# RECIPE APP

This site aims to provide a comprehensive platform for sharing and discovering various recipes. Users can register on the site to submit their own recipes and comment on others. Additionally, the site offers detailed information about different cuisines, ingredients, and cooking methods. Users can explore a wide range of recipes, from appetizers to desserts, and find inspiration for their next meal. The site also features cooking tips and video tutorials to enhance the culinary experience. Enjoy browsing through the diverse collection of recipes and make your cooking journey more delightful.

A screen shot of a computer code

Description automatically generated

The login function is an asynchronous function designed to handle user login requests. It takes req (the request object) and res (the response object) as parameters.

Functionality:

Extracting Data: The function extracts username, password, and email from the req.body. This assumes that the incoming request contains a JSON body with these fields.

Error Handling: The function is wrapped in a try block to handle any potential errors that may occur during execution.

Username-Based Login:

If a username is provided, the function attempts to find a user in the database with the given username using User.findOne({ username }).

If no user is found (!oldUser), it responds with a message indicating that the user doesn't exist.

If a user is found, it compares the provided password with the stored password using bcrypt.compare.

If the password is incorrect (!isPasswordCorrect), it responds with a message indicating invalid credentials.

If the password is correct, it responds with a message indicating a successful login and includes the user data (result: oldUser).

Email-Based Login:

If an email is provided (and username is not provided), the function performs similar steps to the username-based login, but it searches for a user by email instead.

The same checks for user existence and password correctness are performed, and appropriate responses are sent.

Missing Credentials:

If neither username nor email is provided, the function responds with a message asking the user to provide either a username or an email.

Catch Block:

If any error occurs during the process, the catch block captures it and responds with a message indicating that something went wrong.

Summary:

The login function handles user authentication by checking either the username or email against stored user records. It verifies the password using bcrypt and returns appropriate responses based on the validity of the credentials provided.

A screen shot of a computer code

Description automatically generated

The register function is an asynchronous function designed to handle user registration requests. It takes req (the request object) and res (the response object) as parameters.

Functionality:

Extracting Data: The function extracts email, password, name, surname, username, and gender from the req.body. This assumes that the incoming request contains a JSON body with these fields.

Error Handling: The function is wrapped in a try block to handle any potential errors that may occur during execution.

Check for Existing User:

The function searches the database for an existing user with the provided email using User.findOne({ email }).

If a user with the same email already exists (oldUser), it responds with a message indicating that the user already exists and stops further execution.

Hashing the Password:

If the email is not already registered, the function hashes the provided password using bcrypt.hash(password, 12). The number 12 indicates the salt rounds, making the hashing process secure.

Creating a New User:

With the hashed password, the function creates a new user record in the database using User.create. It includes the provided email, hashed password, name, surname, username, and gender in the new user record.

If the user is successfully created, it responds with a message indicating successful registration and includes the newly created user data (result).

Catch Block:

If any error occurs during the process, the catch block captures it and responds with a message indicating that something went wrong, along with the error details.

Summary:

The register function handles the registration of new users. It checks if the email is already registered, hashes the password, creates a new user record, and returns appropriate responses based on the success or failure of these operations.

A screen shot of a computer code

Description automatically generated

The getUser function is an asynchronous function that retrieves a user by their ID.

Extracting ID: It takes id from req.params

Finding the User:

It searches the database for a user with the given ID using User.findById(id).

If the user is not found, it responds with a 404 status and a "User not found" message.

Successful Retrieval:

If the user is found, it responds with a 200 status and the user data.

Error Handling:

Any errors during the process are caught and result in a 500 status and an error message.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

getRecipes

Functionality: Retrieves all recipes.

Details:

Fetches all recipes from the Recipe model.

For each recipe, fetches the corresponding author from the User model.

Combines recipe data with the author data.

Responds with the combined data or an error message if something goes wrong.

getRecipe

Functionality: Retrieves a specific recipe by its ID.

Details:

Fetches the recipe with the specified ID from the Recipe model.

If the recipe is found, fetches the corresponding author from the User model.

Combines the recipe data with the author data.

Responds with the combined data or an error message if something goes wrong or if the recipe is not found.

createRecipe

Functionality: Creates a new recipe.

Details:

Extracts recipe details from the request body.

Creates and saves a new recipe in the Recipe model.

Responds with the created recipe data or an error message if something goes wrong.

deleteRecipe

Functionality: Deletes a specific recipe by its ID.

Details:

Fetches the recipe with the specified ID from the Recipe model.

Checks if the requesting user is the author of the recipe.

Deletes the recipe if the user is authorized.

Responds with a success message or an error message if something goes wrong, if the recipe is not found, or if the user is not authorized.

updateRecipe

Functionality: Updates a specific recipe by its ID.

Details:

Extracts updated recipe details from the request body.

Fetches the recipe with the specified ID from the Recipe model.

Checks if the requesting user is the author of the recipe.

Updates the recipe if the user is authorized.

Responds with the updated recipe data or an error message if something goes wrong, if the recipe is not found, or if the user is not authorized.

FRONTEND:

In the frontend, I used Next.js to build the user interface and Axios to handle HTTP requests. Next.js provides a robust framework for server-side rendering and static site generation, ensuring a fast and optimized user experience. Axios simplifies the process of making API calls, allowing seamless interaction with the backend to fetch and submit data. Together, these technologies enable the creation of a dynamic and responsive recipe site.

Berat Doğan 20202022014