Recipe Finder

In this Recipe Finder Application the is a web-based app that allows users to search for recipes using an autocomplete feature. Users can enter at least three characters to fetch matching recipes, view a dropdown list, and navigate to a detailed recipe page.

The Technologies used here are

Frontend: React.js (for UI rendering and interactivity), CSS (for styling) React Router (for navigation)

Backend: Spring Boot (for REST API), H2 Database (for storing recipe data), Java (for backend logic)

This is the Project structure of the Backend part

PROJECT\SRC\MAIN\JAVA\COM\EXAMPLE\PROJECT_RECIPES

```
|   AppConfig.java
|   ProjectRecipesApplication.java
|
+---config
|       WebConfig.java
|
+---controller
|       RecipeController.java
|
+---entity
|       Recipe.java
|       RecipeResponse.java
|
+---exception
|       GlobalExceptionHandler.java
|
+---repository
|       RecipeRepository.java
|
\---service------RecipeService.java
```

Entity Layer → Represents a table in the database and uses JPA for ORM (Object-Relational Mapping) to map Java objects to database records.

Repository Layer → Handles database operations (CRUD: Create, Read, Update, Delete) and extends JpaRepository, interacting with the database using Spring Data JPA.

Service Layer → Contains business logic, calls the Repository Layer to store/retrieve data, and acts as a bridge between Controller and Repository.

Controller Layer → Exposes APIs like (@GetMapping, @PostMapping, @PutMapping, @DeleteMapping) and calls the Service Layer to process requests.

Exception Handling → Handles errors gracefully and prevents exposing raw exceptions to the client.

Configuration Layer → Stores custom application settings and is used for defining Beans, CORS policies, and Security settings.

This is the Project structure of the Frontend part

Project\src\main\resources\static\Recipe-Frontend\src

```
+---src
| -  App.jsx
|
| -  Index.css
|
| -  Main.jsx
|
| -  RecipeController.java
|
| -  RecipeDetails.jsx
|
| -  SearchBar.jsx
```

App.jsx → Main Application Component Acts as the root component of the React app and renders the SearchBar component and a grid of recipe images and uses react-router-dom for navigation.

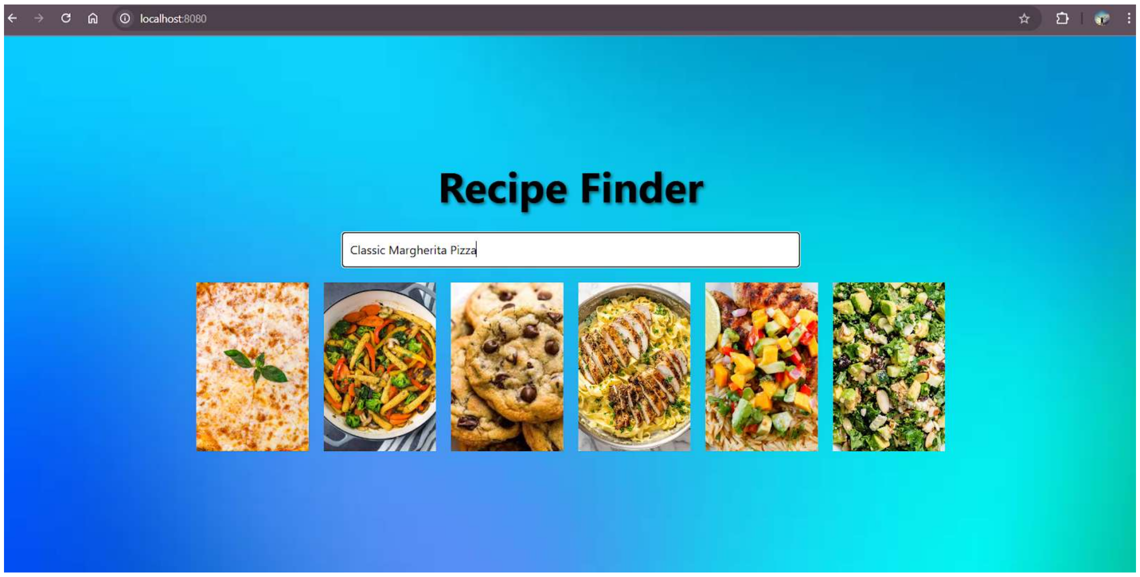
main.jsx → Application Entry Point Initializes and renders the React application configures react-router-dom for handling navigation and mounts the App.jsx component to the HTML root element.

RecipeDetails.jsx → Recipe Details Page Fetches recipe details from the backend using useEffect() and displays recipe information like name, image, cuisine, ingredients, and instructions uses useParams() from react-router-dom to extract the recipe ID from the URL.

SearchBar.jsx → Autocomplete Search Component Contains an input field for searching recipes Implements a autocomplete feature calls the backend API when at least 3 characters are entered displays matching recipes in a dropdown list on selection, navigate to the RecipeDetails.jsx page.

index.css → Styling Contains styles for UI components, including Layout and positioning, Search bar and dropdown menu and Recipe details page styles.

Here how the Frontend UI looks like

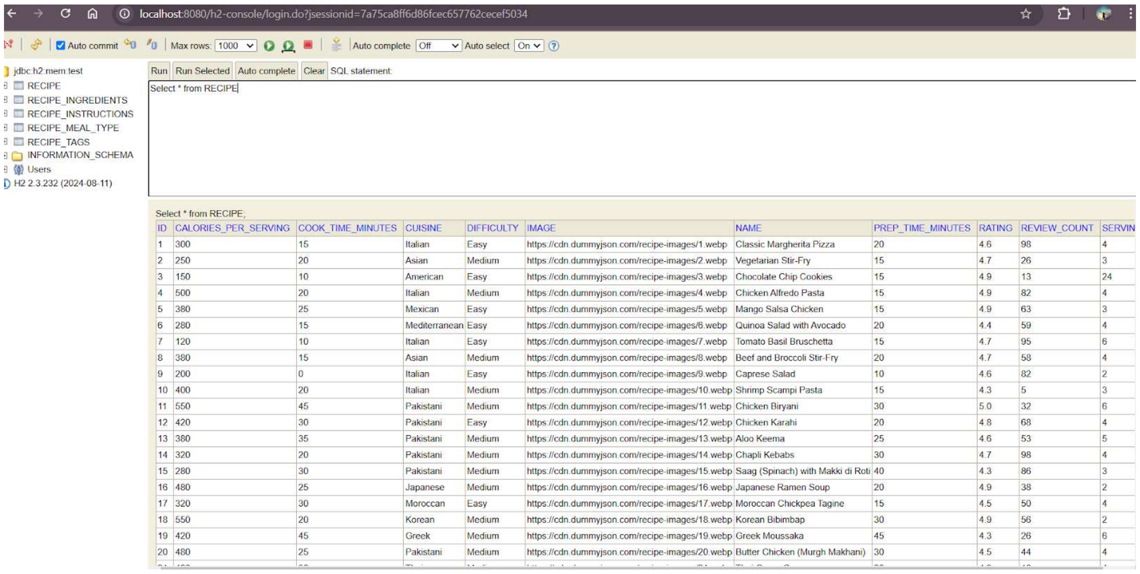


In the Backend

All the data has been stored in the H2 Database

Endpoints are tested using the Postman

This is the complete data stored in the H2 Database

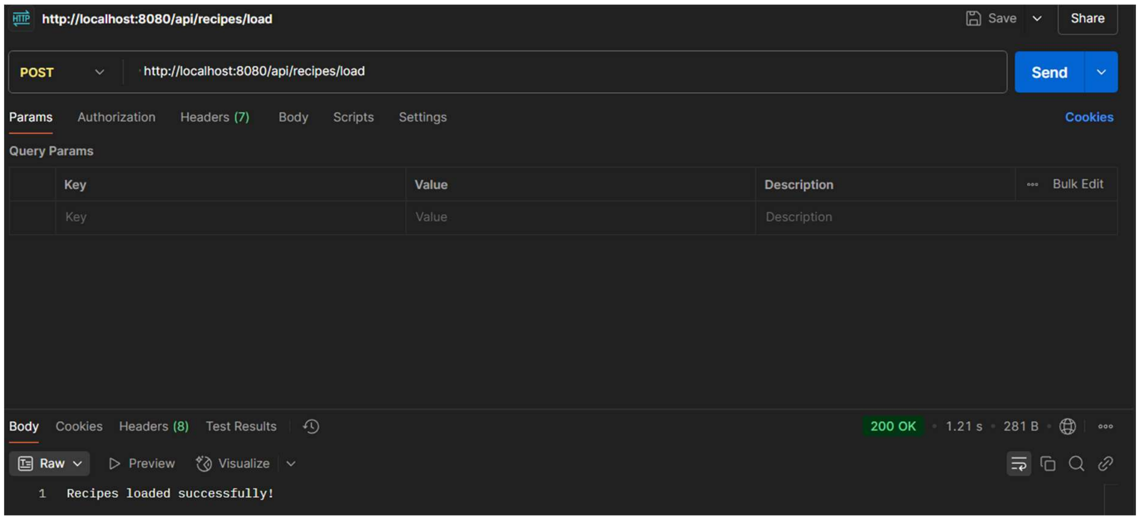


```
Select * from RECIPE
```

Select * from RECIPE;

ID	CALORIES_PER_SERVING	COOK_TIME_MINUTES	CUISINE	DIFFICULTY	IMAGE	NAME	PREP_TIME_MINUTES	RATING	REVIEW_COUNT	SERVIN
1	300	15	Italian	Easy	https://cdn.dummyjson.com/recipe-images/1.webp	Classic Margherita Pizza	20	4.6	98	4
2	250	20	Asian	Medium	https://cdn.dummyjson.com/recipe-images/2.webp	Vegetarian Stir-Fry	15	4.7	26	3
3	150	10	American	Easy	https://cdn.dummyjson.com/recipe-images/3.webp	Chocolate Chip Cookies	15	4.9	13	24
4	500	20	Italian	Medium	https://cdn.dummyjson.com/recipe-images/4.webp	Chicken Alfredo Pasta	15	4.9	82	4
5	380	25	Mexican	Easy	https://cdn.dummyjson.com/recipe-images/5.webp	Mango Salsa Chicken	15	4.9	63	3
6	280	15	Mediterranean	Easy	https://cdn.dummyjson.com/recipe-images/6.webp	Quinoa Salad with Avocado	20	4.4	59	4
7	120	10	Italian	Easy	https://cdn.dummyjson.com/recipe-images/7.webp	Tomato Basil Bruschetta	15	4.7	95	6
8	380	15	Asian	Medium	https://cdn.dummyjson.com/recipe-images/8.webp	Beef and Broccoli Stir-Fry	20	4.7	58	4
9	200	0	Italian	Easy	https://cdn.dummyjson.com/recipe-images/9.webp	Caprese Salad	10	4.6	82	2
10	400	20	Italian	Medium	https://cdn.dummyjson.com/recipe-images/10.webp	Shrimp Scampi Pasta	15	4.3	5	3
11	550	45	Pakistani	Medium	https://cdn.dummyjson.com/recipe-images/11.webp	Chicken Biryani	30	5.0	32	6
12	420	30	Pakistani	Easy	https://cdn.dummyjson.com/recipe-images/12.webp	Chicken Karahi	20	4.8	68	4
13	380	35	Pakistani	Medium	https://cdn.dummyjson.com/recipe-images/13.webp	Aloo Keema	25	4.6	53	5
14	320	20	Pakistani	Medium	https://cdn.dummyjson.com/recipe-images/14.webp	Chapli Kebabs	30	4.7	98	4
15	280	30	Pakistani	Medium	https://cdn.dummyjson.com/recipe-images/15.webp	Saag (Spinach) with Makki di Roti	40	4.3	86	3
16	480	25	Japanese	Medium	https://cdn.dummyjson.com/recipe-images/16.webp	Japanese Ramen Soup	20	4.9	38	2
17	320	30	Moroccan	Easy	https://cdn.dummyjson.com/recipe-images/17.webp	Moroccan Chickpea Tagine	15	4.5	50	4
18	550	20	Korean	Medium	https://cdn.dummyjson.com/recipe-images/18.webp	Korean Bibimbap	30	4.9	56	2
19	420	45	Greek	Medium	https://cdn.dummyjson.com/recipe-images/19.webp	Greek Moussaka	45	4.3	26	6
20	480	25	Pakistani	Medium	https://cdn.dummyjson.com/recipe-images/20.webp	Butter Chicken (Murgh Makhani)	30	4.5	44	4

Method: POST
Now when you load up the loadRecipes Endpoint: http://localhost:8080/api/recipes/load
Expected Response: You should receive a response with a 200 OK status and a message saying "Recipes loaded successfully!".



```
Recipes loaded successfully!
```

Method: GET
Test the getRecipeById Endpoint: http://localhost:8080/api/recipes/{id}
Expected Response: You should receive a response with a 200 OK status and the recipe details.

Method: POST
Test the createRecipe Endpoint: http://localhost:8080/api/recipes
Headers:

Key: Content-Type

Value: application/json

Body:

Json
{

"name": "Masala Dosa",

"ingredients": ["Rice Batter", "Sambar"],

"instructions": ["Make the pan hot and grease some oil on the pan", "Pour some amount of batter in circular form"],

"prepTimeMinutes": 15,

"cookTimeMinutes": 25,

"servings": 3,

"difficulty": "Medium",

"cuisine": "Indian",

"caloriesPerServing": 300,

"rating": 5.0,

"reviewCount": 15,

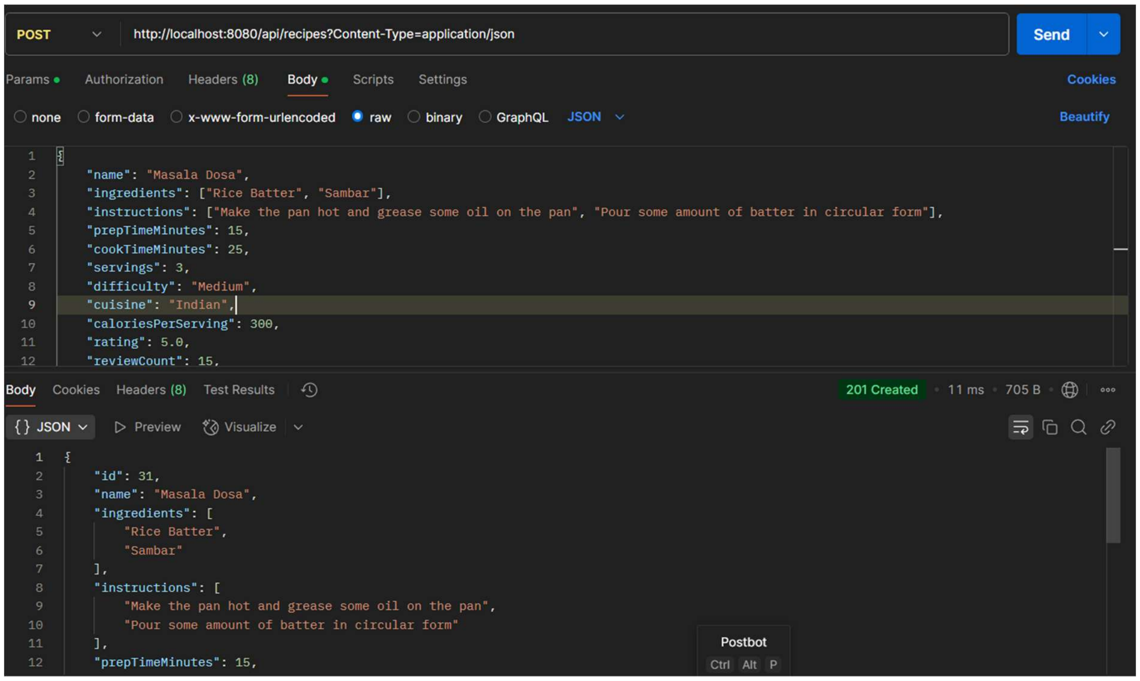
"image": "https://cdn.dummyjson.com/recipe-images/1.webp",

"tags": ["tag1", "tag2"],

"mealType": ["BreakFast", "Dinner"]

}

Expected Response: You should receive a response with a 201 Created status and the details of the newly created recipe.

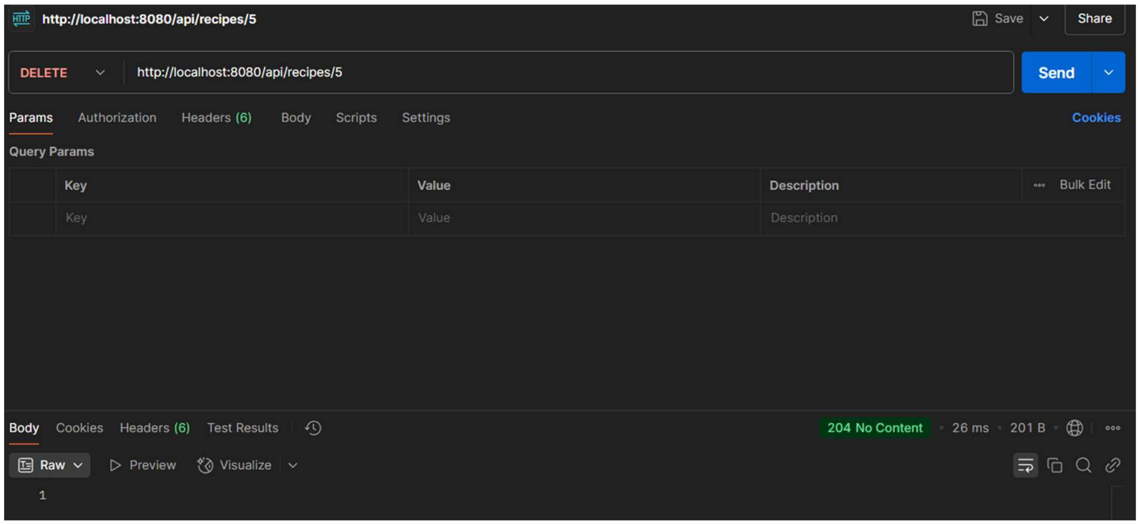

POST http://localhost:8080/api/recipes?Content-Type=application/json

```
{
    "name": "Masala Dosa",
    "ingredients": ["Rice Batter", "Sambar"],
    "instructions": ["Make the pan hot and grease some oil on the pan", "Pour some amount of batter in circular form"],
    "prepTimeMinutes": 15,
    "cookTimeMinutes": 25,
    "servings": 3,
    "difficulty": "Medium",
    "cuisine": "Indian",
    "caloriesPerServing": 300,
    "rating": 5.0,
    "reviewCount": 15,
```

201 Created • 11 ms • 705 B

```
{
    "id": 31,
    "name": "Masala Dosa",
    "ingredients": [
        "Rice Batter",
        "Sambar"
    ],
    "instructions": [
        "Make the pan hot and grease some oil on the pan",
        "Pour some amount of batter in circular form"
    ],
    "prepTimeMinutes": 15,
```

Method: DELETE
Test the deleteRecipe Endpoint: http://localhost:8080/api/{id}


DELETE http://localhost:8080/api/recipes/5

204 No Content • 26 ms • 201 B

How to Run the Application

Navigate to the frontend project folder and Install dependencies npm install

Start the React application: npm run dev

Backend Setup Clone the repository. Navigate to the backend project folder and run the backend application using:

mvn spring-boot:run

The backend will start at http://localhost:8080 and you will be able to see the Frontend UI.