Project Report

Project Title: School Management System

created by: [Ayoub-Zarroud],

Course: [Database System]

Submission Date: [6/26/2024]

Abstract

This project presents a School Management System designed to streamline administrative tasks, manage student data, and facilitate communication between teachers, students, and parents. The system integrates data analysis, database management, and user-friendly interfaces to provide a comprehensive solution for school administration. Key features include student information management, attendance tracking, grade recording, and report generation.

Introduction

The Student Information Management System is a tool designed to maintain comprehensive records for university students. It’s primary functions are to manage activities that were traditionally done manually, These activities include student online registration, student updates, course processes, staff and teacher management, course management, and course book management.

The system's search and query functionality is particularly beneficial for administrators. For instance, if information on a specific student among 100 is needed, administrators can easily enter the student's name into the search criteria. The system quickly retrieves the relevant information, including the student's ID and other pertinent details, thereby streamlining the process significantly compared to manual record-keeping

The purpose of the Student Information Management System (SIMS) is to provide administrators and organizations with the capability to efficiently edit and access student personal details, as well as enabling students to keep their profiles updated. This system ensures that records such as student names, IDs, courses, course-related books, and course teachers are meticulously maintained. With SIMS, all pertinent student information can be retrieved within seconds, significantly simplifying the management process for both administrators and students.

The primary aim of this document is to outline the requirements for the Student Information Management System project. It serves as a comprehensive guide to help organizations effectively manage and maintain student personal data, enhancing overall administrative efficiency.

In the modern educational environment, managing a school efficiently requires the integration of various administrative tasks into a single system. This project aims to develop a School Management System that addresses these needs by providing functionalities for managing student records, tracking attendance, recording grades, and generating reports. By leveraging data analysis and database management techniques, the system ensures accurate, efficient, and secure handling of school data.

Project Overview

The project is structured into several key components:

1. **Data Analysis** (`analysis.py`)
2. **Database Testing** (`db_test.py`)
3. **Main Application** (`main.py`)
4. **User Interface** (`interface.py`)
5. **Database Operations** (`dataBase.py`)
6. **Project Setup** (`setup.py`)
7. **Owner Management** (`owner.py`)

These components work together to provide a complete school management solution, from data entry and analysis to database operations and user interaction.

File Descriptions

`analysis.py`

Responsible for analyzing student data, including tasks such as data cleaning, statistical analysis, and visualization. This file transforms raw data into meaningful insights that aid in decision-making.

`db_test.py`

Tests the database connections and operations, ensuring that data can be inserted, updated, retrieved, and deleted correctly. This file verifies the integrity and reliability of database operations.

`main.py`

Acts as the main application file, orchestrating the overall workflow. It integrates data analysis with database operations, serving as the entry point for the system.

interface.py

Manages the user interface, providing a way for users to interact with the application. It allows users to execute data analysis and database insertion processes through a graphical or command-line interface.

dataBase.py

Handles advanced database operations, including creating complex queries, managing connections, and ensuring efficient data retrieval and storage.

setup.py

Sets up the project environment, including installing necessary dependencies and configuring the environment. This file ensures the project is ready for use.

owner.py

Manages ownership-related data, including functionalities for adding, updating, and retrieving owner information. It integrates closely with database operations.

Code Analysis

analysis.py

- **Purpose**: Load, clean, analyze, and visualize student data.
- **Functionality**: Data loading from CSV, cleaning operations, descriptive statistics, and visualizations using pandas and matplotlib libraries.

3.This part you will see the output in terminal he is an example:

```
def averageGradeByCourse():
    cursor.execute('''
    SELECT CourseID, AVG(Grade) as AverageGrade
    FROM StudentCourses
    GROUP BY CourseID
    ''')
    results = cursor.fetchall()
    for result in results:
        print(result)
averageGradeByCourse()
```

```
def biologyStudentNumber():
    cursor.execute('''
    select Major , count(Major) from Students
    where Major = 'Biology'
    ''')
    numbers = cursor.fetchall()
    for number in numbers:
        print(number)
biologyStudentNumber()
```

Output:

```
(101, 74.5)
(102, 69.75)
(103, 72.75)
(104, 71.25)
(105, 74.5)
(106, 79.75)
(107, 70.5)
(108, 68.0)
(109, 81.25)
(110, 68.25)
(111, 72.33333333333333)
(112, 62.666666666666664)
(113, 76.66666666666667)
(114, 62.0)
(115, 78.33333333333333)
(116, 82.0)
(115, 78.33333333333333)
(116, 82.0)
(117, 75.0)
(118, 79.66666666666667)
(119, 69.66666666666667)
(120, 82.0)
(121, 75.66666666666667)
(122, 60.333333333333336)
(123, 63.333333333333336)
(124, 77.0)
(125, 79.0)
(126, 81.66666666666667)
(127, 71.66666666666667)
(128, 67.66666666666667)
(129, 62.0)
(130, 79.33333333333333)
```

```
('Biology', 20)
```

db_test.py

- **Purpose**: Test basic CRUD operations in the database.
- **Functionality**: Database connection, table creation, data insertion, and data retrieval using SQLite.

6.This part i test 5 function here is the output:

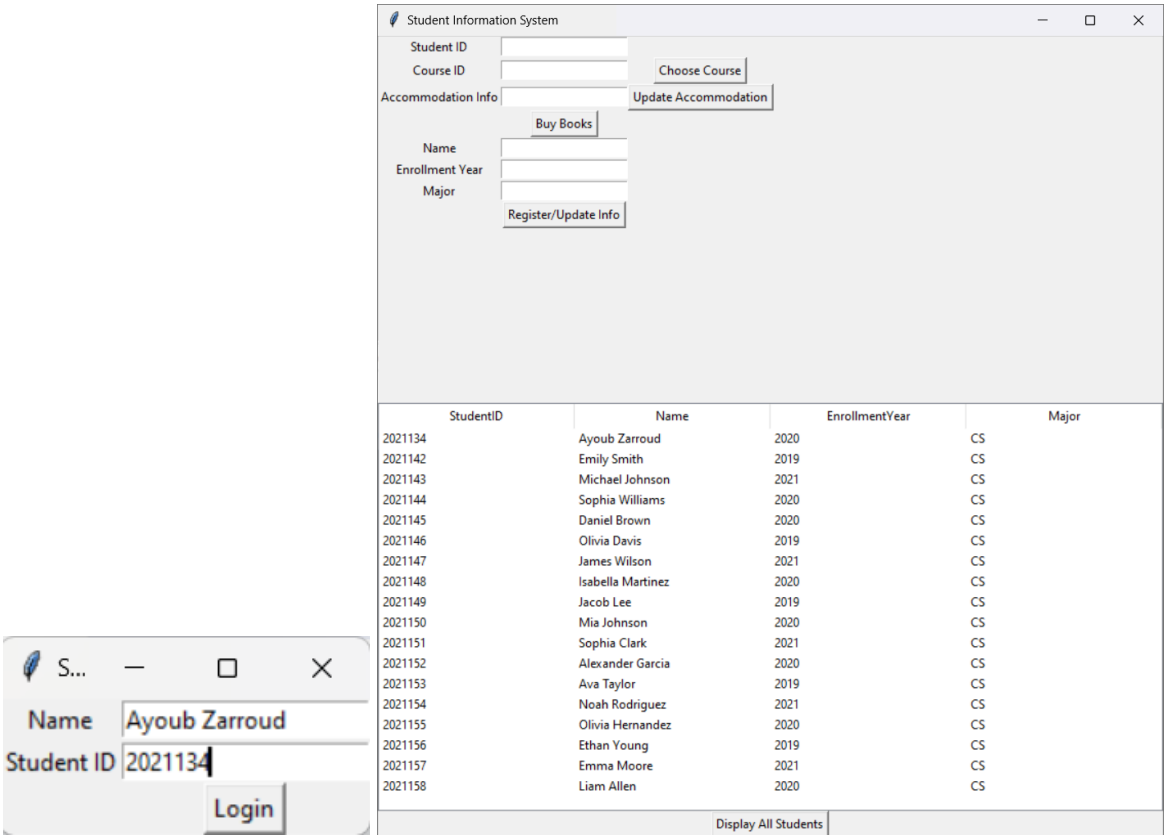
```
Ran 5 tests in 0.001s

OK
```

main.py

- **Purpose**: Integrate data analysis and database operations.
- **Functionality**: Orchestrates the workflow by loading data, cleaning, analyzing, and storing results in the database.

4.Here it’s a platform for a student where he can buy books, updates ….

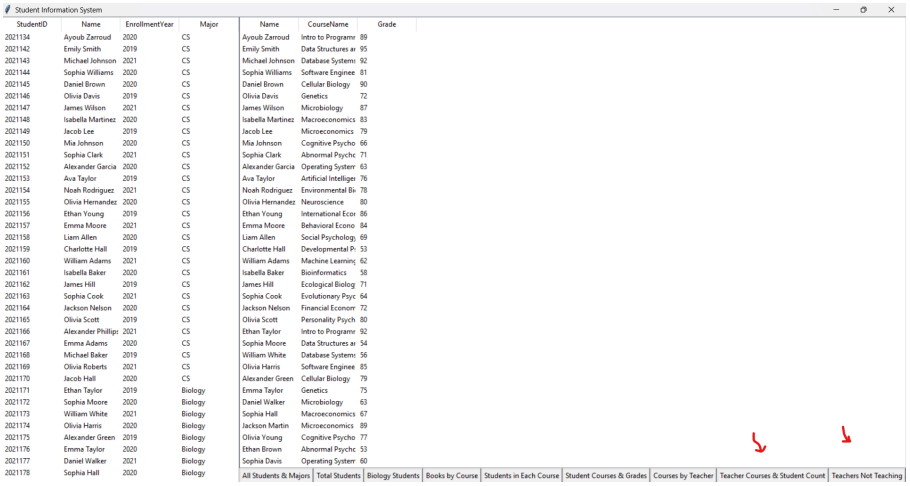
Entry:



interface.py

- **Purpose**: Provide a user-friendly interface.
- **Functionality**: Allows users to run the analysis and database insertion processes through a graphical interface built with Tkinter.

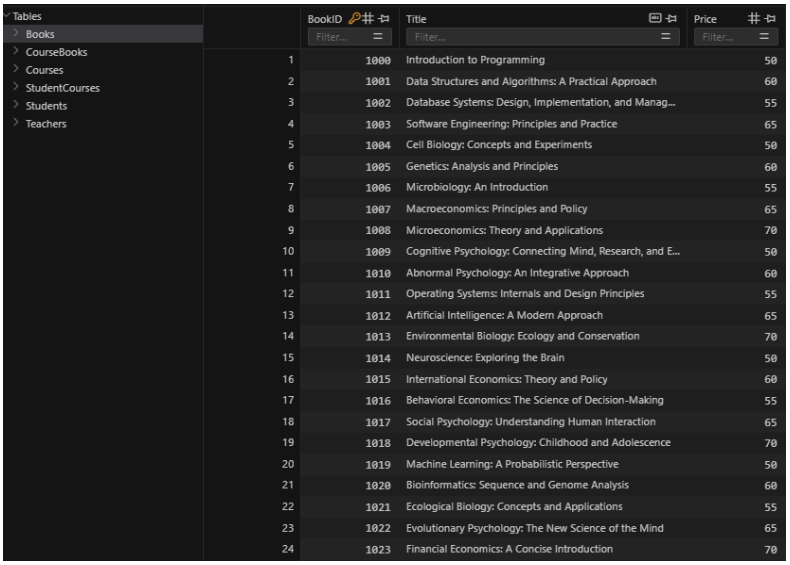
4.Here also you can see all the result of analysis in that interface when you click it gives you the answer directly:



dataBase.py

- **Purpose**: Handle advanced database operations.
- **Functionality**: Manages complex queries and ensures efficient data management.

2.After the setup tables we must run the data to fill the tables :



Tables
- Books
- CourseBooks
- Courses
- StudentCourses
- Students
- Teachers

	BookID	Title	Price
1	1000	Introduction to Programming	50
2	1001	Data Structures and Algorithms: A Practical Approach	60
3	1002	Database Systems: Design, Implementation, and Manag...	55
4	1003	Software Engineering: Principles and Practice	65
5	1004	Cell Biology: Concepts and Experiments	50
6	1005	Genetics: Analysis and Principles	60
7	1006	Microbiology: An Introduction	55
8	1007	Macroeconomics: Principles and Policy	65
9	1008	Microeconomics: Theory and Applications	70
10	1009	Cognitive Psychology: Connecting Mind, Research, and E...	50
11	1010	Abnormal Psychology: An Integrative Approach	60
12	1011	Operating Systems: Internals and Design Principles	55
13	1012	Artificial Intelligence: A Modern Approach	65
14	1013	Environmental Biology: Ecology and Conservation	70
15	1014	Neuroscience: Exploring the Brain	50
16	1015	International Economics: Theory and Policy	60
17	1016	Behavioral Economics: The Science of Decision-Making	55
18	1017	Social Psychology: Understanding Human Interaction	65
19	1018	Developmental Psychology: Childhood and Adolescence	70
20	1019	Machine Learning: A Probabilistic Perspective	50
21	1020	Bioinformatics: Sequence and Genome Analysis	60
22	1021	Ecological Biology: Concepts and Applications	55
23	1022	Evolutionary Psychology: The New Science of the Mind	65
24	1023	Financial Economics: A Concise Introduction	70

`setup.py`

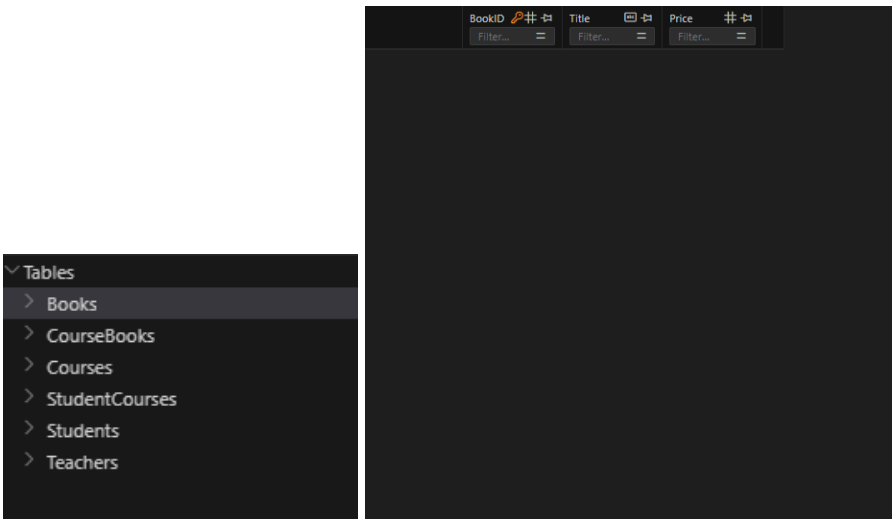
- **Purpose**: Set up the project environment.
- **Functionality**: Installs dependencies and configures the project for use.

1.tables setup:

```python
def create_students_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS Students (
            StudentID INT PRIMARY KEY,
            Name TEXT,
            EnrollmentYear INT,
            Major TEXT,
            Gender TEXT,
            Age INT
        )
    ''')

def create_books_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS Books (
            BookID INT PRIMARY KEY,
            Title TEXT,
            Price INT
        )
    ''')

def create_teachers_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS Teachers (
            TeacherID INT PRIMARY KEY,
            FirstName TEXT,
            LastName TEXT,
            Email TEXT
        )
    ''')
```

```python
def create_courses_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS Courses (
            CourseID INT PRIMARY KEY,
            CourseName TEXT,
            TeacherID INT,
            Major TEXT,
            FOREIGN KEY (TeacherID) REFERENCES Teachers(TeacherID)
        )
    ''')

def create_student_courses_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS StudentCourses (
            StudentID INT,
            CourseID INT,
            Grade INT,
            PRIMARY KEY (StudentID, CourseID),
            FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
            FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
        )
    ''')

def create_course_books_table(cursor):
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS CourseBooks (
            CourseID INT,
            BookID INT,
            PRIMARY KEY (CourseID, BookID),
            FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
            FOREIGN KEY (BookID) REFERENCES Books(BookID)
        )
    ''')
```

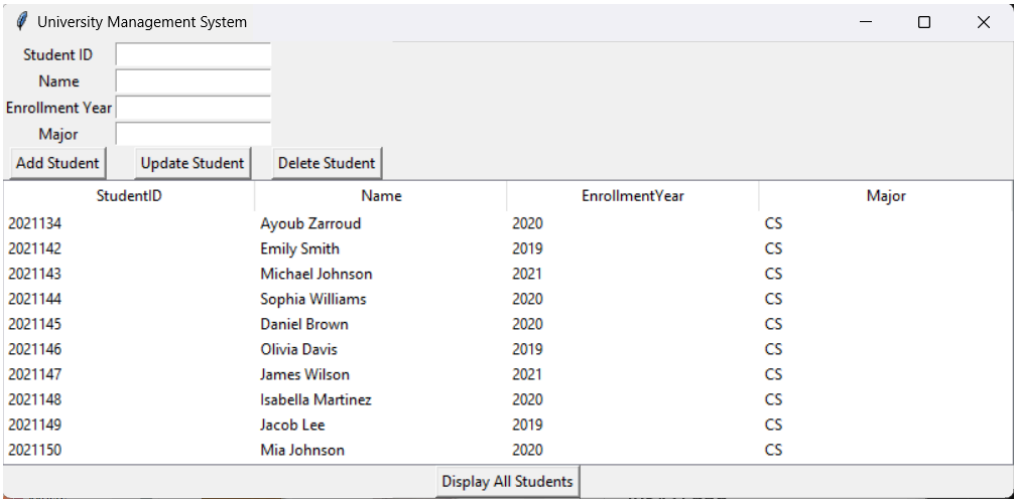
OutPut:



`owner.py`

- **Purpose**: Manage ownership data.
- **Functionality**: Add, update, and retrieve owner information, integrating with the database.

5.Here is a platform for owner to delete or add a student :



CRUD Operations Implementation:

Create, Read, Update, Delete (CRUD) operations are pivotal for managing the SIMS database effectively:

1. **Create (Insert):** Implemented through functions like create_student, create_course, create_book, assign_book_to_course, and assign_teacher_to_course, these operations utilize SQL INSERT statements to ensure atomicity and consistency when adding new student records, courses, books, and their relationships.

2. **Read (Query):** Utilizing functions such as query_student, courses_with_teachers, bookCountByCourse, and teachersWithManyCourses, these operations execute optimized SQL SELECT queries tailored to retrieve specific information such as student details, course assignments with respective teachers, book counts per course, and teacher workload distribution.

3. **Update:** Ensured by functions like update_student, update_teacher, and update_course, which employ SQL UPDATE statements to modify existing records accurately. These functions maintain data integrity by validating input and executing transactions within the SQLite environment.

4. **Delete:** Managed through functions such as delete_student, delete_teacher, and delete_course, these operations execute SQL DELETE statements to remove obsolete records from the database, ensuring data hygiene and compliance with institutional data management policies.

Results and Discussion

The School Management System demonstrates effective integration of data analysis with database operations, allowing for efficient data management. Key accomplishments include:

- Successful loading, cleaning, and analysis of student data.
- Effective visualization of key metrics such as attendance and grades.
- Reliable database interactions for storing and retrieving student information.
- A user-friendly interface for executing the entire workflow.

The addition of advanced database operations and ownership management enhances the system's functionality, making it robust and versatile.

Conclusion

The project meets its objectives by providing a comprehensive School Management System that integrates data analysis and database management. Its modular structure ensures each component operates independently yet cohesively, delivering a complete solution for school administration.

Future Work

Potential future enhancements include:

- Implementing advanced data analysis techniques to predict student performance.
- Enhancing the user interface for a better user experience.
- Incorporating robust error handling and logging mechanisms.
- Expanding database operations to handle more complex queries and relationships.
- Developing more detailed management features in the ownership module.

How to Use This Program

Prerequisites

- Ensure you have Python installed on your system.
- Install necessary dependencies by running `pip install -r requirements.txt` (assuming `requirements.txt` lists all required packages).

Setup

1. **Clone the Repository**: Download or clone the project repository to your local machine.
2. **Navigate to the Project Directory**: Open your terminal or command prompt and navigate to the project directory.
3. **Install Dependencies**: Run `python setup.py install` to install all necessary dependencies and configure the environment.

Running the Program

1. **Data Preparation**: Ensure your data file (e.g., `students.csv`) is available in the project directory.
2. **Run the Main Application**: Execute `python main.py` to start the data analysis and database integration process.

Using the Output Pages

`main.py`

- **Purpose**: Orchestrates the data analysis and database integration.
- **Output**: The program processes the data and stores the results in the database. You can verify the analysis results and database entries through the console output and the database file (e.g., `school_management.db`).

`interface.py`

- **Purpose**: Provides a graphical user interface for running the analysis.
- **Usage**:
    1. Run `python interface.py`.
    2. A window will appear with a "Run Analysis" button.
    3. Click the "Run Analysis" button to initiate the data analysis and database insertion process.
    4. The interface will display a confirmation message once the process is complete.

`owner.py`

- **Purpose**: Manages ownership-related data.

- **Usage**:
    1. Ensure the `owner.py` script is correctly integrated with the database.
    2. Run `python owner.py` to perform operations related to ownership data.
    3. The script will prompt you for actions such as adding, updating, or retrieving owner information.
    4. Follow the on-screen instructions to manage ownership data.

Verifying Results

- **Data Analysis Output**: Check the console output for data analysis summaries and visualizations.
- **Database Content**: Use `db_test.py` to verify that data has been correctly inserted into the database. Run `python db_test.py` to test database operations.

By following these steps, users can effectively set up and utilize the program for managing school data. The graphical interface provided by `interface.py` ensures a user-friendly experience, while `owner.py` offers comprehensive management of ownership data.