



Culinary Canvas

The is submitted in partial fulfillment
of the requirements of the degree of
Master of Computer Applications
(MCA)

by

Darshan Jain (2023-M-11081998)

Under the Supervision of
Prof. Devyani Kamble



May 2025

School of Engineering

Ajeenkya DY Patil University, Pune



AJEENKYA
D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY

School of
Engineering

May 13, 2025

CERTIFICATE

This is to certify that the dissertation entitled “**Culinary Canvas**” is a bonafide work of “**Darshan Jain (2023-M-11081998)**” submitted to the School of Engineering, **Ajeenkya D Y Patil University, Pune** in partial fulfilment of the requirement for the award of the degree of “**Master of Computer Applications**”.

Prof. Devyani Kamble

Supervisor

Internal-Examiner/s

External Examiner

Dr. Prashant Kumbharkar

Dean-School of Engineering

Prof. Devyani Kamble May 13, 2025 Assistant Professor

Supervisor's Certificate

This is to certify that the dissertation entitled **“Culinary Canvas”** submitted by **Darshan Jain(2023-M-11081998)** , is a record of original work carried out by him/her under my supervision and guidance in partial fulfillment of the requirements of the degree of **Master of Computer Applications** at **School of Engineering, Ajeenkya DY Patil University, Pune, Maharashtra-412105**. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Prof. Devyani
Kamble
Supervisor

Dr. Uttam
Deshmukh
HOD

Declaration of Originality

I, **Darshan Jain (2023-M-11081998)**, hereby declare that this dissertation entitled “**Culinary Canvas**” presents my original work carried out as a master student of School of Engineering, Ajeenkya D Y Patil University, Pune, Maharashtra. To the best of my knowledge, this dissertation contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of Ajeenkya D Y Patil University, Pune or any other institution. Any contribution made to this project by others, with whom I have worked at Ajeenkya D Y Patil University, Pune or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections “Reference” or “Bibliography”. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I am fully aware that in case of any non-compliance detected in future, the Academic Council of Ajeenkya D Y Patil University, Pune may withdraw the degree awarded to me on the basis of the present dissertation.

Date: May 14, 2025

Place: Lohegaon, Pune

Darshan Jain

Acknowledgement

I would like to express our sincere gratitude to Prof. Devyani Kamble for her unwavering support, invaluable guidance, and profound mentorship throughout the development journey of the Culinary Canvas. Prof. Devyani Kamble's deep expertise, keen insights, and constructive feedback have been the cornerstone of our project's success.

Her dedication to nurturing our understanding, refining our methodologies, and fostering a collaborative environment has significantly enriched our learning experience and propelled us towards achieving our goals. And also I want to extend my gratitude to Dr. Uttam Deshmukh and Prof. Smita Jain.

Additionally, our warm thanks extend to Ajeenkya D Y Patil University, Pune, Maharashtra for providing us with the opportunity to carry out this prestigious project and enhance our learning in various technical fields.

Date : May 13, 2025

Darshan Jain (2023-M-11081998)

Place: Lohegaon, Pune

Index

Sr. No.	Contents	Page No.
Chapter 1	INTRODUCTION	8
	1.1 Existing System	9
	1.2 Problem Definition- Need of Computerization	9
Chapter 2	PROPOSED SYSTEM	10
	2.1 Proposed System	10
	2.2 Objectives of System	12
	2.3 Operating Environment – Hardware and Software	15
Chapter 3	ANALYSIS AND DESIGN	16
	3.1 Module List	16
	3.2 Flow Chart, ERD (ERD ,Flow chart Diagram)	17
	3.3 Screen Shots	19
	3.4 Source Code	22
Chapter 4	USER MANUAL	36
	4.1 User Manual	36
	4.2 Menu Explanation	37
Chapter 5	CONCLUSION	39

	5.1 Limitations & Drawbacks	43
	5.2 Future Enhancement	44
	5.3 Conclusion	46
	5.4 References & Bibliography	47

Chapter 1

INTRODUCTION

Culinary Canvas is a user-friendly web application made for people who love food and want to share or discover new recipes. This platform lets users create, save, and manage recipes in a clean and organized way. Whether you are a beginner or an experienced cook, you can easily upload your recipes, edit them anytime, and even mark your favorites. The site is built to be easy to use on all devices, from desktops to phones.

Users can view recipe cards that show cooking time, ingredients, and instructions in a neat layout. They can also delete or edit their own recipes and use a heart icon to favorite dishes they like. The design is simple but modern, with responsive cards and attractive animations. It helps users not only store their recipes but also enjoy the visual experience of browsing through them, just like flipping through a digital cookbook.

1.1 Existing System

Traditional recipe sharing and management is often done through blogs, social media platforms, or handwritten notes. While these platforms may allow for posting and browsing, they lack structure, organization, and ease of use for day-to-day recipe storage or retrieval. Most users face challenges in searching, editing, or sorting through multiple recipes. Furthermore, these methods lack interactive features such as favoriting recipes or managing user accounts for a personalized experience. As a result, recipe sharing remains fragmented and inefficient for many cooking enthusiasts.

Some food apps and websites exist, but they are often packed with ads or locked behind subscription models. Many don't let users add or manage their own recipes easily. There's also limited support for editing or organizing recipes into personalized categories. The need for a simple, user-centered platform that offers both flexibility and a clean, responsive design is what this project aims to address.

1.2 Problem Definition – Need of Computerization

Modern web development tools enable rich user experiences and can solve common issues in recipe sharing. The main challenges this system addresses are:

1. Organization:

Without a centralized system, recipes are often lost, scattered across devices, or written on paper. This app allows users to manage, edit, and browse recipes efficiently from one place.

2. Accessibility:

Responsive design ensures users can access their recipe collection from desktops, tablets, and smartphones, making cooking more convenient anytime, anywhere.

3. User Interaction:

Features like "Add to Favorites," profile management, and recipe cards with images enhance user experience and encourage regular engagement.

4. Personalization:

Users can edit and update their own recipes, making it easy to adjust ingredients or steps based on personal taste.

5. Visual Appeal:

Unlike plain-text formats, this system offers visually engaging cards and animations, turning cooking into a more inspiring digital experience.

Chapter 2

PROPOSED SYSTEM

2.1 Proposed System

The proposed system is a responsive web application that allows users to create, view, manage, and share cooking recipes in an organized and user-friendly manner. It removes the need for unstructured recipe storage and replaces it with a modern interface where users can interact with the content easily. The system supports features like marking favorite recipes, editing shared content, and deleting outdated posts. It aims to make cooking and recipe sharing a smooth and enjoyable experience for all users.

Key features of the proposed system include:

1. Recipe Submission:

Users can add their own recipes with ingredients, steps, and an image. Each recipe can be saved and later updated or removed based on the user's needs.

2. Recipe Browsing & Cards:

Recipes are displayed in stylish, responsive cards. These cards can expand to show detailed ingredients and instructions when clicked.

3. Favorites System:

Users can mark recipes as favorites. Favorite data is stored locally, allowing easy access and updates.

4. Edit and Delete:

Recipes can be edited or removed anytime, especially in the user's profile view. This gives full control over shared content.

5. **Responsive Design:**

The layout adjusts from 3 cards to 2 or 1, depending on screen size. This ensures smooth viewing on laptops, tablets, and phones.

6. **Interactive UI Effects:**

Features such as hover animations, expanded content display, custom buttons, and visually enhanced elements improve the user experience and engagement.

7. **Scalable and Modular:**

The system is built with scalable front-end tools and structured component design, making future enhancements easier.

Overall, this recipe management system helps users store and share cooking recipes in a simple, engaging, and responsive platform. It improves the cooking and sharing experience by offering features that make food-related content easier to manage and interact with.

2.2 Objectives of System

The objectives of the proposed Recipe Sharing Web Application are as follows:

1. Simplify Recipe Management:

The system allows users to easily create, edit, view, and delete recipes. Each recipe can include a title, image, ingredients, and cooking steps, helping users manage their collection effectively.

2. Enhance User Engagement:

Attractive cards and interactive hover effects make the browsing experience enjoyable. Features like typewriter animations and visual highlights help retain user interest.

3. Support Responsive Browsing:

The layout adjusts between one, two, or three cards depending on screen size. This makes the platform accessible and smooth on mobiles, tablets, and desktops.

4. Allow Favorites and Personalization:

Users can mark recipes as favorites and store them locally. This provides personalized access to frequently used or liked recipes.

5. Encourage Sharing and Updates:

Users can share their unique recipes and edit or update them anytime. This helps foster a sense of contribution and collaboration among users.

6. Provide a Clean UI/UX:

The platform uses a simple, intuitive interface with clear buttons and organized sections. This makes it easy for users of all ages to navigate and use the system.

7. Improve Access and Control:

Users can view recipes in a list or card view, expand them for details, and manage content efficiently. This offers control and flexibility.

8. Use Lightweight Technologies:

The system relies on lightweight tools like local storage and basic JavaScript, ensuring fast loading, low memory usage, and good performance without server-side complexity.

Requirements

To ensure the proposed system meets the needs of its users effectively, it's essential to define the requirements from the user's perspective. These requirements cover the features and functionalities expected by users when using a modern recipe-sharing web application. Below are the key user requirements:

1. User-Friendly Interface

- Easy navigation with clearly labeled buttons and sections.
- Clean layout that helps users focus on recipes and content.

2. Responsive Design

- Fully functional across devices including mobiles, tablets, and desktops.
- Adjusts card layout automatically to one, two, or three columns based on screen size.

3. Recipe Management

- Users can add, edit, delete, and view recipes.
- Recipe fields include title, ingredients, preparation steps, and an image.

4. Search and Filter Functionality

- Ability to search recipes by title or ingredient.
- Filter recipes by category, date added, or favorites.

5. Favorites and Bookmarking

- Users can mark favorite recipes.
- Favorites are stored locally for easy future access.

6. Attractive Visual Effects

- Hover animations and visual highlights make the UI engaging.
- Custom hover effects on profile images and buttons improve interactivity.

7. Sharing Features

- Ability to share recipes with others.
- Buttons for copying or sending recipe links directly.

8. Data Storage and Security

- Local data storage for user entries.
- Simple and secure handling of user-generated content.

9. Feedback and Interaction

- Option to add comments or feedback on recipes.
- Expandable sections to reveal full content in an organized way.

10. Scalability and Performance

- Lightweight design ensures quick loading even with many recipes.
- Designed to support increased usage and new features over time.

Overall, these user requirements aim to provide a seamless, engaging, and feature-rich experience that encourages users to cook, share, and explore a wide variety of recipes.

2.3 Operating Environment – Hardware and Software

The successful deployment and performance of the Recipe Sharing Web Application depend on compatibility with specific hardware and software configurations. Below is an overview of the recommended operating environment to ensure smooth development, testing, and usage.**2.4.1**

2.4.1 Software and Hardware Requirements

Software Requirements:

- Operating System: Compatible with Windows, macOS, or Linux
- Programming Languages: HTML, CSS, JavaScript
- Frontend Framework: Bootstrap (for responsive design and UI components)
- Code Editor or IDE: Visual Studio Code (or any modern text editor)
- Version Control: Git with GitHub (for version tracking and collaboration)
- Browser: Chrome, Firefox, or Edge (latest versions recommended)

Hardware Requirements:

- Processor: Intel Core i5 or AMD Ryzen 5 (or higher)
- RAM: Minimum 8 GB (recommended 16 GB for smoother development)
- Storage: Minimum 256 GB SSD (recommended 512 GB or more)
- Graphics: Integrated or basic dedicated graphics for UI testing
- Display: 1080p resolution for better UI design visibility

This configuration ensures optimal performance for both development and user-side deployment of the web application, supporting responsiveness, fast rendering, and smooth interface interactions.

CHAPTER 3

ANALYSIS AND DESIGN

Analysis and Design for the Recipe Sharing Web Application focuses on understanding the needs of users who wish to share, browse, and save cooking recipes. The analysis helps identify how users interact with recipes, what data they need, and how to present it in an intuitive way. The design process then builds the structure, layout, and features to ensure smooth navigation, good performance, and a visually appealing experience. The system is designed to be responsive and user-friendly for a wide range of users.

3.1 Module List

To develop the application efficiently, the system is broken down into smaller functional modules. Each module has a specific role to ensure the application is organized, scalable, and easy to manage. Below is the proposed module list:

1. User Management Module:

- **Manages user registration, login, and profile settings.**
- **Includes secure authentication, password recovery, and session handling.**

2. Recipe Submission Module:

- Allows users to submit new recipes using a form.
- Supports title, ingredients, preparation steps, and image uploads.

3. Recipe Display Module:

- Displays recipes in a card layout using responsive design.
- Includes filtering, sorting, and viewing recipe details.

4. Recipe Sharing Module:

- Enables sharing of recipes through social platforms or links.

- Integrates WhatsApp API for easy direct sharing.

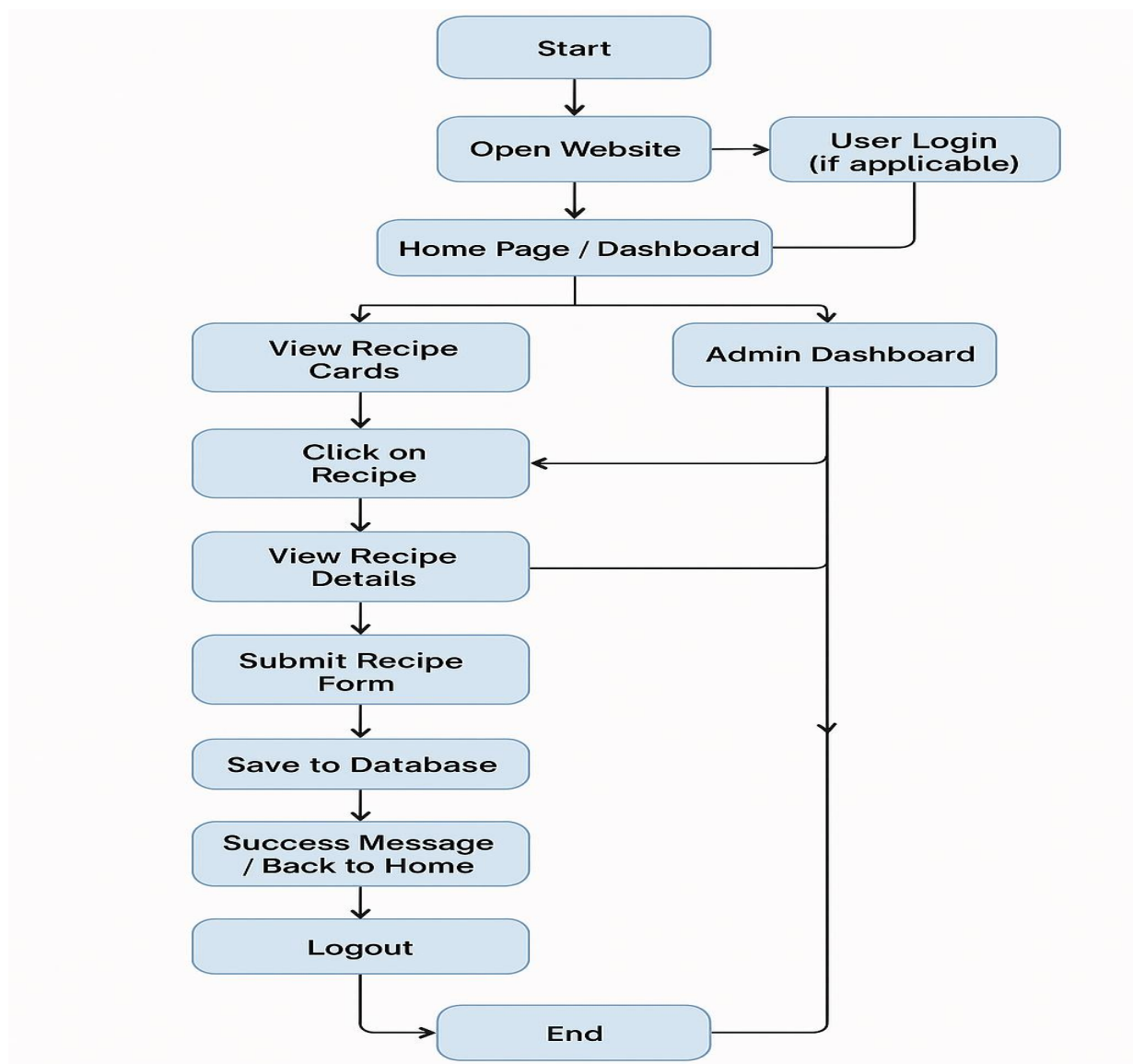
5. Like Module:

- Allows users to like recipes .
- Enhances community engagement and feedback.

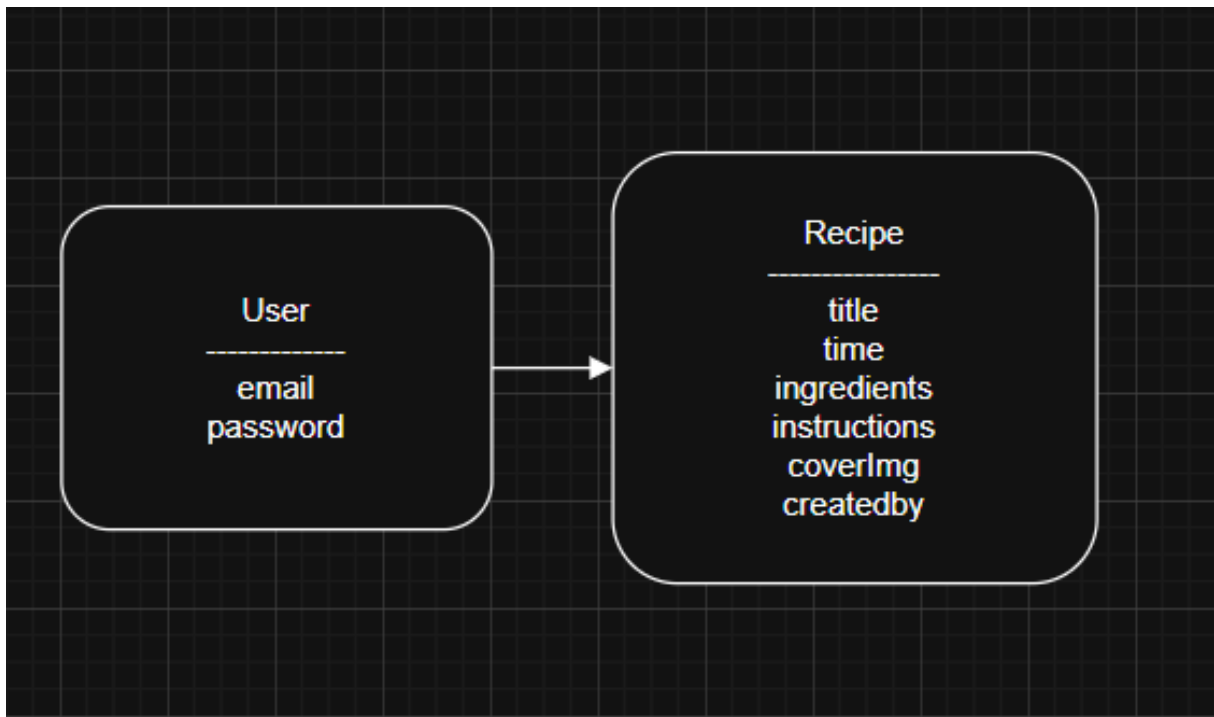
6. Responsive UI Module:

- Ensures the layout adjusts from 3 to 2 to 1 cards depending on screen size.
- Enhances mobile usability and experience.

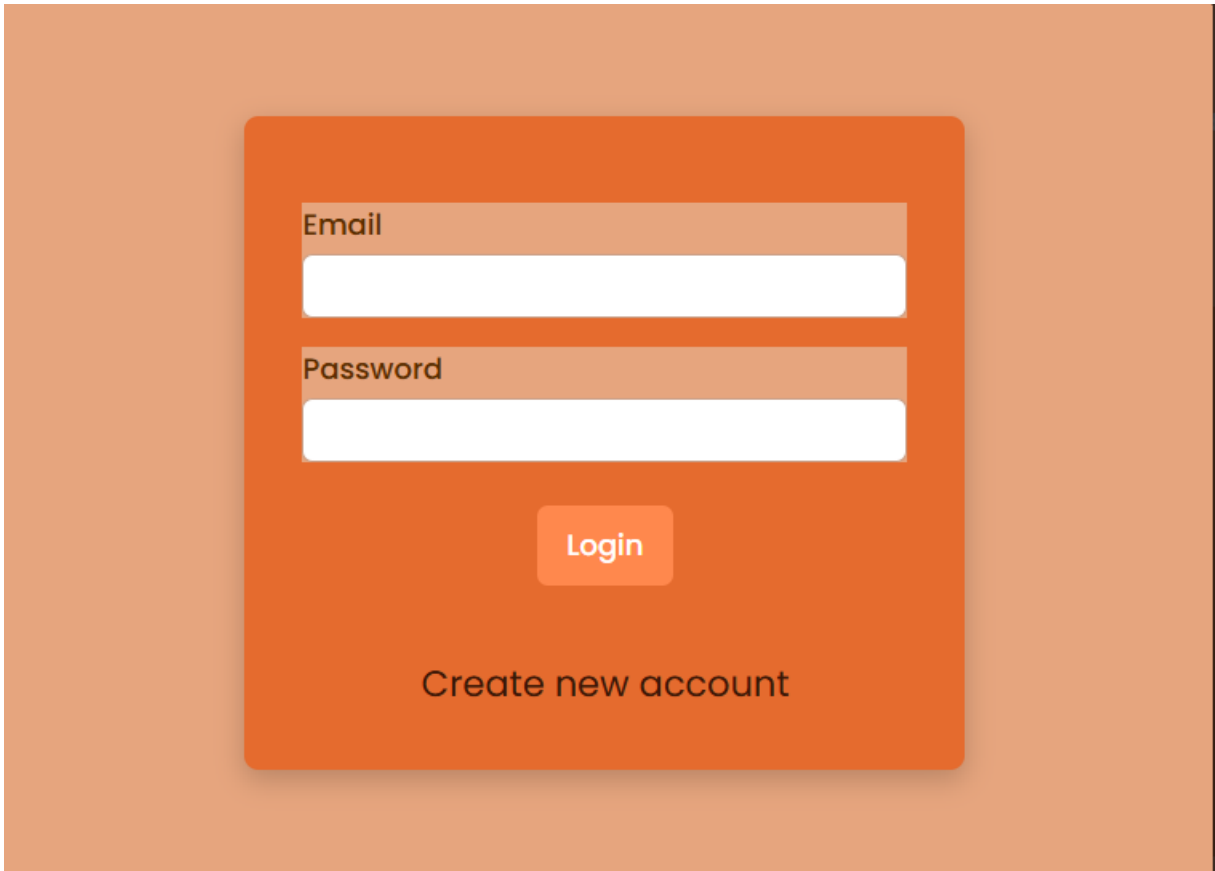
3.2.2 Flow Chart



3.2.4 ERD DIAGRAM

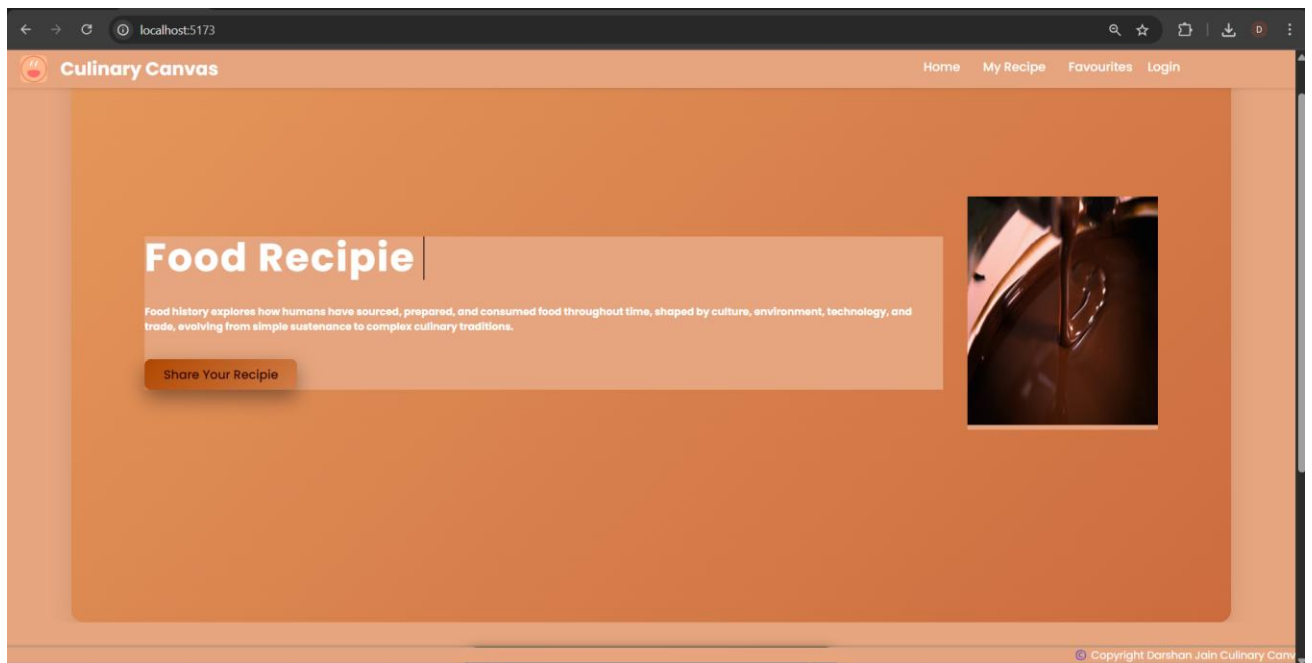


3.3.1 Screen Shot – Login Screen

A login screen mockup with an orange background. In the center is a darker orange rounded rectangle containing two white input fields. The top field is labeled 'Email' and the bottom field is labeled 'Password'. Below these fields is an orange 'Login' button. At the bottom of the central rectangle is the text 'Create new account'.

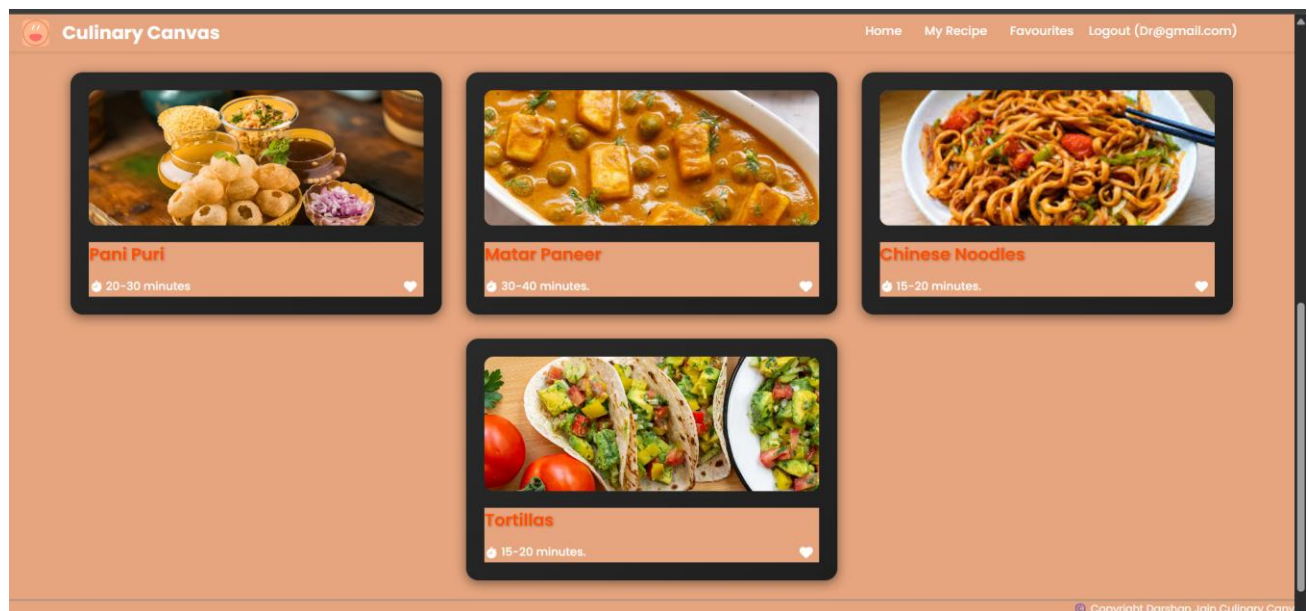
Login Screen

3.3.2 Screen Shot – Home Page



Home Page

3.3.3 Screen Shot – Cards View



Cards View

3.3.4 Screen Shot – Add Recipe

The screenshot shows the 'Add Recipe' form in the Culinary Canvas application. The form is centered on a light orange background. It contains the following fields and elements:

- Title:** A text input field.
- Time:** A text input field.
- Ingredients:** A large text area for listing ingredients.
- Instructions:** A large text area for providing cooking instructions.
- Recipe Image:** A section with a 'Choose File' button and the text 'No file chosen'.
- Add Recipe:** An orange button at the bottom of the form.

The top navigation bar includes the 'Culinary Canvas' logo and links for 'Home', 'My Recipe', 'Favourites', and 'Logout (Dr@gmail.com)'. A copyright notice 'Copyright Darshan Jain Culinary Canvas' is visible in the bottom right corner.

Inserting new Recipe

3.3.5 Screen Shot – Edit Recipe

The screenshot shows the 'Edit Recipe' form in the Culinary Canvas application. The form is centered on a light orange background. It contains the following fields and elements:

- Title:** A text input field containing 'Chinese Noodles'.
- Time:** A text input field containing '15-20 minutes'.
- Ingredients:** A large text area for listing ingredients.
- Instructions:** A large text area for providing cooking instructions.
- Recipe Image:** A section with a 'Choose File' button and the text 'No file chosen'.
- Edit Recipe:** An orange button at the bottom of the form.

The top navigation bar includes the 'Culinary Canvas' logo and links for 'Home', 'My Recipe', 'Favourites', and 'Logout (Dr@gmail.com)'. A copyright notice 'Copyright Darshan Jain Culinary Canvas' is visible in the bottom right corner.

Edit Recipe

3.4 Source Code

App.jsx

```
import React from "react";
import './App.css'
// import './Main.css'
import Home from './pages/Home'
import {createBrowserRouter,RouterProvider} from "react-router-dom"
import MainNavigation from "./components/MainNavigation";
import axios from "axios";
import AddFoodRecipie from "./pages/AddFoodRecipie";
import EditRecipie from "./pages/EditRecipie";
const getAllRecipies=async()=>>{
  let allRecipies=[]
  await axios.get('http://localhost:5000/recipie').then(res=>{
    allRecipies=res.data
  })
  return allRecipies
}

const getMyRecipie=async()=>>{
  let user = JSON.parse(localStorage.getItem("user"))
  let allRecipies=await getAllRecipies()
  return allRecipies.filter(item=>item.createdBy===user._id)
}

const getFavRecipies=()=>>{
  return JSON.parse(localStorage.getItem("fav"))
}

const router= createBrowserRouter([
  {path:"/",element:<MainNavigation/>,children:[
    {path:"/",element:<Home/>,loader:getAllRecipies},
    {path:"/myRecipie",element:<Home/>,loader:getMyRecipie},
    {path:"/favRecipie",element:<Home/>,loader:getFavRecipies},
    {path:"/addRecipie",element:<AddFoodRecipie/>},
    {path:"/editRecipie/:id",element:<EditRecipie/>}
  ]}
])
export default function App(){

  return(
    <>
    <RouterProvider router={router}></RouterProvider>
    </> )}
```

App.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');

* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
  background-color: #e6a57e
  color: #fff;
}
.profileImg:hover {
  transform: scale(1.1) rotate(3deg);
  filter: brightness(1.2) contrast(1.1);
  box-shadow: 0 0 20px #FF4E00, 0 0 40px rgba(255, 78, 0, 0.6);
  border: 3px solid #fff;
  cursor: pointer;
}
#btnShareRecipie{
  background: linear-gradient(135deg, #b04701, #b746006c);
  color: rgb(54, 8, 8);
  box-shadow: 0 10px 25px rgba(0,0,0,0.4);
}
#btnShareRecipie:hover{
  box-shadow: 12px 12px 6px black;
  transition:all 0.3s ease-in;
  color: rgb(255, 228, 201);
  border-radius: 40px;
}
#IconSvg {
  height:35px;
  width:40px;
}
.navbar{
  width: 100vw;
  display: flex;
  justify-content: space-between;
  align-items: center;
  box-shadow: 1px 0 5px rgba(0, 0, 0, 0.2);
  position: fixed;
  top:0;
  z-index: 10;}
```

```

.leftBox{
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-left: 15px;
  gap: 5px;
}
.rightBox{
  width: 40%;
  height: 50px;
}
.list{
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 10px;
  color: white;
  gap: 10px;
}
header ul li{
  list-style: none;
  cursor: pointer;
}
li a{
  /* color: black; */
  transition: all 0.3s;
  padding: 2px 10px;
  border-radius: 10px;
}
.login:hover{
  color: #FF4E00;
}
li .login{
  font-size: 16px;
  font-weight: 500;
}
header h2{
  margin-left: 10px;
}
section{
  height: 100vh;
  width: 90%;
  margin: 0 auto;
}

```



```

.home{
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 4rem 6rem;
  background: linear-gradient(135deg, #e6985c, #cc6c3f);
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.1);
  gap: 2rem;
}
.home p {
  font-size: 1.2rem;
  color: #333;
  max-width: 600px;
  line-height: 1.6;
  animation: fadeInUp 1s ease 0.3s both;
}
.home .left h1{
  margin-bottom: 1rem;
}
.home button{
  /* color: white; */
  border:none;
  /* background-color: #213547; */
  width: 200px;
  padding: 0.5rem;
  /* border-radius: ; */
}
.home a:hover{
  color: white;
  background-color: #2c4860;
}
.home .left h5{
  margin:2rem 0;
}
.bg svg{
  position: absolute;
  bottom: 0;
}
.card-container{
  display: flex;
  flex-wrap: wrap;
  gap: 2rem;
  justify-content: center;
  padding: 2rem;}

```

```

.IconImage{
  border-radius: 5px;
}
.card{
  background: linear-gradient(145deg, #2a2a2a, #1f1f1f);
  border-radius: 16px;
  padding: 1.5rem;
  width: 30%; /* ~3 cards in a row */
  min-width: 280px;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.5);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
  color: white;
  cursor: pointer;
  position: relative;
}
.card:hover{
  transform: translateY(-5px) scale(1.02);
  box-shadow: 15px 15px 20px rgba(255, 78, 0, 0.8);
}
.card img.IconImage {
  width: 100%;
  height: 180px;
  object-fit: cover;
  border-radius: 12px;
  margin-bottom: 1rem;
}
.card-body{
  display: flex;
  flex-direction: column;
  gap: 0.8rem;
}
.title{
  font-size: 1.4rem;
  font-weight: 600;
  color: #FF4E00;
  text-shadow: 1px 1px 3px rgba(0,0,0,0.3);
}
.icons svg {
  cursor: pointer;
  transition: transform 0.3s ease, color 0.3s ease;
}
.icons svg:hover {
  color: #FF4E00;
  transform: scale(1.2);
}

```

```

.icons {
  display: flex;
  justify-content: space-between;
  align-items: center;
  color: #ccc;
  font-size: 1.1rem;
}
.card-body .icons{
  display: flex;
  color: #ee713b;
  justify-content: space-between;
  align-items: center;
  /* border-radius: 8px; */
  padding-right: 8px;
}
.card-body .timer{
  display: flex;
  padding: 2px;
  align-items: center;
  gap: 5px;
  font-size: 15px;
  font-weight: 500;
}
.action{
  display: flex;
  font-size: 19px;
  justify-content: space-between;
  align-items: center;
  width: 45px;
}
.editIcon{
  color: black;
  font-size: 25px;
}
.action a:hover{
  color: black;
}
.deleteIcon{
  color: rgb(239,53,53);
  font-size: 20px;
  cursor: pointer;
}
.modal{
  position: fixed;
  top: 34%;

```

```

    z-index: 10;
    padding: 2rem;
    border-radius: 6px;
    border:none;
    left:30%;
    width: 40%;
}
.backdrop{
    position: fixed;
    top:0;
    left: 0;
    width: 100%;
    height: 100vh;
    background-color: rgba(0, 0, 0, 0.75);
    z-index: 9;
}
.form{
    width: 100%;
    max-width: 400px;
    margin: 2rem auto;
    text-align: center;
    background-color:#e56b2e;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}
.form-header{
    background-color: transparent;
    text-align: center;
}
.form-control{
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    margin-top: 1rem;
}
.form-control label {
    padding-left: 5px;
    margin-bottom: 4px;
    color: #5a2c00;
    font-weight: 500;
}
.form-control .input{
    width: 100%;
    border: 1px solid #caa088;

```

```

border-radius: 6px;
height: 36px;
padding: 0 10px;
font-size: 15px;
background-color: #fff;
color: #333;
}
.form-control .input:focus {
  outline: 2px solid #ff884d;
}
.form-control .input-textarea{
  border: 1px solid rgb(156 163 175);
  border-radius: 4px;
  flex-basis: 70%;
  padding: 2px;
  width: 100%;
  background-color: white;
}
.form button {
  margin-top: 1.5rem;
  padding: 10px 16px;
  border: none;
  background-color: #ff884d;
  color: white;
  font-size: 16px;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.5s;
}
.form button:hover {
  background-color: #e56b2e;
  box-shadow: 12px 12px 25px #562106c4,0px 25px 25px #562106c4 ;
}
.form p {
  margin-top: 1rem;
  cursor: pointer;
  color: #3a1706;
  font-size: 20px;
  border-bottom: 1px solid transparent;
  display: inline-block;
  transition: border-color 0.3s;
  background-color:#e56b2e;
}
.form p:hover {
  border-color: #ff884d;}

```

```

.form .error{
margin-top: 0.5rem;
color: #d10000;
font-size: 13px;
}
.container{
display: flex;
justify-content: center;
align-items: center;
width: 100vw;
height: 100vh;
}
.add-recipe{
width: 100%;
}
.recipe{
margin-bottom: 2rem;
padding-bottom: 2rem;
}
.footer{
width:100vw;
height: 30px;
color: #386f6b9c;
background-color: #497a7750;
position: fixed;
bottom: 0;
text-align: end;
margin-top: 1rem;
padding-top: 2px;
font-size: 14px;
}
.outer-container{
width: 70%;
margin: 5rem auto;
}

.outer-container .profile{
display: flex;
gap: 1rem;
align-items: center;
}
.outer-container .title{
margin: 1rem 0;
text-transform: uppercase;
}

```

```

}
.outer-container .recipe-details{
  display: flex;
  gap: 3rem;
  align-items: justify;
  margin-top: 1rem;
}
.active{
  color:white;
  /* color:black !important; */
  /* background-color: #d4f6e8; */
}
@media screen and (max-width: 500px) {
  .form {
    padding: 1.2rem;
    border-radius: 6px;
    width: 90%;
  }

  .form-control {
    gap: 0.5rem;
  }

  .form button {
    width: 100%;
  }
}

```

Add Food Recipe code

```

import axios from 'axios'
import React, { useState } from 'react'
import { useNavigate } from 'react-router-dom'

export default function AddFoodRecipie() {
  const [recipeData, setRecipeData] = useState({ })
  const navigate = useNavigate()
  const onChange = (e) => {
    // console.log(e.target.files[0])
    let val = (e.target.name === "ingredients") ? e.target.value.split(",") : (e.target.name ===
"file") ? e.target.files[0] : e.target.value
    setRecipeData(pre => ({ ...pre, [e.target.name]: val }))
  }
  const onSubmit = async (e) => {
    e.preventDefault()
    console.log(recipeData)
    await axios.post("http://localhost:5000/recipe", recipeData, {
      headers: {
        'Content-Type': 'multipart/form-data',

```

```

        'authorization': 'bearer ' + localStorage.getItem("token")
    }
  })
  .then(() => navigate("/"))
}
return (
  <>
    <div className='container'>
      <form className='form' onSubmit={onHandleSubmit}>
        <div className='form-control'>
          <label>Title</label>
          <input type="text" className='input' name="title"
onChange={onHandleChange}></input>
        </div>
        <div className='form-control'>
          <label>Time</label>
          <input type="text" className='input' name="time"
onChange={onHandleChange}></input>
        </div>
        <div className='form-control'>
          <label>Ingredients</label>
          <textarea type="text" className='input-textarea' name="ingredients"
rows="5" onChange={onHandleChange}></textarea>
        </div>
        <div className='form-control'>
          <label>Instructions</label>
          <textarea type="text" className='input-textarea' name="instructions"
rows="5" onChange={onHandleChange}></textarea>
        </div>
        <div className='form-control'>
          <label>Recipe Image</label>
          <input type="file" className='input' name="file"
onChange={onHandleChange}></input>
        </div>
        <button type="submit">Add Recipe</button>
      </form>
    </div>
  </>
)
}

```


Connection To DataBase code

```
const mongoose=require("mongoose");

const connectdb=async()=>>
{
  await mongoose.connect(process.env.CONNECTION_STRING)
  .then(()=>console.log("Connected To DB...."))
}

module.exports=connectdb
```

Authentication Middleware

```
const jwt= require("jsonwebtoken")

const verifyToken=async(req,res,next)=>{

  let token = req.headers["authorization"]

  if (token){

    token=token.split(" ")[1]

    jwt.verify(token,process.env.SECRET_KEY,(err,decoded)=>{

      if(err){

        return res.status(400).json({ message:"Invalid TOken"})

      }

      else{

        console.log(decoded)

        req.user=decoded

      }

    })

    next()

  }

  else{
```

```

        return res.status(400).json({ message: "Invalid" })
    }
}

module.exports = verifyToken

```

Recipe Controller

```

const { response } = require("express");
const recipie = require("../models/recipe");
const Recipes = require("../models/recipe");
const multer = require("multer")

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, './public/images')
  },
  filename: function (req, file, cb) {
    const filename = Date.now() + '-' + file.filename
    cb(null, filename)
  }
})

const upload = multer({ storage: storage })

//fetch recipe all
const getRecipes = async (req, res) => {
  const recipes = await Recipes.find();
  return res.json(recipes);
};

const getRecipe = async (req, res) => {
  const recipie = await Recipes.findById(req.params.id);
  res.json(recipie);
};

const addRecipe = async (req, res) => {
  const { title, ingredients, instructions, time } = req.body;
  if (!title || !ingredients || !instructions || !time) {
    res.json({ message: "Required Fields Can't Be Empty" });
  }

  const newRecipe = await Recipes.create({
    title,

```

```

    ingredients,
    instructions,
    time,
    coverImg:req.file.filename,
    createdBy:req.user.id
  });
  return res.json(newRecipie);
};

const editRecipie = async (req, res) => {
  const { title, ingredients, instructions, time } = req.body;
  let recipie = await Recipies.findById(req.params.id);
  try {
    if (recipie) {
      let coverImg=req.file?.filename?req.file.filename:recipie.coverImg
      await Recipies.findByIdAndUpdate(req.params.id, {...req.body,coverImg}, { new: true });
      res.json({ title, ingredients, instructions, time });
    }
  } catch (error) {
    return res.status(404).json({
      message:" Id Not Found " +error,

    })

  }
};

const deleteRecipie = async(req, res) => {
  try {
    await Recipies.deleteOne({_id:req.params.id})
    res.json({ status:"Ok"})
  } catch (error) {
    return res.status(400).json({ message:"Error"})
  }
};

module.exports = {
  getRecipies,
  getRecipie,
  addRecipie,
  editRecipie,
  deleteRecipie,
  upload
};

```

CHAPTER 4

USER MANUAL

4.1 User Manual

1. Getting Started

1. System Requirements

Before installing the application, ensure that your device meets the following minimum requirements:

A modern web browser (e.g., Chrome, Firefox, Safari, Edge)

- Stable internet connection
- Recommended screen resolution: 1280x720 or higher.

2. Access Instructions

Follow these steps to access the application:

1. Open your preferred web browser.
2. In the address bar, type the website URL: [<https://culinarycanvas.com>]
3. Press **Enter** to load the application.
4. If required, **log in** using your credentials.
5. Begin exploring the features and tools provided by the application.

2.Adding a User

1. Open the web application in your browser (e.g., Google Chrome, Firefox).
2. On the homepage, click on the “**Sign Up**” or “**Register**” button.
3. Fill in the registration form with the following details:
 - Email Address
 - Password
4. Click the “**Register**” button to submit your information.
5. Upon successful registration, you will be redirected to the login page.
6. Use your registered email and password to log in.

3. Updating the Recipe

3.1 Accessing the Edit Recipe Page

1. Login to the application using your account.
2. On the home page or recipe listing, navigate to the recipe you wish to update.
3. Click on the “Edit” button or link associated with that recipe.
4. This will redirect you to the Edit Recipe form page (URL format: /edit/:id).

3.2 Editing Recipe Details

Once you're on the Edit Recipe page:

1. You will see a form with the following fields pre-filled with the recipe’s current data:
 - Title
 - Time
 - Ingredients (comma-separated)

- Instructions
 - Recipe Image (optional, you can upload a new one)
2. Update any of the fields you want to change:
 - Ingredients should be entered as a comma-separated list (e.g., salt,sugar,water)
 - You may upload a new image if you want to replace the old one

3.3 Submitting the Update

1. After making your changes, click the “Edit Recipe” button at the bottom of the form.
2. The application will send your updated data to the backend via an authenticated PUT request.

CONCLUSION

5.1 Limitations & Drawbacks

1. Offline Functionality

Since the application is web-based, it requires a stable internet connection. Users will not be able to access or modify recipes in offline environments.

2. Image Upload Restrictions

Recipe image uploads rely on the browser's file handling capabilities and server-side support. Large file sizes or unsupported formats may cause upload errors.

3. Basic Error Handling

The current application may not provide detailed error messages, which can confuse users when something goes wrong (e.g., API failure, form validation errors).

4. Limited Scalability

The system may face performance issues if the database grows significantly without optimization (e.g., no pagination or indexing in place for large recipe collections).

5. Authentication Dependency

Some features like adding, editing, or deleting recipes require user login. If token-based authentication fails or expires, users may be logged out unexpectedly.

6. No User Roles or Permissions

The application currently lacks role-based access control. Any logged-in user may be able to modify recipes, which limits the granularity of user privileges.

7. No Mobile Optimization

If the frontend is not fully responsive, users on mobile devices might face usability issues, such as form misalignment or text overflow.

5.2 Future Enhancements

1. Responsive Mobile Optimization

- Redesign the user interface with mobile-first responsiveness to improve accessibility and usability across devices of all screen sizes.
- Implement touch-friendly interactions, especially for forms and buttons, to ensure smooth mobile usage.
- Add Progressive Web App (PWA) features to allow users to "install" the site and access it like a native app.

2. Offline Functionality

- Implement service workers and local storage to allow users to create or view recipes offline.
- Enable automatic synchronization of data with the server once the internet connection is restored.

3. Search and Filter Capabilities

- Introduce search functionality by recipe name, ingredient, or cooking time.
- Add filter options such as preparation time, dietary preferences, or difficulty level to help users narrow down recipe results.

4. Commenting and Rating System

- Enable users to rate recipes and leave comments to provide feedback and suggestions.
- Display average ratings and most-liked recipes for community-driven insights.

5. Advanced Recipe Editing Tools

- Introduce rich-text formatting for instructions (bold, numbered steps, links to techniques).
- Allow users to embed images or video clips within the recipe steps for better guidance.
- Include analytics to track recipe popularity, user activity, and submission trends.

6. Social Sharing and Collaboration

- Enable one-click sharing of recipes to platforms like WhatsApp, Instagram, and Facebook.
- Let users collaborate by co-authoring or editing recipes together in shared cookbooks.

5.3 Conclusion

In conclusion, the **Recipe Management Web Application** is designed to provide a simple, intuitive, and efficient platform for users to create, manage, and explore a wide range of food recipes. By offering features such as recipe submission, editing, image upload, and categorization by ingredients and time, the system enhances the overall cooking and food-sharing experience for users.

This web-based application eliminates the need for installation and offers easy accessibility from any internet-connected device. With a clean interface and user-friendly forms, it caters to both beginners and experienced users alike. The integrated authentication system ensures data privacy and allows users to manage their own recipe collections securely.

As we continue to develop this system, our goal is to incorporate advanced functionalities such as personalized dashboards, machine learning-powered suggestions, and real-time collaboration to further enrich the user experience. Feedback from users will play a crucial role in shaping these enhancements and ensuring the platform evolves to meet their evolving culinary needs.

Whether you're a home cook sharing your favorite dish, a food blogger managing your recipe collection, or a curious user looking for inspiration, this application serves as a digital kitchen companion. We thank you for exploring and using our system, and we look forward to improving it further to support creativity, sharing, and learning in the world of cooking.

5.4 References & Bibliography:

- "React Documentation." Meta, <https://reactjs.org/docs/getting-started.html>. Accessed 1 Jan 2025.
- "Express.js Documentation." OpenJS Foundation, <https://expressjs.com/en/starter/installing.html>. Accessed 25 Jan 2025.
- "MongoDB Documentation." MongoDB, <https://www.mongodb.com/docs/>. Accessed 3 March 2025.
- "Multer Documentation – Handling File Uploads in Node.js." <https://github.com/expressjs/multer>. Accessed 4 April 2025.
- "Axios – Promise Based HTTP Client." <https://axios-http.com>. Accessed 15 April 2025.