Ass 2

November 25, 2024

0.0.1 Extended Assignment: Task Management System with Flask API and SQLite3

Purpose: This extension builds on your existing **Task Management System** by integrating a **Flask API** and using **SQLite3** for persistent storage. You will expose task management functionalities via API endpoints, replace the existing CSV storage with a database, and enhance the system's usability and scalability.

You are required to implement the project within a virtual environment (venv) and create a requirements.txt file listing all the packages used in your project for easy setup and reproducibility. Organize your project effectively by placing all source code files in a dedicated src folder to maintain a clear and structured directory layout.

0.0.2 Components and Scoring Breakdown:

1. **Database Integration with SQLite3** (4 points)
2. **Flask API Implementation** (4 points)
3. **Documentation Update** (4 points)
4. **Oral Defense** (8 points)

0.0.3 Instructions

1. Database Integration with SQLite3 (4 points)

- **Modify the `Task` class**:
  - Use SQLite3 for task data storage instead of CSV files.
  - Design the database table based on the attributes of a task (e.g., title, due date, status, etc.).
  - Implement methods for interacting with the database, such as:
    - `save_to_db`: Saves a task to the database.
    - `load_from_db`: Loads a task from the database by its ID.
    - `update_in_db`: Updates an existing task in the database.
    - `delete_from_db`: Deletes a task from the database.
- **Modify the `TaskManager` class**:
  - Update it to handle CRUD operations using SQLite3 for task data.
  - Use the methods in the `Task` class for database interactions.

2. Flask API Implementation (4 points)

- **Build a Flask API** that allows interaction with the task management system.
- Implement the following RESTful API endpoints:
  - `POST /tasks`:
    - Create a new task (PersonalTask or WorkTask) based on the request data.
    - Example Request Payload:

```
{
  "type": "personal",  // or "work"
  "title": "Task Title",
  "due_date": "YYYY-MM-DD",
  "description": "Short description",
  "priority": "low"
}
```

  - `GET /tasks`:
    - Retrieve all tasks, or filter by task type (e.g., `?type=personal`).
  - `GET /tasks/<int:task_id>`:
    - Retrieve a specific task by its ID.
  - `PUT /tasks/<int:task_id>`:
    - Update task details, such as status, priority, or team members.
    - Example Request Payload to update a task's description:

```
{
  "description": "This assignment is an extension of Assignment 1."
}
```

  - `DELETE /tasks/<int:task_id>`:
    - Delete a task by its ID.
  - `GET /tasks/pending`:
    - Fetch all tasks that are marked as "pending".
  - `GET /tasks/overdue`:
    - Fetch all overdue tasks based on the current date.

3. Documentation Update (4 points) (Provide clear and well-structured documentation in a format of your choice, ensuring it effectively communicates your work and is easy to understand.)

- **Update the README file** to include:
  - **API Documentation**:
    - Document the endpoints, request formats, and example responses.
  - **Database Schema**:
    - Provide a description of the database schema (e.g., table structure, column names, etc.).
  - **Setup Instructions**:
    - Include virtual environment setup and instructions for using requirements.txt.
    - Explain how to run the Flask application locally.
  - **Example Usage**:
    - Provide example `curl` commands or shell scripts for interacting with the API:
      - Creating tasks (with both valid and invalid data).
      - Retrieving tasks (all tasks, filtered tasks, by task ID).
      - Updating tasks and verifying changes.

- Deleting tasks and handling non-existent IDs.

4. Oral Defense (8 points)

- **Be prepared to present and discuss**:
  1. **Architecture Overview (2 points)**:
     - Explain how Flask interacts with SQLite3 in your system.
     - Describe how the original task class design integrates into the extended system.
  2. **Improvements (2 points) - OPEN TOPIC**:
     - Discuss any improvements or additional features you would suggest for the system (e.g., error handling).
  3. **Q&A Session (4 points)**:
     - Be ready to answer questions about your solution.

0.0.4 Assignment Goals

This extension will enhance your skills in:

- **Backend development**: Using Flask to build APIs and interact with databases.
- **Database management**: Designing schemas and performing CRUD operations with SQLite3.
- **Documentation and presentation**: Effectively communicating your design and implementation.

Good luck with the extension, and happy coding!