**David Ajibola 100735374**

**Geetharaman Murugan 100890042**

**Report On Flask-Based Student Database API**

The code represents a basic implementation of a Flask-based API for managing student records stored in an SQLite database. The API allows users to perform CRUD (Create, Read, Update, Delete) operations on student records. The API provides endpoints to create, retrieve, update, and delete student records, as well as retrieve all student records.

Here is a summary of the main components and functionalities of the code:

**Database Setup:**

An SQLite database file named "student.db" is used to store the student records.

A "students" table is created in the database to store student information, including fields like student\_id (primary key), first\_name, last\_name, dob (date of birth), and amount\_due (amount due, specified as REAL data type).

**Create Sample Record:**

A sample student record is inserted into the "students" table using a standard SQL INSERT statement.

**API Endpoints:**

/students (POST): This endpoint allows users to create a new student record. Users can send a JSON object in the request body containing the required fields (first\_name, last\_name, dob, amount\_due) to create a new record. The student\_id is automatically generated as an auto-incrementing primary key.

/students/<int:student\_id> (GET): This endpoint allows users to retrieve a specific student record by providing the student\_id as part of the URL. The API fetches the record from the database and returns it in JSON format.

/students/<int:student\_id> (PUT): This endpoint allows users to update an existing student record by providing the student\_id as part of the URL and sending a JSON object in the request body containing the updated fields (first\_name, last\_name, dob, amount\_due).

/students/<int:student\_id> (DELETE): This endpoint allows users to delete a specific student record by providing the student\_id as part of the URL.

/students (GET): This endpoint allows users to retrieve all student records present in the database. The API fetches all records from the "students" table and returns them as a JSON list.

**API Functionality:**

The API uses the Flask framework to handle HTTP requests and responses.

For each endpoint, the appropriate database operations are performed using SQLite, such as insertion, selection, update, and deletion, based on the requested operation.

The data retrieved from the database is transformed into JSON format and returned as the response to the client.

**Running the API:**

The code includes an if \_\_name\_\_ == '\_\_main\_\_': block that starts the Flask development server when the script is executed directly.

**API Usage:**

To interact with the API, users can use HTTP methods like POST, GET, PUT, and DELETE on the specified endpoints to manage student records.

**Improvements and Considerations:**

**Error Handling:** The API could benefit from better error handling and validation of user input to provide meaningful responses in case of invalid or missing data.

**Authentication and Authorization:** Currently, there's no authentication or authorization mechanism implemented, making the API publicly accessible to anyone. To secure the API and restrict access to authorized users, authentication and authorization mechanisms should be added.

**Data Validation:** The API should validate the incoming data to ensure that the required fields are present and that data formats (e.g., date of birth) are valid.