MARIAN COLLEGE
KUTTIKKANAM
AUTONOMOUS
MAKING COMPLETE

PG DEPARTMENT OF COMPUTER APPLICATIONS

SHOPPING LIST MAKER API

USING REST FRAMEWORK & REACT

https://boisterous-choux-f8ab67.netlify.app/

SUBMITTED BY

NAME : SAJAN PAPPACHEN
ROLL NO : 23PMC148
DOS : 25 APRIL 2024

SUBMITTED TO

Mr. SATHEESH KUMAR S
ASST. PROFESSOR

TABLE OF CONTENTS

1. ## SUMMARY OF SHOPPING LIST CREATION APP

   The Shopping List app is designed to help users plan their shopping, promoting cost-effective spending habits. By providing an intuitive interface for creating and managing shopping lists, the app encourages users to plan their purchases and avoid impulse buys. This hassle-free tool allows users to easily add items, update quantities, and monitor costs without requiring registration or login, making it accessible to everyone. The goal is to empower users to save money and shop more efficiently through better planning.

FRONT END LINK : https://boisterous-choux-f8ab67.netlify.app/

2. ## EXPLANATION OF ARCHITECTURE

   The app employs a React front end and a Django REST Framework (DRF) back end. The React front end offers a seamless user experience, featuring interactive components for creating, editing, and deleting shopping list items. The back end, powered by Django, manages data storage and retrieval through a RESTful API, allowing the app to persistently store shopping list items and manage CRUD operations. The architecture supports real-time updates, enabling users to interact with the app without reloading the page, enhancing overall responsiveness and user satisfaction.

3. REASON FOR CHOOSING THE ARCHITECTURE & FRAMEWORK

I have chosen React front end with a Django REST Framework (DRF) back end. For me, this offers several key advantages that make it an excellent choice for this type of application. Here's why these technologies and components were selected:

Front End: React

Interactivity and Responsiveness: React allows the creation of interactive and responsive user interfaces. It uses a virtual DOM, which enables fast updates and smooth user experiences. This is crucial for a shopping list app where users need to quickly add, edit, and delete items without page reloads.

Component-Based Structure: React's component-based architecture allows for modular development. Each part of the interface (e.g., item list, edit form, delete button) can be developed and maintained independently, promoting code reusability and easier debugging.

State Management: React's state management capabilities, through useState and useEffect hooks, make it easier to

handle dynamic data in real-time, like filtering items based on search terms or updating the list upon addition or deletion of an item.

Back End: Django REST Framework

Robustness and Scalability: Django is a mature and scalable framework that provides a strong foundation for building RESTful APIs. It includes built-in authentication, permissions, and other features that enhance security and reliability.

Rapid Development: Django's design philosophy of "batteries included" provides many utilities out of the box, allowing for faster development of back-end services, including CRUD operations and data validation.

RESTful API: DRF makes it simple to create RESTful endpoints for front-end interaction. This design allows for a clear separation of concerns, ensuring that the front end communicates with the back end through well-defined APIs, making the architecture more maintainable and extendable. In summary, the combination of React for the front end and Django REST Framework for the back end creates an architecture that is interactive, responsive, and robust, allowing for a seamless user experience while providing a

scalable and secure back end. This setup supports the primary goal of the project: to provide users with an easy-to-use, efficient tool for creating shopping lists and planning purchases.

4. API END POINTS

1. GET /api/items/
Description: Retrieves a list of all items in the database. Useful for displaying the full shopping list to users.

2. POST /api/items/
Description: Creates a new item in the database. Takes input such as item name, quantity, and expected price.

3. GET /api/items/{id}/
Description: Retrieves a specific item by its unique ID. Useful for getting detailed information about a single item or editing it.

4. PUT /api/items/{id}/
Description: Updates an existing item identified by its ID. Takes input for new values like item name, quantity, and price.

5. DELETE /api/items/{id}/

Description: Deletes an item from the database by its ID. This endpoint removes the item from the shopping list.

6. GET /api/items/search/
Description: Searches for items by a specific query, typically based on the item's name. Useful for implementing search functionality in the front end.

GET Requests: Used to retrieve data without changing the state of the server. These endpoints are typically used to display information or search for specific items.

POST Requests: Used to create new resources on the server. For example, to add a new shopping list item.

PUT Requests: Used to update existing resources. For example, to edit the details of a specific item.

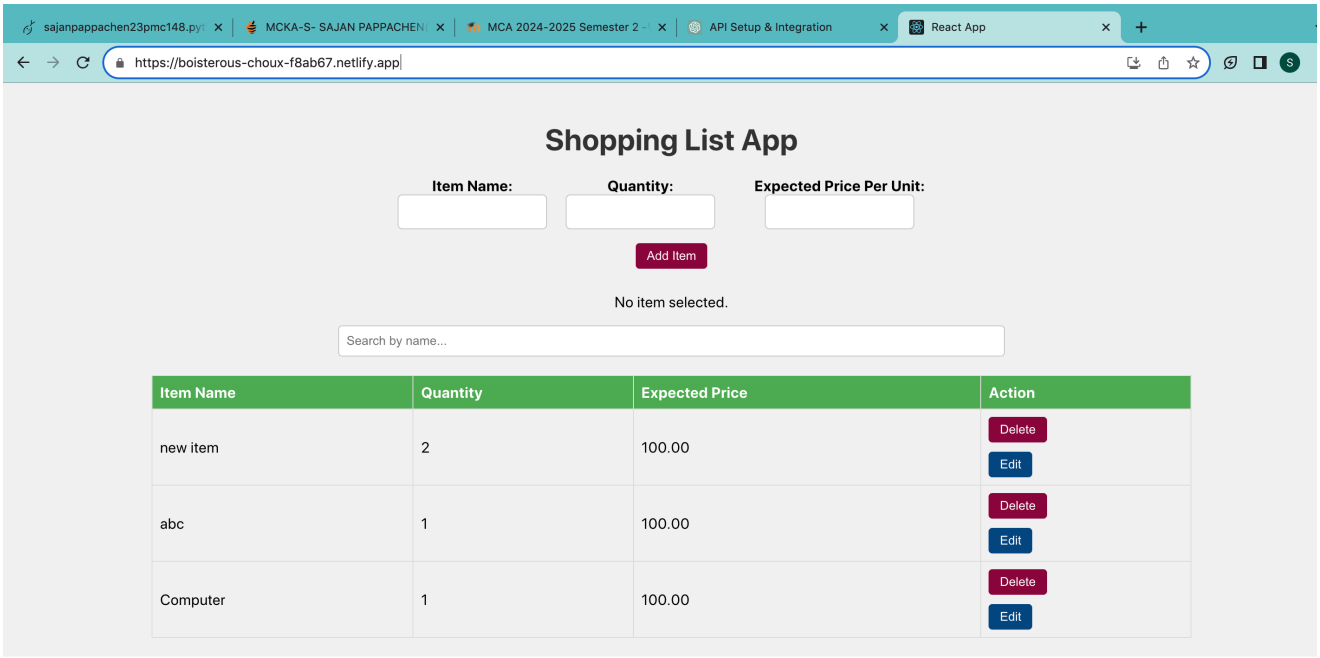
DELETE Requests: Used to remove resources from the server. For example, to delete an item from the shopping list.

Implementation Details

These endpoints are implemented in the Django REST Framework and interact with the back-end database. The front end (in React) communicates with these endpoints to

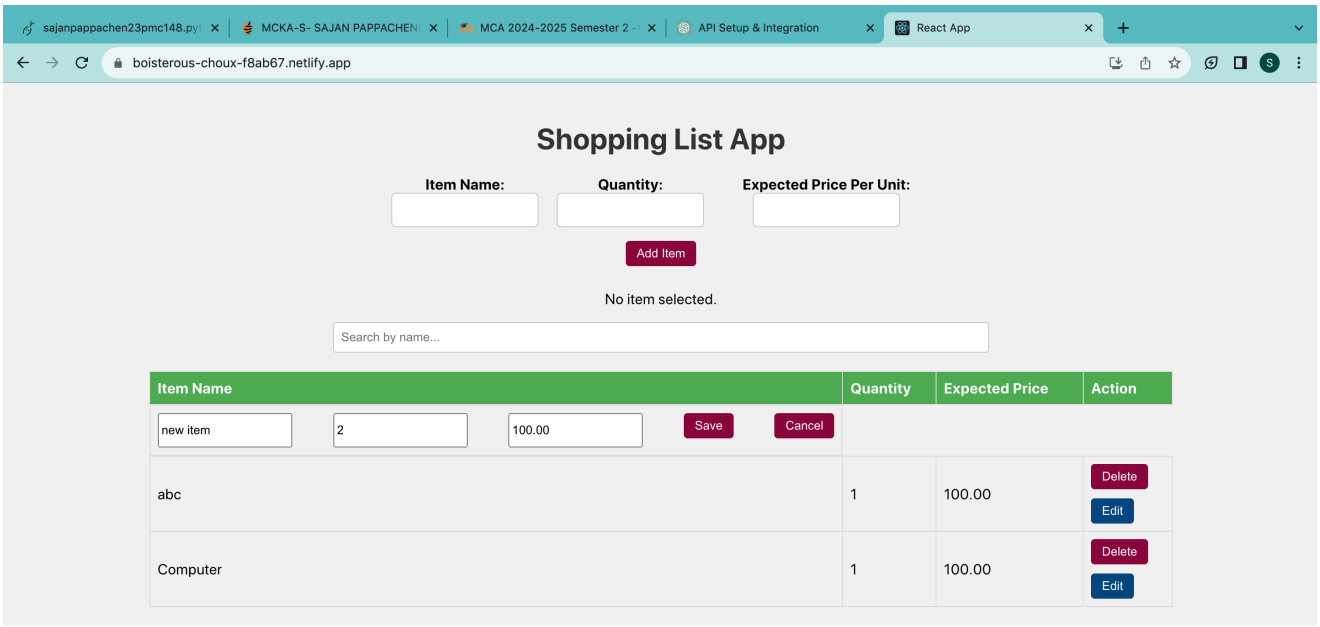
perform various operations, like fetching the list of items, adding new items, updating existing items, and deleting items. This setup enables a clear separation of concerns, making the front end responsible for user interaction and the back end responsible for data management and business logic.

5. SCREENSHOTS

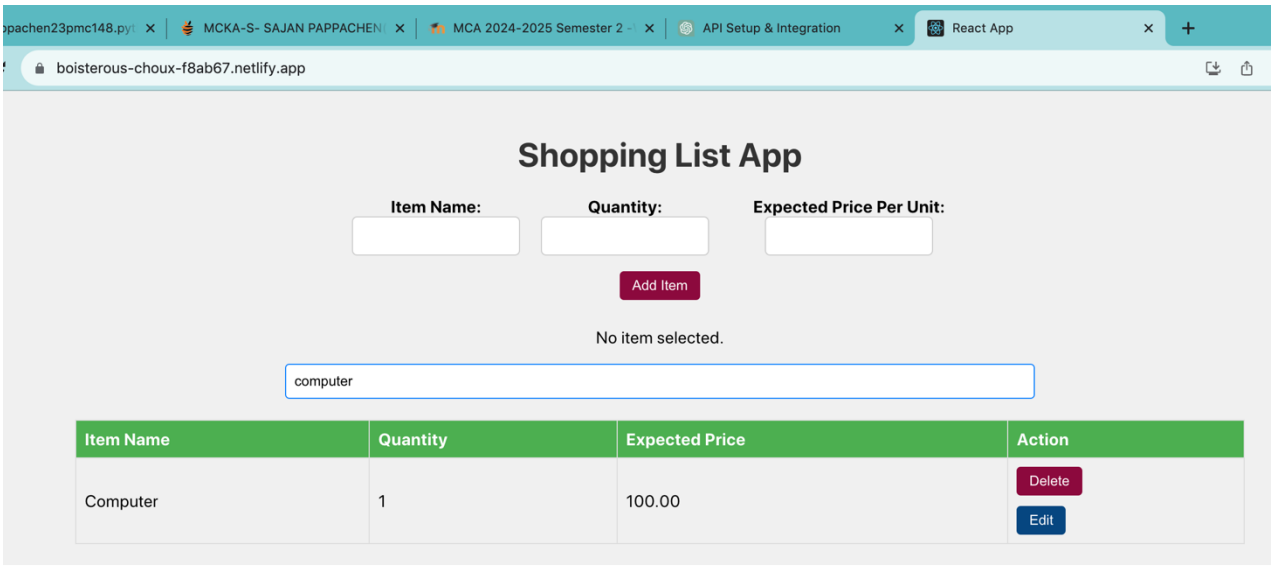


HOSTED FRONT END – LIST OF ITEMS

Link : https://boisterous-choux-f8ab67.netlify.app/



EDIT A SHOPPING ITEM VIEW



SEARCH AN ITEM IN THE LIST

6. SOURCE CODE LINK

Front End : https://github.com/sajanpappi/shoppinglist-REST-API-Frontend

Back End : https://github.com/sajanpappi/shoppinglist-REST-API-Backend

7. HOSTIG DETAILS

The shopping list application's back-end is hosted on PythonAnywhere and the front-end is hosted on Netlify, a platform designed for deploying React applications.

8. FULLY QUALIFIED URL OF DEPLOYED APPLICATION

Link : https://boisterous-choux-f8ab67.netlify.app/

9. REFERENCES

a. https://legacy.reactjs.org/tutorial/tutorial.html

b. https://youtu.be/7AkTv4Sl9S4

c. ChatGPT