**PROJECT REPORT**

**ON**

# RECIPE APPLICATION USING REACTJS

Submitted to

OSMANIA UNIVERSITY

*In partial fulfillment of the requirements**for the award of*

## BACHELOR OF ENGINEERING

**IN**

## COMPUTER SCIENCE AND TECHNOLOGY (AI & ML)

**Submitted By**

**AMOGH JOSHI 245321748066**

**Under the esteemed guidance of**

**Mrs. DEEPIKA**

**ASSISTANT PROFESSOR**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AIML)**

# NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

**2023-2024**

# NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

**CERTIFICATE**

*This is to certify that the Mini project work entitled* “**RECIPE APPLCATION USING REACT JS”** *is a bonafide work carried out by* **AMOGH JOSHI (245321748066)**of III-year V semester **Bachelor of Engineering** *in CSE(***AIML)** *by Osmania University, Hyderabad during the academic year* **2023-2024** *is a record of bonafide work carried out by them*. *The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

## Internal Guide Head of Department

Mrs. M. Deepika Dr. T. Prem Chander

Assistant Professor Associate Professor

**External**

# NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES) Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

# DECLARATION

# 

We hereby declare that the Mini Project Report entitled, “**RECIPE APPLICATION USING REACT JS**” submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

**Date:**

**AMOGH JOSHI 245321748066**

# ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. T. Prem Chander, Head of the Department,** CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Mrs. M. Deepika, Assistant Professor** for her technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

# ABSTRACT

React.js is a declarative, efficient, and flexible JavaScript library for building user interfaces. It allows developers to create interactive UIs by breaking them into reusable components. These components manage their state and can be composed to build complex user interfaces. There are numerous online recipe apps available, catering to various preferences, dietary needs, and culinary interests. The goals of a recipe app are often aligned with providing a seamless and enriching experience for users interested in cooking and discovering new recipes.

## TABLE OF CONTENTS

**CHAPTER TITLE PAGE NO.**

**ACKNOWLEDGEMENT I**

**ABSTRACT II**

**LIST OF FIGURES V**

**LIST OF TABLES V**

1. **INTRODUCTION**
   1. PROBLEM STATEMENT  **1**
   2. MOTIVATION **1**
   3. SCOPE  **1-2**
   4. OUTLINE  **2**
2. **LITERATURE SURVEY**
   1. EXISTING SYSTEM **3**
   2. PROPOSED SYSTEM **3**
3. **SOFTWARE REQUIREMENT SPECIFICATION**
   1. OVERALL DESCRIPTION **4**
   2. OPERATING ENVIRONMENT **4**
   3. FUNCTIONAL REQUIREMENTS **4 – 5**
   4. NON – FUNCTIONAL REQUIREMENTS **5 – 7**

1. **SYSTEM DESIGN** 
   1. USE-CASE DIAGRAM **8 – 9**
   2. CLASS DIAGRAM **10**
   3. SEQUENCE DIAGRAM **11**
   4. ACTIVITY DIAGRAM **12**
2. **IMPLEMENTATION** 
   1. SAMPLE CODE **13 –20**
3. **TESTING**
   1. TEST CASES  **21 22**
4. **SCREENSHOTS**  **23 28**
5. **CONCLUSION AND FUTURE SCOPE 29– 30**

**BIBLIOGRAPHY 31**

**APPENDIX A:** TOOLS AND TECHNOLOGY  **32**

|  |  |  |
| --- | --- | --- |
| **List of Figures** |  | |
| **Figure No.** | **Name of Figure** | **Page No.** |
| 1. | Use case Diagram | 8 – 9 |
| 2. | Class Diagram | 10 |
| 3.  4. | Sequence Diagram  Activity Diagram | 11  12 |

|  |  |  |
| --- | --- | --- |
| **List of Tables** |  | |
| **Table No.** | **Name of Table** | **Page No.** |
| 1. | Testcases | 21– 22 |

**CHAPTER – 1**

**INTRODUCTION**

## PROBLEM STATEMENT

In this project, we aim to create a recipe app that redefines the culinary experience by seamlessly blending functionality with a personalized and immersive user interface. Despite the abundance of existing recipe platforms, our goal is to engineer a solution that not only caters to a diverse range of culinary preferences, dietary restrictions, and user interests but also fosters a vibrant community of culinary enthusiasts. This entails the development of an intuitive interface, supported by robust backend systems capable of intricate recipe organization and sophisticated recommendation algorithms. Additionally, we plan to integrate interactive features such as dynamic meal planning tools and seamless social sharing functionalities, striking a delicate balance between simplicity and comprehensive functionality. Ultimately, our objective is to inspire users to embark on a journey of culinary exploration and innovation, igniting their passion for cooking and discover.

**1.2 MOTIVATION:**

Imagine a scenario where you encounter an unfamiliar ingredient dish and wish to quickly identify it with just a click. Object detection algorithms, particularly in the realm of computer vision, play a pivotal role in addressing such needs. Much like the convenience offered by Google Lens, where unknown objects can be identified swiftly, integrating similar functionality into a recipe app can greatly enhance user experience. Object detection technology not only aids in culinary exploration but also finds applications across various domains, making it a valuable asset for users seeking efficient identification and categorization of objects.

# 1.3 SCOPE:

The scope of the recipe app encompasses various facets of the culinary journey, ranging from recipe discovery and meal planning to ingredient management and social interaction. Key features include a vast database of recipes spanning different cuisines and dietary preferences, intelligent search functionality for quick and precise recipe retrieval, personalized recommendations based on user preferences and browsing history, meal planning tools to facilitate efficient meal preparation and grocery shopping, interactive cooking guides and tutorials to assist users at every step of the cooking process, and social sharing capabilities for users to connect, collaborate, and exchange culinary experiences with others. Additionally, the app may incorporate features for ingredient substitution, nutritional analysis, and user-generated content to enhance the overall user experience and foster a sense of community among users.

# 1.4 OUTLINE:

The recipe app's outline entails a thorough journey from conception to deployment, beginning with research and planning to identify user needs and competitors, followed by designing an intuitive user interface and incorporating interactive features like personalized recommendations and social sharing. Development involves building robust backend infrastructure and implementing frontend components using technologies like React.js, with integration of object detection algorithms for advanced features like ingredient recognition. Comprehensive testing ensures reliability and usability, leading to deployment across app stores and promotion through marketing strategies, with ongoing monitoring and iteration post-launch to enhance user experience and engagement.

# CHAPTER – 2

# LITERATURE SURVEY

## EXISTING SYSTEM:

* There are several apps and numerous real-time websites specialize in creating recipes and finding ingredients by searching for specific items.
* Users can typically filter recipes based on dietary preferences, cuisine types, and other criteria.

Here are some of the well-known examples,  
->All Recipes  
->Yummly  
->Super Cook etc...

## PROPOSED SYSTEM:

## Our recipe app is a revolutionary step-by-step cooking guide designed for both beginners and seasoned cooks.

## Additionally, it boasts a searchable database of delicious dishes, a scaling tool to adjust servings, and a handy timer.

## Whether you're a novice chef or an experienced foodie, our app provides everything you need to cook with confidence and create culinary masterpieces in your own kitchen.

# CHAPTER - 3

**SOFTWARE REQUIREMENTS SPECIFICATION**

## 3.1 Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

## 3.2. Operating Environment:

Software Requirements:

Operating System : Windows 7 (Min)

Front End : React Js , Express

Back End : Node Js , MongoDB

Hardware Requirements:

Processor : Intel CORE i5 10th gen

Speed : 2.9 GHz (Min)

RAM : 4 GB (Min)

Hard Disk : 128 GB (Min)

## Functional Requirements:

**User Functionality:**

* The user will be able to upload images for their own recipes .
* The user will be able to insert images up to 200 mb in size.

## 

## Admin Functionality:

* The admin manages the website.
* The admin can make changes to the website such as modifying the UI and making it more interactive than earlier.
* The admin can implement a better algorithm if at all a better algorithm is created in future.

## Non-Functional Requirements:

* + 1. **Performance Requirements:**

Performance requirements refer to static numerical requirements placed on the interaction between the users and the software.

### Response Time:

Average response time shall be less than 5 sec.

### Recovery Time:

In case of system failure, the redundant system shall resume operations within 30 secs. Average repair time shall be less than 45 minutes.

### Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

### Capacity:

The system accommodates many users at a time.

### Utilization of Resources:

The system returns a result after a search is provided by the user.

**3.4.2 Safety Requirements:**

-NA-

* + 1. **Security Requirements:**

The model will be running on a secure website i.e., an HTTPS website and also on a secure browser such as Google Chrome, Brave, etc.

* + 1. **Software Quality Attributes:**

### Reliability:

The system shall be reliable i.e., in case the webpage crashes, progress will be saved.

### Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

### Security:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

### Maintainability:

The model shall be designed in such a way that it will be very easy to maintain it in future. Our model is a web-based system and will depend much on the web server. However, the web application will be designed using React Js and proper database modeling along with extensive documentation which will make it easy to develop, troubleshoot and maintain in future.

### Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily.

### Scalability:

The system will be designed in such a way that it will be extendable. If more species or algorithms are going to be added in the system, then it would easily be done.

The same system can also be developed to become a mobile application rather than just a website.

# CHAPTER-4

# SYSTEM DESIGN

## Use Case Diagram:

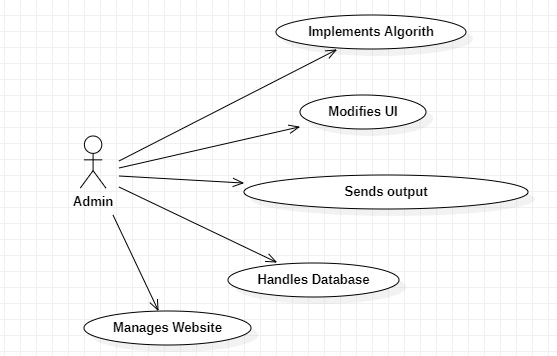
# 

Input Recipes

Input images

Input recipes

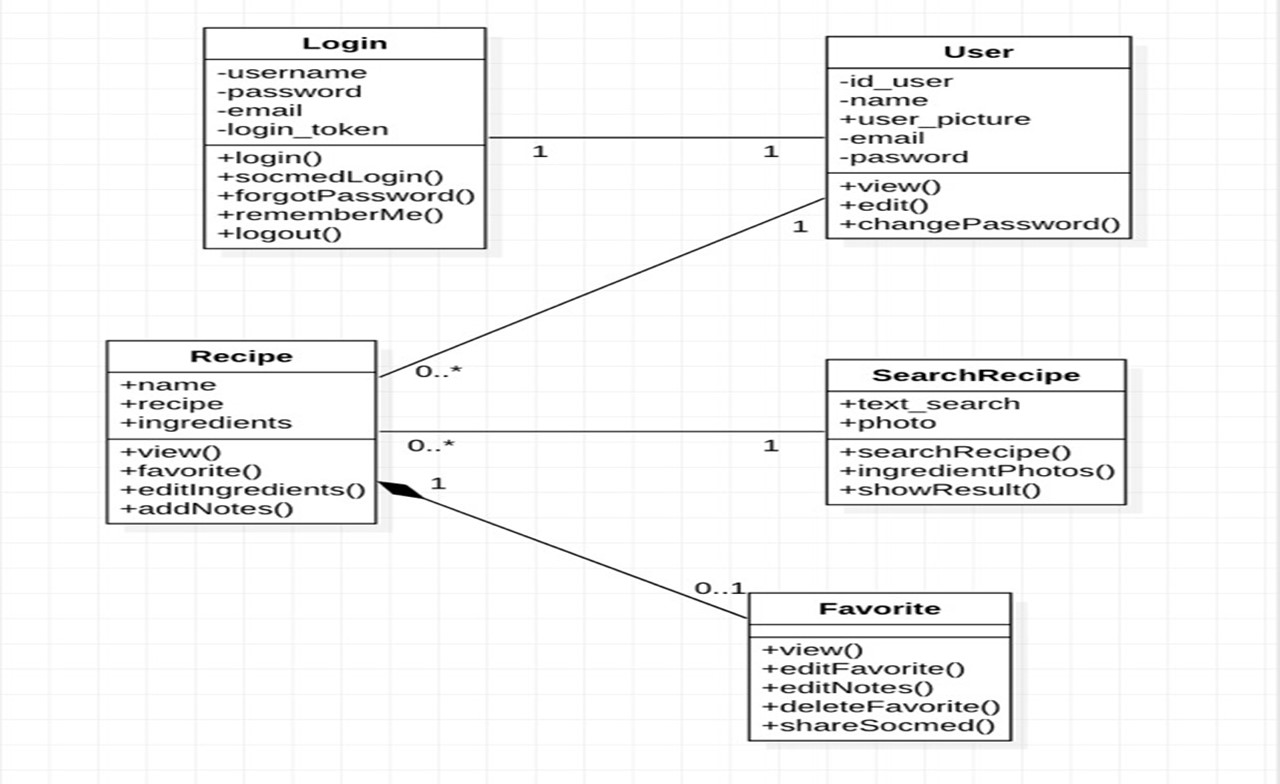
**Fig.4.1 : Use Case diagram for User**



**Fig.4.1.2 : Use Case diagram for User**

## 

## Class Diagram:



**Fig.4.2 : Class diagram for Application**

## Sequence Diagram:

## 

**Fig4.3 : Sequence Diagram for Application.**

## Activity Diagram:

**Fig4.4 : Activity Diagram for Application.**

**CHAPTER-5**

**IMPLEMENTATION**

**5.1 SAMPLE CODE**

**APP.JS**

const express = require('express');

const expressLayouts = require('express-ejs-layouts');

const fileUpload = require('express-fileupload');

const session = require('express-session');

const cookieParser = require('cookie-parser');

const flash = require('connect-flash');

const app = express();

const port = process.env.PORT || 3000;

require('dotenv').config();

app.use(express.urlencoded( { extended: true } ));

app.use(express.static('public'));

app.use(expressLayouts);

app.use(cookieParser('CookingBlogSecure'));

app.use(session({

secret: 'CookingBlogSecretSession',

saveUninitialized: true,

resave: true

}));

app.use(flash());

app.use(fileUpload());

app.set('layout', './layouts/main');

app.set('view engine', 'ejs');

const routes = require('./server/routes/recipeRoutes.js')

app.use('/', routes);

app.listen(port, ()=> console.log(`Listening to port ${port}`));

**Index.ejs**

<div class="row flex-lg-row-reverse align-items-center g-5 py-4 mb-4">

<div class="col-12 col-lg-6">

<img src="/img/hero-image.png" width="607" height="510" class="d-block mx-lg-auto img-fluid" loading="lazy" alt="Cooking With Node.js">

</div>

<div class="col-12 col-lg-6">

<h1 class="display-5 fw-bold mb-3">Huge selection of delicious recipe ideas</h1>

<p class="lead">

Explore our huge selection of delicious recipe ideas including; easy desserts, delicious vegan and vegetarian dinner ideas, gorgeous pasta recipes, quick bakes, family-friendly meals and gluten-free recipes.

</p>

<div class="d-grid gap-2 d-md-flex justify-content-md-start">

<a href="/explore-latest" class="btn btn-primary btn-dark btn-lg px-4 me-md-2">Explore Latest</a>

<a href="/explore-random" class="btn btn-outline-secondary btn-lg px-4 me-md-2">Show Random</a>

</div>

</div>

</div>

<!-- Categories Start -->

<div class="row row-cols-2 row-cols-lg-6 g-2 g-lg-3 py-4">

<% if(categories != '') { %>

<% categories.forEach(function(category, index) { %>

<a href="/categories/<%- category.name %>" class="col text-center category\_\_link">

<div class="category\_\_img shadow">

<img src="img/<%- category.image %>" alt="<%- category.name %>" loading="lazy">

</div>

<div class="pt-1"><%- category.name %></div>

</a>

<% }) %>

<% } %>

<a href="/categories" class="col text-center category\_\_link">

<div class="category\_\_img shadow">

<img src="img/view-all.jpg" alt="View All Categories" loading="lazy">

</div>

<div class="pt-1">View All</div>

</a>

</div>

<!-- Categories End -->

<!-- Latest Start -->

<section class="pb-4 pt-4">

<div class="d-flex mb-2 align-items-center">

<h2>Latest Recipes</h2>

<a href="/explore-latest" class="ms-auto">View More</a>

</div>

<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">

<% if(food != '') { %>

<% food.latest.forEach(function(recipe, index){ %>

<a href="/recipe/<%= recipe.\_id %>" class="col text-center category\_\_link">

<div class="category\_\_img category\_\_img--large shadow">

<img src="/uploads/<%= recipe.image %>" alt="<%= recipe.name %>" loading="lazy">

</div>

<div class="pt-1"><%= recipe.name %></div>

</a>

<% }) %>

<% } else { %>

<p>No items found.</p>

<% } %>

</div>

</section>

<!-- Latest End -->

<!-- Thai Start -->

<section class="pb-4 pt-4">

<div class="d-flex mb-2 align-items-center">

<h2>Thai Recipes</h2>

<a href="/categories/Thai" class="ms-auto">View More</a>

</div>

<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">

<% if(typeof food.thai !== 'undefined' && food.thai.length > 0) { %>

<% food.thai.forEach(function(recipe, index){ %>

<a href="/recipe/<%= recipe.\_id %>" class="col text-center category\_\_link">

<div class="category\_\_img category\_\_img--large shadow">

<img src="/uploads/<%= recipe.image %>" alt="<%= recipe.name %>" loading="lazy">

</div>

<div class="pt-1"><%= recipe.name %></div>

</a>

<% }) %>

<% } else { %>

<p>No items found.</p>

<% } %>

</div>

</section>

<!-- Thai End -->

<!-- American Start -->

<section class="pb-4 pt-4">

<div class="d-flex mb-2 align-items-center">

<h2>American Recipes</h2>

<a href="/categories/American" class="ms-auto">View More</a>

</div>

<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">

<% if(food.american != '') { %>

<% food.american.forEach(function(recipe, index){ %>

<a href="/recipe/<%= recipe.\_id %>" class="col text-center category\_\_link">

<div class="category\_\_img category\_\_img--large shadow">

<img src="/uploads/<%= recipe.image %>" alt="<%= recipe.name %>" loading="lazy">

</div>

<div class="pt-1"><%= recipe.name %></div>

</a>

<% }) %>

<% } else { %>

<p>No items found.</p>

<% } %>

</div>

</section>

<!-- American End -->

<!-- Chinese Recipes -->

<section class="pb-4 pt-4">

<div class="d-flex mb-2 align-items-center">

<h2>Chinese Recipes</h2>

<a href="/categories/American" class="ms-auto">View More</a>

</div>

<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">

<% if(food.chinese != '') { %>

<% food.chinese.forEach(function(recipe, index){ %>

<a href="/recipe/<%= recipe.\_id %>" class="col text-center category\_\_link">

<div class="category\_\_img category\_\_img--large shadow">

<img src="/uploads/<%= recipe.image %>" alt="<%= recipe.name %>" loading="lazy">

</div>

<div class="pt-1"><%= recipe.name %></div>

</a>

<% }) %>

<% } else { %>

<p>No items found.</p>

<% } %>

</div>

</section>

<!-- American End -->

<!-- Submit - Start -->

<section class="px-4 py-5 my-5 text-center">

<img src="/img/publish-recipe.png" class="d-block mx-auto mb-4 img-fluid" alt="Publish your recipe for FREE today" width="566" height="208" loading="lazy">

<h1 class="display-5 fw-bold">Publish your recipe for FREE today</h1>

<div class="col-lg-6 mx-auto">

<p class="lead mb-4">Publish your Recipe in front of thousands of people for free.</p>

<div class="d-grid gap-2 d-sm-flex justify-content-sm-center">

<a href="/submit-recipe" class="btn btn-primary btn-dark btn-lg">Submit Recipe</a>

</div>

</div>

</section>

<!-- Submit - End -->

**Database.js**

const mongoose = require('mongoose');

mongoose.connect("mongodb+srv://Name:password@cluster0.co9jvfi.mongodb.net/?retryWrites=true&w=majority");

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:'));

db.once('open', function(){

console.log('Connected')

});

// Models

require('./Category');

require('./Recipe');

# CHAPTER – 6

# TESTING

## 6.1 TEST CASES

**Test Case to check whether the required Software is installed on the systems**

|  |  |
| --- | --- |
| **Test Case ID:** | 1 |
| Test Case Name: | Required Software Testing |
| Purpose: | To check whether the required Software is installed on the systems |
| Input: | Enter Node command |
| Expected Result: | Should Display the version number |
| Actual Result: | Displays Node version |
| Failure | If the Node environment is not installed, then the Deployment fails |

**Table 6.1.1 Node Installation verification**

**Test Case to check node Integration Testing**

|  |  |
| --- | --- |
| **Test Case ID:** | 2 |
| Test Case Name: | Programs Integration Testing |
| Purpose: | To ensure that all the modules work together |
| Input: | All the modules should be accessed. |
| Expected Result: | All the modules should be functioning properly. |
| Actual Result: | All the modules should be functioning properly. |
| Failure | If any module fails to function properly, the implementation fails. |

|  |  |
| --- | --- |
| **Test Case ID:** | 3 |
| Test Case Name: | Collect Dataset and Load the Dataset |
| Purpose: | Check Dataset is collected, and the data is stored |
| Input: | Provide Dataset as input |
| Expected Result: | Dataset is collected and view the Dataset and store the Dataset |
| Actual Result: | Load the Dataset and view the Dataset and store |
| Failure | If the dataset is not loaded, it will throw an error. |

**Table 6.1.2 Node Programs Integration**

**Table 6.1.3 Collect Dataset and Load the Dataset**

**Test Case to check working of the commands provided by the user**

|  |  |
| --- | --- |
| **Test Case ID:** | 4 |
| Test Case Name: | Proper execution of the commands. |
| Purpose: | To check whether the system performs desired action for the commands provided. |
| Input: | Provide sonar frequencies. |
| Expected Result: | The system performs the action specified in the command. |
| Actual Result: | The system performs the action specified in the command. |
| Failure | If the system doesn’t understand the command it will not perform the action. |

**Table 6.1.4 Execution of commands**

# CHAPTER – 7

# SCREENSHOTS

# 

# Fig 7.1 : Home Page for Application.

# 

**Fig 7.2 : Search Page for Application.**

# 

**Fig 7.3 : Recipe Page for Application.**

# 

**Fig 7.4 : Explore Page for Application.**

# 

**Fig 7.3 : Submit Page for Application.**

# CHAPTER - 8

# CONCLUSION AND FUTURE SCOPE

**Conclusion:**

In conclusion, the recipe application developed using ReactJS offers a user-friendly platform for users to discover, save, and share recipes. Throughout the project, we successfully implemented key features such as a searchable recipe database, and a responsive user interface. The application provides an engaging experience for both novice and experienced cooks, catering to a wide range of culinary preferences.

**Future Scope**:

While the current version of the recipe application fulfills its primary objectives, there are several avenues for future enhancement and expansion: -

* Enhanced Recommendation System: Implement machine learning algorithms to provide personalized recipe recommendations based on user preferences, dietary restrictions, and past interactions with the application.
* Social Integration: Integrate social features such as user profiles, follower/following mechanisms, and the ability to share recipes with friends and family via social media platforms.
* Advanced Search Filters: Incorporate advanced search filters, allowing users to narrow down recipe results based on criteria such as cooking time, difficulty level, ingredients on hand, and nutritional information.
* Multimedia Content: Enhance the user experience by including multimedia content such as step-by-step video tutorials, high-resolution images, and user-generated content like cooking tips and variations.
* Localization and Internationalization: Extend support for multiple languages and regional cuisines to make the application accessible to a global audience.
* Integration with Smart Devices: Explore integration with smart kitchen appliances and devices, enabling users to seamlessly follow recipes and control cooking processes directly from the application.

**BIBLIOGRAPHY**

1. Smith, John. "The Art of Cooking: A Comprehensive Guide." Culinary Publishing, 2020.
2. Brown, Sarah. "The Joy of Baking: Delicious Recipes for Every Occasion." Baking House Press, 2019.
3. Patel, Rajesh. "Nutrition and You: Understanding the Role of Ingredients in Healthy Cooking." Health Foods Publishing, 2018.
4. Williams, Emily. "The Ultimate Cookbook: 1000 Recipes to Elevate Your Culinary Skills." Epicurean Publications, 2021.
5. Johnson, Michael. "Food Photography: Capturing the Beauty of Culinary Creations." Visual Arts Press, 2020.
6. Miller, Jessica. "The Science of Flavor: Exploring the Chemistry Behind Delicious Dishes." Gastronomy Press, 2017.
7. Garcia, Maria. "Cultural Cuisine: Exploring the Diversity of Global Flavors." World Foods Publishing, 2019

## 

**APPENDIX A:** TOOLS AND TECHNOLOGIES

* REACT JS: It is a JavaScript library for building user interfaces, known for its component-based architecture and efficient rendering.
* EXPRESS: It is a minimalist web application framework for Node.js, designed for building robust and efficient web applications and APIs with simplicity and flexibility in mind.
* NODE JS: Node.js is a powerful, event-driven JavaScript runtime built on Chrome's V8 engine, ideal for scalable and high-performance network applications. Its non-blocking I/O model makes it efficient for handling concurrent operations, making it a popular choice for building server-side applications.
* MONGO DB: It is a popular NoSQL database known for its flexibility and scalability, perfect for handling unstructured data efficiently. Its document-oriented model and dynamic schema make it ideal for modern applications requiring agile data management
* WINDOWS 11: Windows 11 was used as the operating system.