**Documentation for the Recipe Planner Application**

**Project Overview**

The Recipe Planner Application is a web-based platform designed to manage and organize recipes. Users can create, view, and manage recipes and their respective ingredients. This project emphasizes incorporating robust testing methodologies, including unit tests, component tests, and automated testing, to ensure high code quality and functionality.

**Objectives**

* Develop additional features to enhance the application, such as a search bar for filtering recipes by name.
* Improve the testing coverage for backend and frontend components.
* Automate test reporting and include code coverage analysis.
* Explore and implement Test-Driven Development (TDD).
* Conduct and document code reviews for constructive feedback and collaborative improvement.

**Planning**

**Features Developed**

1. **Search Bar**:
   * A feature to filter recipes by name dynamically on the frontend.
2. **Fix Submit Button**:
   * Addressed issues with the "Submit" button on the Add Recipe page, ensuring successful form submission to the backend.

**Testing Goals**

* Achieve high test coverage for critical components of the application.
* Automate test execution and reporting.
* Validate features end-to-end, including API interactions and UI functionality.

**Development Approach**

* Emphasis on iterative development with regular feedback.
* Use of TDD to identify edge cases early.
* Collaboration through pull requests with active comments and suggestions.

**Architecture**

**Application Layers**

1. **Frontend**:
   * Built with React.js, featuring components for managing and displaying recipes.
   * Integration with backend APIs for data retrieval and submission.
2. **Backend**:
   * Spring Boot application providing REST APIs for recipe management.
   * Includes a service layer for business logic and a mapper layer for data transformations.
3. **Database**:
   * Persistent storage for recipes and ingredients.

A white rectangle with black text

Description automatically generated

**Test Concept**

*Please* *refer to the test concept document for detailed testing approaches. It is located in Test Concept on Github in:  
Project LB2/TestConcept/*[*TestConceptRecipePlannerNikolaMinh.docx*](https://github.com/NikolaBogosavljevic/NiBog_M450/blob/main/Project%20LB2/TestConcept/TestConceptRecipePlannerNikolaMinh.docx)

**Code Reviews and Pull Requests**

**Process**

* Each team member submitted a minimum of three pull requests during development.
* Pull requests were actively reviewed, with comments and suggestions provided to ensure code quality.
* Focus areas included:
  + Adherence to coding standards.
  + Identification of potential edge cases.
  + Improvement of test coverage and logic.

**Reflection**

Code reviews fostered collaboration and knowledge sharing among team members. Constructive feedback improved the overall quality and maintainability of the codebase.

**Reflection on TDD**

**Benefits**

* Encouraged writing tests before implementing features, leading to more thoughtful design.
* Helped identify edge cases early in development.
* Improved confidence in the codebase as all features were backed by tests.

**Challenges**

* Initial setup and adapting to the TDD workflow required additional time and effort.
* Writing meaningful tests for complex components and business logic was challenging but rewarding.
* **Writing meaningful tests for complex components and business logic was challenging but rewarding.**

**Testing Evidence**

**Postman API Testing**

1. Add Recipe:A screenshot of a computer

   Description automatically generated
   * *Figure 1: API Test for Adding a Recipe using Postman.*
2. Retrieve All Recipes: A screenshot of a computer

   Description automatically generated
   * *Figure 2: API Test for Retrieving All Recipes using Postman.*
3. Retrieve Recipe by ID: A screenshot of a computer

   Description automatically generated
   * *Figure 3: API Test for Retrieving a Recipe by ID using Postman.*

JUnit Test Results

* + Mapper Tests: *A screenshot of a computer program

    Description automatically generatedFigure 4: JUnit Test Results for RecipeEntityMapper.*
  + Controller, Service, and Mapper Tests: *A screen shot of a computer program

    Description automatically generatedFigure 5: Overall JUnit Test Results for Backend Components.*

**Conclusion**

The Recipe Planner Application provided us with a valuable opportunity to focus on testing practices and methodologies. This project allowed us to apply Test-Driven Development (TDD), write extensive tests for backend and frontend components, and integrate testing into our workflow.

One of the most significant challenges we faced was setting up a CI/CD pipeline to automate the execution of tests and generate reports. Overcoming this challenge gave us a deeper understanding of modern testing workflows and automation tools. Using frameworks like JUnit, Mockito, and Cypress, we ensured that both unit tests and end-to-end tests were thoroughly implemented.

Collaborating through GitLab and conducting structured code reviews enhanced the quality of our tests and the overall product. This project reinforced the importance of testing in the development process and taught us practical skills that we are confident will be invaluable in future projects.