## Advance Web Technology

**CRUD Operations**



**Project Supervisor: Sir Safiullah**

**Submitted By:**

Maryam Shehzadi 20006105016

BS-IT BATCH 6

## Session

## (2020-2024)

## University of Management and Technology

## Sialkot, Pakistan

**Question: 01**

**Develop a web application utilizing JavaScript and Bootstrap framework, integrated with a RESTful**

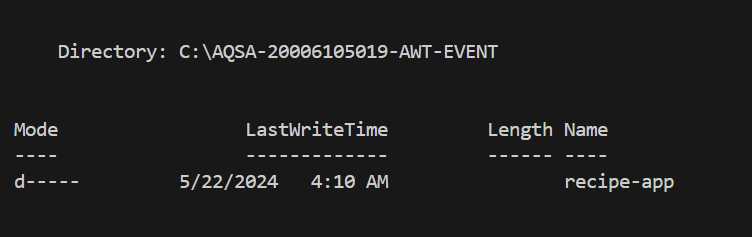
**API to facilitate CRUD (Create, Read, Update, Delete) operations. & quot?**

**ANS:**

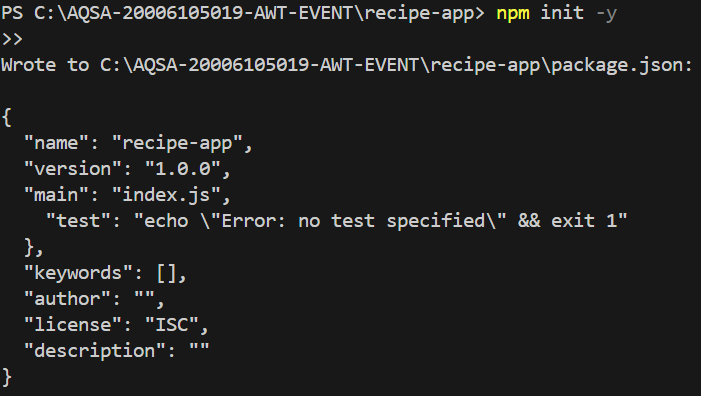
**Step 1: Set Up Your Project Environment**

* Install Node.js and npm
* Create a Project Directory: Use commands like **mkdir recipe-app** and cd recipe-app.
* Initialize Your Project: Run **npm init -y** to create a **package.json** file.
* Install json-server: Execute **npm install json-server** for creating a mock REST API.

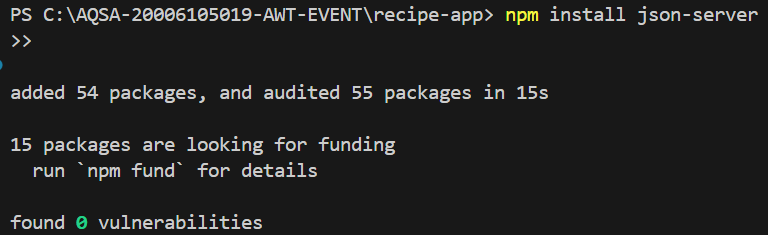
Create project directory

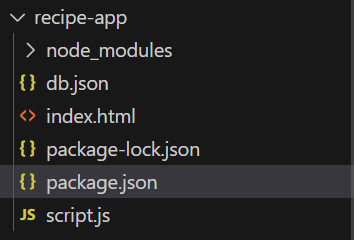
Change directory



Run **npm init -y** to create a **package.json** file

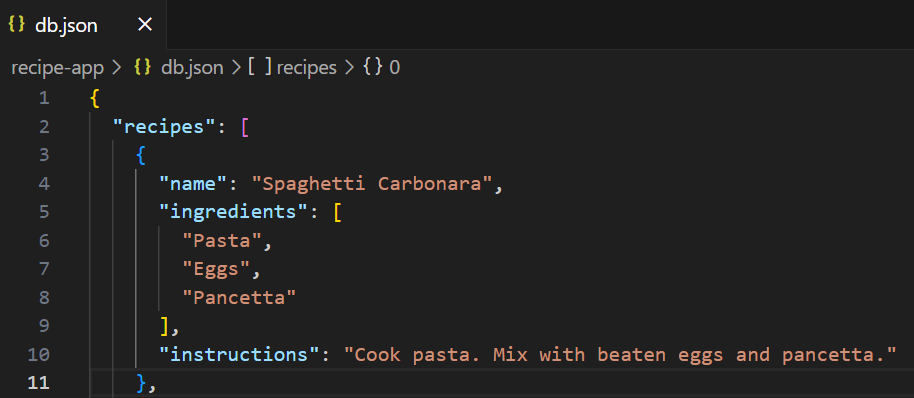


creating a mock REST API.

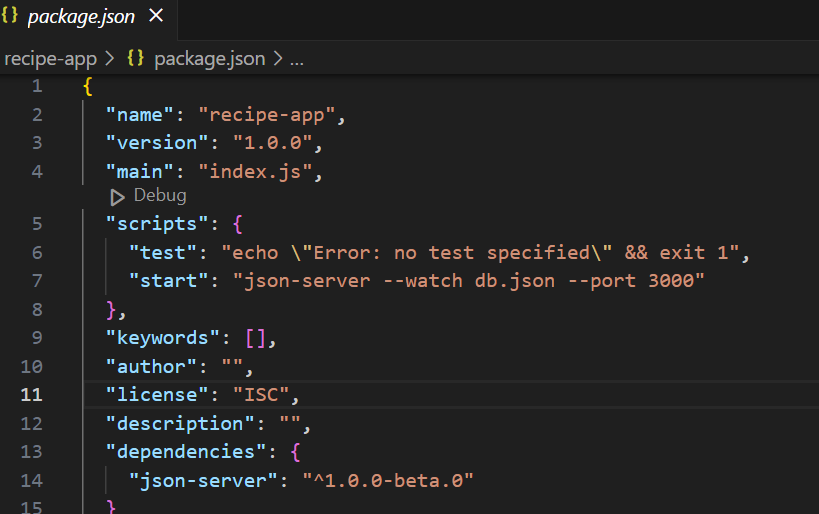


**Step 2: Create and Configure Your Mock Database**

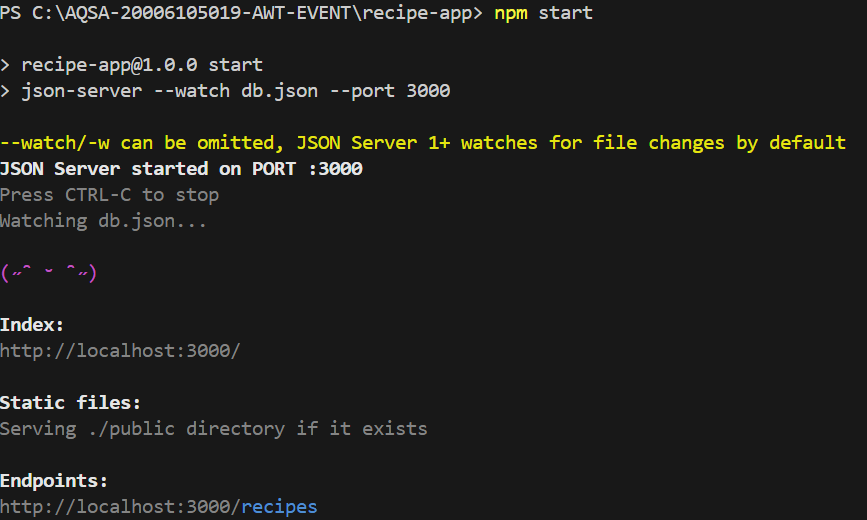
* Create a **db. json File**: This file will act as your mock database.
* Populate **db. json:** Add initial data such as sample recipes.
* Configure json-server: Add a script in **package. json** to run the server, e.g., **"start": "json**-**server --watch db. json --port 3000".**



Mock database, add initial data such as recipes.



Add a script in **package. json** to run the server



**Javascript File:**

**Step 3: Write Your JavaScript for CRUD Operations**

* Create a JavaScript File: Name it script.js.
* Fetch Data from the API: Write a function to GET data from your server and display it.
* Implement Create, Read, Update, and Delete Functions: Use fetch with POST, GET, PUT, and DELETE methods**.**
* Create Forms for Data Input: Make forms for submitting new data and updating existing data.
* Add Event Listeners: Set up listeners for form submissions and button clicks.
* Dynamically Update the UI: Ensure your UI updates when data is added, changed, or removed.

document.getElementById('recipe-form').addEventListener('submit', function(event) {

    event.preventDefault();

    const name = document.getElementById('name').value;

    const ingredients = document.getElementById('ingredients').value.split(',');

    const instructions = document.getElementById('instructions').value;

    const id = document.getElementById('recipe-id').value; // hidden input for id (used for updates)

    const url = id ? `http://localhost:3000/recipes/${id}` : 'http://localhost:3000/recipes';

    const method = id ? 'PUT' : 'POST';

    fetch('http://localhost:3000/recipes', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json',

        },

        body: JSON.stringify({ name, ingredients, instructions }),

    }).then(() => {

        document.getElementById('recipe-form').reset();

        fetchRecipes();

    });

});

function fetchRecipes() {

    fetch('http://localhost:3000/recipes')

        .then(response => response.json())

        .then(recipes => {

            const list = document.getElementById('recipes-list');

            list.innerHTML = '';

            recipes.forEach(recipe => {

                const item = document.createElement('li');

                item.textContent = `${recipe.name}: ${recipe.ingredients.join(', ')}`;

                const deleteButton = document.createElement('button');

                deleteButton.textContent = 'Delete';

                deleteButton.onclick = () => deleteRecipe(recipe.id);

                const updateButton = document.createElement('button');

                updateButton.textContent = 'Update';

                updateButton.onclick = () => loadRecipeForUpdate(recipe);

                item.appendChild(deleteButton);

                item.appendChild(updateButton);

                list.appendChild(item);

            });

        });

}

function deleteRecipe(id) {

    fetch(`http://localhost:3000/recipes/${id}`, {

        method: 'DELETE',

    }).then(() => fetchRecipes());

}

function loadRecipeForUpdate(recipe) {

    document.getElementById('name').value = recipe.name;

    document.getElementById('ingredients').value = recipe.ingredients.join(',');

    document.getElementById('instructions').value = recipe.instructions;

    document.getElementById('recipe-id').value = recipe.id; // assuming you have a hidden input for storing id

}

document.addEventListener('DOMContentLoaded', fetchRecipes);

**HTML File:**

* Create index.html file.
* Open Your Application: Load index.html in a web browser and test each CRUD operation.
* Debug: Check for issues and ensure all functionalities work as expected.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Recipe App</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            margin: 0;

            padding: 20px;

            color: #333;

        }

        h1 {

            color: #5A5A5A;

        }

        form {

            background: rgb(206, 135, 185);

            padding: 20px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0,0,0,0.1);

        }

        input[type="text"], textarea {

            width: 100%;

            padding: 8px;

            margin-top: 6px;

            margin-bottom: 16px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        button {

            background-color: #f0e8ef;

            color: rgb(227, 193, 193);

            border: none;

            padding: 10px 20px;

            text-align: center;

            text-decoration: none;

            display: inline-block;

            font-size: 16px;

            margin: 4px 2px;

            cursor: pointer;

            border-radius: 4px;

        }

        ul {

            list-style-type: none;

            padding: 0;

        }

        li {

            background: white;

            padding: 10px;

            margin-top: 8px;

            border-radius: 8px;

            box-shadow: 0 2px 4px rgba(0,0,0,0.1);

        }

    </style>

</head>

<body>

    <h1>Recipe App</h1>

    <form id="recipe-form">

        <input type="text" id="name" placeholder="Recipe name" required>

        <textarea id="ingredients" placeholder="Ingredients" required></textarea>

        <button type="submit">Add Recipe</button>

    </form>

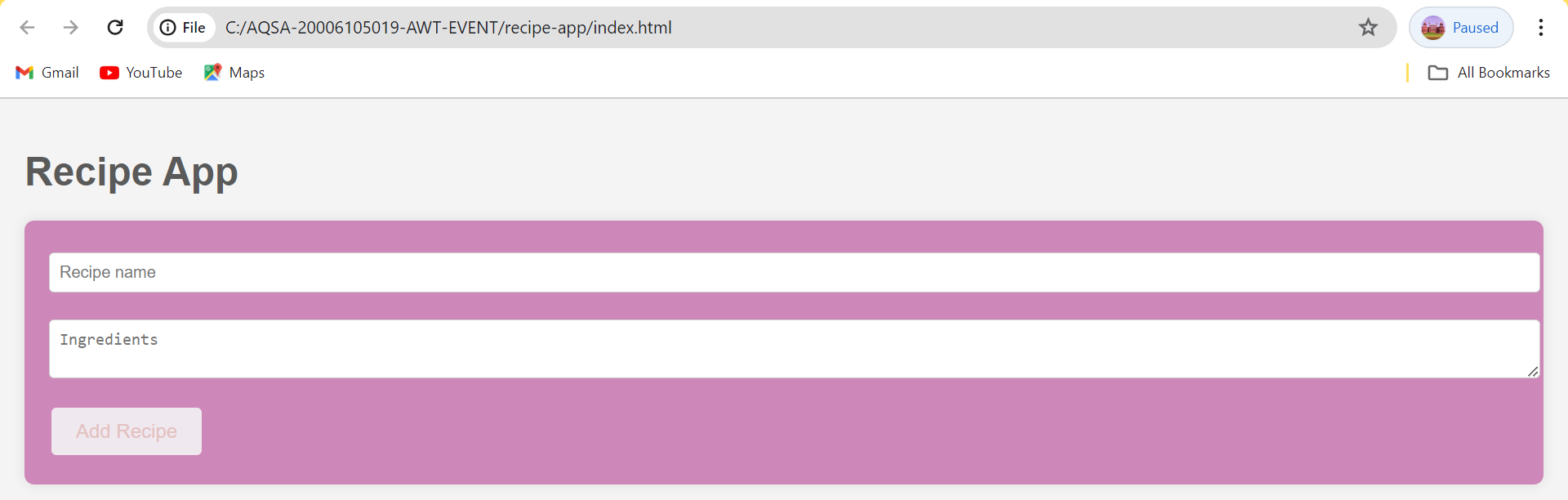
    <ul id="recipes-list"></ul>

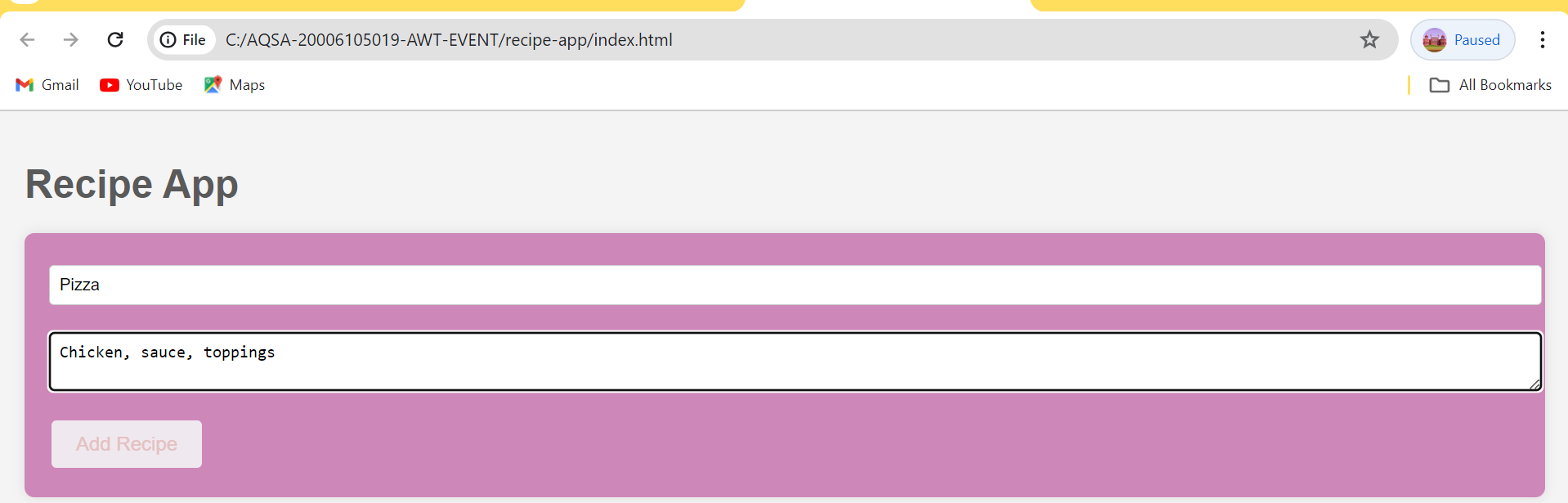
    <script src="script.js"></script>

</body>

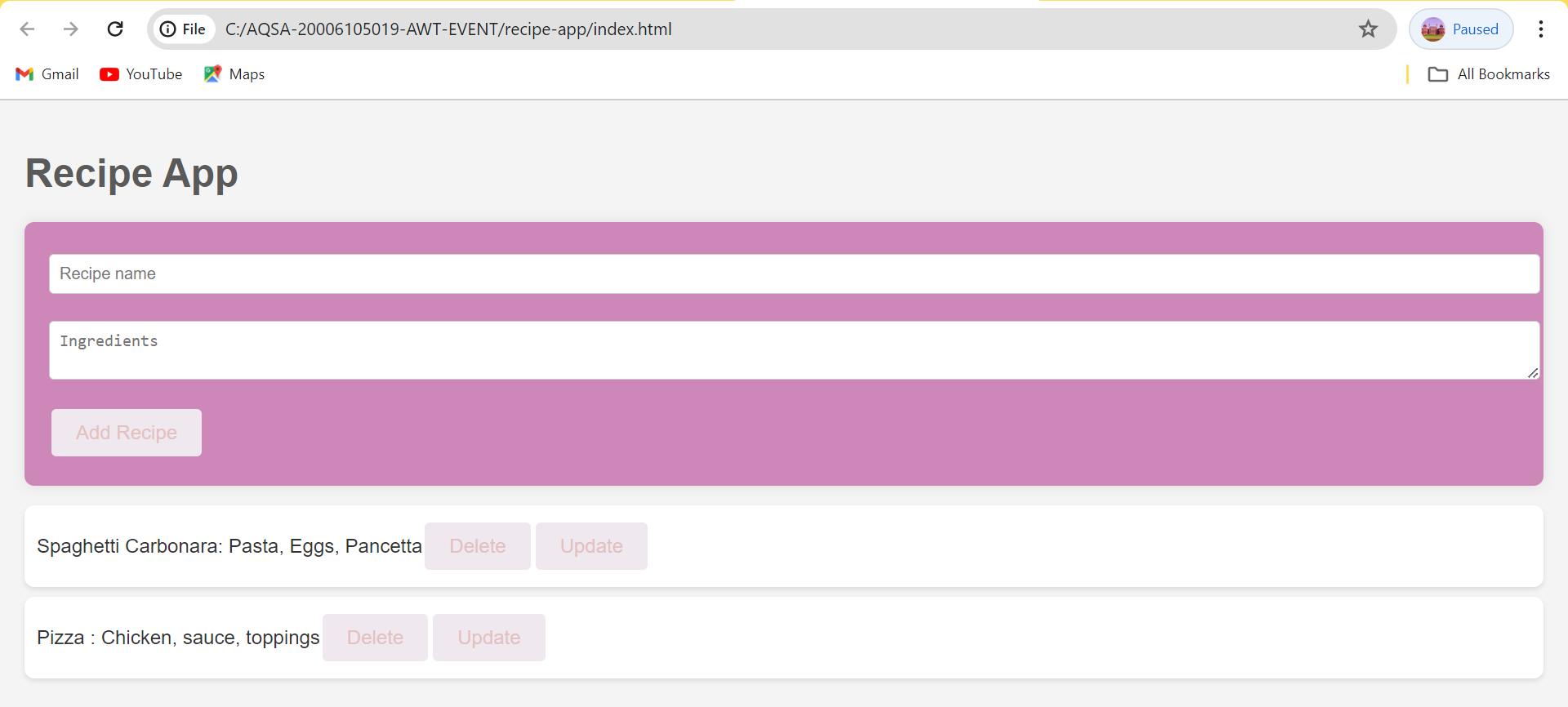
</html>

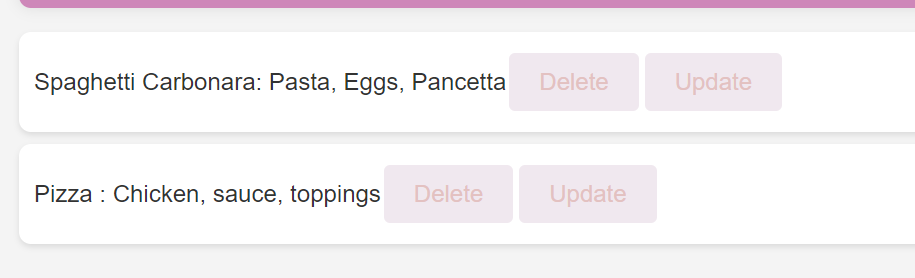
Homepage Screenshot:



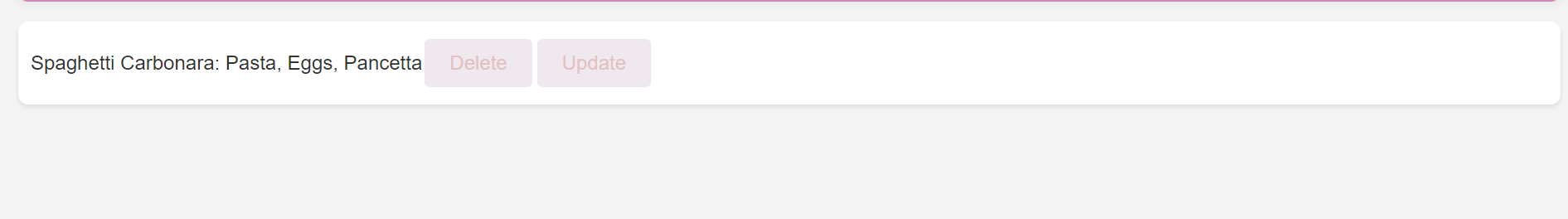


Add recipe

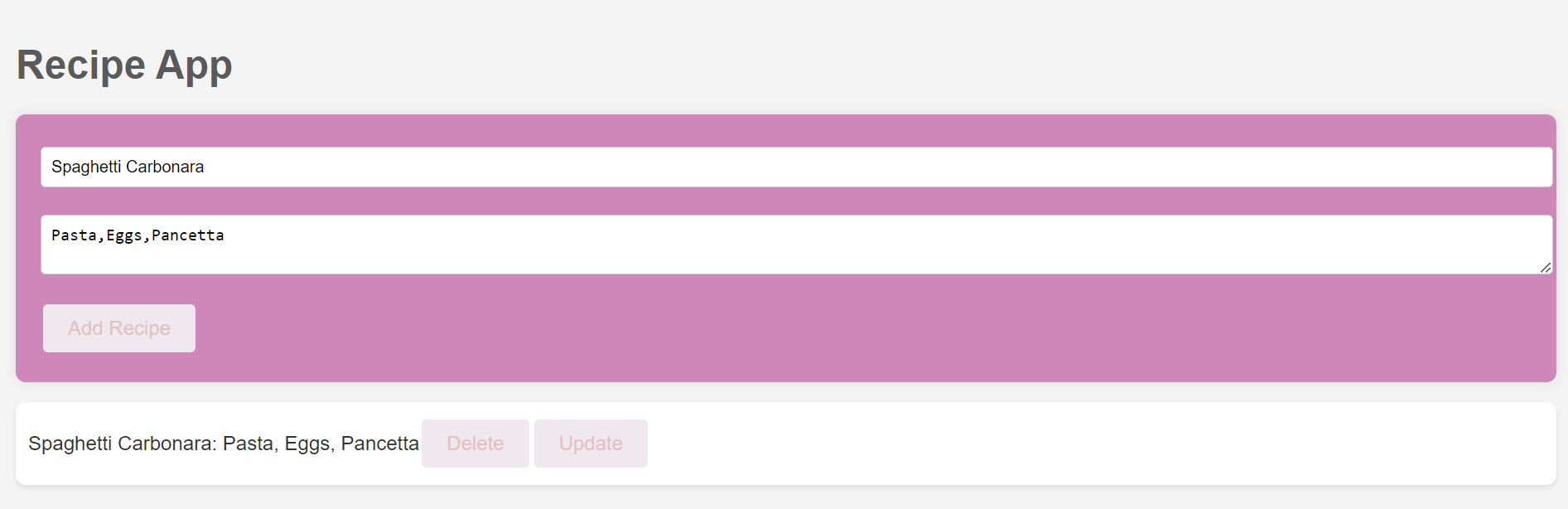




Delete, update, read Recipes.



One recipe delete.



Update and read recipe.