

PROFESSIONAL TRAINING REPORT- II

Entitled

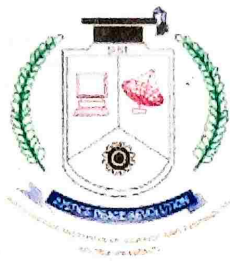
FOOD RECIPE GENERATOR

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and
Engineering with specialization in Artificial Intelligence and Robotics

by

MARUPAKA DHANUSH

41612012



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI – 600119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training is the bonafide work of **Mr. MARUPAKA DHANUSH(41612012)** who carried out the project entitled **FOOD RECIPE GENERATOR** under my supervision from January 2024 to May 2024.

Internal Guide

Dr. SONIA JENIFER RAYEN, M.Tech., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 07-05-2024

Internal Examiner

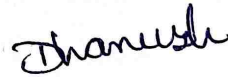
External Examiner

DECLARATION

I, **Mr. MARUPAKA DHANUSH(41612012)**, hereby declare that the Professional Training Report-II entitled **FOOD RECIPE GENERATOR** done by me under the guidance of **Dr. SONIA JENIFER RAYEN, M.Tech., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Artificial Intelligence and Robotics.

DATE: 07/05/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

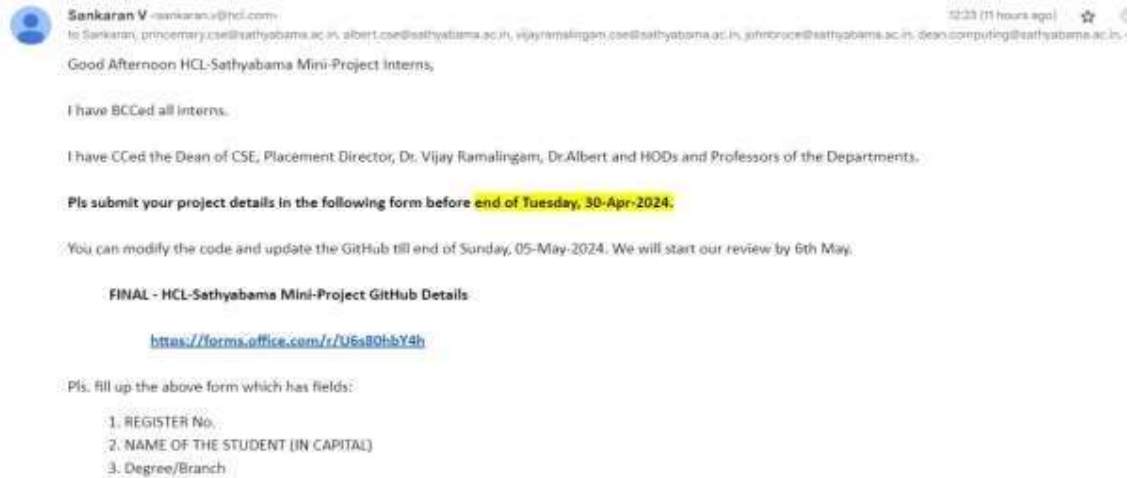
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. Sonia Jenifer Rayen, M.Tech., Ph.D.**, for his/her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-1 professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

Internship Proof



ABSTRACT

The Food Recipe Generator Application represents a pioneering culinary platform that seamlessly integrates API technologies to provide users with an unparalleled cooking experience. Evolving beyond conventional recipe searches, this innovative application now offers a multitude of advanced features to cater to diverse tastes and preferences. Users can effortlessly navigate through an extensive database of recipes, filtering options by meal types, dietary considerations, dish categories, and specific cuisines, thereby ensuring a tailored culinary journey. Whether seeking inspiration for a quick breakfast, a nutritious lunch, or an elaborate dinner, the app's intuitive interface facilitates effortless exploration. Moreover, with the ability to refine searches based on dietary requirements such as highprotein, low-carb, or other specialized diets, users can align their meals with their health and wellness objectives using Gemini Api. Through its usercentric design and comprehensive functionality, the Food Recipe Generator empowers individuals of all skill levels to unleash their culinary creativity, discover new flavors, and embark on a delightful gastronomic adventure

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	vii
	INTRODUCTION	
1	1.1 Overview	1
	LITERATURE SURVEY	
2	2.1 survey	2
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	6
	3.2	6
	3.2.1 Hardware Requirements	
	3.2.2 Software Requirements	
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	8
	Proposed Product	8
4.1	4.1.1 Ideation Map/Architecture Diagram	10
	4.1.2 Internal or Component design structure	11
	Features	12
4.2	4.2.1 Novelty of the Project	13
	4.2.2 Program code	14
5	CONCLUSION	26
	REFERENCES	27
	APPENDIX	28
	A. Research Paper	30
	B. Source Code	35

LIST OF FIGURES

Figure No.	Figure Name	Page No
4.1	Ideation Map	09
4.2	Landing Page	10
4.3	SEARCH OPTION	10
4.4	Customization Option	11
4.5	INGREDIENTS AND DETAILS	11
4.6	Gemini bot response	12
4.7	Recipes Generated	12

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The "Food Recipe Generator" project stands at the forefront of culinary innovation, offering a revolutionary approach to meal planning and recipe discovery. With the exponential growth of digital technology, traditional recipe books and cooking shows have evolved into dynamic digital platforms that cater to the modern-day food enthusiast. The Food Recipe Generator represents a paradigm shift in the way we approach cooking, leveraging cutting-edge technologies and sophisticated algorithms to revolutionize the culinary experience. At its core, the Food Recipe Generator is a user-centric platform designed to empower individuals with the tools and resources they need to explore, experiment, and indulge in the art of cooking. Whether you're a seasoned chef looking to expand your culinary repertoire or a novice cook eager to embark on a culinary journey, this platform offers something for everyone. By harnessing the power of data-driven insights and personalized recommendations, the Food Recipe Generator transcends traditional recipe platforms, offering users a truly immersive and engaging cooking experience. One of the key features that sets the Food Recipe Generator apart is its intuitive interface, which seamlessly guides users through the recipe discovery process. From browsing through a vast collection of recipes to customizing search parameters based on dietary preferences and nutritional requirements, users have complete control over their culinary journey. Additionally, the platform's intelligent recommendation engine analyzes user behavior and preferences to deliver highly relevant and personalized recipe suggestions, ensuring that every cooking endeavor is met with success and satisfaction. Moreover, the Food Recipe Generator is more than just a repository of recipes; it's a dynamic ecosystem that fosters creativity, exploration, and community engagement. Through interactive features such as user-generated content, social sharing capabilities, and collaborative recipe creation tools, the platform encourages users to connect, collaborate, and inspire one another on their culinary adventures. As we continue to innovate and evolve, the Food Recipe Generator remains committed to its mission of redefining the way we cook and eat. By harnessing the power of technology, data, and community-driven insights, we aim to empower individuals to unleash their inner chef, explore new flavors, and create memorable culinary experiences that nourish the body, mind, and soul.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

1.SMART CUISINE RECIPE FINDER:

Authors: Abhijith V , Athira Sunil , Sanjay Sajeevan, Suhana T R, Vidya Hari

Details: Department of Computer Science ,A P J Abdul Kalam Technological University, KMEA Engineering College , Edathala , Aluva .

People get easily tired of eating the same things again and again. They then resort to restaurant/ junk food which affects their health as the vegetables and oil used by them are of suboptimal quality. The proposed system suggests recipes according to the fruits and vegetables available to the user. User scans the available fruits and vegetables and android application displays a list of possible recipes. The application is personalized, which means the recipes displayed are filtered according to the user's preferences and health conditions. For example, the proposed application does not display a spicy recipe to a user for elderly people. This makes our application more health friendly. It quickly displays a list of possible recipes. Another interesting aspect of the proposed project is using Machine Learning approach. Detection is one of the challenging and interesting problems in machine learning. This is done using Tensorflow ,which is free open source library.

2.FOODIE: An Android Application

Authors: Rahul Dandekar , Sanket Gadkari , Mayuri Sadudia , Sayali Kamble

Details: Department of Information Technology Padmabhushan Vasantdada Patil Pratishthan's College of Engineering, Mumbai.

In today's hectic world of work loaded life, there is no time for an individual to give a dedicated schedule for making his daily meals. You just want to relax after returning from work, either by ordering in a takeaway or heading out to a delicious establishment near you. At times it becomes a very tedious work for humans to think on, for what to make at dinner in such limited materials available. The food industry is an underserved market. The possibilities are wide-open, and the competition is scarce.

Entrepreneurs can easily make a quick turn around by providing a much needed service. We are hence, developing an Android Application which would make the task of preparing meals a less tensed for an individual. The system is an innovative new recipe search engine which finds recipes you can make with the ingredients you currently have at hand. The system is an Android application - Foodie, which will display the recipes according to the inputs given by text, voice and image and filter them accordingly to their variety and list as per the most appropriate recipes first.

3.RECIPE WORLD FOOD RECIPE APPLICATION

Author: Siavash Motalebi

This paper presents a food recipe Android application that helps users find and view different food recipes based on different categories, as well as allowing them to add their own recipes to the database. The users can filter the list of recipes based on the ingredients used in the recipe, the prep time, cook time, ready time, and the number of servings. The app aspires to run efficiently, while having an intuitive and simple design that provides the user all the necessary functionalities. This project uses both databases and web-services to design and implement an online food recipe Android app. The app is intended to have an intuitive and simple user interface that allows the user to find and view different food recipes based on different inputs such as, the type of food, the cuisine, and the food course.

4.Recipe Recommendation System

Authors: Alkatai Gholve, Nidhi Mishra, Revati Gaikwad, Shruti Patil

Details: Student, Computer Department, Vishwakarma Institute of Information Technology, Pune, India

The project proposed a recipe recommendation system that makes use of images of ingredients to recommend a recipe to a person who doesn't know anything about the contents and its proportion in a particular recipe. This system is also beneficial for health-conscious people. With the changing living manners, diet habits are changed and work load has increased which resulted in various diseases, such as diabetes, BP, problems related to heart and so on. All these diseases can be controlled by avoiding uneven and unhealthy food. So it's important to understand that what is the proper diet

and in how much quantity it should be taken. Our main aim is to recommend recipes to maintain their health for people with disease and without disease which will satisfy the needs of user. To give the recommendations, recommended system uses the user's profile, their favourite food and details of that food. Clients from various countries, belonging to various cultures are contributing in terms of a large number of new recipes on the web all over the world. Every recipe contains of so many distinct elements. Thus, user might not be able to identify all the ingredients or contents. In this paper we propose a recommendation system for recipe using Convolutional Neural Network (CNN) which is used for supervised learning in order to analyse the data. Recommended Recipe contains diverse ingredients, cooking procedure, categories and so on.

5. Food Recipe Recommendation Based on Ingredients Detection Using Deep Learning

Authors: Md. Kishor Morol, Md. Shafaat Jamil Rokon, Ishra Binte Hasan, A. M. Saif, Rafid Hussain Khan, Shuvra Smaran Das

Details: American International University-Bangladesh, Bangladesh

Food is essential for human survival, and people always try to taste different types of delicious recipes. Frequently, people choose food ingredients without even knowing their names or pick up some food ingredients that aren't obvious to them from a grocery store. Knowing which ingredients can be mixed to make a delicious food recipe is essential. Selecting the right recipe by choosing from a list of ingredients is very difficult for a beginner cook. However, it can be a problem even for experts. There is the constant use of machine learning in our everyday lives. One such example is recognizing objects through image processing. In this paper, we implemented a model for food ingredient recognition and designed an algorithm for recommending recipes based on recognized ingredients. We made a custom dataset consisting of 9856 images belonging to 32 different food ingredient classes. A Convolution Neural Network (CNN) model was used to identify food ingredients, and for recipe recommendations, we used machine learning. We achieved an accuracy of 94%, which is quite impressive.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

Our proposed project is a web-based Food Recipe Generator application that runs directly on a local server. The software prerequisites for this project include

- Node Package Manager (npm)
- React JS
- JavaScript (JS)
- HTML5
- CSS3.

The Food Recipe Generator is a web-based application designed to cater to users' culinary interests by providing a diverse range of recipes directly sourced from external APIs. Unlike traditional recipe platforms, this application doesn't rely on a backend database to store recipe data. Instead, it leverages API fetch functionality to dynamically retrieve recipe information based on user queries. The core objective of the Food Recipe Generator is to empower users with a seamless and efficient means of discovering new recipes aligned with their preferences. By utilizing the search functionality, users can input keywords or specific ingredients they wish to explore, prompting the application to communicate with external APIs and fetch relevant recipe data. This data is then seamlessly integrated into the user interface, allowing users to browse through a curated selection of recipes tailored to their search criteria. Upon accessing the application, users are greeted with an intuitive interface featuring a prominent search bar prompting them to input their desired recipe queries. Upon submitting a search request, the application initiates the API fetch process, retrieving pertinent recipe details such as ingredients, cooking instructions, and nutritional information. These details are presented to the user in an organized and visually appealing manner, facilitating easy browsing and selection. To enhance user engagement and satisfaction, Food Recipe Generator may incorporate additional features such as recipe filtering options to refine results based on dietary preferences, or cooking difficulty levels. Furthermore, interactive functionalities such as user ratings,

reviews, and favorite saving capabilities may be integrated to foster community interaction and provide valuable feedback on recipe experiences.

3.2 REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

- **Processor:** The Intel Core i3 processor is a dual-core CPU known for its efficient performance and reliability. It is capable of handling everyday computing tasks with ease, making it suitable for general use. With its balanced combination of processing power and energy efficiency, the Core i3 processor provides a smooth user experience for tasks such as web browsing, office productivity, and multimedia consumption.
- **Memory (RAM):** With 8GB of RAM (Random Access Memory), this system offers ample memory for multitasking and running multiple applications simultaneously. The generous amount of RAM ensures smooth performance, allowing users to switch between tasks seamlessly without experiencing slowdowns or lag. Whether it's browsing the web, editing documents, or streaming media, the 8GB of RAM provides sufficient memory to handle everyday computing needs efficiently.
- **Hard Disk:** The 1TB (Terabyte) hard disk provides ample storage space for storing a large collection of files, including documents, photos, videos, and software applications. With its vast storage capacity, users can comfortably store their multimedia libraries, personal files, and software installations without worrying about running out of space. Additionally, the 1TB hard disk offers fast data access speeds, enabling quick retrieval of files and efficient data transfer operations. Overall, the spacious hard disk provides ample room for storing and organizing digital content, making it suitable for both personal and professional use.

3.2.2 SOFTWARE REQUIREMENTS

Operating System:

- **Linux or Windows**

Linux and Windows are popular operating systems used for hosting servers and running applications. Linux is open-source and known for stability, security, and flexibility, while Windows offers a user-friendly interface and compatibility with various software.

Coding Languages:

- **Sass (Syntactically Awesome Style Sheets):**

Sass is a preprocessor scripting language that compiles into CSS. It extends CSS by adding features like variables, mixins, and nesting, making it easier to write and manage stylesheets.

- **Gemini API:**

The Gemini API is provided by the Gemini cryptocurrency exchange platform. It allows developers to integrate trading functionalities, market data, and account management into their applications using RESTful and WebSocket endpoints.

- **React JS:**

React JS is a JavaScript library for building user interfaces. It's efficient, scalable, and widely used for creating interactive web applications.

- **JavaScript (JS):**

JavaScript is a programming language used for adding interactivity to web pages. It's supported by all modern web browsers and enables dynamic content and user interactions.

- **HTML5 (Hypertext Markup Language):**

HTML5 is the latest version of the markup language used to structure web content. It introduces new features for multimedia, forms, and responsive design.

- **CSS3 (Cascading Style Sheets):**

CSS3 is the latest version of the styling language used to customize the appearance of HTML elements. It offers advanced styling options like animations, transitions, and flexible layouts.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

The Food Recipe Generator Project provides the forward-looking initiative poised to redefine the culinary landscape. Through technology-driven innovation and a commitment to user satisfaction, this project empowers individuals to embark on a culinary journey that is personalized, efficient, and filled with delightful gastronomic experiences.

The Food Recipe Generator project is to create a software web based application or platform that assists users in discovering, planning, and preparing meals efficiently and effectively. The project aims to address various needs and goals related to cooking and meal planning. Here are the primary objectives of the project:

- **Personalization:** The recipe suggestions to individual user preferences, including cuisine type, dietary restrictions, and ingredient availability.
- **Convenience:** Provide users with quick and easy access to a wide variety of recipes, reducing the time and effort required for meal planning and preparation.
- **Minimize Food Waste:** Help users make the most of their available ingredients by suggesting recipes that utilize what they have on hand, thus reducing food waste.
- **Cooking Guidance:** Provide step-by-step cooking instructions and tips to ensure that users can successfully prepare the selected dishes, regardless of their culinary skill level.
- **Nutritional Awareness:** Offer detailed nutritional information for each recipe to promote healthy eating and enable users to make informed dietary choices.
- **Global Cuisines:** Include a diverse range of global cuisines and recipes, allowing users to explore and experience different culinary traditions and flavors.
- **Saving and Deletion:** Offer the user to save the recipes interested for future use and references and get all the saved recipes by the particular user in the saved section.

- **Filter:** Get the filter based search option to maximize the user experience while searching the recipies.
- **User Engagement:** Keep users engaged and satisfied with the application by continuously improving the user interface, and features based on feedback and evolving culinary trends.
- **Scalability and Accessibility:** Ensure that the application can accommodate a growing user base and is accessible across various devices and platforms.

4.1.1 Ideation Map/System Architecture

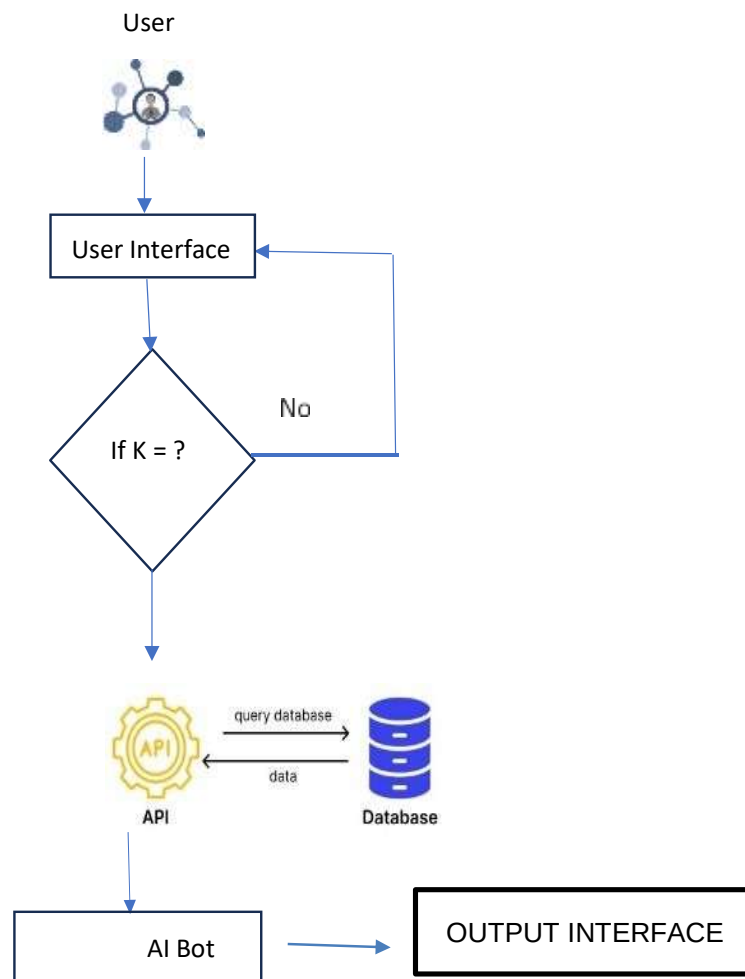


Fig.4.1 IDEATION MAP

4.1.3 Internal or Component design structure :

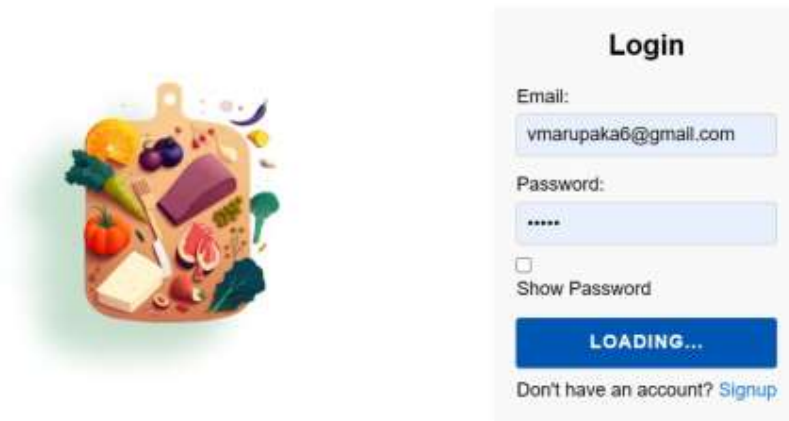


Fig.4.2 LANDING PAGE

4.2 FEATURES

1.Search Option:

User have a chance to search the favourite recipie in the search menu using relevant keyword and get the instances for the sea .



Fig.4.3 SEARCH OPTION

2. Customization Theme Option:

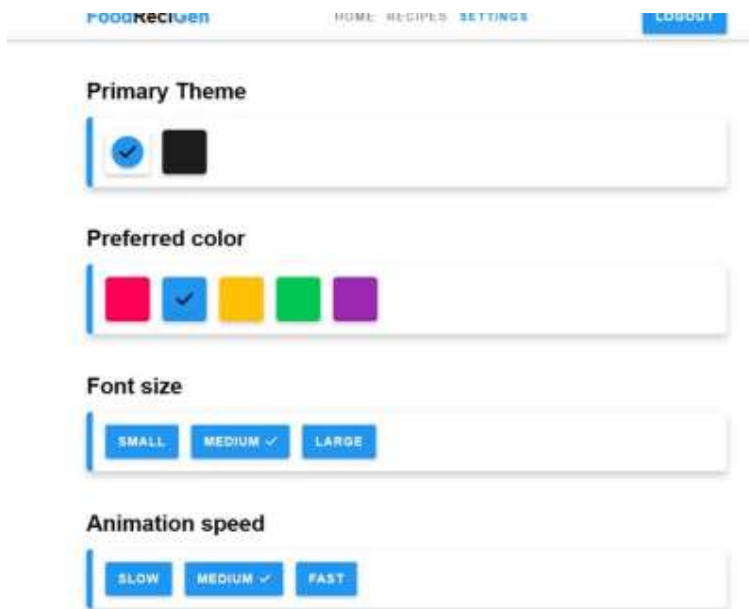


Fig.4.4 CUSTOMIZATION OPTIONS

3. Get Instructions/Details

The user can able to see all the Instructions and the Ingredients about the recipe in the recipe cards as shown.



Fig.4.5 INGREDIENTS AND DETAILS

4. Gemini Generated recipe:

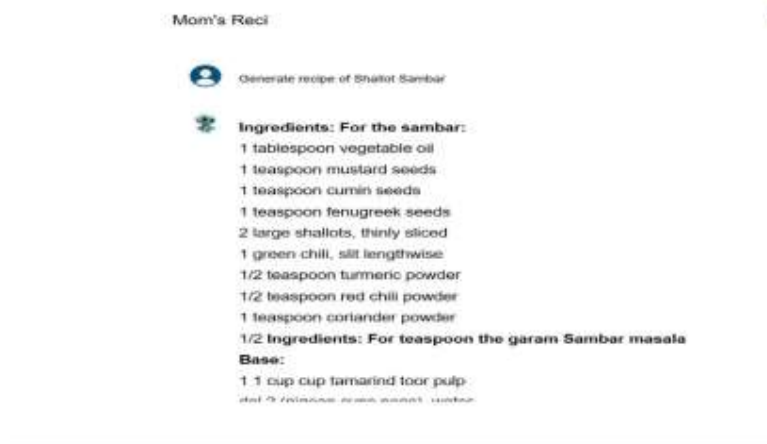


Fig.4.6 gemini generated recipe

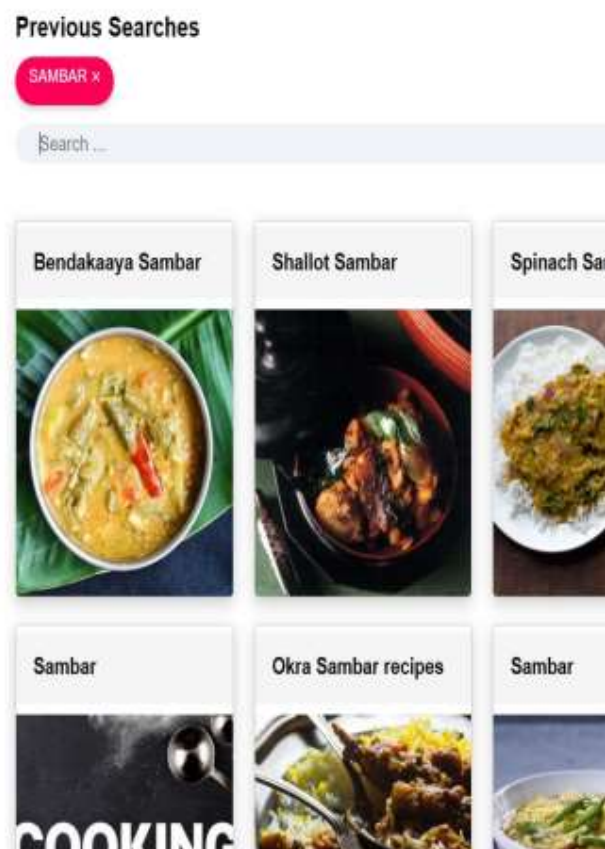


Fig.4.7 recipe Generated

4.2.1 Novelty of the proposal

The proposed Food Recipe Generator Application introduces a novel paradigm in the culinary landscape, amalgamating cutting-edge technologies with culinary expertise to deliver a personalized cooking experience unlike any other. At the core of its novelty lies an array of features designed to revolutionize how users approach meal planning and preparation. One of the standout features of this application is its integration of meal planning tools alongside the generation of shopping lists based on selected recipes. By offering a seamless and comprehensive solution for organizing meals, users can streamline their culinary endeavors, saving time and effort in the process. This holistic approach empowers users to plan their meals more efficiently, ensuring they have all the necessary ingredients at hand. Furthermore, the application distinguishes itself by its extensive support for a diverse range of global cuisines. Users can explore and experiment with recipes from various cultural backgrounds, broadening their culinary horizons and fostering a deeper appreciation for culinary diversity. Whether craving Italian pasta, Indian curry, or Japanese sushi, the application provides a treasure trove of culinary delights to explore. Central to the application's appeal is its user-friendly interface, which caters to individuals of all culinary skill levels. From novice cooks to seasoned chefs, users can easily navigate the interface and access a wealth of resources to enhance their cooking experience. Intuitive features and advanced functionalities ensure a seamless user experience, encouraging users to unleash their creativity in the kitchen. In addition to its innovative features for meal planning and recipe exploration, the Food Recipe Generator Application sets itself apart by integrating the Gemini API response generator . This integration enhances the user experience by providing a seamless channel for users to interact with the application and access recipe recommendations. By providing users with access to comprehensive recipe details, the application empowers them to make informed choices about their meals, promoting healthier eating habits and culinary experimentation. In essence, the Food Recipe Generator Application represents a paradigm shift in the culinary landscape, offering a multifaceted platform that combines convenience, diversity, and education. By harnessing the power of technology and culinary expertise, the application empowers users to embark on a culinary journey filled with exploration, creativity, and culinary delight.

4.2.2 Program Code

App.Js File:

```
import {
  BrowserRouter as Router,
  Switch,
  Route
} from "react-router-dom"
import LoginForm from "../components/Login"
import Footer from "../components/Footer";
import Home from "../pages/Home";
import Recipes from "../pages/Recipes";
import Settings from "../pages/Settings";
import Navbar from "../components/Navbar";
import DetailedRecipePage from "../components/DetailedRecipePage";
import SignUpForm from "../components/SignUpForm";
function App() {
  return (
    <>
      <Navbar/>
      <div className="container main">
        <Switch>
          <Route exact path="/login" component={LoginForm}/>
          <Route exact path="/" component={Home} />
          <Route exact path="/recipes" component={Recipes} />
          <Route exact path="/recipes/:title" component={DetailedRecipePage} />
          <Route exact path="/signup" component={SignUpForm} />
          <Route exact path="/settings" component={Settings} />
        </Switch>
      </div>
      <Footer />
    </>
  )
}
export default App;
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import { BrowserRouter as Router } from 'react-router-dom';
import "../styles/index.scss";

// Import createRoot function from react-dom
import { createRoot } from 'react-dom';

// Use createRoot to render your application
createRoot(document.getElementById('root')).render(
  <Router>
    <React.StrictMode>
      <App />
```

```

    </React.StrictMode>
  </Router>
);

```

DetailedRecipe.css and js

```

import React, { useState } from 'react';
import ContextProvider from '../context/Context';
import PropTypes from 'prop-types';
import { useHistory } from 'react-router-dom';
import './DetailedRecipePage.css'; // Import custom CSS file for styling

```

```

import Main from '../components/Main';

```

```

const DetailedRecipePage = ({ location }) => {
  const history = useHistory();
  const [isFavorite, setIsFavorite] = useState(false);

```

```

  const handleGoBack = () => {
    history.goBack();
  };

```

```

  if (!location.state) {
    return <div>Error: Recipe details not found.</div>;
  }

```

```

  // Access properties from location.state
  const { title, calories, image, ingredients } = location.state;

```

```

  return (
    <div className="detailed-recipe">
      <button className="delete-btn btn" onClick={handleGoBack}>
        {/* Back arrow icon */}
        ⬅️ Go back
      </button>

      <h2>{title}</h2>
      <div className="recipe-details-container">
        <div className="recipe-image-container">
          <img src={image} alt={title} />
        </div>
        <div className="recipe-info">
          <p className="calories">Calories: {calories}</p>

          </div>

```

```

    </div>
    <ContextProvider><Main title={title}/></ContextProvider>

```

```

    </div>
  );
};

DetailedRecipePage.propTypes = {
  location: PropTypes.object.isRequired,
};

export default DetailedRecipePage;

```

Login.js

```

import React, { useState } from 'react';
import { Redirect, Link } from 'react-router-dom';
import Cookies from 'js-cookie';
import './index.css';

const Login = ({ history }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [showSubmitError, setShowSubmitError] = useState(false);
  const [errorMsg, setErrorMsg] = useState("");
  const [isPasswordChecked, setIsPasswordChecked] = useState(false);
  const [isLoading, setIsLoading] = useState(false); // New state for loading

  const onSuccessLogin = (jwtToken) => {
    Cookies.set('jwt_token', jwtToken, { expires: 30 });
    history.replace('/');
  };

  const onSubmitFailure = (errorMsg) => {
    setErrorMsg(errorMsg);
    setShowSubmitError(true);
    setIsLoading(false); // Set loading to false on failure
  };

  if (response.ok) {
    onSuccessLogin(data.jwtToken);
  } else {
    if (data && data.error_msg) {
      onSubmitFailure(data.error_msg);
    } else {
      onSubmitFailure('An unexpected error occurred');
    }
  }
} catch (error) {
  console.error('Error:', error);
  onSubmitFailure('An unexpected error occurred');
}
};

```

```

    </div>
  );
};

```

```
export default Login;
```

```

<div className='main'>
  <div className="nav">
    <p className='spell-title'>Mom's Reci</p>
    
    </div>
    <div className="main-container">
      {showResult ? (
        <div className="result">
          <div className='result-title'>
            
            <p>{recentPrompt}</p>
          </div>
          <div className="result-data">
            
            {loading ? (
              <div className="loader">
                <hr className="animated-bg" />
                <hr className="animated-bg" />
              </div>
            ) : (
              <p dangerouslySetInnerHTML={{ __html: resultData }}></p>
            )}
          </div>
        </div>
      ) : (
        <>
          {userData && (
            <div className="greet">
              <p><span>Hello, {userData.name}</span></p>
              <p>I may give whatever you want....</p>
            </div>
          )}
          <div className="search-box">
            <input onChange={(e) => setInput(e.target.value)} value={`Generate recipe of
${title}`} type="text" placeholder={`Generate recipe of ${title}`} />
            <div>
              <img src={assets.gallery_icon} width={30} alt="" />

```

```

        <img src={assets.mic_icon} width={30} alt="" />
        {input ? <img onClick={() => onSent()} src={assets.send_icon} width={30}
alt="" /> : null}
      </div>
    </div>
  </>
)}

</div>
</div>
);

```

export default Main;

Navbar.js

```

import React from "react";
import { Link, useLocation } from "react-router-dom";
import Cookies from 'js-cookie';
import { useState } from "react";
import Sidebar from "./Sidebar";

import { faHome, faList, faCog } from "@fortawesome/free-solid-svg-icons";

export default function Navbar() {
  const onClickLogout = () => {
    Cookies.remove('jwt_token');
    window.location.href = '/login'; // Redirect to login page after logout
  };
  const [showSidebar, setShowSidebar] = useState(false);
  const location = useLocation();
  const jwtToken = Cookies.get('jwt_token');

  function closeSidebar() {
    setShowSidebar(false);
  }

  // Render Navbar only if the user is logged in (jwtToken exists)
  return (
    <>
      {jwtToken && (
        <div className="navbar container">
          <Link to="/" className="logo">Food<span>Reci</span>Gen</Link>
          <div className="nav-links">
            {links.map(link => (
              <Link className={location.pathname === link.path ? "active" : ""}
to={link.path} key={link.name}>{link.name}</Link>
            ))}
          </div>
          <div onClick={() => setShowSidebar(true)} className={showSidebar ?
"sidebar-btn active" : "sidebar-btn"}>

```

```

        <div className="bar"></div>
        <div className="bar"></div>
        <div className="bar"></div>
    </div>
    {/* Example usage of onClickLogout */}
    <button className='btn' onClick={onClickLogout}>Logout</button>
</div>
    })
    {showSidebar && <Sidebar close={closeSidebar} links={links} />}
</>
);
}

```

PreviousSearches.js

```

import React, { useState, useEffect } from 'react';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faSearch, faTimes } from '@fortawesome/free-solid-svg-icons';

export default function PreviousSearches({ onSearchSelect }) {
  const [search, setSearch] = useState("");
  const [previousSearches, setPreviousSearches] = useState([]);

  useEffect(() => {
    const storedSearches = localStorage.getItem('previousSearches');
    if (storedSearches) {
      setPreviousSearches(JSON.parse(storedSearches));
    }
  }, []);

  const handleSearchSelect = (search) => {
    onSearchSelect(search);
  };

  const handleSearchInputChange = (e) => {
    setSearch(e.target.value);
  };

  const handleDeleteSearch = (index) => {
    const updatedSearches = [...previousSearches];
    updatedSearches.splice(index, 1);
    setPreviousSearches(updatedSearches);
    localStorage.setItem('previousSearches', JSON.stringify(updatedSearches));
  };

  return (
    <div className="previous-searches section">
      <h2>Previous Searches</h2>
      <div className="previous-searches-container">
        {previousSearches.map((searchTerm, index) => (
          <div key={index} className="search-item">

```

```

        <span onClick={() =>
handleSearchSelect(searchTerm)}>{searchTerm}</span>
        <button onClick={() => handleDeleteSearch(index)} className="delete-
btn">
            <FontAwesomeIcon icon={faTimes} />
        </button>
    </div>
    )})
</div>
<form onSubmit={handleSearchSubmit}>
    <div className="search-box">
        <input type="text" placeholder="Search ..." value={search}
onChange={handleSearchInputChange} />
        <button type="submit" className="btn">
            <FontAwesomeIcon icon={faSearch} />
        </button>
    </div>
</form>
</div>
);
}

```

RecipeCard.js

```

import React from 'react';
import { Link } from 'react-router-dom';
import PropTypes from 'prop-types';
import { useHistory } from 'react-router-dom';
import './recipecard.css'
const RecipeCard = ({ title, calories, image, ingredients }) => {
    const history = useHistory();
    className='recipe-title'>{title}</h2>
        <div className='recipe-image'>
            <img src={image} alt={title} />
        </div>

    </div>
    );
};

```

```

RecipeCard.propTypes = {
    title: PropTypes.string.isRequired,
    calories: PropTypes.number.isRequired,
    image: PropTypes.string.isRequired,
    ingredients: PropTypes.arrayOf(PropTypes.shape({
        text: PropTypes.string.isRequired
    })).isRequired,
};

```

```

export default RecipeCard;
import { createContext, useState } from "react";

```

```

import runChat from "../config/gemini";
export const Context = createContext();
const ContextProvider = (props) => {
  const [prevPrompts, setPrevPrompts] = useState([]);
  const [input, setInput] = useState("");
  const [recentPrompt, setRecentPrompt] = useState("");
  const [showResult, setShowResult] = useState(false)
  const [loading, setLoading] = useState(false)
  const [resultData, setResultData] = useState("")

  function delayPara(index, nextWord) {
    setTimeout(function () {
      setResultData(prev => prev + nextWord)
    }, 75 * index);
  }

  const onSent = async (prompt) => {

    setResultData("")
    setLoading(true)
    setShowResult(true)
    let response;
    if (prompt !== undefined) {
      response = await runChat(prompt);
      setRecentPrompt(prompt)
    }
    else {
      setPrevPrompts(prev => [...prev, input]);
      setRecentPrompt(input)
      response = await runChat(input);
    }
    let responseArray = response.split('**');
    let newArray = "";
    for (let i = 0; i < responseArray.length; i++) {
      if (i === 0 || i % 2 !== 1) {
        newArray += responseArray[i]
      }
      else {
        newArray += "<b>" + responseArray[i] + "</b>"
      }
    }
    console.log(newArray);
    responseArray = newArray.split('*').join("</br>").split(" ");
    for (let i = 0; i < responseArray.length; i++) {
      const nextWord = responseArray[i];
      delayPara(i, nextWord + " ")
    }
    setLoading(false);
    setInput("")
  }
}

```

```

const newChat = async () => {
  setLoading(false);
  setShowResult(false);
}
const contextValue = {
  prevPrompts,
  setPrevPrompts,
  onSent,
  setRecentPrompt,
  recentPrompt,
  showResult,
  loading,
  resultData,
  input,
  setInput,
  newChat
}

return (
  <Context.Provider value={contextValue}>
    {props.children}
  </Context.Provider>
)
}

export default ContextProviderR
import React from 'react';
import ContextProvider from '../context/Context'; // Import your context provider
component
import HeroSection from '../components/HeroSection';
import ImproveSkills from '../components/ImproveSkills';
import QouteSection from '../components/QuoteSection';
import Main from '../components/Main'
export default function Home() {
  return (
    <ContextProvider> { /* Wrap your Main component with the context provider */}
    <div>
      <HeroSection />
      <ImproveSkills />
      <QouteSection />

    </div>
    </ContextProvider>
  );
}
import React, { useState, useEffect } from 'react';
import RecipeCard from "../components/RecipeCard";
import PreviousSearches from "../components/PreviousSearches";
import { Rings } from 'react-loader-spinner';

```

```

import './Recipes.css'
const Recipes = () => {
  const app_id = "158540c6";
  const app_key = "4cc951f3ecf74138534e62890c0537ce";

  const [recipes, setRecipes] = useState([]);
  const [search, setSearch] = useState("");
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const getRecipes = async () => {
    setLoading(true);
    setError(null);

    try {
      const response = await
fetch('https://api.edamam.com/api/recipes/v2?type=public&q=${search}&app_id=${app_i
d}&app_key=${app_key}');
      if (!response.ok) {
        throw new Error('Failed to fetch recipes');
      }
      const data = await response.json();
      setRecipes(data.hits);
    } catch (error) {
      setError(error.message);
    } finally {
      setLoading(false);
    }
  };
  export default Recipes;
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome"
import { faCheck } from "@fortawesome/free-solid-svg-icons"
export default function Settings(){
  const [settings, setSettings] = useState({
    "--background-color": "#fff",
    "--background-light": "#fff",
    "--primary-color": "rgb(255, 0, 86)",
    "--shadow-color": "rgba(0,0,0,0.2)",
    "--text-color": "#0A0A0A",
    "--text-light": "#575757",
    "--font-size": "16px",
    "--animation-speed": 1
  })
  useEffect(() => {
    const root = document.documentElement
    for(let key in settings){
      root.style.setProperty(key, settings[key])
    }
  }, [settings])

  const primaryColors = [

```

```

    "rgb(255, 0, 86)",
    "rgb(33, 150, 243)",
    "rgb(255, 193, 7)",
    "rgb(0, 200, 83)",
    "rgb(156, 39, 176)"
  ]
  const fontSizes = [
    {
      title: "Small",
      value: "12px"
    },
    {
      title: "Medium",
      value: "16px"
    },
    {
      title: "Large",
      value: "20px"
    }
  ]
  const animationSpeeds = [
    {
      title: "Slow",
      value: 2
    },
    {
      title: "Medium",
      value: 1
    },
    {
      title: "Fast",
      value: .5
    }
  ]
  const [primaryColor, setPrimaryColor] = useState(0)
  const [fontSize, setFontSize] = useState(1)
  const [animationSpeed, setAnimationSpeed] = useState(1)
  return (
    <div>
      <div className="section d-block">
        <h2>Primary Theme</h2>
        <div className="options-container">
          <div className="option dark" onClick={() => changeTheme(1)}>
            { theme === "dark" && (
              <div className="check">
                <FontAwesomeIcon icon={faCheck} />
              </div>
            ) }
          </div>
        </div>
      </div>
    </div>
  )

```

```

    </div>
  </div>
  <div className="section d-block">
    <h2>Preferred color</h2>
    <div className="options-container">
      { primaryColors.map((color, index) => (
        <div key={index} className="option light" style={{backgroundColor:
color}} onClick={() => changeColor(index)}>
          { primaryColor === index && (
            <div className="check">
              <FontAwesomeIcon icon={faCheck} />
            </div>
          ) }
        </div>
      ) ) }
    </div>
  </div>
  <div>
    <div className="section d-block">
      <h2>Font size</h2>
      <div className="options-container">
        { fontSizes.map((size, index) => (
          <button key={index} className="btn" onClick={() =>
changeFontSize(index)}>
            {size.title}
            { fontSize === index && <span><FontAwesomeIcon icon={faCheck}
/></span> }
          </button>
        ) ) }
      </div>
    </div>
    <div className="section d-block">
      <h2>Animation speed</h2>
      <div className="options-container">
        { animationSpeeds.map((speed, index) => (
          <button key={index} className="btn" onClick={() =>
changeAnimationSpeed(index)}>
            {speed.title}
            { animationSpeed === index && <span><FontAwesomeIcon
icon={faCheck} /></span> }
          </button>
        ) ) }
      </div>
    </div>
  </div>
)
}

```

CHAPTER 5

CONCLUSION

The Recipe application represents a significant advancement in the realm of culinary technology, offering users a comprehensive and personalized cooking experience. By harnessing the power of the Gemini API, we have created a platform that empowers users to explore a vast array of recipes tailored to their preferences and available ingredients. The incorporation of innovative features such as meal planning tools, shopping list generation, and advanced filtering options elevates the user experience, streamlining the meal preparation process and promoting culinary creativity. Moreover, our commitment to education ensures that users not only discover new recipes but also enhance their culinary skills and knowledge. Through the provision of cooking tips, nutritional information, and educational resources, we aim to empower users to become more confident and proficient cooks. With a focus on user engagement and accessibility, the Recipe application boasts an intuitive interface and inclusive design, welcoming users of all backgrounds and skill levels. As we continue to refine and expand the application, we remain dedicated to our mission of making cooking enjoyable, accessible, and educational for everyone. In essence, the Recipe application represents more than just a digital cookbook—it is a comprehensive culinary companion that inspires creativity, fosters learning, and facilitates a deeper appreciation for the art of cooking. The Recipe application serves as an invaluable educational resource, providing users with a wealth of culinary knowledge and expertise. From informative articles on cooking techniques to in-depth nutritional analyses of ingredients, users can expand their culinary horizons and sharpen their culinary skills with every interaction. Whether you're a novice cook embarking on your culinary journey or a seasoned chef looking to refine your craft, the Recipe application offers a wealth of resources to help you succeed in the kitchen. Recipe application remains committed to innovation and excellence, continuously evolving to meet the changing needs and preferences of our users. With ongoing updates and enhancements, we strive to elevate the cooking experience, empower culinary enthusiasts, and foster a community of passionate food lovers united by their shared love of good food and great recipes.

REFERENCES

1. Florian Pecune, Lucile Callebort, Stacy Marsella, "A Recommender System for Healthy and Personalized Recipe Recommendations", University of Glasgow, Scotland: UK, September 2023.
2. Zhu L, Spachos P, Pensini E, Plataniotis KN (2021) Deep learning and machine vision for food processing: a survey. *Curr Res Food Sci* 4:233–249.
3. ajayogi JR, Manjunath G, Shobha G (2019) Indian food image classification with transfer learning. In: 2019 4th international conference on computational systems and information technology for sustainable solution (CSITSS), vol 4. IEEE, pp 1–4.
4. Fangda Han, Ricardo Guerrero, Vladimir Pavlovic; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 1450-1458.
5. Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. 2018. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 35--44.
6. Technopedia. What is Concurrency? | Definition from Technopedia. Retrieved May 11, 2018 from techopedia.
7. Katherine Chou, Xavier Ducrohet, Tor Norbye. 2013. Android Developers Blog: Android Studio: An IDE built for Android. Retrieved May 5, 2018 from [Link].
8. Keiji Yanai, Takuma Maruyama and Yoshiyuki Kawano,"A Cooking Recipe Recommendation System with Visual Recognition of Food Ingredients", The University of Electro-Communications, Tokyo: Japan, in 2021.
9. API: API data that used to generate data based on user choice is generated by edamam API generator from edamam API.
10. Icons: Icons that used in this project is used from Flaticon.
11. Recipe Generator using Deep Learning ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022-.

APPENDIX

RESEARCH PAPER

Similarity Report

PAPER NAME

**Enhancing Conversational AI Developing
an Intelligent Chatbot with ReactJS and
Gemini AI API.docx**

WORD COUNT

2857 Words

CHARACTER COUNT

17812 Characters

PAGE COUNT

4 Pages

FILE SIZE

338.0KB

SUBMISSION DATE

Apr 29, 2024 10:28 AM GMT+5:30

REPORT DATE

Apr 29, 2024 10:28 AM GMT+5:30

● 16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 12% Internet database
- 10% Publications database
- Crossref database
- Crossref Posted Content database
- 13% Submitted Works database

● Excluded from Similarity Report

- Small Matches (Less than 10 words)

Summary

● 16% Overall Similarity

Top sources found in the following databases:

- 12% Internet database
- 10% Publications database
- Crossref database
- Crossref Posted Content database
- 13% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	jptcp.com Internet	2%
2	Melbourne Institute of Technology on 2024-03-31 Submitted works	2%
3	University of Teesside on 2024-01-12 Submitted works	1%
4	Siddhant Meshram, Namit Naik, Megha VR, Tanmay More, Shubhangi ... Crossref	1%
5	ijert.org Internet	1%
6	ijisrt.com Internet	1%
7	CSU, San Jose State University on 2023-05-16 Submitted works	1%
8	Purdue University on 2024-03-26 Submitted works	<1%

Sources overview

A. RESEARCH PAPER :

Enhancing Conversational AI: Developing an Intelligent Chatbot with ReactJS and Gemini AI API

Kakara Venkata Nithish Sarma Kasiralla Praveen Kumar Pandu Krishna Chaitanya Bunga DhanushMarupakka Dr.Sonia Jenifer Rayen,M.Tech.,Ph.d.,
Nithish.sharma99@gmail.com praveenpandu3039@gmail.com krishnachaitanyabunga@gmail.com vmarupakka6@gmail.com sonijr1@gmail.com
Sathyabama University Sathyabama University Sathyabama University Sathyabama University Sathyabama University

Abstract—This study explores the creation of an AI-driven chatbot with advanced natural language processing utilizing the Gemini AI API and front-end programming using ReactJS. Data storage and user authentication are handled using MongoDB. The conversational experience is enhanced by important features including user authentication, voice-to-text input, code creation, and text production. Text creation techniques, voice-to-text integration, code generating processes, and MongoDB authentication are all covered in detail. Future directions include task automation, IoT integration, enhanced language understanding, interactive learning, voice-to-text translation, augmented reality integration, multi-language support, sentiment analysis, contextual understanding, personalization, and integration with external APIs for real-time data. With these improvements, the chatbot should be easier to use, more flexible, and able to accommodate a wider range of user requirements and preferences.

Keywords—Chatbot, MongoDB, API, Gemini AI, ReactJS, Node.js

I. INTRODUCTION

Conversational AI has become a key technology in today's digital world, completely changing the way people communicate with machines. Using ReactJS and the Gemini AI API, this research paper explores the creation of an artificial intelligence chatbot and provides a thorough analysis of the project named "Building an AI Chatbot with ReactJS and Gemini AI API." By combining these technologies, a comprehensive platform for fluid human-computer interaction is provided, representing a major breakthrough in the fields of natural language processing and user interface design.

The combination of AI and human-computer interaction has brought out a new age of ease and innovation in the field of modern technology. The construction process of an artificial intelligence chatbot is explored in this study article, which is appropriately named "Building an AI Chatbot with ReactJS and Gemini AI API." This project is essentially an amalgam of state-of-the-art technology designed to improve user experience and natural language processing. Through the utilization of the Gemini AI API and ReactJS, this chatbot is a paradigm change in the way people engage with digital systems.

The foundation of this project is ReactJS, which was selected for its unmatched adaptability and effectiveness in creating dynamic front-end user interfaces. Its robust ecosystem and modular design offer a strong base on which to construct engaging and responsive user interfaces, which are necessary to give a smooth conversational experience. The Gemini AI API, a complex toolkit that gives the chatbot significant natural language understanding skills, is a great addition to ReactJS. The Gemini AI API bridges the gap between human and machine intelligence by enabling the chatbot to understand user inquiries with exceptional accuracy and fluency using machine learning and deep learning methods.

We go into the nuances of the features put in place and the technology used as we set out on this investigation to fulfill the dream of an AI-driven chatbot. Every facet of the chatbot's development is carefully examined, from voice-to-text input integration and code generating features to user authentication methods that make use of MongoDB. We simplify the challenges of integrating ReactJS with the Gemini AI API to produce a fluid conversational interface by providing in-depth implementation insights.

Furthermore, this research paper serves as a springboard for envisioning the future directions of AI-driven conversational interfaces. By proposing feature developments such as multi-language support, sentiment analysis, and integration with external APIs, we strive to push the boundaries of what is possible in AI-powered interaction. Ultimately, this introduction sets the stage for a comprehensive exploration of the transformative potential of AI chatbots in reshaping the digital landscape and enhancing user experiences across diverse domains.

Provides a starting point for a thorough analysis of the features included, the technology used, and the minute implementation details that went into making this AI chatbot. We hope to shed light on the revolutionary possibilities of AI-driven conversational interfaces and offer insights into the future paths of this emerging industry through our investigation.

II. RELATED WORKS

However, manual interactions pose challenges, risking

customer and partner relationships. Leveraging AI, particularly conversational AI and chatbots, offers automation solutions. proposed AI-driven approach streamlines onboarding, providing real-time project status, multilingual support, cost reduction, and 24/7 availability. Personalized chatbots tailored to user logins ensure efficient data updates. [1]Deployed on the cloud, this solution benefits service providers, customers, and trading partners, enhancing efficiency and fostering stronger business relationships.

By providing developers with a strong foundation to build intelligent AI assistants, the Rasa framework revolutionizes customer-business communication by enabling sophisticated natural language interpretation and dialogue management. "Conversational AI with Rasa" walks readers through the process of creating a variety of chatbots, including task-oriented, FAQbased, and knowledge graph-driven models, by exploring the fundamental architecture and guiding principles of Rasa. This book covers a wide range of strategies for developing efficient conversational agents, from setting up and training chatbots to using form-based dialogue management and answer pickers. To provide a thorough knowledge of utilizing Rasa for next-generation AI assistants, readers also learn how to configure Rasa, [2]deploy systems to production environments, and apply best practices for creating effective and scalable chat systems.

In the post-Moore age, chiplet technology emerges as a key [3]solution that makes it possible to integrate an increasing number of transistors onto a single accelerator, hence satisfying the increasing computing needs driven by the fast breakthroughs in AI. But these developments also have drawbacks, including greater packaging costs and Die-to-Die interface prices; moreover, they need large space, consume more power, and provide less bandwidth than on-chip interconnects. We propose Gemini, a layer-centric encoding approach coupled with a flexible hardware template, to overcome these issues and optimize chiplet technology, allowing co-exploration of mapping and architecture for large-scale DNN chiplet accelerators.

including Microsoft Copilot, ChatGPT, GPT-4, BlenderBot-3, and Gemini. The report offers vital information into the current level of chatbot technology and helps installers make educated decisions by clarifying the capabilities of various chatbots.[4] The research provides a thorough evaluation of chatbot efficacy by assessing elements including emotive presentation, undesirable content management, and response to human interaction. Aspects such as conversation applicability, costeffectiveness, data quality, relevancy of replies, bias control, and emotional expression are painstakingly analyzed through stringent review processes, which include particular questions catered to each chatbot.

InvestingIQ serves easily accessible information and quantitative analysis tools for navigating the stock market. Providing current and historical stock price data, alongside company news and activity details, the platform equips investors with essential insights. Incorporating machine learning models like Facebook's

Prophet and Random Forest for stock price predictions, along with sentiment analysis for assessing public perception,[5] InvestingIQ empowers investors to make informed decisions. Additionally, metrics derived from company balance sheets aid in evaluating performance, enabling investors to assess potential investment opportunities efficiently and confidently.

This study explores the growing use of AI chatbots in e-commerce with the goal of lowering labor costs and improving service efficiency. Nonetheless, opinions of intelligent customer service continue to differ among consumers. The study explores system parameters and user characteristics and their effects on user trust and service perception.[6] A trust research model for AI chatbots is developed using social presence theory as a basis. It shows that while cognitive reactance has a negative effect on social presence, perceived personalization, media richness, and prior usage experience favorably affect social presence.

In this study, a unique chatbot that answers commonly asked questions (FAQs) is introduced. Its capabilities go beyond text-based queries to include the ability to see images. While FAQs are often managed by people, this study suggests an automated approach to improve productivity. The chatbot can read user-provided photos and deliver pertinent answers in addition to written responses[7]. Users can switch to a human agent who can handle the query and update the database if they are still not pleased with the chatbot's replies. This novel method optimizes knowledge management procedures and user happiness by seamlessly integrating automated and human-assisted support.

The effectiveness of rapid engineering, a well-known Generative AI approach, is examined in this work, specifically in relation to models based on natural language processing. The research presents a query transformation module to maximize conversational answer generation with low data needs, with a focus on Korean-based LLMs. [8]The QTM improves LLM comprehension by converting input prompts into structured inquiries, which results in more precise and organic replies. The efficacy of the suggested method is demonstrated by the validation using two Korean LLMs, SKT GPT-2 and Kakaobrain KoGPT-3, which exhibit an impressive 11.46% average improvement in sentence creation.

As well as the expanding popularity of voice commands on gadgets like Google Home and Amazon Echo, highlight how dependent we are on speech recognition technologies. The Speech API has made it possible for users to effortlessly integrate voice input into online apps. The online Speech API makes advantage of contemporary online capabilities to facilitate the creation of interactive websites that appeal to a wide variety of users. [9]This project focuses on using chatbots to improve the usability and accessibility of rich online applications by offering instructional help. Additionally, the API has the ability to support those with impairments or injuries, highlighting its potential for accessible digital interactions.

Because public transit is so widely used, effective navigation aids are essential, especially for visitors to new places. In response to this requirement, our AI chatbot proves to be a useful tool, utilizing deep learning and NLP to enable smooth communication between humans and machines. The chatbot retrieves pertinent data from its database by asking intelligent questions to validate users' present position and intended destination. Utilizing Python's Keras library in conjunction with Tkinter for graphical user interface,[10]

In the current era Chatbots improve user experience and productivity by providing quick, accurate answers to often requested queries, in contrast to traditional methods. Chatbots use NLP and AI to enable natural human-machine interactions that promote language acquisition and understanding. NLP techniques like pattern matching enable chatbots to comprehend user requests and provide relevant answers.[11] Text-to-text, text-to-speech, and speech-to-speech chatbots are just a few of the varieties that have developed over time to meet a variety of communication needs and preferences.

III. FUNCTIONALITIES INTEGRATED

- **User Authentication:** MongoDB offers robust support for user authentication, providing a secure mechanism to control access to databases and resources. The authentication process involves the creation of user accounts with associated roles and privileges, which are then verified during connection attempts.
- **Voice-to-text-input:** Instead of inputting text into the chatbot interface, users may utilize voice instructions thanks to the voice-to-text input functionality. By offering a realistic hands-free communication option, it improves the user experience and is especially helpful for individuals with limited mobility or in circumstances where typing is not practicable.
- **Code-Generation:** The code-generation feature enables the chatbot to analyze user inquiries and generate relevant code snippets as responses. This functionality caters to users seeking assistance with programming tasks or troubleshooting issues, providing them with actionable solutions in the form of executable code.
- **Text generation:** Users may have natural, engaging discussions with the chatbot by integrating the text generation function, and they will receive timely, useful replies that are customized to meet their wants and inquiries. This improves the user experience in general and raises the usefulness and efficacy of the chatbot across a range of applications.

IV. CAPABILITY ADVANCEMENTS

- **Multi-language Support:** Boost the chatbot's ability to comprehend and answer to user

inquiries in a variety of languages, successfully serving a varied and international user base.

- **Sentiment analysis:** is the process of using algorithms to identify the emotional tone and sentiment that users are conveying in their messages. This allows the chatbot to respond to users' messages with empathy and understanding.
- **Integration with External APIs:** Enhance the chatbot's ability to provide consumers with seamless real-time information and services by integrating it with external application programming interfaces (APIs), such as those for news feeds, weather updates, and ecommerce data.
- **Contextual Understanding:** Improve the chatbot's ability to remember context throughout exchanges, enabling more seamless and organic interactions with users across longer time frames.
- **Voice and Video Calling:** Extend the chatbot's capabilities to include voice and video calling. This will allow consumers to connect with the chatbot more effectively by using both audio and visual channels.
- **Task Automation:** By integrating the chatbot seamlessly with relevant service providers, you may enable it to do activities on behalf of users, such as making bookings, arranging appointments, or completing orders. This will improve convenience and streamline user workflows.
- **Improved Natural Language Understanding:** By utilizing sophisticated NLP techniques, such as utilizing pre-trained language models and fine-tuning on domain-specific data, the chatbot's ability to understand natural language will be continuously improved. This will result in more accurate and contextually relevant user interactions.
- **Interactive Learning:** Implement interactive learning systems that let users to provide input on the chatbot's answers, gradually improving its precision and effectiveness.
- **Voice-to-Text Translation:** Allow users to speak with the chatbot by using voice commands to enter text. This feature translates spoken words into text for smooth processing, making the chatbot more accessible and user-friendly for all users.
- **Examine how augmented reality:** technology may be used with chatbots to create immersive experiences. This might involve adding interactive AR overlays and virtual avatars to real-world environments to increase

user engagement and interaction opportunities.

V.OUTPUT & RESULTS



Fig.5.7 The user interface of the proposed Chatbot



Fig.5.8 The interactive phase of Chatbot

VII. CONCLUSION & FUTURE WORKS:

The creation of the chatbot, which makes use of the Gemini AI API and ReactJS, has greatly improved accessibility and user engagement across a range of fields. Advanced features like sentiment analysis, multilingual support, and interaction with other APIs have expanded the breadth of user involvement and provide individualized, real-time replies. The use of voice-to-text translation and augmented reality integration has enhanced user experiences and improved the chatbot's usability even more. By means of ongoing improvements to interactive learning processes and natural language comprehension, chatbots have developed into a flexible and effective means of communication and task automation.

In the future, there are several opportunities for the functionality and capabilities of the chatbot to be further enhanced and innovative. To allow more complex and contextually appropriate replies, future study might concentrate on improving contextual comprehension. Further investigation into IoT device integration may also open up new avenues for customized help and home automation. Richer and more immersive user interactions may be possible with more study and development in voice and video calling features.

Furthermore, the continuous progress in natural language processing and machine learning algorithms presents prospects for further refining the chatbot's natural language comprehension and answer generating skills. Furthermore, investigating cuttingedge technologies like virtual reality (VR) and

augmented reality (AR) may provide new paths for developing engaging chatbot experiences that further conflate virtual and real-world interactions. In general, the chatbot's future rests on its ability to innovate and adapt further to satisfy users' changing demands and preferences in a world that is becoming more digitally linked and automated.

REFERENCES

1. G. R, M. S. Sodhi and K. S. Kumar, "Supply Chain Partner Onboarding using Conversational AI and NLP in B2B Context," 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-4, doi: 10.1109/CONECCT55679.2022.9865838.
2. iaoquan Kong; Guan Wang; Alan Nichol, Conversational AI with Rasa: Build, test, and deploy AI-powered, enterprise-grade virtual assistants and chatbots , Packt Publishing, 2021.
3. J. Cai et al., "Gemini: Mapping and Architecture Co-exploration for Large-scale DNN Chiplet Accelerators," 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Edinburgh, United Kingdom, 2024, pp. 156-171, doi: 10.1109/HPCA57654.2024.00022.
4. S. Arya, A. Bhaskar and K. Gupta, "Conversational Ai: A Treatise About Vying Chatbots," 2024 2nd International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 2024, pp. 929-934, doi: 10.1109/ICDT61202.2024.10489545.
5. "InvestingIQ – Algorithmic Trading Dashboard Powered by AI & ML Built Using ReactJS & Python," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1 - 13 , doi : 10 . 11 0 9 / ICCCNT56998.2023.10306592.
6. F. Min, Z. Fang, Y. He and J. Xuan, "Research on Users' Trust of Chatbots Driven by AI: An Empirical Analysis Based on System Factors and User Characteristics," 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 2021, pp. 55-58, doi: 10.1109/ICCECE51280.2021.9342098.
7. R. Vannala, S. B. Swathi and Y. Puranam, "AI Chatbot For Answering FAQ's," 2022 IEEE 2nd International Conference on Sustainable Energy and Future Electric Transportation (SeFeT), Hyderabad, India, 2022, pp. 1-5, doi: 10.1109/SeFeT55524.2022.9908774.
8. D. Park, G. -t. An, C. Kamyod and C. G. Kim, "A Study on Performance Improvement of Prompt

Engineering for Generative AI with a Large Language Model," in Journal of Web Engineering, vol. 22, no. 8, pp. 1187-1206, November 2023, doi: 10.13052/jwe1540-9589.2285.

9. P. V. K. V. Prasad, N. V. Krishna and T. P. Jacob, "AI CHATBOT using Web Speech API and Node.js," 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 360-362, doi: 10.1109/ICSCDS53736.2022.9760803.
10. M. Dharani, J. V. S. L. Jyostna, E. Sucharitha, R. Likitha and S. Manne, "Interactive Transport Enquiry with AI Chatbot," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 1271-1276, doi: 10.1109/ICICCS48265.2020.9120905.
11. A. Rane, C. Ranade, H. Bandekar, R. Jadhav and V. Chitre, "AI driven Chatbot and its Evolution," 2022 5th International Conference on Advances in Science and Technology (ICAST), Mumbai, India, 2022, pp. 170-173, doi: 10.1109/ICAST55766.2022.10039515.

B. SOURCE CODE

APP.JSX :

```
import React, { useState } from 'react' import { Switch,Route } from
'react-router-dom' import Signup from './components/signup/signup'
import Home from './components/Home' import Login from
'./components/Login/login' import ProtectedRoute from
'./components/ProtectedRoute' import PrivateRoute from
'./components/PrivateRoute/private' const App = () => { return (  <>
  <Switch>

    <PrivateRoute exact path="/signup" component={Signup} />

    <PrivateRoute exact path="/login" component={Login} />

    <ProtectedRoute exact path="/" component={Home } />

  </Switch>

</>
)
}
export default App
```

PACKAGE.JSON:

```
{
  "name": "gemini-clone",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@google/generative-ai": "^0.2.1",
```

```

    "axios": "^0.21.4",
    "bootstrap": "^5.3.3",
    "js-cookie": "^3.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^6.1.6",
    "react-router-dom": "^5.3.4"
  },
  "devDependencies": {
    "@types/react": "^18.2.56",
    "@types/react-dom": "^18.2.19",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.56.0",
    "eslint-plugin-react": "^7.33.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "vite": "^5.1.4"
  }
}
MAIN.JSX:

```

```

import React from 'react' import ReactDOM from
'react-dom/client' import App from './App.jsx'
import './index.css'
import { BrowserRouter } from 'react-router-dom' import
ContextProvider from './context/Context.jsx'

```

```

ReactDOM.createRoot(document.getElementById('root')).render(
  <BrowserRouter>
    <ContextProvider>
      <App />
    </ContextProvider>
  </ContextProvider>
)

```

```
</BrowserRouter>
```

```
)
```

GEMINI.JS :

```
import {
  GoogleGenerativeAI,
  HarmCategory,
  HarmBlockThreshold, } from
"@google/generative-ai"; const MODEL_NAME
= "gemini-1.0-pro";
// Paste Your API KEY Below

const API_KEY = "AIzaSyDwjLUrNfEgCXoaylZqd-UP1eFRLF6i8g"; async
function runChat(prompt) {
  const genAI = new GoogleGenerativeAI(API_KEY);

  const model = genAI.getGenerativeModel({ model: MODEL_NAME }); const
generationConfig = { temperature: 0.75, topK: 1, topP: 1,
  maxOutputTokens: 2048,

  };

  const chat = model.startChat({
generationConfig, safetySettings,
history: [
  ],

  });

  const result = await chat.sendMessage(prompt); const
response = result.response; return response.text();
}

export default runChat;
```

