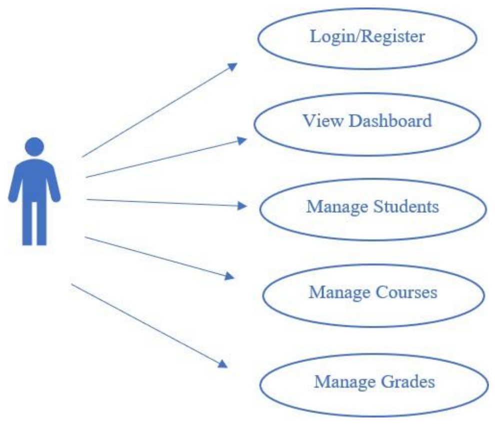

Fig.2. Architecture Details

3. Technical Implementation

The **Student Management System** is built using a combination of **JavaFX** for the GUI and **MySQL** for database, with the system's architecture based on the principles of **Object-Oriented Programming (OOP)**:

- **Abstraction:** Abstraction: The system uses abstraction to separate complex tasks into simpler components, like the `AbstractButtonAction` class for button actions and `BasePage` for common page elements, promoting code reusability and easier maintenance.
- **Encapsulation:** Information such as student names, course IDs, and grades is encapsulated within their respective classes, ensuring that sensitive data is only accessible via getter and setter methods, providing a layer of security.
- **Inheritance:** Pages such as `DashboardPage`, and `AddStudentPage` inherit from a common `BasePage` class, reducing redundancy and increasing code reusability.
- **Polymorphism:** Methods are customized through method overriding to handle different actions based on the context.

- **Database Design:**
    - **MySQL Database:** The system operates with a MySQL database, facilitating the storage and management of various data components.
    - **CRUD Operations:** Each page performs basic CRUD (Create, Read, Update, Delete) operations for students, courses, and grades. For example, when adding a course, the system inserts a new record into the `courses` table using a parameterized SQL query to prevent SQL injection.
- **Core Components and Classes:**
    - **StudentManagementSystem (Main Class):** The entry point for the application, responsible for launching the GUI and managing the primary window and scene transitions.
    - **LoginPage & RegisterPage:** Handle user authentication by providing forms for login and registration. The login page checks user credentials against the database, while the registration page allows new users to sign up.
    - **DashboardPage:** Acts as the central hub of the application, displaying key student data such as the total number of enrolled students, gender-wise breakdown, and trends over time using charts.
    - **AvailableCoursesPage:** Manage courses in the Student Management System.
    - **AddStudentPage:** Provides a form for adding , deleting or updating new students to the database, with fields such as name, gender, enrollment year, and department.
    - **GradesPage:** Allows administrators to input , update or delete grades for students in specific courses, providing a means to track academic progress.
- **JavaFX Design:**
    - **User Interface:** The GUI is developed using JavaFX, utilizing a variety of components such as **TextField**, **Button**, **TableView**, and **GridPane**. For example, the course management page uses a **TableView** to display courses in a tabular format, with options to add, update, or delete records.
    - **Events and Actions:** Event handlers are attached to buttons (e.g., **Add**, **Clear**, **Delete**, **Update**) to trigger appropriate actions, such as adding or deleting students or courses, or clearing form fields.

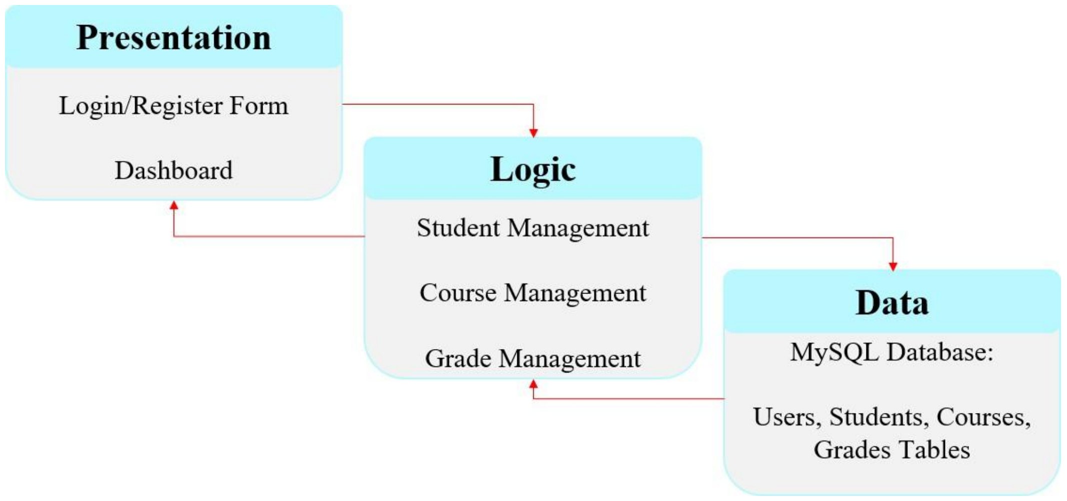

Fig.3. Architecture Details

4. Data Handling

The system relies on a MySQL database to store and retrieve data. The database schema is organized into several tables:

- **Students Table:** Stores personal information such as student ID, name, gender, contact details, and enrollment status.
- **Courses Table:** Contains course-specific data such as course ID, name, department, and instructor.
- **Grades Table:** Keeps track of students' academic performance by linking students with their respective courses and grades.
- **User**: Manages user authentication with username, email, and password.

SQL queries are written to perform various database operations. For example, when adding a new student, a `INSERT` statement is executed to insert the student's data into the `students` table. Similarly, updates to student or course information are performed using `UPDATE` statements, and deletions use `DELETE` statements. These operations ensure that data is handled efficiently and securely.

5. Conclusion

The **Student Management System** is a robust solution for managing student records, courses, and grades. Its modular design, based on **Object-Oriented Programming (OOP)** principles, allows for efficient maintenance and future expansion. By integrating **JavaFX** for the front-end and **MySQL** for the back-end, the system ensures both user-friendly interactions and secure data handling.

Future improvements could include:

- **Mobile Compatibility:** Adapting the system for mobile devices to increase accessibility for students and administrators.
- **Additional Features:** Implementing advanced features such as course scheduling, notifications, and detailed reporting for better academic tracking.
- **Enhanced Security:** Strengthening security by adding features like multi-factor authentication (MFA) and encryption for sensitive data.

This system serves as a strong foundation for educational institutions seeking to streamline administrative processes and enhance student management.