PROJECT REPORT OF CO-OP PROJECT AT INDUSTRY (MODULE-1)

ON

Culinary-Crafts-Food-App

submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By:
Ishaan Singla (2210992582)

Submitted To: .
Dr. Preeti Saini

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHITKARA UNIVERSITY
CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB, INDIA

CONTENTS

DECLARATION

We hereby declare that the project work titled, **Culinary-Crafts-Food-App** submitted as part of Bachelor's degree in CSE, at Chitkara University, Punjab, is an authentic record of our own work carried out under the supervision of Dr. Preeti Saini.

Signature(s):

Acknowledgement

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing mean opportunity to undergo Integrated Project as the part of the curriculum.

We also extend our sincere appreciation to **Dr. Preeti Saini** Who provided her valuable suggestions and precious time in accomplishing our Istntegrated project report.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we share dour day-to day experience and received lots of suggestions that improve our quality of work.

1. Abstract

CulinaryCrafts is a MERN-stack (MongoDB, Express.js, React.js, Node.js) web application designed as a complete food companion. It allows users to browse recipes, generate custom recipes instantly using Google Gemini AI, chat in real time with other food enthusiasts via Socket.IO (with image sharing), buy ingredients, manage personal shopping lists, and maintain a functional cart.

The platform features user authentication, persistent chat, and saving of AI-generated recipes, all wrapped in a clean and responsive UI. Built with modularity and scalability in mind, CulinaryCrafts combines modern web technologies and third-party services to deliver an engaging, practical, and enjoyable cooking experience.

Keywords: MERN Stack, Gemini AI, Socket.IO, Real-time Chat, Recipe Generation, Full-Stack Application

2. Introduction

CulinaryCrafts is a full-stack web application designed to be an all-in-one platform for food lovers. It combines recipe discovery, AI-powered recipe generation, social interaction, ingredient shopping, and personalized shopping lists into a single user-friendly app. Built using the MERN stack (MongoDB, Express, React, Node.js), the application aims to make cooking more fun, accessible, and social.

The main goal of this project was to create a practical food application that solves real-world problems like finding recipes, planning grocery shopping, and connecting with other cooking enthusiasts. We included modern features like real-time chat and AI-generated recipes to make it stand out. Whether you're a beginner looking for simple instructions or an experienced cook wanting to share your creations, CulinaryCrafts has something for everyone.

The project was developed by a team of four members and integrates several modern technologies to deliver a smooth and interactive experience.

3. System Requirements

3.1. Hardware Requirements

- **Processor**: Intel Core i5 or equivalent (minimum)
- **RAM**: 8 GB or higher
- **Storage**: At least 1GB free space for development

3.2. Software Requirements

- **Operating System**: Windows 10/11, macOS, or Linux
- **Node.js**: Version 14 or higher
- **Npm**: Version 6 or higher
- **MongoDB**: Local installation or MongoDB Atlas account
- **Browser**: Latest version of Chrome, Firefox, or Edge
- **Code Editor**: VS Code (recommended)

3.3. Additional API Keys

- Google Gemini API key for the AI recipe generation feature

4. Methodology

We followed an iterative development approach throughout the project. The development process was divided into phases:

- **Planning and Requirement analysis**: Discussed features, created user stories, and finalized the tech stack (MERN).
- **Design Phase:** Designed UI wireframes, planned database schema, and decided on component structure in React.
- **Development:** worked in parallel – frontend team focused on React components and UI, backend team set up Express server, MongoDB models, and Socket.IO for real-time chat.
- **Integration**: Connected frontend with backend APIs, integrated Gemini AI for recipe generation.
- **Deployment Preparation**: Tested locally with full setup (frontend + backend).
- We used Git for version control

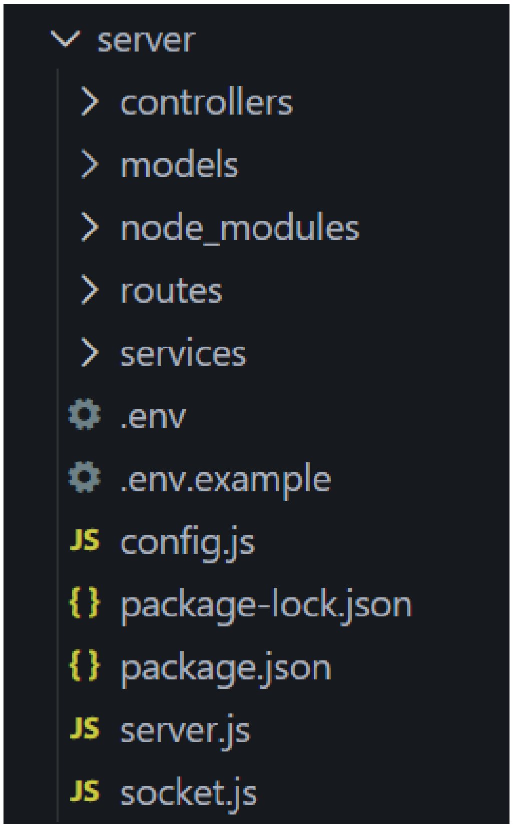
5. Software Design

5.1. The application follows a client-server architecture:

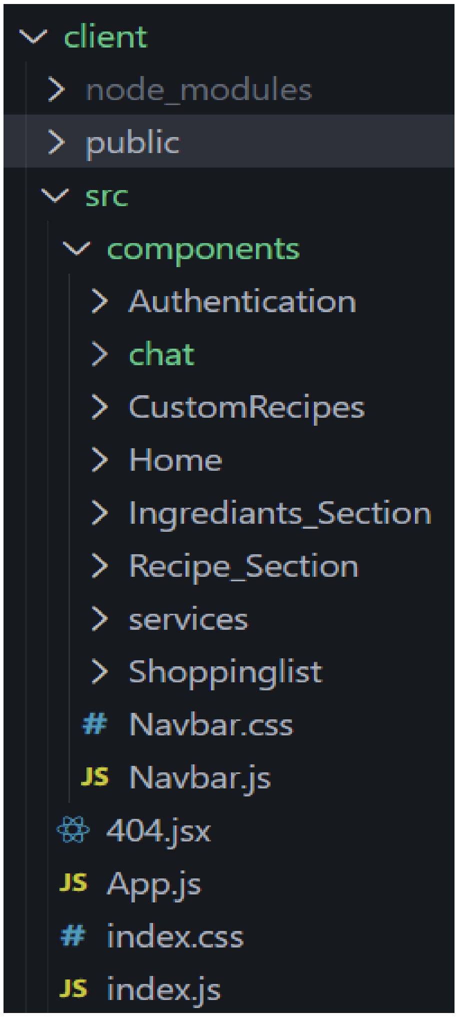
- **Frontend:** React.js with functional components and hooks for state management. We used React Router for navigation between pages (Home, Recipes, Socials, Ingredients, Cart, etc.).
- **Backend**: Node.js with Express.js for creating RESTful APIs. Socket.IO was integrated for real-time bidirectional communication in the chat feature.
- **Database**: MongoDB (NoSQL) for flexible storage of users, recipes, ingredients, and card data.

5.2. Key design patterns:

- Used JWT & Context API for global state (like user authentication and card)



```
v server
    > controllers
    > models
    > node_modules
    > routes
    > services
    .env
    .env.example
    config.js
    package-lock.json
    package.json
    server.js
    socket.js
```



```
v client
  > node_modules
  > public
  v src
    v components
      > Authentication
      > chat
      > CustomRecipes
      > Home
      > Ingrediants_Section
      > Recipe_Section
      > services
      > Shoppinglist
      Navbar.css
      Navbar.js
    404.jsx
    App.js
    index.css
    index.js
```

6. Implementation Details

6.1. Frameworks and Technologies

- **Frontend**: React, React Router , CSS (custom styling), LocalStorage for persisting chat messages and cart items
- **Backend**: Node.JS , Express.js, MongoDB, Mongoose ODM
- **Database :** MongoDB atlas
- **Real-time communication**: Socket.IO
- **AI integration** : Google Gemini API for generating recipes based on user input

6.2. Core Logic

- **User Authentication**: we implemented JWT-based authentication, but for simplicity in demo, some parts use session/local storage.
- **AI recipe Generation**:

  When a user wants a custom recipe, they simply type what they feel like cooking (e.g., "chicken biryani with less spice" or "vegan chocolate cake") in the Generate Recipe section.

Here's what happens step-by-step behind the scenes:

- The user types their request and clicks "Generate".
- The frontend (React) captures the input text and sends it to our backend via a POST request to the API
- On the backend (Express route), we take the user's input and create a proper prompt for the Gemini AI. We append extra instructions to make the output structured and cooking-friendly, something like:

  "Generate a complete recipe for [user-input]. Include a title, list of ingredients with quantities, and numbered step-by-step cooking instructions. Make it beginner-friendly"

- This crafted prompt is sent to Google's Gemini API using their official endpoint and our API key.
- Gemini responds with a nicely formatted recipe in text/markdown format.

        - The backend receives this response and sends it back to the frontend.
        - The React component displays the generated recipe beautifully with proper headings, bullet points for ingredients, and numbered steps.
        - The whole process usually takes 2-4 seconds and feels instant to the user.

- **Real-time Chat with Socket.IO**:

  The socials section has a community chat where users can talk and share pictures of their food in real time.

  Here’s how it works:

    - When any user opens the Socials page, their browser creates a WebSocket connection to our Node.js server using Socket.IO.
    - Whenever a user types a message or uploads an image and presses send:
        - The message (along with username and optional image URL) is sent directly through the Socket.IO connection (not a normal HTTP request).
        - The server receives this event (sendMessage) and immediately broadcasts it to all connected users using io.emit('receiveMessage', data)
    - Every user’s browser that is currently on the Socials page receives the new message instantly and appends it to the chat window.
    - To make messages survive page refresh, we also save every new message in the browser’s localStorage. On page load, the app reads from localStorage and shows the previous chat history.

- **Database Design and schema:**
    - **User Collection**

      Stores everyone who signs up or logs in

        - Username: unique name you choose
        - Email: email (must be valid and unique)
        - Password: stored after hashing , this is used for login/sign up and to know who is doing what in the app.
    - **Custom-recipe Collection**

      Saves all the recipes that users generate using the AI

        - Username: who generated this recipe

    - Recipe_name : title of the dish
    - Generated_content : the full recipe text that Gemini returned
    - Instructions : extra notes or steps
    - Dietary_preference : vegetarian / non-veg / vegan / flexitarian (we have a dropdown ready for this)
    - createdAt : when it was generated
- **Todos Collection**

  Powers the personal Shopping List / To-do checklist.

    - todo → the item name (e.g., “2 tomatoes”, “500g chicken”)
    - completed → true/false when you tick it off
    - Each user gets their own list that they can check off while shopping or cooking.
- **Social Collection**

  Stores every message and picture posted in the real-time chat (Socials section).

    - user → username of the person who sent it
    - message → the text they typed
    - picture → uploaded picture (if any)

7. Testing and Validation

We performed multiple types of testing

- **Unit Testing**: Tested individual React components and API endpoints using console logs and manual checks.
- **Integration Testing**: verified that frontend successfully communicates with backend (e.g. adding to cart updates database).
- **Functionality Testing**: Tested all major features – recipe browsing, AI generation, chat persistence, adding/removing from cart.
- **Responsive Testing**: Checked UI on different screen sizes using chrome DevTools.
- **Edge Cases**: Empty inputs in AI generation, no internet for local storage features, multiple users chatting simultaneously.

8. Results

The application successfully delivers all planned features:

- Users can browse and view detailed recipes.
- AI-powered recipe generation works effectively using Gemini API.
- Real-time chat functions smoothly with image upload support.
- Shopping and cart system allows easy ingredient management.
- Responsive design works well on mobile and desktop.

The project demonstrated successful integration of MERN stack with third-party services (Gemini and Socket.IO).

9. Conclusion

CulinaryCrafts turned out to be a comprehensive food application that combines discovery, creation, shopping, and community interaction in one platform. We successfully built a functional, interactive, and visually appealing app that meets the initial objectives.

10. Future Scope

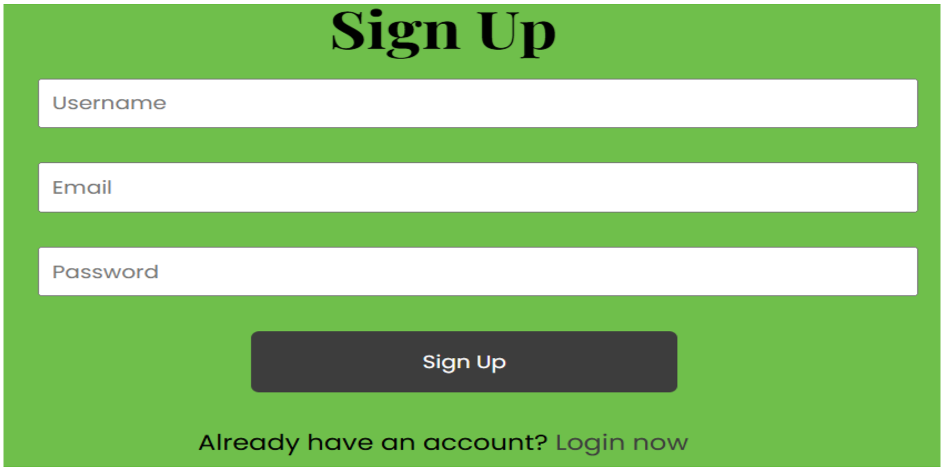
- Allow users to upload and share their own recipes in some privacy folder.
- Implement actual payment gateway for checkout.
- Add recipe rating and comment system.
- Integrate push notifications for chat.
- Mobile app version using React Native.
- Recommendation engine based on user preferences.

11.References

- Gemini API : https://ai.google.dev/gemini-api/docs
- reactJS : https://react.dev/learn
- MongoDB : https://www.mongodb.com/docs/
- SocketIO : https://socket.io/docs/v4
- JWT : https://developers.docusign.com/platform/auth/jwt/
- Mongoose : https://mongoosejs.com/docs/
- expressJS : https://expressjs.com/
- NodeJS : https://nodejs.org/api/all.html

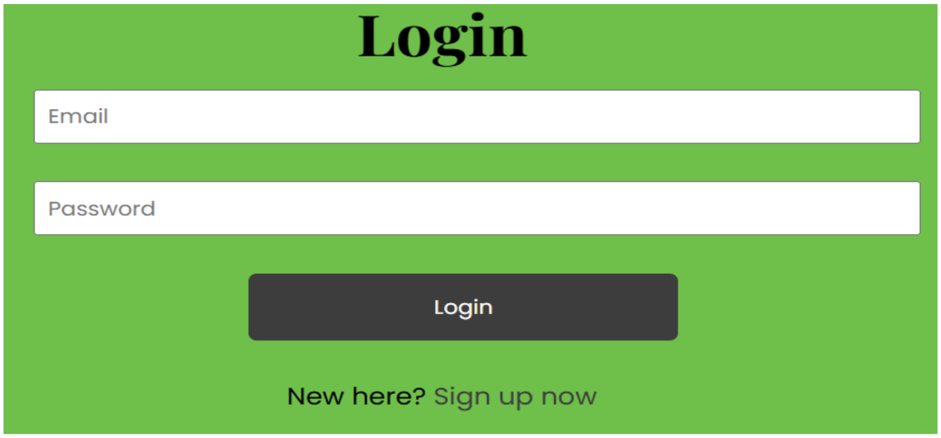
12.Appendices

12.1. Appendix A : Screenshots



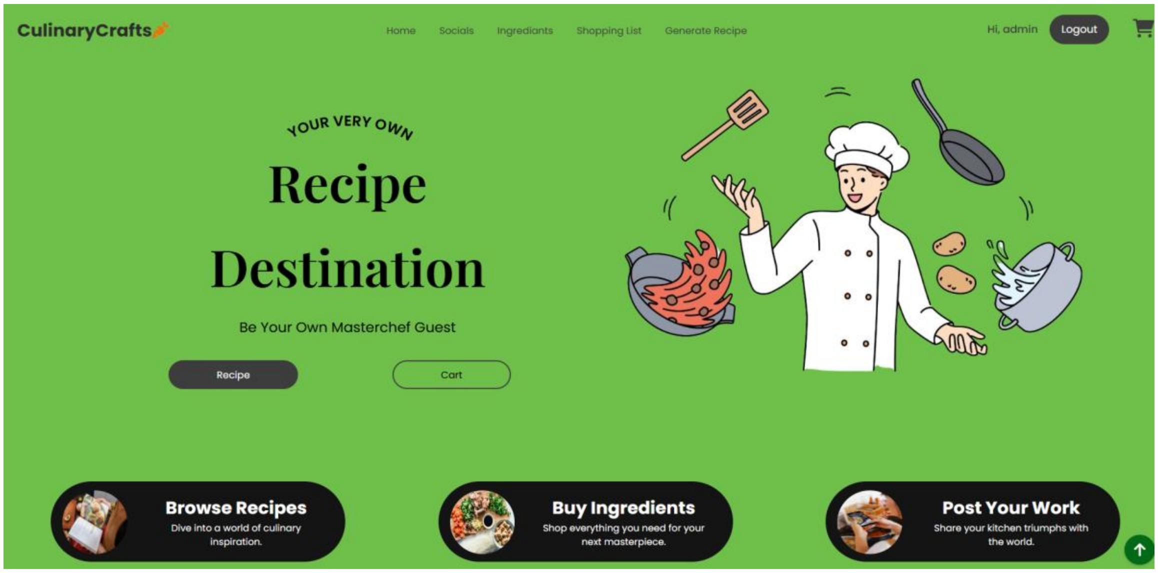
Sign Up

Already have an account? Login now



Login

New here? Sign up now



YOUR VERY OWN

Recipe

Destination

Be Your Own Masterchef Guest

Browse Recipes

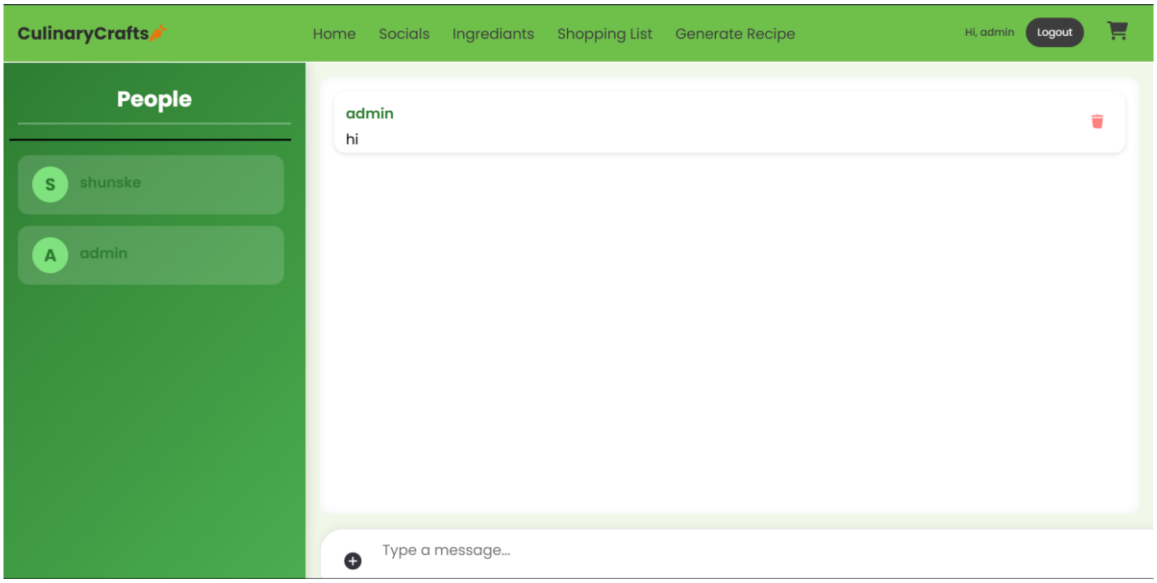
Dive into a world of culinary inspiration.

Buy Ingredients

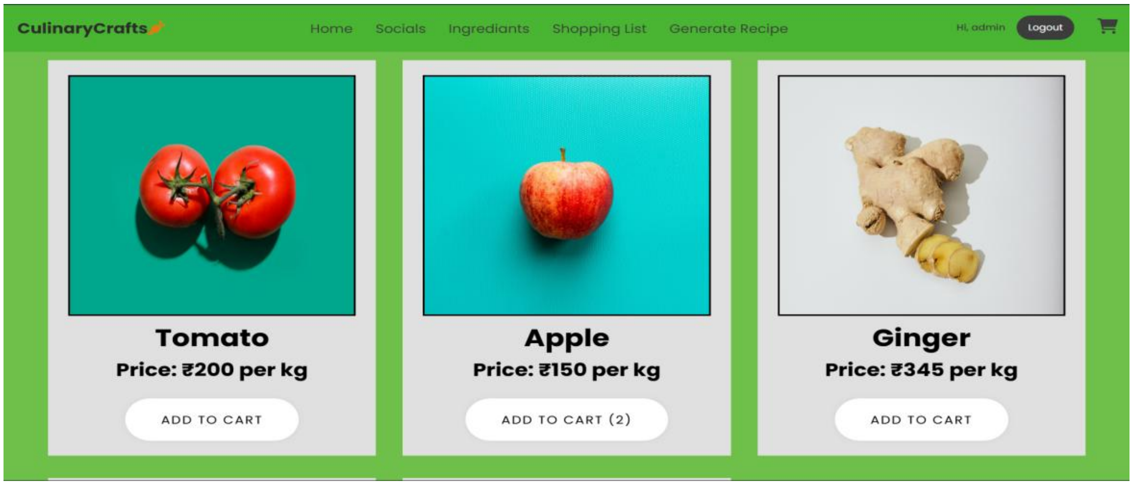
Shop everything you need for your next masterpiece.

Post Your Work

Share your kitchen triumphs with the world.



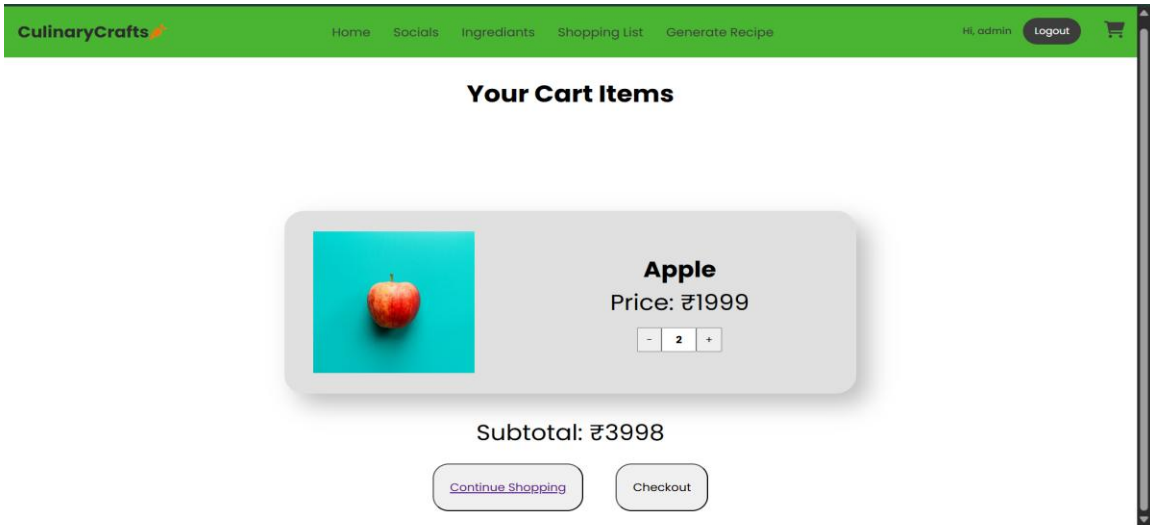
admin
hi



Tomato
Price: ₴200 per kg
ADD TO CART

Apple
Price: ₴150 per kg
ADD TO CART (2)

Ginger
Price: ₴345 per kg
ADD TO CART

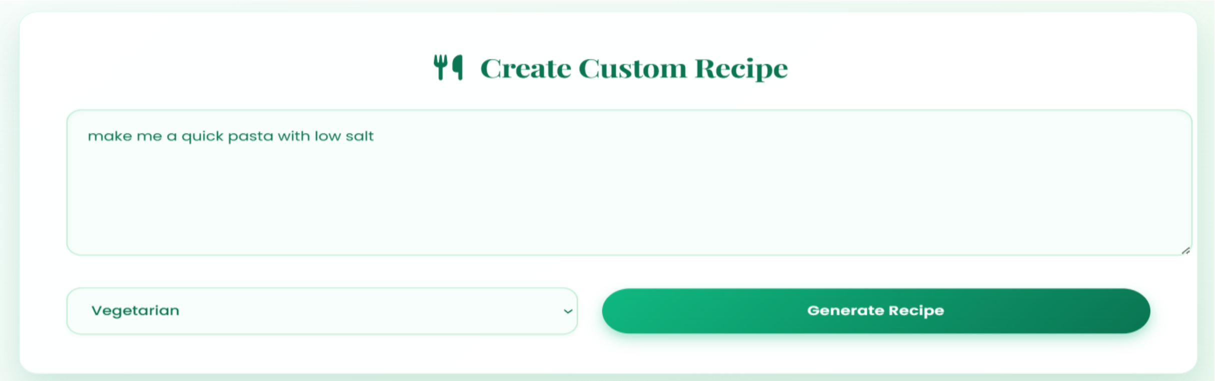


Your Cart Items

Apple
Price: ₴1999
- 2 +

Subtotal: ₴3998

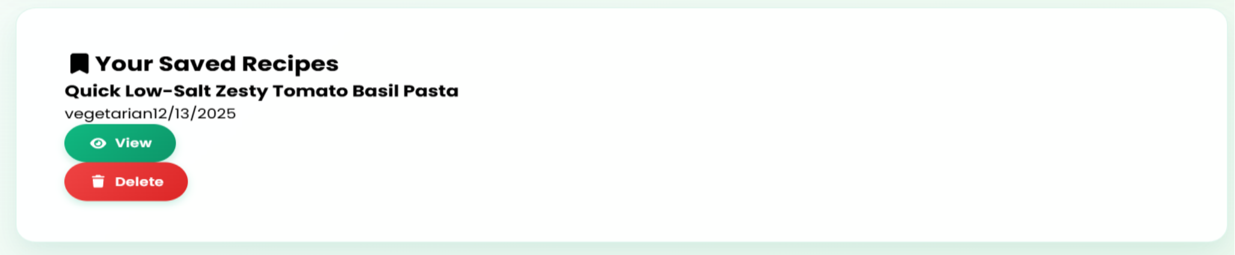
Continue Shopping
Checkout



Create Custom Recipe

make me a quick pasta with low salt

Vegetarian



Your Saved Recipes

Quick Low-Salt Zesty Tomato Basil Pasta

vegetarian12/13/2025

12.2. Appendix B : Github Repository

- **Frontend** :
  - ➢ https://github.com/Ishaan282/CulinaryCrafts-food-app/tree/main/client

- **Backend**:
  - ➢ https://github.com/Ishaan282/CulinaryCrafts-food-app/tree/main/server