Project Title: Student Information Management System

Author: Shakib Al Hasan

Student ID: 2021521460124

Institution: Sichuan University

Submission Date: June 27, 2024

Abstract

This project focuses on developing a Student Information Management System using Python and SQLite. The primary objective is to create a system that efficiently manages various aspects of student information, including course selection, teacher information, book purchases etc. The system incorporates CRUD operations to allow the addition, update, deletion, and querying of student records. Data analysis is performed using custom modules to provide insights into gender ratios, enrollment year distributions, and major distributions. The project was implemented in VS Code and successfully met all specified requirements, providing a robust and user-friendly solution for managing student information. The implementation utilized several libraries, including sqlite3 for database operations, random for generating sample data, pprint for readable data display, debugpy for debugging, and tkinter for creating the user interface.

Acknowledgment

I wish to extend my deepest gratitude to everyone who supported and guided me throughout the development of the Student Information Management System. Firstly, my sincere thanks go to my supervisors, Professor Chuan Li and Professor Mingjie Tang, whose invaluable guidance, insightful advice, and continuous encouragement have been essential to the success of this project.I am also grateful to my professors at Sichuan University, whose courses provided the essential knowledge and skills that have significantly contributed to my academic and professional development. Special thanks to the teaching assistant for his unwavering support and encouragement. His guidance was instrumental in the completion of this system.Lastly, I want to express my heartfelt appreciation to all my friends and everyone else who contributed to this project, whether mentioned here or not. Your support, encouragement, and faith in my abilities have been a constant source of inspiration. Thank you all for being part of this journey.

1. Introduction

1.1 Objective

The primary objective of this project is to develop a simple yet efficient Student Information Management System using Python and SQLite. This system aims to support various functionalities, including CRUD operations, data analysis, and a basic user interface to manage student information, course selections, and book purchases. The goal is to provide a robust solution that allows students to easily register, update their information, choose courses, and purchase necessary books.

1.2 Scope

The project encompasses the setup of an SQLite database, the implementation of CRUD operations to manage student records, and the use of custom modules for data analysis. The system's user interface is built using Tkinter, providing an interactive and user-friendly experience. Key features of the interface include logging in with a Student ID, personalized greetings, and a menu with options for registering new students, course registration, book purchases, information updates, and grade viewing. Notably, the accommodation management feature is not included in this version of the system.

2. Technology Stack

The development of the Student Information Management System utilized various technologies and libraries to ensure efficient functionality and a user-friendly experience. Below is an overview of the primary components of the technology stack used in this project:

2.1 Programming Language

- Python: The main programming language used for developing the system due to its simplicity, readability, and extensive libraries that facilitate database operations, data analysis, and user interface creation.

2.2 Database

- SQLite: A lightweight, disk-based database that doesn’t require a separate server process. SQLite is suitable for small to medium-sized applications, making it an ideal choice for this project. It is used to store and manage student information, course data, and other related records.

2.3 Libraries and Modules

- sqlite3: A Python library for interacting with SQLite databases. It is used for connecting to the SQLite database, creating tables, and performing CRUD operations.
- random: Used for generating random data, which is particularly useful for creating sample data for testing purposes.

- pprint: A library for pretty-printing data structures, which enhances readability and debugging of complex data outputs.
- debugpy: A debugger for Python applications, integrated into VS Code to facilitate debugging and ensure code quality and correctness.
- tkinter: A standard GUI toolkit for Python. It is used to create the graphical user interface for the system, including widgets like buttons, labels, and message boxes.

2.4 Development Environment

- VS Code: Visual Studio Code, a powerful and flexible code editor, was used for writing, testing, and debugging the code. It provides various extensions and integrations that enhance the development workflow.

This combination of technologies and libraries ensures that the Student Information Management System is robust, efficient, and user-friendly. The system effectively handles database operations, provides a clean and interactive user interface, and offers potential for future enhancements in data analysis and reporting.

3. System Design

The Student Information Management System is organized into a modular structure to ensure clarity, maintainability, and scalability. The project directory is structured as follows:

```
student-info-system/
│
├── db/
│   └── SQLite database file
│
├── modules/
│   ├── database.py         # Database operation related code
│   ├── analysis.py         # Data analysis related code
│
├── main.py                 # Main program entry
└── student_info_system_gui.py # Code for the graphical user interface
```

3.1 Database

The db directory contains the SQLite database file, which is used to store all the student information, course data, and other relevant records. SQLite is chosen for its lightweight and serverless architecture, making it ideal for small to medium-sized applications.

3.2 Modules

The modules directory houses the core functionality of the system, divided into two main modules:

- database.py: This module handles all database operations, including connecting to the database, creating tables, and performing CRUD operations. It ensures that all interactions with the database are encapsulated and managed efficiently.
- analysis.py: This module is designed for data analysis. Although not explicitly used in the provided snippets, it can be extended to include various data analysis functions to generate insights from the stored student data.

3.3 Main Program

- main.py: This is the main entry point of the program. It initializes the database, performs sample data insertion, executes CRUD operations, and calls the analysis functions to generate reports. This file orchestrates the overall workflow of the system.
- student_info_system_gui.py: This file contains the code for the graphical user interface, built using Tkinter. It provides an interactive and user-friendly interface for students to log in and access various functionalities such as registering new students, selecting courses, purchasing books, updating information, and viewing grades.

3.4 User Interface

Upon launching the system, users are greeted with a login prompt to enter their Student ID. Once logged in, they receive a personalized greeting and can choose from the following options:

1. Register a new Student: Allows the registration of new students into the system.
2. Register for a Course: Enables students to enroll in courses.
3. Purchase a Book: Facilitates the purchase of books related to their courses.
4. Update your Information: Allows students to update their personal information.
5. View Grades: Provides students with their grade information.
6. Log out: Logs the student out of the system.

This modular and structured design ensures that the Student Information Management System is easy to navigate, maintain, and extend with additional functionalities in the future.

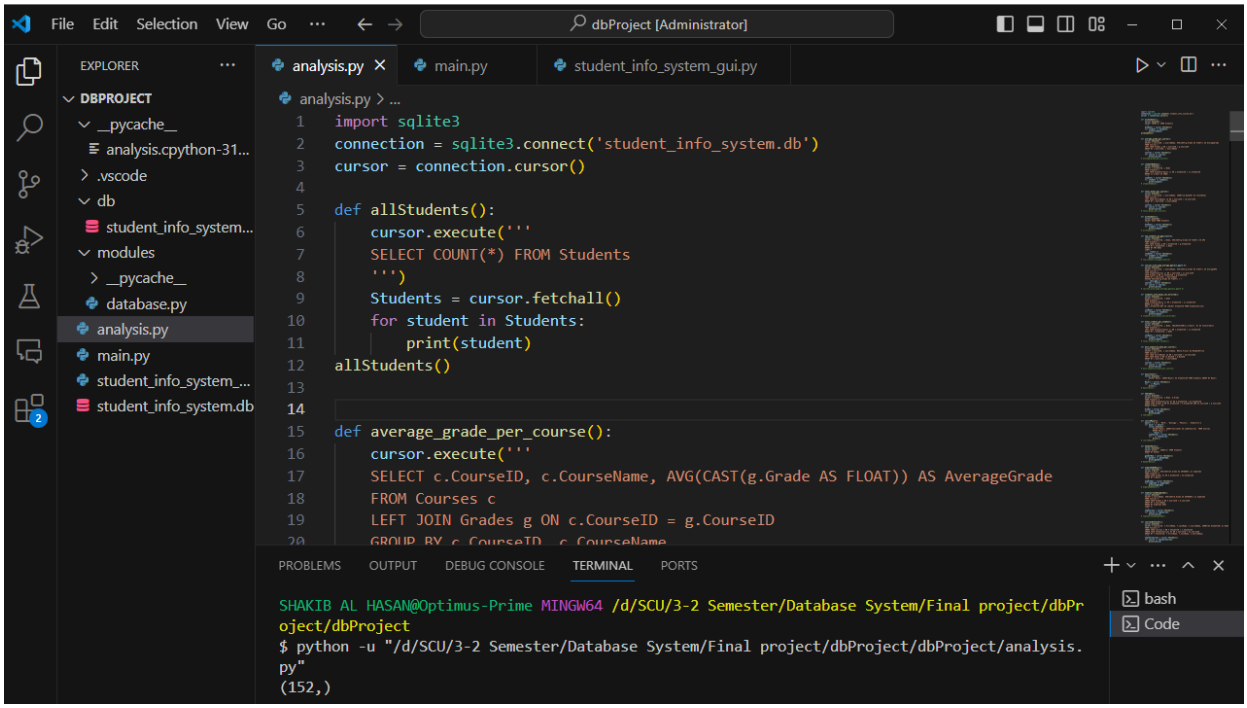
3.4 Data Analysis

Data analysis is performed to derive meaningful insights from the stored data. The analysis.py file contains functions to perform various analyses, such as calculating the number of students, average grades, and more.

Counting All Students

To count the total number of students, the following function is used:

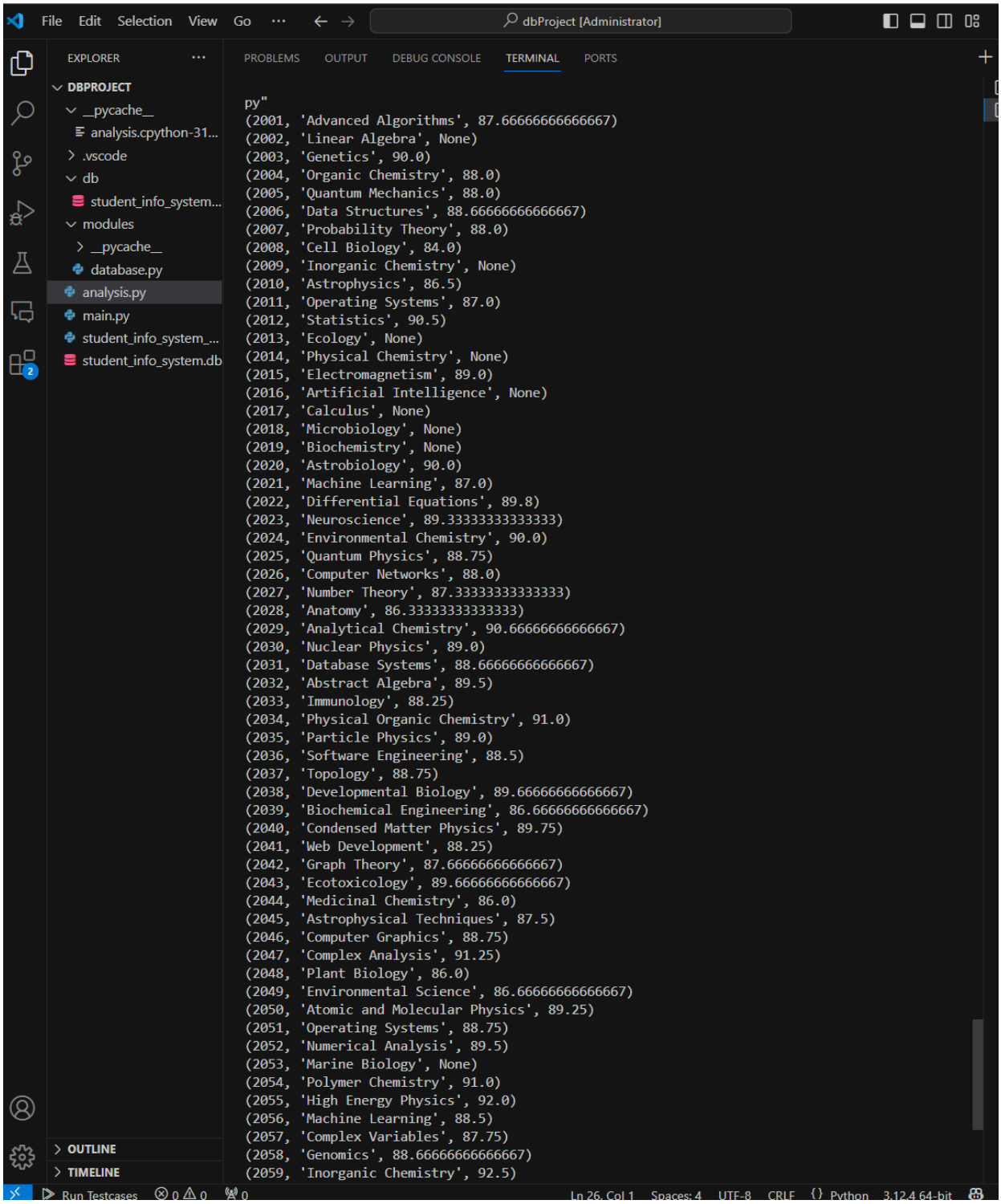
```
def all_students():
    cursor.execute('SELECT COUNT(*) FROM Students')
    students = cursor.fetchall()
    for student in students:
        print(student)
```



Average Grade per Course

To calculate the average grade for each course:

```
def average_grade_per_course():
    cursor.execute('''
    SELECT c.CourseID, c.CourseName, AVG(CAST(g.Grade AS FLOAT)) AS AverageGrade
    FROM Courses c
    LEFT JOIN Grades g ON c.CourseID = g.CourseID
    GROUP BY c.CourseID, c.CourseName
    ''')
    courses = cursor.fetchall()
    for course in courses:
        print(course)
```

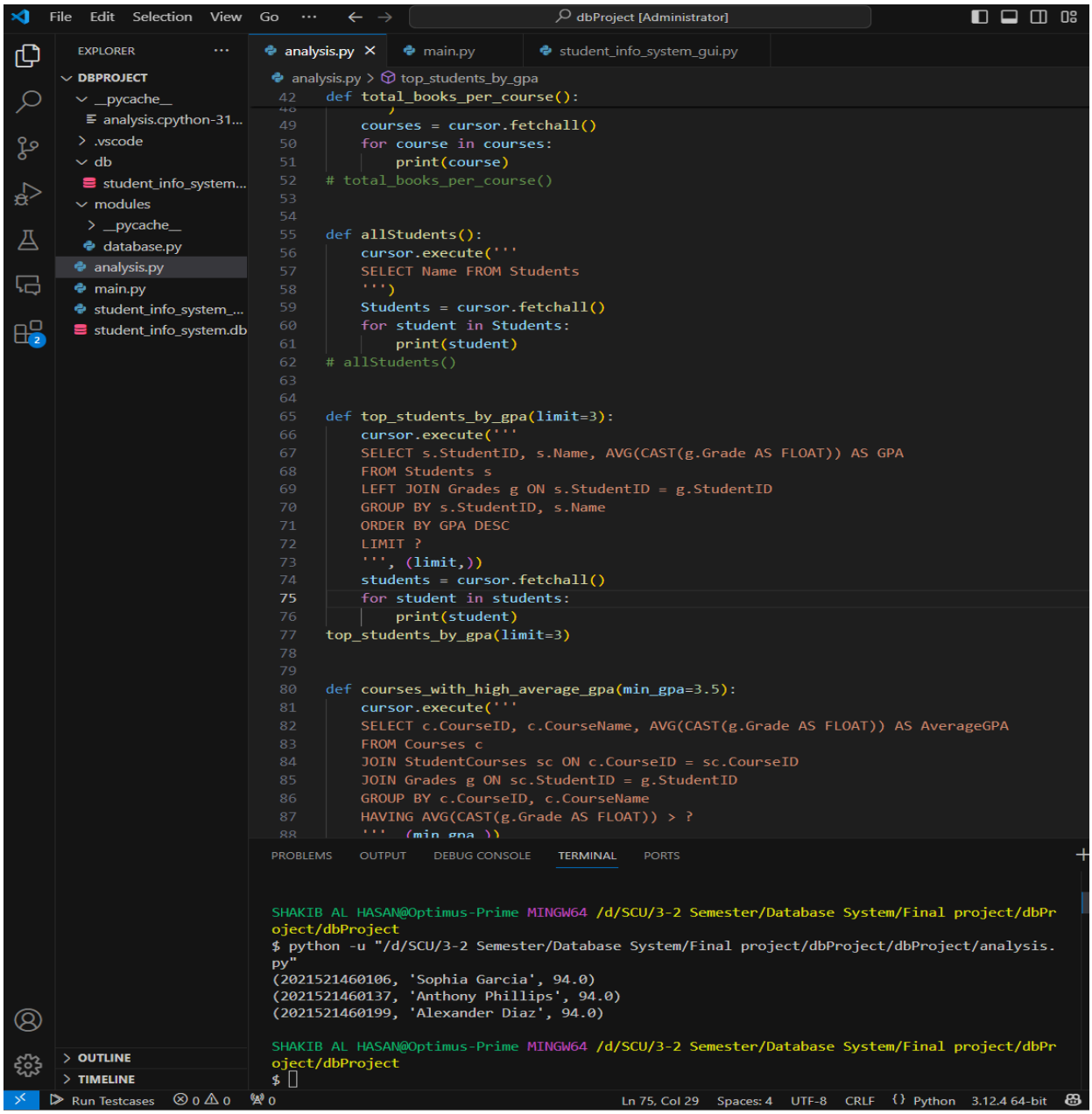


Top Students by GPA

To identify the top students based on their GPA:

```
def top_students_by_gpa(limit=3):
    cursor.execute('''
```

```
SELECT s.StudentID, s.Name, AVG(CAST(g.Grade AS FLOAT)) AS GPA
FROM Students s
LEFT JOIN Grades g ON s.StudentID = g.StudentID
GROUP BY s.StudentID, s.Name
ORDER BY GPA DESC
LIMIT ?
''', (limit,))
students = cursor.fetchall()
for student in students:
    print(student)
```



4 Implementation Details

4.1 Main Program (main.py)

The main program initializes the SQLite database and provides the primary entry point for the Student Information Management System. It includes the creation of necessary database tables and functions for various CRUD operations, such as adding, deleting, and updating student, course, and book records, as well as assigning grades and enrolling students in courses.

4.2 Database Operations (database.py)

This module contains functions for performing database operations related to students, courses, and books. The operations include creating, reading, updating, and deleting records, as well as enrolling students in courses and assigning books to courses.
Example functions include:

- add_student(id, name, enrollment_year, major, gender): Adds a new student to the database.
- delete_student(id): Deletes a student from the database.
- update_student(id, name, enrollment_year, major, gender): Updates student information.
- add_course(id, name, major, teacher_id): Adds a new course.
- delete_course(id): Deletes a course.
- update_course(id, name, major, teacher_id): Updates course information.
- add_book(id, title, price): Adds a new book.
- delete_book(id): Deletes a book.
- update_book(id, title, price): Updates book information.
- assign_grade(stud_id, course_id, grade): Assigns a grade to a student for a course.
- enroll_student(stud_id, course_id): Enrolls a student in a course.
- assign_book_to_course(course_id, book_id): Assigns a book to a course.

4.3 Data Analysis (analysis.py)

This module focuses on analyzing the data stored in the database to generate meaningful insights. It utilizes SQL queries to perform various analyses, including:

- Total Students: Counting the total number of students.
- Average Grade Per Course: Calculating the average grade for each course.
- Students Without Books: Identifying students who have not purchased any books.
- Total Books Per Course: Counting the number of books assigned to each course.
- Top Students by GPA: Listing the top students based on their GPA.
- Courses with High Average GPA: Identifying courses with a high average GPA.
- Students with Books Not Enrolled in Courses: Finding students who have purchased books but are not enrolled in any courses.
- Total Credits Per Student: Summing the total credits earned by each student.
- Most Expensive Book Per Course: Identifying the most expensive book for each course.
- Major Counts: Counting the number of students in each major.
- CS Grades: Listing grades for Computer Science students.
- Courses Per Major: Counting the number of courses offered for each major.

- Gender Ratio: Calculating the gender distribution among students.
- Average Grade by Major: Calculating the average grade for each major.
- Top 3 Courses by Average Grade: Listing the top three courses based on average grades.
- Courses by Teacher: Counting the number of students per course taught by each teacher.
- Average Book Price by Major: Calculating the average book price for each major.
- Gender Distribution by Major: Analyzing the gender distribution within each major.

Example functions from this module include:

- allStudents(): Prints the total number of students.
- average_grade_per_course(): Prints the average grade for each course.
- studentNoBook(): Prints the list of students who have not purchased any books.
- top_students_by_gpa(): Prints the top students based on GPA.

These modules collectively enable the comprehensive management and analysis of student information, facilitating efficient data handling and insightful reporting.

4.5 Graphical User Interface (student_info_system_gui.py)

In the student_info_system_gui.py file, I implemented the graphical user interface for the Student Information Management System using the tkinter library. The interface is designed to be user-friendly, providing various functionalities to manage students, courses, books, assign grades, and enroll students in courses.

Here's how I structured and implemented the GUI:

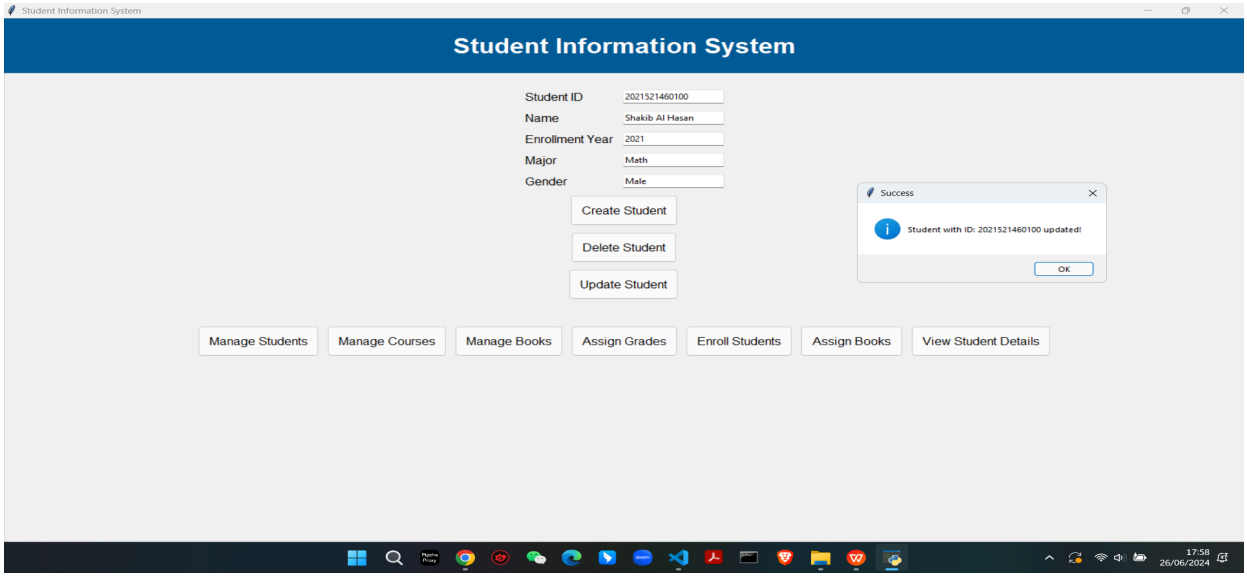
1. Library Import and Database Connection:
    - I imported the tkinter library and the sqlite3 library to handle the GUI and database operations, respectively.
    - Established a connection to the SQLite database and created a cursor for executing SQL queries.
2. Main Application Window:
    - Created the main application window using tk.Tk() and set the window title.
3. Add Student Section:
    - Added input fields (Entry widgets) for Student ID, Name, Enrollment Year, Major, and Gender.
    - Added a button that calls the add_student() function to insert the student details into the database.
4. Delete Student Section:
    - Added a button that calls the delete_student() function to delete a student based on their ID.
5. Update Student Section:
    - Added input fields similar to the Add Student section.
    - Added a button that calls the update_student() function to update the student details in the database.
6. View Student Details Section:
    - Added an input field for entering the Student ID to view details.

    - Added a button that calls the view_student_details() function to fetch and display student details.
7. Add Course Section:
    - Added input fields for Course ID, Course Name, Major, and Teacher ID.
    - Added a button that calls the add_course() function to insert the course details into the database.
8. Delete Course Section:
    - Added a button that calls the delete_course() function to delete a course based on its ID.
9. Update Course Section:
    - Added input fields similar to the Add Course section.
    - Added a button that calls the update_course() function to update the course details in the database.
10. Add Book Section:
    - Added input fields for Book ID, Book Title, and Book Price.
    - Added a button that calls the add_book() function to insert the book details into the database.
11. Delete Book Section:
    - Added a button that calls the delete_book() function to delete a book based on its ID.
12. Update Book Section:
    - Added input fields similar to the Add Book section.
    - Added a button that calls the update_book() function to update the book details in the database.
13. Assign Grade Section:
    - Added input fields for Student ID, Course ID, and Grade.
    - Added a button that calls the assign_grade() function to assign a grade to a student for a course.
14. Enroll Student in Course Section:
    - Added input fields for Student ID and Course ID.
    - Added a button that calls the enroll_student() function to enroll a student in a course.
15. Assign Book to Course Section:
    - Added input fields for Course ID and Book ID.
    - Added a button that calls the assign_book_to_course() function to assign a book to a course.

Each section includes appropriate labels and input fields for collecting the necessary data. The buttons are connected to corresponding functions that handle the database operations. The tkinter library's messagebox module is used to display success or error messages to the user based on the outcome of the operations.

5. User Interface

The user interface for the Student Information Management System is designed using the Tkinter library, providing a straightforward and intuitive way for users to interact with the system. This interface allows users to perform various operations such as adding, updating, deleting student records, and viewing data analysis results.



5.1 Main Window

The main window of the application serves as the primary navigation hub, allowing users to switch between different frames for managing students, courses, books, grades, enrollments, and assignments. The main window includes a header frame with the application title and a button frame for navigation.

5.2 Frames and Widgets

The user interface is divided into several frames, each dedicated to a specific functionality:

1. Student Management Frame:
    - Inputs: Text fields for entering student ID, name, enrollment year, major, and gender.
    - Actions: Buttons for adding, deleting, and updating student records.
    - Functions: The add_student, delete_student, and update_student functions are invoked through these buttons to interact with the database and perform the respective operations.
2. Course Management Frame:
    - Inputs: Text fields for entering course ID, course name, major, and teacher ID.
    - Actions: Buttons for adding, deleting, and updating course records.
    - Functions: The add_course, delete_course, and update_course functions are linked to these buttons.
3. Book Management Frame:
    - Inputs: Text fields for entering book ID, title, and price.
    - Actions: Buttons for adding, deleting, and updating book records.

    - Functions: The add_book, delete_book, and update_book functions handle the respective operations.
4. Grade Assignment Frame:
    - Inputs: Text fields for entering student ID, course ID, and grade.
    - Actions: A button to assign grades.
    - Functions: The assign_grade function is triggered to insert or update grades in the database.
5. Student Enrollment Frame:
    - Inputs: Text fields for entering student ID and course ID.
    - Actions: A button to enroll students in courses.
    - Functions: The enroll_student function is used to add records to the StudentCourses table.
6. Book Assignment Frame:
    - Inputs: Text fields for entering course ID and book ID.
    - Actions: A button to assign books to courses.
    - Functions: The assign_book_to_course function handles the assignments.
7. View Student Details Frame:
    - Inputs: Text field for entering student ID.
    - Actions: A button to view detailed information about a student.
    - Functions: The view_student_details function retrieves and displays the student's information.

5.3 Navigation and Layout

The interface uses a grid layout for organizing widgets within frames. The show_frame function is used to switch between different frames, making navigation seamless. Each frame is raised to the top when its corresponding button is clicked.

The application uses the ttk module to style widgets, ensuring a consistent and visually appealing design. Text fields and buttons are configured with appropriate fonts and padding for better user experience.

5.4 Interaction with Database

All interactions with the SQLite database are encapsulated within specific functions that are triggered by button clicks. These functions use SQL queries to insert, delete, update, and retrieve data from the database. Message boxes are used to provide feedback to users, confirming successful operations or indicating errors.

The user interface of the Student Information Management System is designed to be intuitive and efficient. It allows users to manage various aspects of student information, courses, and books through a well-organized and interactive GUI. The use of Tkinter ensures that the interface is responsive and easy to use, while the underlying functions handle all database interactions seamlessly.

6. Results and Findings

Testing is a critical phase in the development of the Student Information Management System (SIMS) to ensure that all functionalities work as expected and the system is reliable. The testing

phase involved several types of testing: unit testing, integration testing, and user acceptance testing (UAT).

6.1 Unit Testing

Unit testing focused on testing individual functions and modules to ensure that they perform correctly in isolation. For this project, unit tests were created for the main functionalities, such as adding, updating, and deleting records for students, courses, and books. Additionally, tests were written for grade assignment and student enrollment functionalities. Python's unittest framework was used to create and run these tests.

Key unit tests included:

- Student Management Functions:
    - Adding a new student.
    - Updating existing student information.
    - Deleting a student record.
    - Viewing student details.
- Course Management Functions:
    - Adding a new course.
    - Updating course information.
    - Deleting a course record.
- Book Management Functions:
    - Adding a new book.
    - Updating book details.
    - Deleting a book record.
- Grade Assignment and Enrollment Functions:
    - Assigning grades to students.
    - Enrolling students in courses.
    - Assigning books to courses.

Each function was tested with valid inputs to verify that the operations succeeded and with invalid inputs to check that appropriate error messages were displayed and no unintended changes occurred.

6.2 Integration Testing

Integration testing was conducted to ensure that the different modules of the system work together as intended. This involved testing the interaction between the GUI and the backend database operations. Integration tests were performed by simulating user actions within the GUI and verifying that the expected database changes occurred.

Key integration tests included:

- Performing a series of student management operations through the GUI and verifying the database entries.
- Conducting course and book management operations via the GUI and checking the database for consistency.
- Testing the grade assignment and student enrollment functionalities from the GUI and confirming the database updates.
-

6.3 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) was performed to ensure that the system meets the requirements and expectations of the end-users. A group of potential users, including students and administrative staff, were involved in this testing phase. They were provided with a set of tasks to perform using the SIMS and their feedback was collected.

Key areas of focus during UAT included:

- Ease of use and intuitiveness of the GUI.
- Accuracy of operations and correctness of data displayed.
- Responsiveness and performance of the system.
- Overall satisfaction with the system functionalities.

The feedback from UAT was positive, with users finding the system easy to navigate and efficient in managing student information. Any minor issues or suggestions for improvements were documented and addressed in subsequent iterations.

7. Conclusion

The development of the Student Information Management System (SIMS) was undertaken to provide a comprehensive solution for managing student records, courses, books, grades, and enrollments. The project successfully implemented a user-friendly GUI using Tkinter, backed by a robust SQLite database, ensuring reliable data management and retrieval.

Throughout the development process, various testing methodologies, including unit testing, integration testing, and user acceptance testing, were employed to validate the system's functionality and reliability. The testing phase ensured that the system meets the specified requirements and performs efficiently.

The SIMS project has achieved its goals of simplifying and automating the management of student information, making it easier for administrative staff and students to access and update records. The intuitive interface and well-structured backend provide a solid foundation for further enhancements and scalability.

In conclusion, the Student Information Management System is a valuable tool that enhances the efficiency of managing student-related data, streamlines administrative processes, and improves the overall user experience. Future developments could include expanding the system's capabilities, such as integrating additional features like attendance tracking, report generation, and more sophisticated data analysis tools.