**Flask Web (server) Application for User Spending Analysis**

**Overview**

This Flask application provides APIs to analyze user spending data stored in a SQL database. It allows you to retrieve the total spending by a specific user and calculate the average spending by age ranges. Additionally, it demonstrates how to integrate MongoDB for storing users that exceed specific amount of total spending. In that way, the MongoDB can be accessed for getting the user IDs of the users that are eligible for a bonus voucher at the end of the year. *As a bonus, this application will send Telegram messages with calculated statistics to the store management and will also have unit tests for the API endpoints.*

### Prerequisites

Before developing this application, ensure you have the following components set up (and access to):

1. SQL deployed (SQLite) Database that will be used for getting the user data
2. MongoDB: Set up a MongoDB database for storing user data (first locally, and when everything works fine, hosted)
3. DBBrowser (SQLite)
4. MySQLWorkbench 8.0
5. Python 3.x
6. Modules installed:
   * pymongo
   * sqlalchemy
   * mysql-connector-python
   * requests
   * jsonify
   * json

### API Endpoints

#### 1. Retrieve Total Spending by User

* **Endpoint:** /total\_spent/<int:user\_id>
* **Method:** GET
* **Description:** Retrieves the total spending for a specific user based on their user ID.
* **Parameters:**
  + user\_id (integer): The unique user ID for the user.
* **Response:**
  + JSON object containing the user ID and total spending.

#### 2. Calculate Average Spending by Age Ranges

* **Endpoint:** /average\_spending\_by\_age
* **Method:** GET
* **Description:** Calculates the average spending for different age ranges and sends the results to specific Telegram users.
* **Response:**
  + JSON object containing average spending for age ranges (18-24, 25-30, 31-36, 37-47, >47)

#### 2. Write user data to MongoDB

* **Endpoint:** /write\_to\_mongodb
* **Method:** POST
* **Description:** This API endpoint allows clients to submit user data that exceeds specific amount of spending in JSON format, which is then inserted into a MongoDB collection.
* **Format of the input JSON data (example):   
  {   
   “user\_id”: 1,** “total\_spending”: 2000  
  }
* **Response:**
  + Upon successful insertion of the user data into the MongoDB collection, the API will return a JSON response with a success message and the HTTP status code 201 Created. If any errors occur during the insertion process, an error message will be returned with the appropriate HTTP status code and details of the error.

## Integration with SQL database

The integration with SQL database in the Flask application involves managing two tables: user\_info and user\_spending. These tables are used to store user information and spending data, respectively. The integration ensures that data is efficiently stored and retrieved, enabling various data analysis functionalities within the application.

### Database Tables

#### 1. user\_info Table

* **Purpose:** The user\_info table stores user-related information such as user IDs, names, emails and ages
* **Columns:**
  + user\_id: A unique identifier for each user.
  + name: The name of the user.
  + email: The email address of the user.
  + age: The age of the user.

#### 2. user\_spending Table

* **Purpose:** The user\_spending table is used to store data related to user spending, including the amount of money spent by each user and the corresponding year.
* **Columns:**
  + user\_id: A unique identifier that links the spending data to a specific user.
  + money\_spent: The amount of money spent by the user.
  + year: The year in which the spending data is recorded.

### Integration with MongoDB

This application also integrates with MongoDB to store user data, including user IDs and total spending for the users that exceed specific amount of spending. MongoDB is used to efficiently store the user data. The database’s name is *users\_vouchers* and the collection’s name *vouchers.*

### *Bonus 1- Integration with Telegram*

*The application sends calculated statistics (average spending by age ranges) to the store management where the spending is recorded on Telegram using the Telegram Bot API. Make research and find a way to integrate this feature in this web application.*

***Bonus 2 – Write unit tests for the API endpoints***

*Every application needs to be tested before it is deployed to production. Write specific unit tests for the API endpoints to test all the possible use-cases.*

### Data calculation and statistics for the API endpoints

The application needs to calculate total spending by user and average spending by age range without reading the entire database.

The Flask application uses efficient SQL queries to efficiently retrieve and calculate data from the SQLite database:

* **Total Spending by User:** The API endpoint /total\_spent/<int:user\_id> allows clients to retrieve the total spending for a specific user based on their user ID.
* **Average Spending by Age Ranges:** The API endpoint /average\_spending\_by\_age calculates the average spending for different age ranges. It uses SQL queries **to join** the user\_info and user\_spending tables and aggregates data to determine average spending within predefined age ranges.

Add comments to describe the application and use functions where possible for maintaining clean code and project architecture.