**# Recipe Explorer API**

A Flask-based API for searching and filtering recipes with a modern frontend interface.

**## Features**

- Search recipes by cuisine, calories, rating, and cooking time

- Pagination support

- Detailed recipe information display

- Modern and responsive UI

- JSON-based data storage

**## Setup Instructions**

**### System Requirements**

This project requires:

- Python 3.8 or higher (Python 3.10 recommended)

- pip (Python package manager)

- SQLite (automatically included with Python)

- A modern web browser (Chrome, Firefox, Safari, or Edge)

**### Installation Steps**

1. Clone the repository:

```bash

git clone https://github.com/Livingstone2103/Recipe\_Data\_Collection.git

cd Recipe\_Data\_Collection

```

2. Create a virtual environment:

```bash

python -m venv venv

```

3. Activate the virtual environment:

```bash

# Windows:

venv\Scripts\activate

# macOS/Linux:

source venv/bin/activate

```

4. Install dependencies:

```bash

pip install -r requirements.txt

```

5. Initialize the database:

```bash

python app.py

```

6. Load sample data (optional):

```bash

python load\_data.py

```

7. Start the application:

```bash

python app.py

```

The application will be available at `http://127.0.0.1:5000`

**## API Endpoints**

**### 1. GET /api/recipes**

Returns a paginated list of recipes with the following parameters:

- `page`: Page number (default: 1)

- `limit`: Number of items per page (default: 10)

Example usage:

```bash

curl http://127.0.0.1:5000/api/recipes

```

**### 2. GET /api/recipes/search**

Search and filter recipes with multiple parameters:

- `page`: Page number (default: 1)

- `limit`: Items per page (default: 15)

- `cuisine`: Cuisine name (e.g., "Italian")

- `calories`: Calorie filter using operators (<, >, <=, >=)

- `rating`: Rating filter using operators

- `total\_time`: Cooking time filter using operators

Example search:

```bash

curl "http://127.0.0.1:5000/api/recipes/search?cuisine=Italian&calories=<500"

```

**## Frontend Features**

**### Search Filters**

- Cuisine: Text search (e.g., "Italian", "Chinese")

- Calories: Numeric filter with operators

- Rating: Rating filter with operators

- Total Time: Cooking time filter with operators

**### Pagination**

- Displays 15 items per page

- Shows page numbers with ellipsis

- Maintains current page selection

- Shows total results

**### Recipe Display**

Each recipe shows:

- Title (truncated if too long)

- Cuisine

- Rating (0-5 stars)

- Total cooking time

- Number of servings

**## Project Structure**

```

Recipe\_Data\_Collection/

├── app.py              # Main Flask application

├── load\_data.py        # Script to load sample data

├── requirements.txt    # Python dependencies

├── static/

│   ├── css/

│   │   └── styles.css # Frontend styles

│   ├── js/

│   │   └── script.js  # Frontend JavaScript

├── templates/

│   └── index.html     # Main HTML template

└── README.md          # This file

```

**### 2. GET /api/recipes/search**

Search and filter recipes.

Parameters:

- `page`: Page number (default: 1)

- `limit`: Items per page (default: 15)

- `cuisine`: Cuisine name (e.g., "Italian")

- `calories`: Calorie filter with operators (<, >, <=, >=)

- `rating`: Rating filter with operators

- `total\_time`: Cooking time filter with operators

Example:

```bash

curl "http://127.0.0.1:5000/api/recipes/search?cuisine=Italian&calories=<500"

```

**## Frontend Features**

**### Search Filters**

- Cuisine: Text search (e.g., "Italian", "Chinese")

- Calories: Numeric filter with operators

- Rating: Rating filter with operators

- Total Time: Cooking time filter with operators

**### Pagination**

- 15 items per page

- Page numbers with ellipsis

- Maintains current page selection

- Shows total results

**### Recipe Display**

Each recipe shows:

- Title (truncated if too long)

- Cuisine

- Rating (0-5 stars)

- Total cooking time

- Number of servings

**## Project Structure**

```

recipe-explorer/

├── app.py              # Main Flask application

├── load\_data.py        # Script to load sample data

├── requirements.txt    # Python dependencies

├── static/

│   ├── css/

│   │   └── styles.css # Frontend styles

│   ├── js/

│   │   └── script.js  # Frontend JavaScript

├── templates/

│   └── index.html     # Main HTML template

└── README.md          # This file

```

**## Usage Examples**

**### Search Recipes**

1. Search by cuisine:

```

GET /api/recipes/search?cuisine=peach

```

2. Search by calories (less than 500):

```

GET /api/recipes/search?calories=<500

```

3. Search by rating (greater than 4):

```

GET /api/recipes/search?rating=>4

```

**### Pagination**

1. Get second page with 15 items:

```

GET /api/recipes/search?page=2&limit=15

```

**## Error Handling**

The API includes comprehensive error handling:

- Invalid parameters

- Database errors

- Invalid JSON data

- Pagination errors

- Filter validation

**## Contributing**

1. Fork the repository

2. Create your feature branch

3. Commit your changes

4. Push to the branch

5. Create a Pull Request

**## License**

This project is licensed under the MIT License - see the LICENSE file for details.

# Recipe*\_Data\_*Collection