Testing plan 2024

JUNE 11

HBO-ICT
Authored by: Joeri Harreman

Contents

Test Plan

User Stories:

User Registration:

As a user, I want to register on the website with my details.

Paths:

- Happy Path: A user successfully registers with valid data.
- Unhappy Path: Registration fails with invalid data.

User Login:

As a user, I want to log in to the website with my credentials.

Paths:

- Happy Path: A user successfully logs in with valid credentials.
- Unhappy Path: Login fails with invalid credentials.

Recipe Index View:

As a user, I want to view the list of recipes available on the website.

Paths:

- Happy Path: Authenticated user views the list of recipes.
- Unhappy Path: Unauthenticated user is redirected to the login page.

Recipe Show View:

As a user, I want to view the details of a specific recipe.

Paths:

- Happy Path: Authenticated user views a specific recipe.
- Unhappy Path: Unauthenticated user is redirected to the login page or the recipe does not exist.

System Tests per User Story

User Story 1: User Registration

- Scenario 1 (Happy Path): User successfully registers with valid data.
- Scenario 2 (Unhappy Path): Registration fails with invalid data.

User Story 2: User Login

- Scenario 1 (Happy Path): User successfully logs in with valid credentials.
- Scenario 2 (Unhappy Path): Login fails with invalid credentials.

Unit Tests per User Story

User Story 3: Recipe Index View

- Scenario 1 (Happy Path): The authenticated user views the list of recipes.
- Scenario 2 (Unhappy Path): The unauthenticated user is redirected to the login page.

User Story 4: Recipe Show View

- Scenario 1 (Happy Path): Authenticated user views a specific recipe.
- Scenario 2 (Unhappy Path): Unauthenticated user is redirected to the login page or the recipe does not exist.

Applying the V-Model

User Story 1: User Registration

- Requirement Gathering: Users need to register on the website.
- System Analysis: The system must provide a registration form and store user data.
- Software Design: Design the registration process in the UserController.
- Module Design: Implement the registration logic and validation.
- Coding: Implement the registration function in the UserController.

Testing Phases:

- Unit Testing: Test the registration function with different input scenarios.
- Integration Testing: Ensure the registration logic works with the user database.
- System Testing: Validate the complete registration process from form submission to data storage.
- Acceptance Testing: Verify that user registration meets the user requirements and expectations.

User Story 2: User Login

- Requirement Gathering: Users need to log in to the website.
- System Analysis: The system must authenticate users with valid credentials.
- Software Design: Design the login process in the UserController.
- Module Design: Implement the login logic and validation.
- Coding: Implement the login function in the UserController.

Testing Phases:

- Unit Testing: Test the login function with different input scenarios.
- Integration Testing: Ensure the login logic works with the user database.
- System Testing: Validate the complete login process from form submission to user authentication.
- Acceptance Testing: Verify that user login meets the user requirements and expectations.

User Story 3: Recipe Index View

- Requirement Gathering: Users need to view a list of recipes.
- System Analysis: The system must retrieve and display a list of recipes.
- Software Design: Design the index view in the RecipeController.
- Module Design: Implement functions to retrieve and display recipes.
- Coding: Implement the index view function in the RecipeController.

Testing Phases:

- Unit Testing: Test the index view function with different data scenarios.
- Integration Testing: Ensure the index view logic works with the recipe database.
- System Testing: Validate the complete index view process from data retrieval to display.
- Acceptance Testing: Verify that the recipe index view meets the user requirements and expectations.

User Story 4: Recipe Show View

- Requirement Gathering: Users need to view details of a specific recipe.
- System Analysis: The system must retrieve and display recipe details.
- Software Design: Design the show view in the RecipeController.
- Module Design: Implement functions to retrieve and display recipe details.
- Coding: Implement the show view function in the RecipeController.

Testing Phases:

- Unit Testing: Test the show view function with different data scenarios.
- Integration Testing: Ensure the show view logic works with the recipe database.
- System Testing: Validate the complete show view process from data retrieval to display.
- Acceptance Testing: Verify that the recipe show view meets the user's requirements and expectations.

Possible Mistakes/Errors Not Detected by Tests

Data Accuracy

- The tests may not detect issues related to the actual accuracy of the data. For example, if the external API provides incorrect data, the tests will not identify this discrepancy as they assume the provided data is accurate.

External Data Manipulation

- The tests may not cover scenarios where the data from the external API is manipulated or tampered with. This could include security vulnerabilities or data integrity issues that arise from external sources.

User Interface Errors

- While the tests check for the presence of specific elements on the page, they may not detect visual or layout issues that affect the user experience. For example, if the recipe list is displayed correctly but the layout is broken, the tests will not identify this issue.

Conclusion

Confidence in System Correctness

- The tests provide a high level of confidence that the system works as expected under normal and error conditions. They cover the critical paths of user registration, login, and viewing recipes, ensuring that the core functionalities are tested.

Assumptions and Limitations

- The tests focus on functional correctness and may not fully cover performance, security, or usability aspects of the system.
- While the tests provide a good indication of system correctness, it is important to continuously monitor and test the system in a production environment to detect and address any issues that may arise.

Overall Conclusion

- Based on the tests conducted, we can conclude that the system is robust and functions correctly under the tested scenarios. However, it is essential to complement these tests with additional testing strategies, such as performance testing, security testing, and user acceptance testing, to ensure comprehensive coverage and reliability of the system

Screenshots

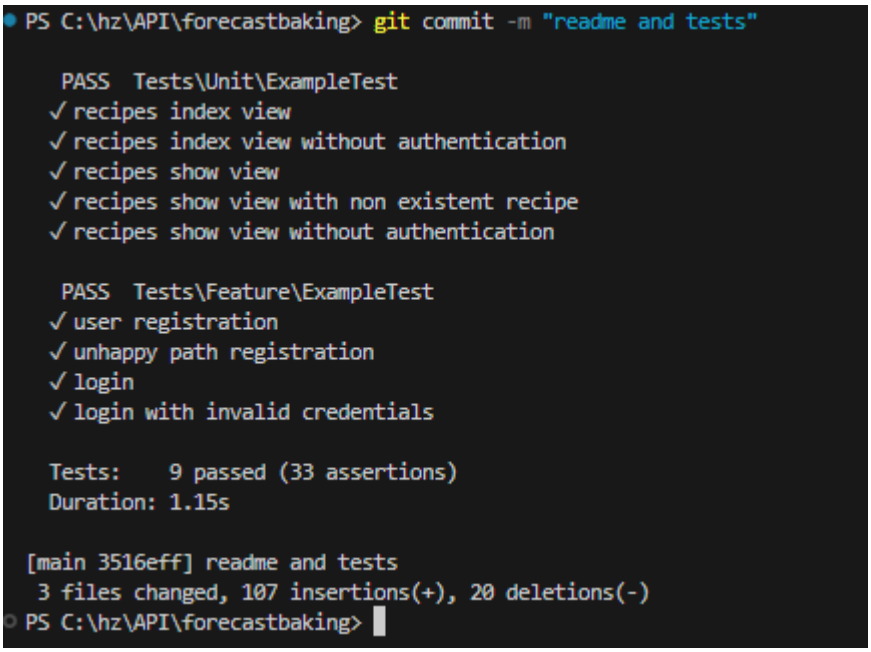

Figure 1, Auto tests when committing.