Naveen M

Reg.no: 212222110029

Saveetha Engineering College

04/06/2025

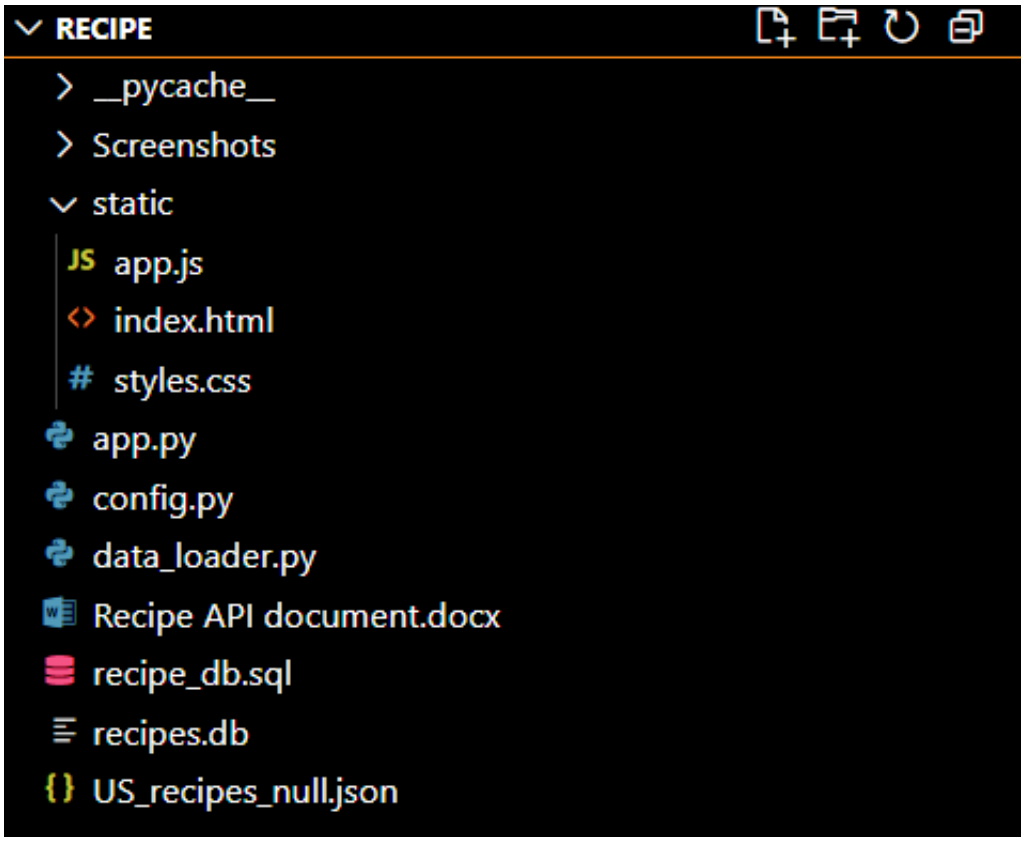
Recipe Application Documentation

1. Overview

This project is a web-based recipe application that allows users to:

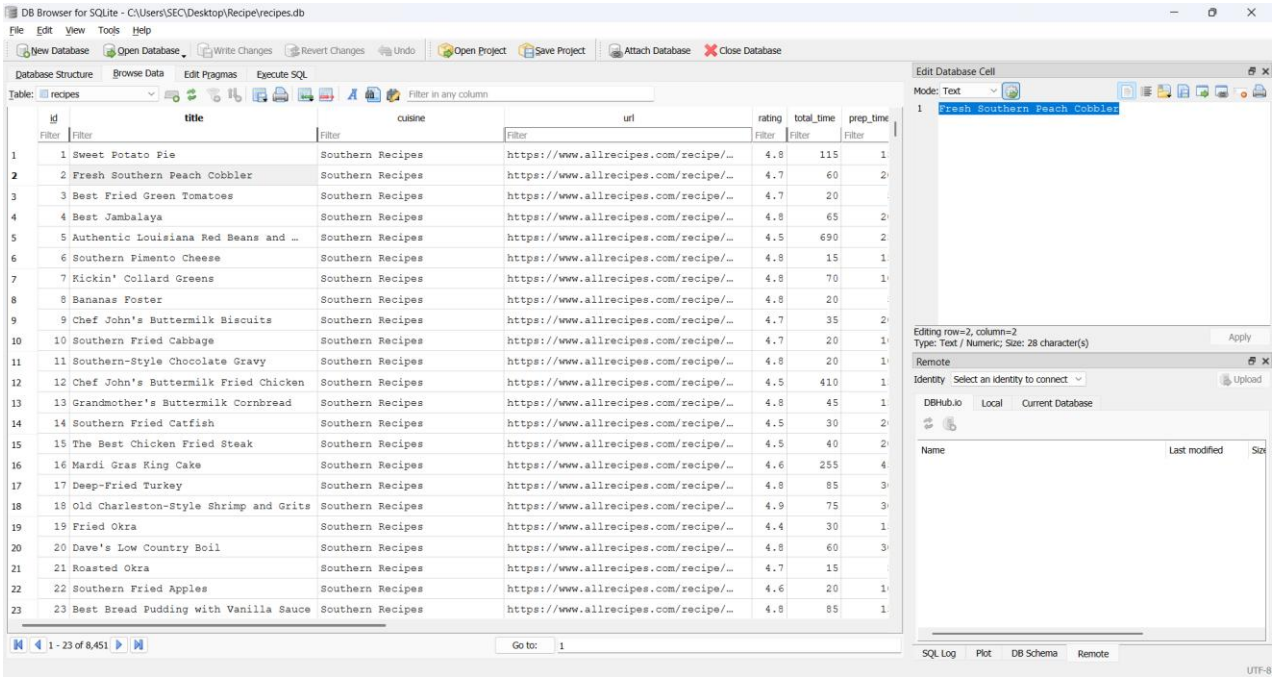
- Browse recipes stored in a SQLite database.
- Search and filter recipes by title and cuisine.
- Paginate results for easier navigation.
- View recipe details such as title, cuisine, rating, and description.
- The application consists of a Flask backend (serving a RESTful API and the frontend) and a modern, responsive frontend.
- The SQLite Database (**recipes.db**) has **8451** data

2. Project Structure



Key Points on Converting JSON to SQLite Database

- Use sqlite3 to interact with the SQLite database and json to parse the JSON file
- Load the JSON data into Python using json.load()
- Establish a connection to the SQLite database file using sqlite3.connect()
- Define the table schema and create it using SQL CREATE TABLE
- Loop through each JSON object and insert its values into the SQLite table using INSERT statements
- Handle Data Types and Structure:
    - **Flat JSON:** Each key-value pair in the JSON object maps directly to a column in the table.
    - **Nested JSON:** Store nested objects or lists as JSON strings (using json.dumps()) or consider normalizing into separate tables



3. System Requirements

- Python 3
- Flask (pip install flask)
- SQLite3 (included with Python)
- Web browser (Chrome, Firefox, Edge, etc.)

4. Setup

4.1. Prepare the Database

Run the data loader script to populate the SQLite database from your JSON file.

The database file (**recipes.db)** will be created in your project folder.

4.2. Start the Flask Application

python app.py

This will start the server at http://localhost:5000.

5. Frontend Features

Search and Filter: Filter recipes by title and/or cuisine.

Pagination: Navigate through pages of recipes using `page` and `limit` controls.

Responsive Design: Works on desktop and mobile devices.

6. Backend API Endpoints

All endpoints are prefixed with /api/recipes

Endpoint	Method
http://localhost:5000/api/recipes	GET
http://localhost:5000/api/recipes /title/Sweet%20Potato%20Pie	GET
http://localhost:5000/api/recipes/cuisine?cuisine=Southern%20Recipes	GET
http://localhost:5000/api/recipes?page=2&limit=5	GET

7. How to Use

Access the application:
Open http://localhost:5000 in your browser.

Browse recipes:
Recipes are displayed in a table. Use the filters to bar to get the results.

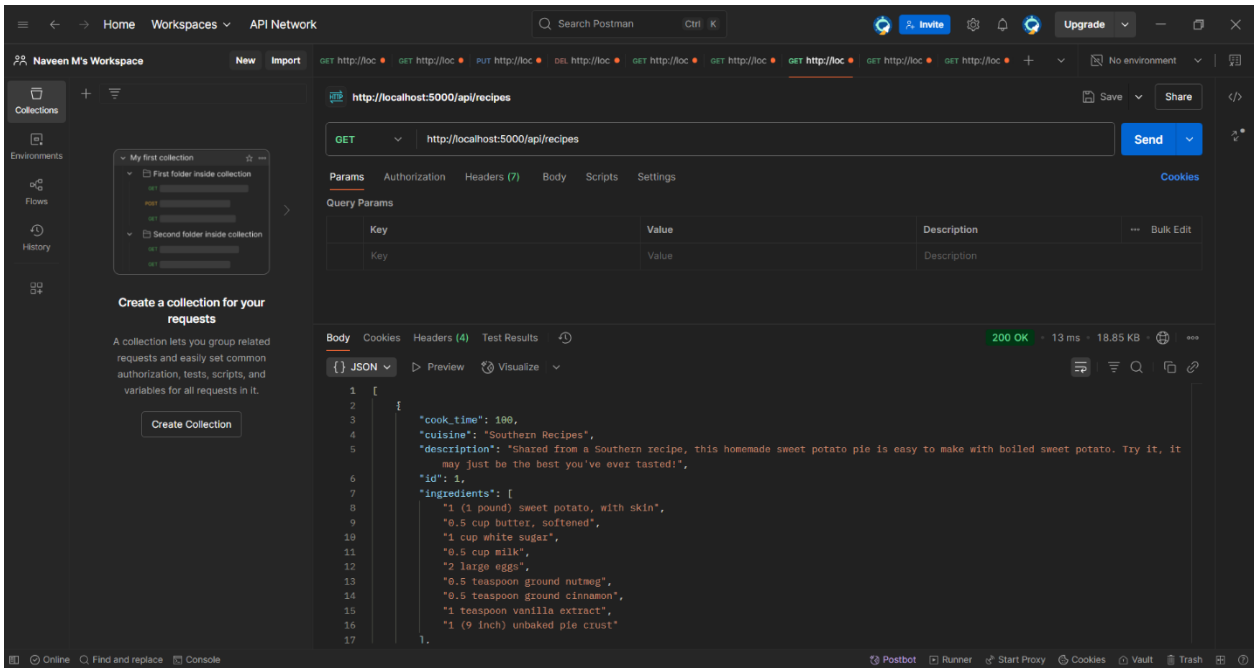
Paginate results:
Use the page and limit inputs to navigate through pages.

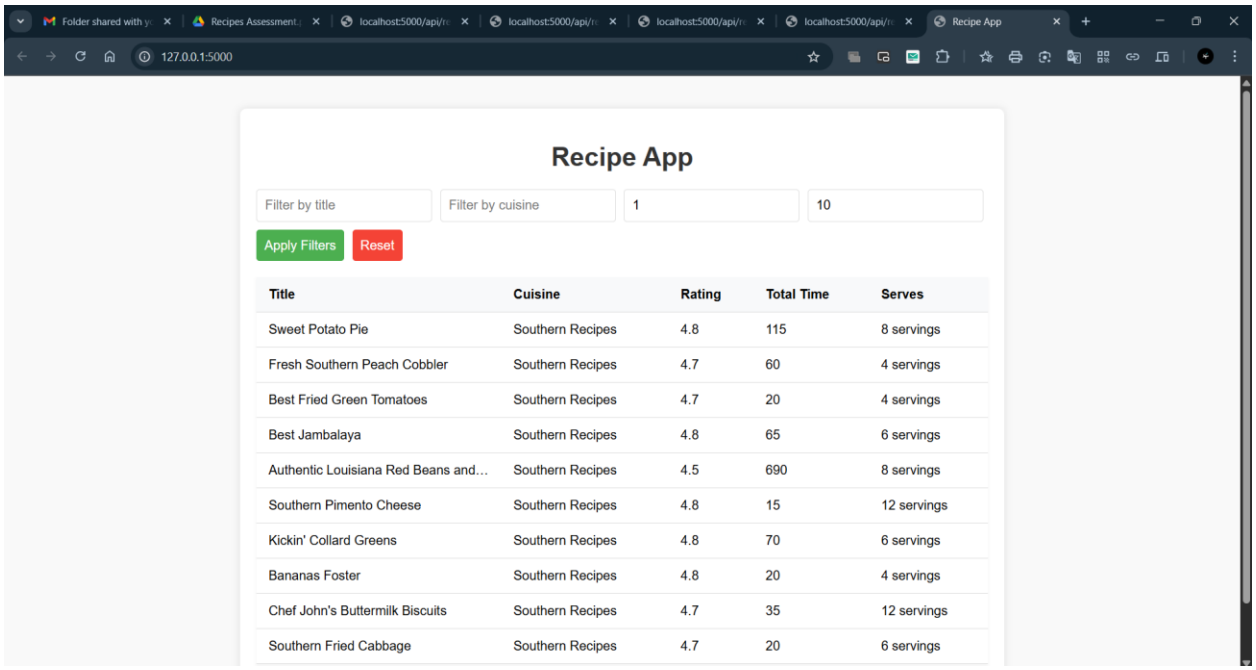
Search and filter:
Enter a title or cuisine to filter the results.

8. Example API Requests

Fetching all recipes:

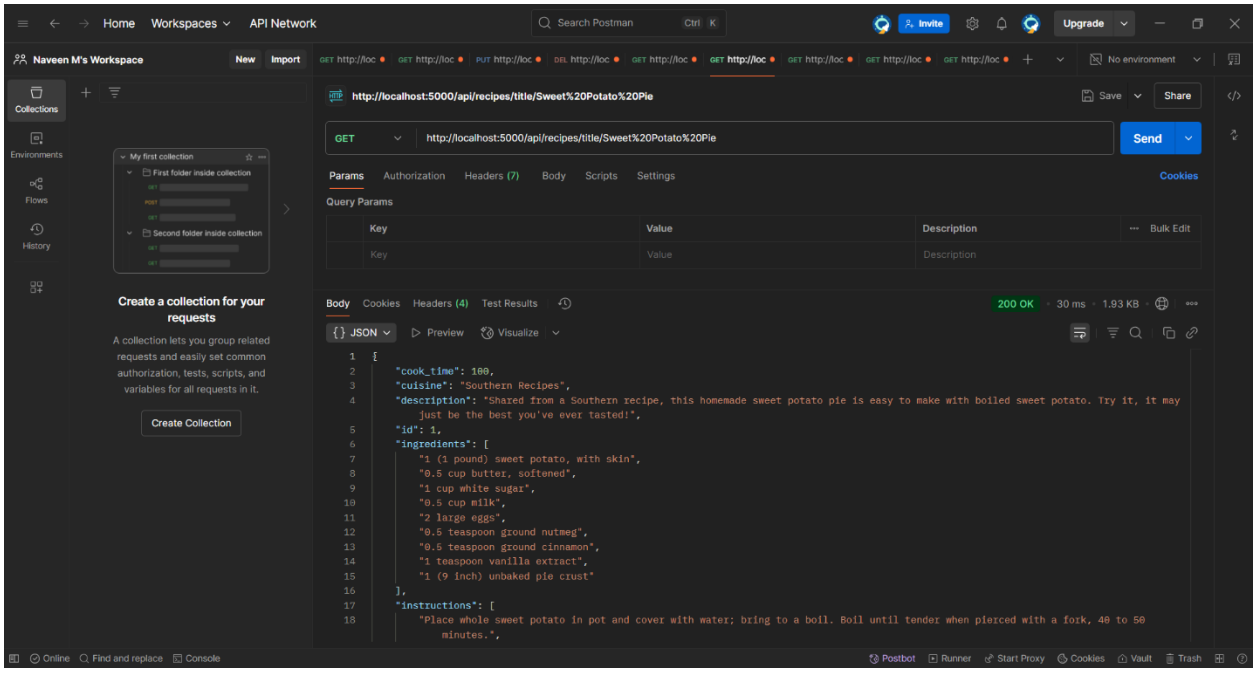
GET http://localhost:5000/api/recipes

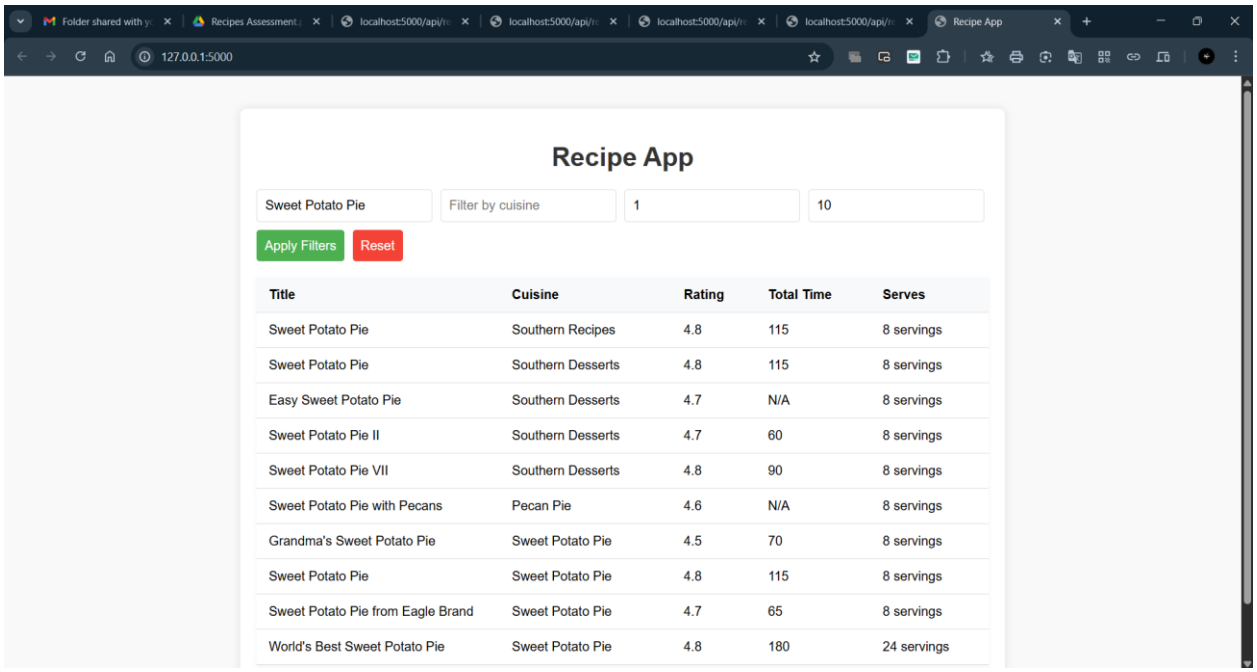




Searching by title:

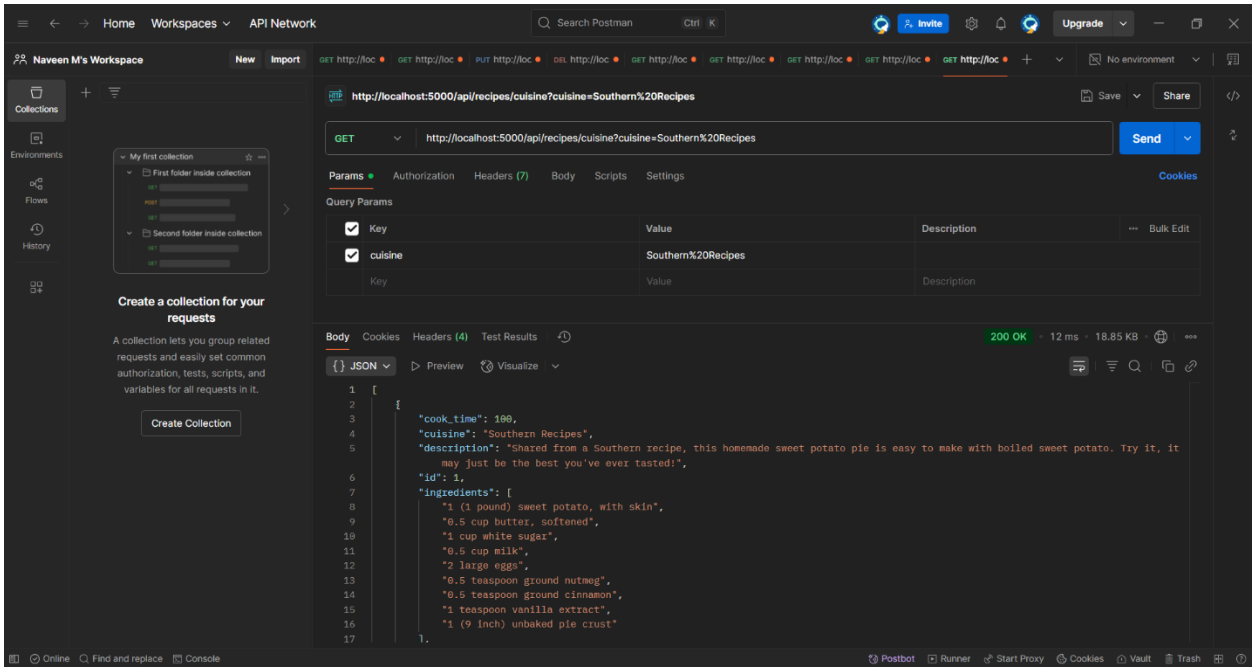
GET http://localhost:5000/api/recipes /title/Sweet%20Potato%20Pie

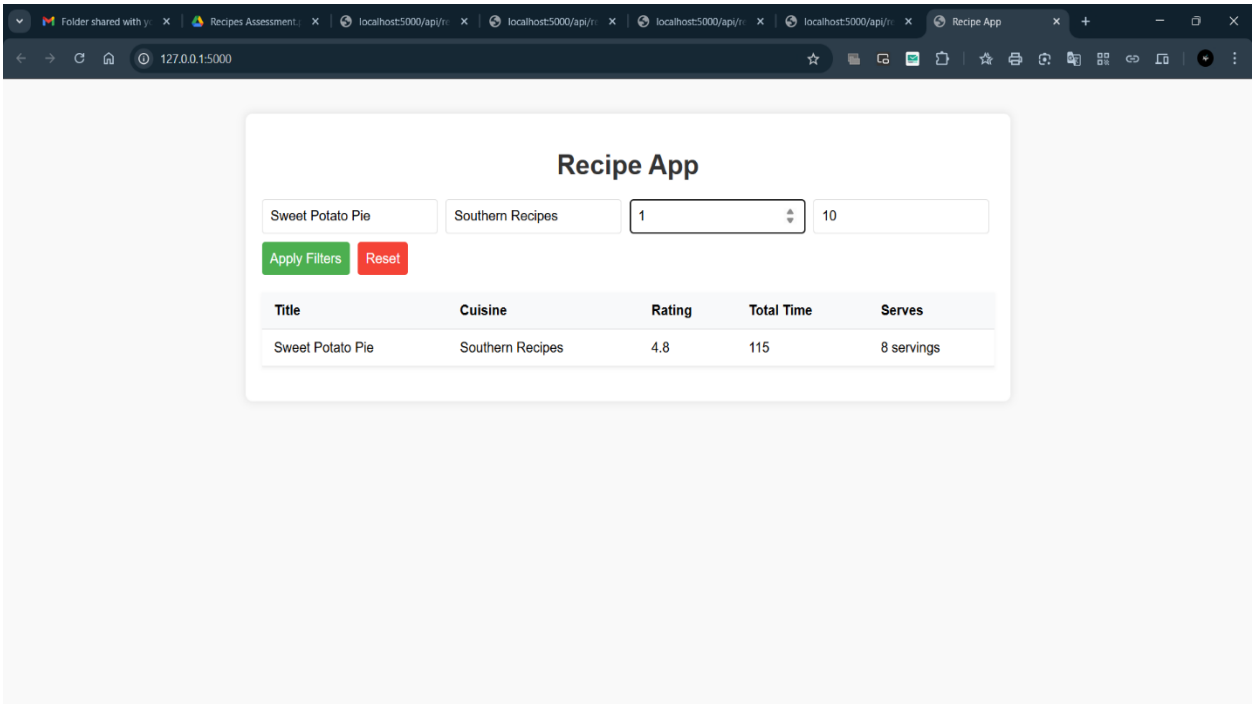




Searching by cuisine

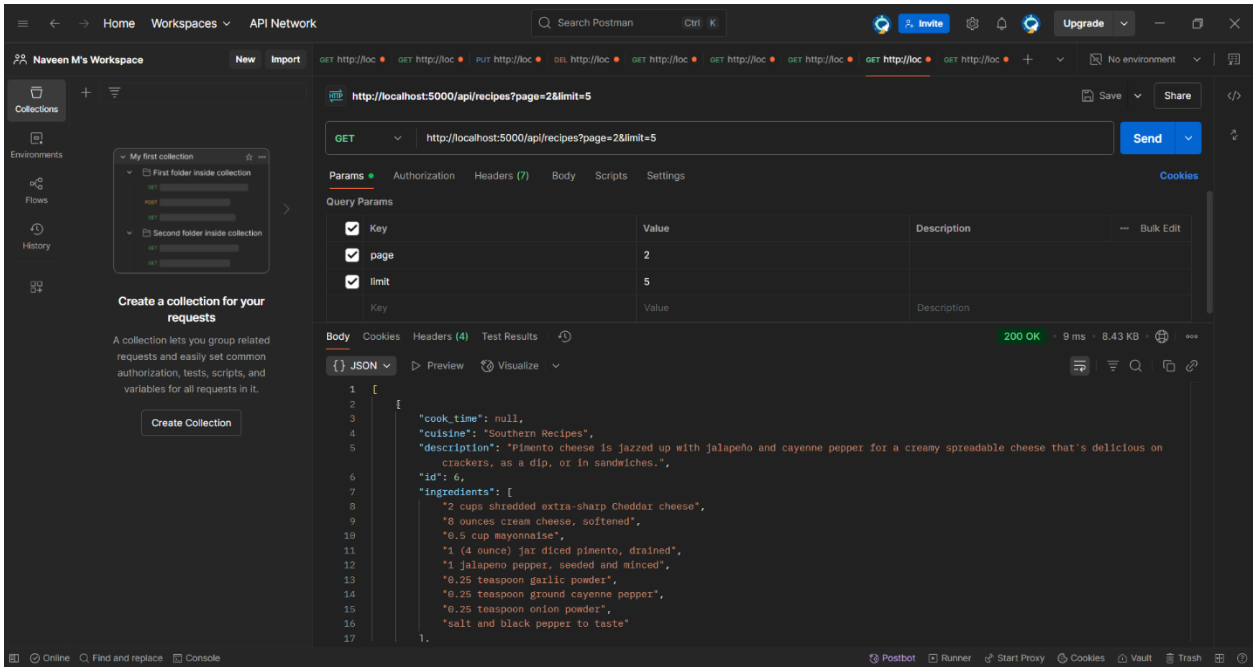
GET http://localhost:5000/api/recipes/cuisine?cuisine=Southern%20Recipes

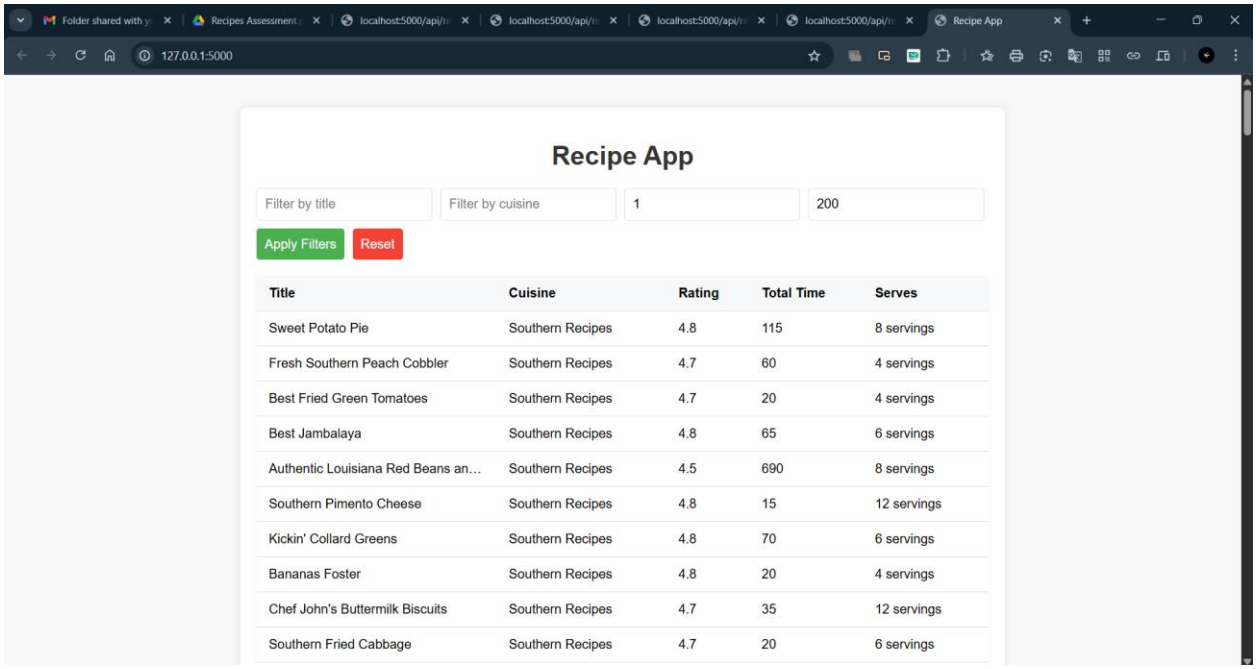


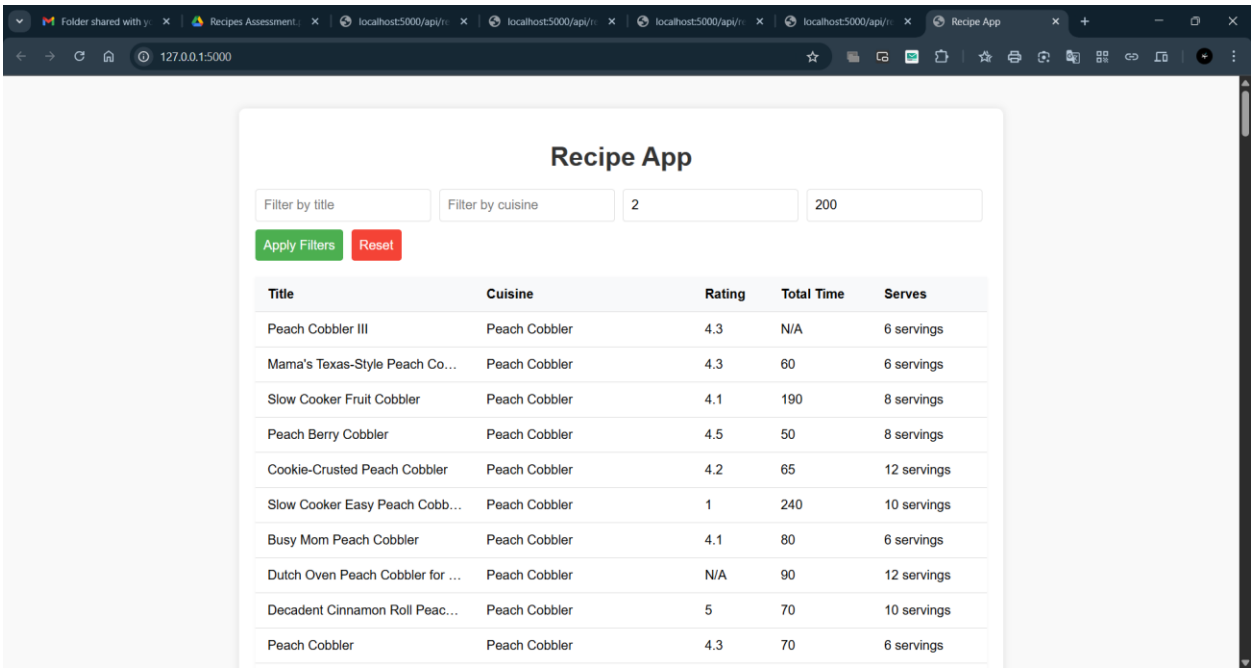


Page and Limiting

GET http://localhost:5000/api/recipes?page=2&limit=5







9. Conclusion

- **Backend:**
  - Uses Flask for routing and API logic.
  - SQLite for data storage.
  - All JSON fields are stored as db using SQLite and parsed back to objects in the API.
- **Frontend:**
  - JavaScript for dynamic content.
  - Responsive CSS for all screen sizes.
  - Fetches data from the backend API using fetch.