# COOK UP: A RECIPE SHARING APPLICATION

**PROJECT REPORT**

**Submitted by:**

**HARSHINI AKSHAYA A S** **220701088**

**HARISH RAGAVENDAR S** **220701087**

**In partial fulfilment for the award of the degree of**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**


**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**

**THANDALAM**

**CHENNAI-602105**

**2023 – 2024**

**BONAFIDE CERTIFICATE**

Certified that this project '**COOK UP: A RECIPE STORAGE APPLICATION'** is the bona fide work of '**HARISH RAGAVENDAR S (220701087)**, **HARSHINI AKSHAYA A S (220701088)**' who carried out this project under my supervision.

**SIGNATURE**
**Dr. R. SABITHA**
**ACADEMIC HEAD**
**PROFESSOR**
Dept. of Computer of Science Engg,
Rajalakshmi Engineering College,
Chennai.

**SIGNATURE**
**Dr. G. DHARANI DEVI**
**ASSOCIATE PROFESSOR,**
Dept. of Computer of Science Engg,
Rajalakshmi Engineering College,
Chennai.

**Submitted of the Practical Examination held on _____**

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ABSTRACT

Cook Up is a web application designed to streamline recipe storage and meal plans. Built with a user-friendly interface, it empowers users to save their favourite recipes and organize them efficiently. The application practices a modern tech stack, including MongoDB, React, and Bootstrap CSS, to deliver a responsive and secure experience.

Cook Up tackles the challenge of managing scattered recipes and forgotten meal routines. Users can create accounts and securely store their culinary. The application integrates seamlessly with MongoDB, a NoSQL database, to store and retrieve recipe data effectively.

**Key functionalities of Cook Up include:**

- User accounts for secure recipe storage
- Recipe saving and organization capabilities
- Integration with MongoDB for data persistence
- Dynamic and responsive interface built with React
- User-friendly design facilitated by Bootstrap CSS

# TABLE OF CONTENTS

# 1. INTRODUCTION

## 1.1 COOK UP: A RECIPE STORAGE APP

Cook Up is a web application that allows users to efficiently manage their recipes and meal plans. Users can perform basic operations such as adding new recipes, organizing them, and viewing their collection. Each recipe in the application has a unique identifier. Users can save a recipe by entering its details, including ingredients and instructions. Each user can store multiple recipes and access them at any time. When a user updates a recipe, the changes are reflected immediately in their collection. Users can also view detailed records of all their saved recipes.

## 1.2 IMPLEMENTATION

The 'COOK UP: A RECIPE STORAGE APPLICATION' project is implemented using React, a JavaScript library, and MongoDB for a database. This user-friendly application is used to address the challenges of organizing and managing personal recipe collections. The system includes functionalities like creating recipes, saving recipes, adding favourites, login authentication, and user profile management with logout options.

Upon launching the application, users are greeted with a login page for secure access. Once logged in, users are landed to the main interface,
A home page which features options such as creating new recipes, saving favourite recipes, and viewing saved recipes. Users can manage their profiles, including updating personal information and logging out securely.

The application interacts with a MongoDB database to store and retrieve recipe data efficiently. The front end is built with React, providing a dynamic and responsive interface that enhances user experience. The design utilizes Bootstrap CSS for a clean and intuitive layout.

## 1.3 FUNCTIONALITIES

**Login Authentication:**

Secure login system to ensure user data privacy. By logging in, the users can access or create their own collections.

**Create Recipe:**

Users can add new recipes by entering details such as ingredients, instructions, and category. Users are also provided with a description text field to enter their recipe plan to their content.

**Saved Recipes:**

Users can view and manage their collection of saved recipes.

**Add Favourites:**

Users can mark recipes as favourites for easy access.

**User Profile Management:**

Options to update user information and securely log out.

# 2.SURVEY OF TECHNOLOGY

## 2.1 SOFTWARE DESCRIPTION

### REACT (JAVASCRIPT):

React is a popular JavaScript library for building user interfaces, particularly single-page applications.

### MONGODB:

MongoDB is a NoSQL database known for its flexibility and scalability. It stores data in JSON-like documents, which makes it to work with data in a format that is natural to use in modern applications.

### BOOTSTRAP CSS:

Bootstrap is a front-end framework that provides pre-designed CSS and JavaScript components. It helps developers quickly create responsive and mobile-first web designs.

### NODE.JS:

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js is used to create the backend server

## 2.2 LANGUAGES

- React.JS (JavaScript Library),
- HTML, CSS,
- JSON (JavaScript Object Notation)

# 3.SYSTEM_REQUIREMENTS AND ANALYSIS

## 3.1 HARDWARE SPECIFICATION:

| | |
|---|---|
| PROCESSOR | - Intel® core™ i5-12400F@ 2.50 GHz |
| RAM | - 8 GB |
| OPERATING SYSTEM | - Microsoft Windows 10/11 |
| HARD DISK | - 1 GB of free space |

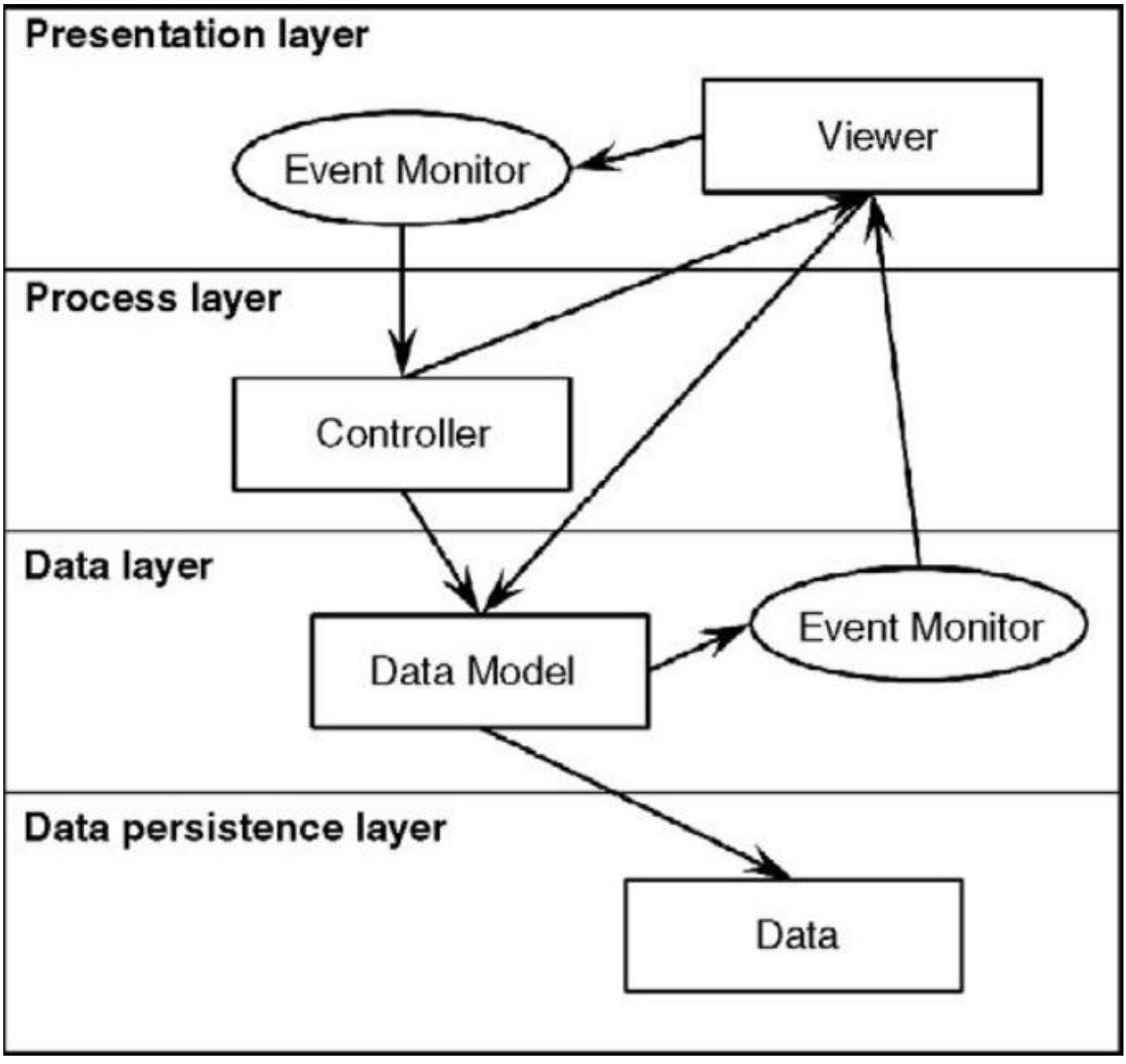
**PRE- REQUISITES:**

NODE.JS PACKAGE
CREATE-REACT-APP PACKAGE
BOOTSTRAP PACKAGE
C/C++ PACKAGE
MONGODB
MONGO SHELL/ ATLAS

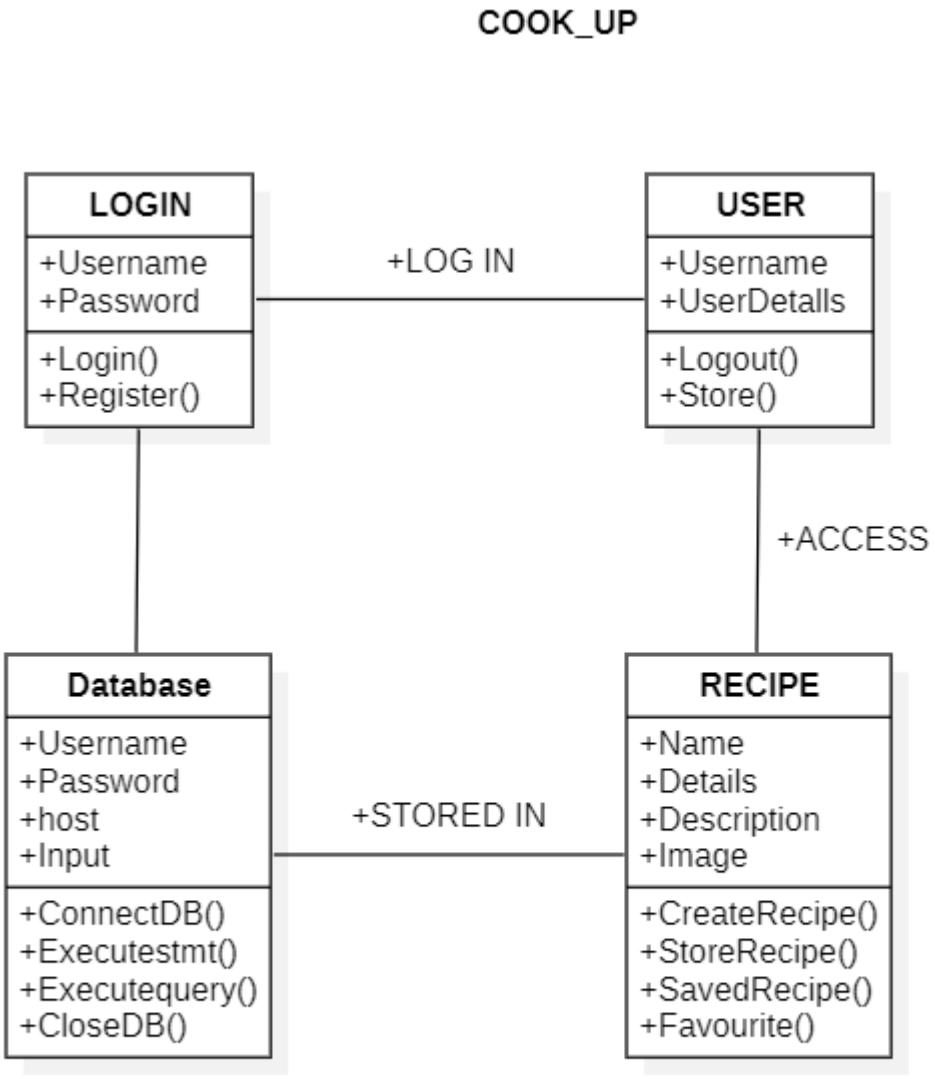
## 3.2 SOFTWARE SPECIFICATION

| | |
|---|---|
| OPERATING SYSTEM | : Microsoft Windows 10 |
| SOFTWARE REQUIRED | : Node.JS, JavaScript (React), MongoDB |

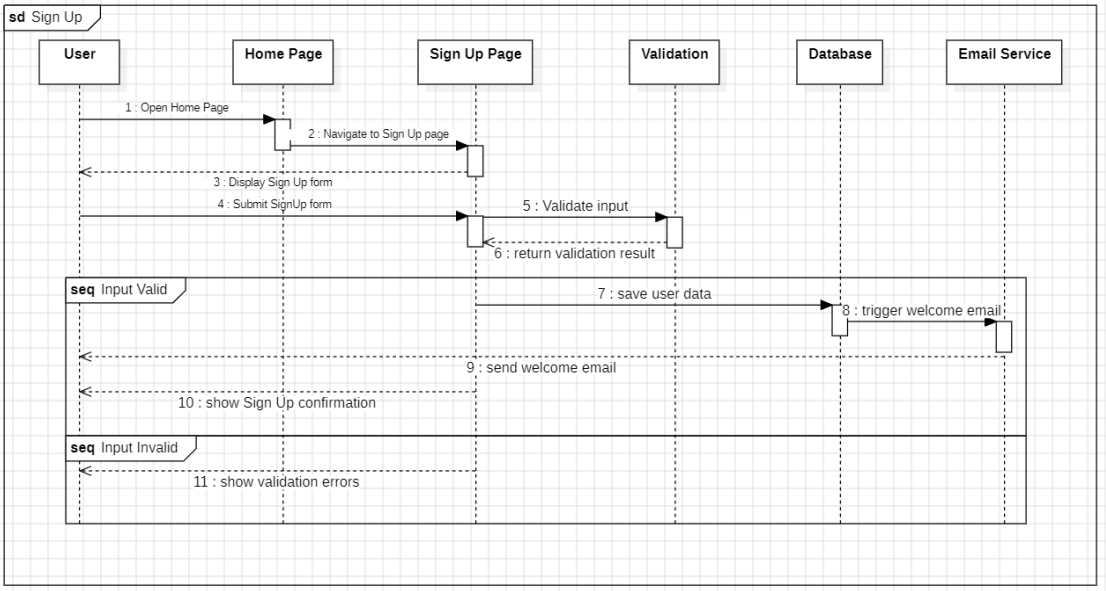
### 3.3 ARCHITECTURE DIAGRAM

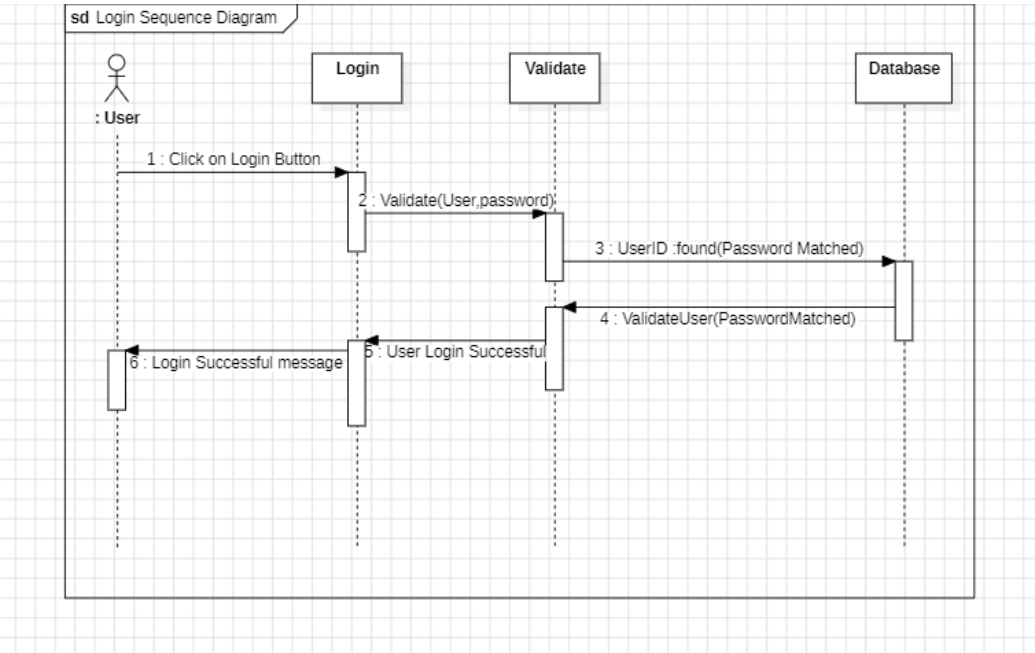


## 3.4 CLASS DIAGRAM

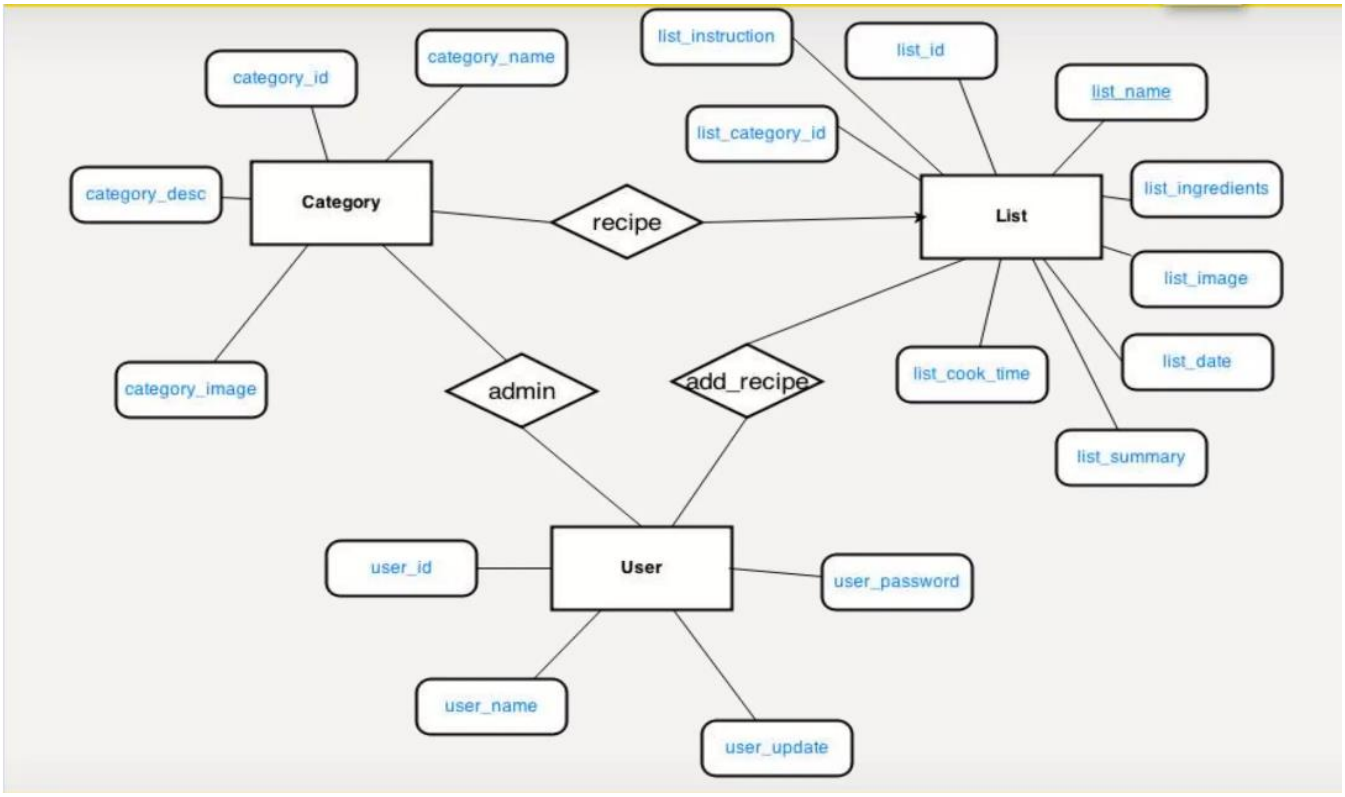


# 3.5 SEQUENCE DIAGRAM



## 3.6 ER DIAGRAM



## 4. PROGRAM CODE

## COOK_UP: A RECIPE STORAGE APPLICATION

### CLIENT

**CreateRecipe.jsx**

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

// Component to create a new recipe
const CreateRecipe = () => {
    // State to store the recipe data
   const [recipe, setRecipe] = useState({
      name: "",
      description:"",
      ingredients:"",
      imageUrl:"",
      userId: window.localStorage.getItem("id")//store user ID in local storage
   })

   // const [file,setFile]=useState(null);

   // Hook from React Router for navigation
   const navigate = useNavigate();

    // Function to handle changes in input fields
   const handleChange = (event) =>{
```

```
    const {name, value} = event.target
    setRecipe({...recipe,[name]:value})
}

// const handleUpload = (e) =>{
//     const formdata = new FormData()
//     formdata.append('file',file)
//     axios.post('http:localhost:3001/upload',formdata)
//     .then(res=>console.log(res))
//     .catch(err=>console.log(err))
// }


// Function to handle form submission
const handleSubmit = (event) =>{
    event.preventDefault()
    // API call to create a new recipe
    axios.post('http://localhost:3001/recipe/create-recipe',recipe)
    .then(result =>{
        // Navigate to home page after successful recipe creation
        navigate('/')
        console.log(result.data)
        alert("recipe created")
    })
    .catch(err => {
        alert(err.response.data.message)
    });
}
```

```
 return (
  <>
    <div className='d-flex justify-content-center align-items-center vh-100'>
     <div className='p-3 border border-3 border-dark w-50'>
        <h3 className='text-center mb-4'>Create Recipe</h3>
        <form onSubmit={handleSubmit}>
          <div className="mb-3">
            <label htmlFor='name' className="form-label">Name</label>
            <input type='text' placeholder='Enter Name' className='form-control' name='name' onChange={handleChange} />
          </div>

          <div className="mb-3">
            <label htmlFor='description' className="form-label">Description</label>
            <input type='text' placeholder='Enter Description' className='form-control' name='description' onChange={handleChange} />
          </div>

          <div className="mb-3">
            <label htmlFor='ingredients' className="form-label">Ingredients</label>
            <input type='text' placeholder='Enter Ingredients' className='form-control' name='ingredients' onChange={handleChange} />
          </div>

          <div className="mb-3">
            <label htmlFor='imageUrl' className="form-label">Image URL</label>
            <input type='text' placeholder='Enter URL' className='form-control' name='imageUrl' onChange={handleChange} />
          </div>
```

```
            {/* <div>
               <input type="file" onChange={e => setFile(e.target.files[0])}/>
               <button onClick={handleUpload}>Upload</button>
            </div> */}



            <button className='mt-1 btn btn-dark w-100 mt-2 mb-3'>Submit</button>
          </form>
        </div>
        </div>
    </>
  )
}

export default CreateRecipe
```

**Home.jsx**

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import { CgProfile } from "react-icons/cg";

// Component to display a list of recipes
const Home = () => {
  // State to store the fetched recipes
  const [recipes,setRecipes] = useState([])
```

```
const [usernames, setUsernames] = useState({});
// Fetch recipes from the API when the component mounts
useEffect(()=>{
  axios.get('http://localhost:3001/recipe/recipes')
  .then(response => {
    setRecipes(response.data); // Set the fetched recipes in the state
    response.data.forEach(recipe => {
      findName(recipe.userId);
    });
  })
  .catch(err => console.log(err))
},[])

const findName = (id)=>{

    axios.get(`http://localhost:3001/auth/find-username/${id}`)
    .then(result =>{
      setUsernames(prevState => ({
        ...prevState,
        [id]: result.data.username // Store the username in the state
      }));
    })
    .catch(err => console.log(err));
}

return (
  <div className='section'>
```

```
    <div className='container'>
      <h2 className='text-center my-4'>Recipes</h2>
      <div className='row'>
        {
          recipes.map(recipe=>(
            <div className='col-lg-4 col-md-4 col-12' key={recipe._id}>
              <div className="d-flex justify-content-center">
                <div className="card mb-4" style={{ width: '18rem',height: '350px' }}
id="recipe">
                  <img src={recipe.imageUrl} className="card-img-top recipe-image"/>
                  <div className="card-body text-center" >
                    <h5 className="card-title"><strong>{recipe.name}</strong></h5>
                    <p className="card-text p-2" style={{ height: '30px', overflowY:
'hidden',fontSize:'12px'}}>Created by {usernames[recipe.userId]}</p>
                    <Link to={`/read-recipe/${recipe._id}`} className='text-decoration-
none'>
                      <button className='btn btn-dark'>View Recipe</button>
                    </Link>
                  </div>
                </div>
              </div>
            </div>
          ))
        }
      </div>
    </div>
  </div>
 )
}
```

export default Home

**Home.css**

```
#nav-bar{
  position: fixed;
  width: 100%;
  top:0;
  left: 0;
  z-index: 100;
  padding: .5rem 5rem;
}

.recipe-image {
    width: 300px;
    height: 300px;
  }

.section{
  width: 100%;
  min-height: 100vh;
  display: flex;
  margin-top: 80px;
  justify-content: center;
}

#recipe img{
```

```
  border:none;
  height: 200px;
  width: 100%;
  object-fit: cover;
}

#myrecipe img{
  border:none;
  height: 200px;
  width: 100%;
  object-fit: cover;
}


#recipe h5{
  font-size: 20px;
  letter-spacing: 2px;
  font-weight: 700;
  color: black;
  text-transform: uppercase;
}

#read-recipe-id{
  padding: 10px 50px;
}
```

```
#read-recipe-id h1{
  font-weight: 800;
  font-size: 35px;
  text-transform: uppercase;
  color: black
}

#read-recipe-id img {
  width: 300px  ;
  height: 300px ;
  object-fit: cover;
  object-position: center;
}
```

**Login.jsx**

```
import React, { useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import axios from 'axios';
import { Link,useNavigate } from 'react-router-dom';

// Component for user login
const Login = () => {
    // State variables to store username and password
    const [username, setUsername] = useState('');
    const[password,setPassword] = useState('');
    const navigate = useNavigate()
```

```
  axios.defaults.withCredentials = true;

  // Function to handle form submission
  const handleSubmit = (e) =>{
     e.preventDefault()

     // Sending login request to the server
     axios.post('http://localhost:3001/auth/login',{username,password})
     .then(res=>{
        console.log(res.data);
        if(typeof res.data.id !== 'undefined'){
           // If login is successful, store user ID in local storage
           window.localStorage.setItem("id",res.data.id)
        }
        else{
           // If login fails, display error message
           alert(res.data.message);
        }
        // Navigate to home page
        navigate('/')
        console.log(res)
     })
     .catch(err=>console.log(err))
  }


return (
```

```
    <div className='d-flex justify-content-center align-items-center vh-100'>
      <div className='p-5 border border-3 w-auto '>
        <h3 className='text-center fw-bold '>LOGIN</h3>
        <form onSubmit={handleSubmit}>
          <div className='mb-3'>
            <label htmlFor='username'>Username</label>
            <input type='text'placeholder='Enter Username' className='form-control'
              onChange={(e)=>setUsername(e.target.value)}
            />
          </div>
          <div className='mb-3'>
            <label htmlFor='username'>Password</label>
            <input type='password' placeholder='Enter Password' className='form-
control'
              onChange={(e)=>setPassword(e.target.value)}
            />
          </div>
          <button className='mt-1 btn btn-dark w-100'>Login</button>
          <Link to="/auth/register"><button className='btn btn-light w-100 mt-2
border'>Register</button></Link>
        </form>
      </div>
    </div>
  )
}

export default Login
```

**MyRecipes.jsx**

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import { MdDelete } from 'react-icons/md';

const MyRecipes = () => {
  const [myrecipes, setMyRecipes] = useState([]);
  const userId = window.localStorage.getItem('id');

  useEffect(() => {
    axios.get(`http://localhost:3001/recipe/myrecipes/${userId}`)
      .then(response => {
        setMyRecipes(response.data); // Set the fetched recipes in the state
        console.log('my recipes fetched');
      })
      .catch(err => console.log(err));
  }, []);

  const deleteRecipe = (recipeId) => {
    if (window.confirm('Are you sure you want to delete this recipe?')) {
      axios.delete(`http://localhost:3001/recipe/deletemyrecipe/${userId}/${recipeId}`)
        .then(response => {
          // Filter out the deleted recipe from the state
          setMyRecipes(myrecipes.filter(recipe => recipe._id !== recipeId));
        })
        .catch(err => console.log(err));
    }
```

```
  };

  return (
    <div className="container" style={{ marginTop: '80px' }}>
      <h2 className="text-center my-4">My Recipes</h2>
      {myrecipes.length === 0 ? (
        <div className="text-center">
          <h3 className="mt-5">You haven't posted any recipes yet!</h3>
          <Link to="/recipe/create-recipe"><button className="btn btn-dark mt-4">
Create Recipe</button></Link>
        </div>
      ) : (
        <div className="row">
          {myrecipes.map(recipe => (
            <div className="col-lg-4 col-md-6 col-sm-12" key={recipe._id}>
            <div className="d-flex justify-content-center">
              <div className="card mb-4" style={{ width: '250px', height: '350px' }}
id="myrecipe">
                <img src={recipe.imageUrl} className="card-img-top" alt="Recipe" />
                <div className="card-body text-center">
                  <h5 className="card-title">
                    <Link to={`/read-recipe/${recipe._id}`} className="text-black text-
decoration-none">
                      <h3>{recipe.name} </h3>
                    </Link>
                  </h5>
                  <button href="#" className="btn btn-dark px-3 m-2"
                    onClick={() => deleteRecipe(recipe._id)}
                  >
```

```
                Delete <MdDelete style={{ fontSize: '20px', cursor: 'pointer', textAlign:
'center' }} />
                </button>
              </div>
            </div>
            </div>
          </div>
        ))}
      </div>
    )}
  </div>
 );
};

export default MyRecipes;
```

**Nav.jsx**

```
import React, { useEffect, useState } from 'react';
import 'bootstrap/dist/js/bootstrap.bundle.min.js';
import { Link, useNavigate } from 'react-router-dom';
import axios from 'axios';

// Navigation Bar component
const Nav = () => {
  const navigate = useNavigate();
  const userId = window.localStorage.getItem("id");
  const isLoggedIn = !!userId; // Check if user is logged in
  const [username,setUsername] = useState("")
  const [showModal, setShowModal] = useState(false);
```

```
    // !!: The double exclamation marks !! are used to convert any value into its boolean equivalent. It's a common JavaScript idiom used to ensure that a value is strictly converted to either true or false.
    // console.log(userId);
    // console.log(isLoggedIn)


    // Function to handle logout
    const handleLogout = () =>{
        if (window.confirm("Are you sure you want to logout?")){window.localStorage.clear(); // Clear user ID from local storage


        // Send logout request to the server
        axios.get('http://localhost:3001/auth/logout')
        .then(result => {
            navigate('/'); // Redirect to home page
            // window.location.reload();
        })
        .catch(err => console.log(err))}
    }


    useEffect(()=>{
        if(isLoggedIn){findName(userId)}
    })


    const findName = (id)=>{


        axios.get(`http://localhost:3001/auth/find-username/${id}`)
        .then(result =>{
          setUsername(result.data.username);
```

```
    })
    .catch(err => console.log(err));
  }

return (
  <div>
    <nav className="navbar navbar-expand-lg navbar-dark bg-dark" id="nav-bar">
      <div className='container-fluid'>

        <Link className='navbar-brand' to="/">
          CookUp
        </Link>
        <button
          className="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarTogglerDemo01"
          aria-controls="navbarTogglerDemo01"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
        <span className="navbar-toggler-icon"></span>
        </button>
        <div className='collapse navbar-collapse' id="navbarTogglerDemo01">
          <ul className='navbar-nav ms-2 me-auto mb-2 mb-lg-0'>
          {isLoggedIn && (
                <>
                  <li className='nav-item'>
```

```
                        <Link className='nav-link text-white' to="/recipe/create-recipe" aria-current="page">
                            Create Recipe
                        </Link>
                    </li>
                    <li className='nav-item'>
                        <Link className='nav-link text-white' to="/myrecipes">
                            My Recipes
                        </Link>
                    </li>
                    <li className='nav-item'>
                        <Link className='nav-link text-white' to="/recipe/saved-recipe">
                            Saved Recipes
                        </Link>
                    </li>
                </>
            )}
        </ul>

        {
            isLoggedIn?
            <>
                {/* <h6 className='text-white mx-2 my-2 d-md-inline d-sm-none'>@{username}</h6>
                <button className='btn btn-outline-light'
                    onClick={handleLogout}
                >
                Logout
```

```
                    </button> */}
                    <li className="nav-item dropdown d-flex align-items-center">
                        <a className="nav-link dropdown-toggle text-white my-1" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
                            @{username}
                        </a>
                        <ul className="dropdown-menu ">
                            <li><button className="dropdown-item" onClick={handleLogout}

                            >Logout</button></li>

                        </ul>
                    </li>
                </>
                :
                <button className='btn btn-dark'>
                    <Link to="/auth/login" className='text-decoration-none text-white'>Login/Register</Link>
                </button>
            }
            </div>
        </div>
    </nav>
  </div>
 )
}

export default Nav
```

**ReadRecipe.jsx**

```
import React, { useEffect, useState } from 'react'
import { useParams } from 'react-router-dom';
import axios from 'axios';
import "../components/Home.css"
import { MdDelete } from "react-icons/md";
import Nav from './Nav';

const ReadRecipe = () => {
    // Get the recipe ID from the URL parameters
    const {id} = useParams();
    // Get the logged-in user ID from local storage
    const userId = window.localStorage.getItem("id");
    // State variables to store the recipe details and saved recipes
    const [recipe,setRecipe] = useState([]);
    const [savedRecipes, setSavedRecipes] = useState([]);

    // Function to fetch the recipe details by ID
    useEffect(()=>{
        const getRecipe = () =>{
            axios.get('http://localhost:3001/recipe/recipe-by-id/'+id)
            .then(result => {
                setRecipe(result.data)
            })
            .catch(err => console.log(err))
        }
```

```
    // Function to fetch the saved recipes for the logged-in user
    const fetchSavedRecipes = () =>{
        axios.get('http://localhost:3001/recipe/saved-recipes/'+userId)
        .then(result => {
            // console.log(result);
            setSavedRecipes(result.data)
        })
        .catch(err => console.log(err))
    }

    // Fetch recipe details and saved recipes when the component mounts
    if(userId){
        fetchSavedRecipes()
    }
    getRecipe()
},[])

// Function to save a recipe
// const savedRecipe = (recipeId) => {
//     axios.put("http://localhost:3001/recipe", { userId, recipeId })
//     .then(result => {
//         setSavedRecipes(result.data.savedRecipes);
//     })
//     .catch(err => console.log(err));
// }



const toggleSavedRecipe = (recipeId) => {
```

```
    if (isRecipeSaved(recipeId)) {
        // If the recipe is already saved, remove it from saved recipes
        axios.delete(`http://localhost:3001/recipe/${userId}/${recipeId}`)
        .then(result => {
            setSavedRecipes(result.data.savedRecipes);
        })
        .catch(err => console.log(err));
    } else {
        // If the recipe is not saved, save it
        axios.put("http://localhost:3001/recipe", { userId, recipeId })
        .then(result => {
            setSavedRecipes(result.data.savedRecipes);
        })
        .catch(err => console.log(err));
    }
}



// Function to check if a recipe is saved by the user
const isRecipeSaved = (id) => {
    // console.log("savedRecipes:", savedRecipes);
    // console.log("recipeId:", id);
    // Check if savedRecipes contains the current recipe ID
    return savedRecipes && savedRecipes.includes(id);
}
```

```
  return (
   <>
   {/* <div className='d-flex justify-content-center container mt-5'>
      <div className='p-2'>
      <img  src={recipe.imageUrl} alt='' className='recipe-image'/>
      </div>
      <div className='p-2'>
      <h2>{recipe.name}</h2>
         {userId && <button className='btn btn-warning'
            onClick={() => toggleSavedRecipe(recipe._id)}
         >
            {isRecipeSaved(recipe._id) ? "Saved" : "Save"}
         </button>}

         <h3>Description</h3>
         <p>{recipe.description}</p>
         <h3>Ingredients</h3>
         <p>{recipe.ingredients}</p>

      </div>
   </div> */}
   <div className='section ' id="read-recipe-id">
      <div className='container-fuild'>
         <div className='row'>
            <div className='col-lg-6 col-md-6 col-12 text-center d-flex justify-content-
center align-items-center'>
               <img src={recipe.imageUrl} className='img-fluid'/>
            </div>
            <div className='col-lg-6 col-md-6 col-12 text-center '>
```

```
                <h1 className='mt-2'>{recipe.name}</h1>
                    {userId && <button className='btn btn-dark mb-2'
                                onClick={() => toggleSavedRecipe(recipe._id)}
                            >
                                {isRecipeSaved(recipe._id) ? "Saved" : "Save"}
                            </button>}
                <h3>Description</h3>
                <p>{recipe.description}</p>
                <h3>Ingredients</h3>
                <p>{recipe.ingredients}</p>
            </div>
        </div>
    </div>
   </div>
   </>
  )
}

export default ReadRecipe
```

**Registration.jsx**

```
import React, { useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import axios from 'axios';
import { Link,useNavigate } from 'react-router-dom';

// Registration component
```

```
const Registration = () => {
    // State variables to store username and password
    const [username, setUsername] = useState('');
    const[password,setPassword] = useState('');

    // Hook from React Router for navigation
    const navigate = useNavigate();

    // Function to handle form submission
    const handleSubmit = (e) =>{
        e.preventDefault()

         // Send registration request to the server
        axios.post('http://localhost:3001/auth/register',{username,password})
        .then(res=>{
            navigate('/auth/login') // Redirect to login page after successful registration
            console.log(res)
        })
        .catch(err=>console.log(err))
    }


  return (
    <div className='d-flex justify-content-center align-items-center vh-100'>
        <div className='p-5 border border-3 w-auto'>
            <h3 className='text-center fw-bold '>REGISTER</h3>
            <form onSubmit={handleSubmit}>
                <div className='mb-3'>
```

```
                <label htmlFor='username'>Username</label>
                <input type='text'placeholder='Enter Username' className='form-control'
                   onChange={(e)=>setUsername(e.target.value)}
                />
            </div>
            <div className='mb-3'>
                <label htmlFor='username'>Password</label>
                <input type='password' placeholder='Enter Password' className='form-
control'
                   onChange={(e)=>setPassword(e.target.value)}
                />
            </div>
            <button className='mt-1 btn btn-dark w-100'>Submit</button>
            <Link to="/auth/login"><button className='btn btn-light w-100 mt-2
border'>Login</button></Link>
          </form>
       </div>
    </div>
  )
}

export default Registration
```

**SavedRecipe.jsx**

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import "../components/Home.css"
import { Link } from 'react-router-dom';
```

```
// SavedRecipe component to display saved recipes
const SavedRecipe = () => {
  // State to store saved recipes
  const [savedrecipes,setSavedRecipes] = useState([])
  // Get the logged-in user ID from local storage
  const userId = window.localStorage.getItem("id");
  useEffect(()=>{

    // Fetch saved recipes when the component mounts
    axios.get('http://localhost:3001/recipe/user-recipes/'+userId)
    .then(response => {
      setSavedRecipes(response.data); // Set the fetched recipes in the state
    })
    .catch(err => console.log(err))
  },[])
  return (
    <>
    {/* <div className='d-flex justify-content-center'>
      <div>
        <h2>Saved Recipes</h2>
        {savedrecipes.length === 0 ? ( // Check if saved recipes array is empty
        <>
          <h3 className="text-center mt-5">You haven't saved any recipes!</h3>

        </>
        ) : (
          savedrecipes.map(recipe => (
```

```
          <div key={recipe._id} className='mt-4 p-3 border'>
            <Link to={`/read-recipe/${recipe._id}`} className='text-decoration-none'>
              <h3>{recipe.name}</h3>
            </Link>
            <img src={recipe.imageUrl} alt={recipe.name} className="recipe-image" />
          </div>
        ))
      )}
    </div>
  </div> */}
  <div className='section'>
    <div className='container'>
      <h2 className='text-center my-4'>Saved Recipes</h2>
      {savedrecipes.length === 0 ?(
        <>
          <h3 className="text-center mt-5">You haven't saved any recipes!</h3>
        </>
      ):(
        <div className='row'>
        {
          savedrecipes.map(recipe=>(
            <div className='col-lg-4 col-md-4 col-12' key={recipe._id}>
              <div className="d-flex justify-content-center">
                <div className="card mb-4" style={{ width: '18rem',height: '320px' }}
id="recipe">
                  <img src={recipe.imageUrl} className="card-img-top"/>
                  <div className="card-body text-center" >
                    <h5 className="card-title"><strong>{recipe.name}</strong></h5>
```

```
                    <Link to={`/read-recipe/${recipe._id}`} className='text-decoration-
none'>
                      <button className='btn btn-dark'>View Recipe</button>
                    </Link>
                  </div>
                </div>
              </div>
            </div>
          ))
        }
      </div>
      )}

    </div>
  </div>
  </>
 )
}

export default SavedRecipe
```

**SERVER**

**Recipe.js**

```
const mongoose = require('mongoose')
const RecipeSchema = new mongoose.Schema({
   name: {
      type: String,
      required: true,
   },
```

```
    description: {
        type: String,
    },
    ingredients: {
        type: String,
    },
    imageUrl: {
        type: String,
    },
    image:{
        type: Object,
    },
    userId: {
        type: mongoose.Schema.Types.ObjectId,
        ref:"User",
        required: true
    }
})

const RecipeModel = mongoose.model("recipes", RecipeSchema)
module.exports = RecipeModel;
```

**User.js**

```
const mongoose = require('mongoose')
const UserSchema = new mongoose.Schema({
    username: {
        type: String,
        required: true,
```

```
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    savedRecipes : [
        {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Recipe"
        }
    ]
})
const UserModel = mongoose.model("users", UserSchema)
module.exports = UserModel;
```

**ROUTES**

**Auth.js**

```
const express = require('express')
const UserModel = require('../models/User')
const bcrypt = require('bcrypt'); //password encryption
const jwt = require('jsonwebtoken')
const cookieParser = require('cookie-parser');
const router = express.Router()
router.post('/register', async (req,res) =>{
    const {username, password} = req.body;
    const user = await UserModel.findOne({username})
    if(user){
```

```
        return res.json({message: 'user existed'})
    }
    const hashpassword = await bcrypt.hash(password,10)
    const newuser = new UserModel({username,password:hashpassword})
    newuser.save()
    return res.json({message:"record saved"})
})

router.post('/login', async (req,res) =>{
    const {username, password} = req.body;
    const user = await UserModel.findOne({username});
    if(!user){
        return res.json({message: 'wrong credentials'})
    }
    const validPassword = await bcrypt.compare(password,user.password);
    if(!validPassword){
        return res.json({message: 'wrong credentials'})
    }
    const token = jwt.sign({id:user._id},"secret")
    res.cookie("token",token)
    return res.json({message: "successfully logged in", id : user._id})

})

router.get('/logout', (req,res)=>{
    res.clearCookie("token")
    res.json({message:"Success"})
})
```

```
router.get('/find-username/:userId',async(req,res)=>{
    try{
        const id=req.params.userId;
        const user = await UserModel.findOne({_id:id})
        if (!user) {
            // If user not found, send 404 status with message
            return res.status(404).json({ message: 'User not found' });
        }
        // console.log(user.username)
        res.json(user);
    }
    catch(error){
        console.error(error);
        res.status(500).json({ message: 'Server Error' });
    }
})
module.exports = router;
```

**reciper.js**

```
const express = require('express')
const RecipeModel = require('../models/Recipe')
const UserModel = require('../models/User')

const router = express.Router()

router.post('/create-recipe',(req,res)=>{
    RecipeModel.create({
```

```
        name: req.body.name,
        description: req.body.description,
        ingredients: req.body.ingredients,
        imageUrl: req.body.imageUrl,
        userId: req.body.userId
    })
    .then(result=>{
        return res.json(result)
    })
    .catch(err=> {
        console.log(err)
        return res.status(500).json({message:'Give all details to create a recipe!'})
    });
})

router.get('/recipes',(req,res)=>{
    RecipeModel.find()
    .then(recipes =>{
        return res.json(recipes)
    })
    .catch(err => res.json(err));
})

router.get('/recipe-by-id/:id',(req,res)=>{
    const id = req.params.id;
    RecipeModel.findById({_id:id})
    .then(result =>{
        return res.json(result)
```

```
    })
    .catch(err => res.json(err));
})

router.get('/saved-recipes/:id',(req,res)=>{
    const id = req.params.id;
    UserModel.findById({_id:id})
    .then(result => {
        if (!result) {
            return res.status(404).json({ error: "User not found" });
        }
        // console.log(result);
        return res.json(result.savedRecipes);
    })
    .catch(err => res.status(500).json({ error: err.message }));
})

router.get('/user-recipes/:id',async (req,res)=>{
    const id = req.params.id;
    try {
        const user = await UserModel.findById({_id:id});
        const recipes = await RecipeModel.find({
            _id : {$in : user.savedRecipes}
        })
        res.status(201).json(recipes);
    }
    catch(err){
        res.status(500).json(err);
```

```
    }
})

router.get('/myrecipes/:id', async (req, res) => {
    const id = req.params.id;
    try {
        const recipes = await RecipeModel.find({
            userId: id
        });
        // console.log(recipes)
        res.status(200).json(recipes); // Sending retrieved recipes in the response
    } catch (err) {
        res.status(500).json(err); // Sending error response in case of an error
    }
});



// router.put('/',async (req,res) =>{
//     const recipe = await RecipeModel.findById({_id: req.body.recipeId})
//     const user = await UserModel.findById({_id: req.body.id})
//     // user.savedRecipes.push(recipe)
//     if (user) {
//         if (!user.savedRecipes) {
//             user.savedRecipes = []; // Initialize savedRecipes array if it doesn't exist
//         }
//         user.savedRecipes.push(recipe);
//     } else {
//         console.log("User not found.");
```

```
//     }
//     user.save()
//     return res.json({savedRecipes : user.savedRecipes})
// })

router.put('/', async (req, res) => {
    try {
        const recipe = await RecipeModel.findById(req.body.recipeId);
        if (!recipe) {
            return res.status(404).json({ error: "Recipe not found" });
        }

        const user = await UserModel.findById(req.body.userId);
        if (!user) {
            return res.status(404).json({ error: "User not found" });
        }

        if (!user.savedRecipes) {
            user.savedRecipes = [];
        }
        user.savedRecipes.push(recipe);
        await user.save();

        return res.json({ savedRecipes: user.savedRecipes });
    } catch (err) {
        console.error(err);
        return res.status(500).json({ error: "Internal server error" });
    }
```

```
});

router.delete('/:userId/:recipeId', async (req, res) => {
    try {
        // console.log('api reached')
        const recipeId = req.params.recipeId;
        const userId = req.params.userId;
        // console.log(recipeId,userId)
        const user = await UserModel.findById(userId);
        if (!user) {
            return res.status(404).json({ error: "User not found" });
        }

        if (!user.savedRecipes || !user.savedRecipes.includes(recipeId)) {
            return res.status(404).json({ error: "Recipe not found in user's saved recipes" });
        }

        user.savedRecipes = user.savedRecipes.filter(savedRecipeId =>
savedRecipeId.toString() !== recipeId);
        await user.save();

        return res.json({ savedRecipes: user.savedRecipes });
    } catch (err) {
        console.error(err);
        return res.status(500).json({ error: "Internal server error" });
    }
});

router.delete('/deletemyrecipe/:userId/:recipeId', async (req, res) => {
```

```
    try {
        const userId = req.params.userId;
        const recipeId = req.params.recipeId;

        // Find the recipe by ID and the user ID
        const recipe = await RecipeModel.findOne({ _id: recipeId, userId: userId });

        if (!recipe) {
            return res.status(404).json({ error: "Recipe not found or not owned by the user"
});
        }

        // Remove the recipe from the database
        await recipe.deleteOne();

        return res.status(200).json({ message: "Recipe deleted successfully" });
    } catch (err) {
        console.error(err);
        return res.status(500).json({ error: "Internal server error" });
    }
});


module.exports = router;
```

**index.js**

```
const express = require("express")
const mongoose = require('mongoose')
const cors = require("cors")
```

```
const userRouter = require('./routes/auth');
const cookieParser = require('cookie-parser');
const recipeRouter = require('./routes/reciper')
const multer = require('multer')

const app = express()
app.use(express.json())
app.use(cors({
    origin:["http://localhost:5173"],
    methods:["GET","POST","PUT","DELETE"],
    credentials: true
}))
app.use(cookieParser())
app.use('/auth',userRouter)
app.use('/recipe',recipeRouter)

mongoose.connect('mongodb://127.0.0.1:27017/RecipeApp-MERN');



// const upload = multer({
//     storage:storage
// })

// app.post('/upload',upload.single('file'),(req,res)=>{
//     console.log(req.file)
// })

app.listen(3001,()=>{
```

```
    console.log("server is running!");
})
```

**package.json**

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.1",
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "express": "^4.19.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.3.1",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^3.1.0"
  }
}
```

# 5. RESULTS AND SNAPSHOTS

CookUp Login/Register

**LOGIN**

Username

Enter Username

Password

Enter Password

Login

Register

CookUp Login/Register

**REGISTER**

Username

Enter Username

Password

Enter Password

Submit

Login

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

Recipes

**ICECREAM**

Created by harish

View Recipe

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

## Create Recipe

Name

Pizza

Description

Our gourmet pizza features a crispy, thin crust topped with a rich tomato sauce, melted r

Ingredients

Crust, tomato sauce, mozzarella, pepperoni, mushrooms, bell peppers, olives, herbs.

Image URL

https://www.foodandwine.com/thmb/Wd4lBRZz3X_8qBr69UOu2m7I2iw=/1500x0/filters:n

Submit

Lighten Color - mdi... All Bookmarks

CookUp Create Recipe My Recipes Saved Recipe @harshiniakshaya

localhost:5173 says

recipe created

OK

Create Recipe

Name

Pizza

Description

Our gourmet pizza features a crispy, thin crust topped with a rich tomato sauce, melted r

Ingredients

Crust, tomato sauce, mozzarella, pepperoni, mushrooms, bell peppers, olives, herbs.

Image URL

https://www.foodandwine.com/thmb/Wd4lBRZz3X_8qBr69UOu2m7I2iw=/1500x0/filters:n

Submit

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

## Recipes

**ICECREAM**

Created by harish

View Recipe

**PIZZA**

Created by harshiniakshaya

View Recipe

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

# ICECREAM

Saved

## Description

This homemade vanilla ice cream recipe is creamy, rich, and bursting with vanilla flavor. It's the perfect treat for hot summer days or any time you're craving something sweet and refreshing.

## Ingredients

2 cups heavy cream 1 cup whole milk 3/4 cup granulated sugar 1 tablespoon pure vanilla extract Pinch of salt

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

# My Recipes

Pizza

Delete

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

# Saved Recipes

ICECREAM

View Recipe

Lighten Color - mdi...

All Bookmarks

CookUp Create Recipe My Recipes Saved Recipe

@harshiniakshaya

localhost:5173 says

Are you sure you want to delete this recipe?

OK Cancel

Pizza

Delete

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

## My Recipes

You haven't posted any recipes yet!

Create Recipe

CookUp Create Recipe My Recipes Saved Recipes @harshiniakshaya

## Recipes

**ICECREAM**

Created by harish

View Recipe

# 6. CONCLUSION

## CONCLUSION:

This program has been created successfully to create a recipe plans storage system/ application called COOK_UP. The results and snapshots of the program for COOK_UP are attached in this document.

## 7. REFERENCES

The below websites helped us in gaining more knowledge on the subject and in completing the project


- https://stackoverflow.com
- https://tutorialspoint.com
- https://youtube.com