Student Management System (SMS)

Project Team: Nikita Karim, Jakub Olsovsky, Daniel Espin, Vadim Cheremukhin, Ronan Banton, Jason Siecienski, Lucas Vas, and Marcos Santiago

Professor Coffman
Thursday, 27 October 2024
Western Connecticut State University
CS350 - Object Oriented Software Engineering

I. Problem Statement

The **Student Management System (SMS)** aims to streamline student data management by offering a user-friendly interface to add, retrieve, update, search, and delete student records. It ensures both integrity and performance, addressing the limitations of manual systems in educational institutions.

Educational institutions often face challenges with outdated or inefficient student information systems, resulting in time-consuming processes and a high potential for errors. The SMS offers fast access through **advanced search capabilities**, minimizing the time spent on record management.

The primary goal of the system is to provide administrators and faculty with a reliable platform for managing student information, ensuring smooth institutional operations. The SMS supports essential functionalities, such as quick search access, performance tracking (GPA and declared majors), and efficient record updates while ensuring data accuracy and preventing redundancy.

Core Objectives:

- Improve the speed and accuracy of retrieving student records.
- Provide **secure** and **organized management** of student data, reducing errors.
- Ensure the system is scalable and flexible to handle increasing student populations.

II. System Requirements

Functional Requirements

1. User Login

The system shall display a login page requiring username and password.
It must authenticate users based on credentials stored in a secure database.

- If valid: Grant access
- If invalid: Display an error message

2. Add Student Records

The system shall provide a form to input student information, including:

- Student ID (unique identifier)

- Name
- Address
- Phone Number
- Email
- Date of Birth
- Major
- GPA

The system shall validate input fields to ensure:

- No fields are left empty
- Email contains the "@" symbol
- GPA is between 0.0 and 4.0

The system shall check that the Student ID is unique before saving.

- If duplicate ID: Display an error message indicating the conflict

Upon successful validation, the system shall save the new student record to the database.

3. Search for Student Records

Provide a search bar to locate records by:

- Student ID
- Student Name (partial or full match)

Display a list of matching student records.

- If no matches: Show "No student records found"

Users can select a student from the search results to view detailed information.

4. Retrieve Student Records

- Display Information:
    - Show the following details:
        - Student ID
        - Name
        - Address
        - Phone Number
        - Email
        - Date of Birth
        - Major
        - GPA

- Real-Time Retrieval: Ensure accurate data is fetched from the database without delay.

5. Update Student Records

Users shall be able to update specific fields, such as:

- Address
- Phone Number
- Email
- Major
- GPA

An "Edit" button shall allow modifications.
The system shall validate the updated data before saving changes to the database.
If no changes are made, users can cancel updates and return to view mode.

6. Delete Student Records

The system shall provide an option to delete student records.

- Ask for confirmation before permanent deletion
- If confirmed: Delete the record and display a success message
- If canceled: Retain the record

Non-Functional Requirements

1. Performance

The system should load and display records within 2-3 seconds for up to 1000 students.
Search results should return within 3-5 seconds for datasets up to 5000 records.

2. Scalability

The system should support up to 10,000 student records without slowdown.
Database design shall focus on essential features, with room for enhancements.

3. Usability

The system should provide a simple and intuitive interface with easy navigation.
Basic tasks (e.g., add or search) should be completed within 3 clicks from the main page.
Design should be clear and uncluttered, using free or basic tools.

4. Reliability

The system should be stable, with retry mechanisms for minor failures.
Data should be saved immediately to prevent accidental loss.

5. Security

The system shall ensure unique Student IDs and validate fields (e.g., valid email format, GPA range).
Basic form validation shall prevent corrupt or invalid data.

6. Maintainability

The system shall be well-structured and documented for easy modification.
Use free version control tools (e.g., GitHub) for collaboration.

7. Portability

The system should run locally.

8. Data Integrity

The system shall ensure Student IDs are unique and prevent saving records if required fields are missing.
It shall handle errors (e.g., failed database connections) with user-friendly messages.

9. Flexibility

The system shall allow small feature additions but focus on core functionality.