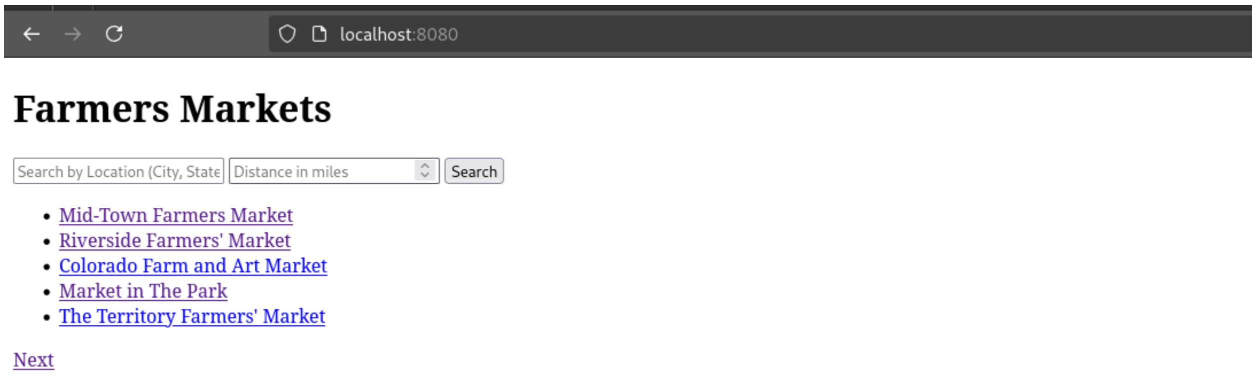
Testing Manual for Farmers Market



localhost:8080

Farmers Markets

- Mid-Town Farmers Market
- Riverside Farmers' Market
- Colorado Farm and Art Market
- Market in The Park
- The Territory Farmers' Market

Next

Test Strategy for Farmers Market

Objectives

The main objectives of the testing strategy are to:

- Ensure the Farmers Market fulfills all specified functional requirements.
- Check the application's behavior under a range of scenarios and edge cases.
- Assess the user interface for both usability and accessibility.
- Verify the application's performance efficiency across varying load conditions.

Introduction

This testing manual provides a step-by-step guide to thoroughly test the Farmers Market Web Application. It outlines the test cases, expected results, and procedures to verify that each component of the application functions correctly. This manual is aimed at developers and testers who need to validate the application's functionality before deployment.

---

1. Environment Setup

Before starting the tests, ensure that your environment is correctly set up.

Prerequisites:

- Java Development Kit (JDK): Ensure that JDK 11 or later is installed.
- Apache Maven: Ensure Maven is installed and properly configured.
- Spring Boot: Verify that the Spring Boot application can be run locally.
- Database Configuration: The database should be populated with test data, and the dbconfig.properties should be correctly configured.

Build and Run the Application:

- Open a terminal and navigate to the root directory of the project.
- Use Maven to build the project:
  ```
  mvn clean install
  ```
- Run the application:
  ```
  java -jar target/farmers-market-application.jar
  ```

2. Testing Scenarios

2.1 Homepage Display

Test Case: Verify that the homepage loads correctly with a list of farmers markets.

- **Steps:**
  - Navigate to http://localhost:8080/.
  - Verify that the page loads without errors.
  - Check that the list of farmers markets is displayed.
- **Expected Result:**
  - The homepage should display a list of farmers markets with their names and addresses.
  - The pagination controls should work, allowing navigation between pages of markets.

2.2 Search Functionality

Test Case: Verify that the search functionality works as expected.

- **Steps:**
  - On the homepage, enter a search query in the search bar (e.g., “New York”).
  - Click the "Search" button.
  - Observe the results displayed.
- **Expected Result:**
  - The application should display farmers markets that match the search query.
  - If no markets match the query, a “No results found” message should be displayed.

2.3 Market Details Page

Test Case: Verify that the market details page displays correct information.

- **Steps:**
  - Click on a market name on the homepage.
  - Verify that the market details page loads with all relevant information (name, address, seasons, products, etc.).
- **Expected Result:**

    - The details page should accurately reflect the data stored in the database for the selected market.
    - If the market does not exist, the application should display a “Market details not found” message.

2.4 Pagination

Test Case: Verify that pagination works correctly on the homepage.

- **Steps:**
    - On the homepage, navigate through the pages using the "Next" and "Previous" buttons.
    - Verify that each page displays the correct list of farmers markets.
- **Expected Result:**
    - The “Next” button should load the next set of markets, and the “Previous” button should load the previous set.
    - Pagination controls should only be available when there are more markets to display.

2.5 Error Handling

Test Case: Verify that the application handles errors gracefully.

- **Steps:**
    - Simulate a database connection failure by providing incorrect database credentials in the dbconfig.properties file.
    - Attempt to load the homepage.
- **Expected Result:**
    - The application should display an appropriate error message indicating the problem.
    - The application should not crash and should recover once the correct credentials are restored.

---

3. Performance Testing

3.1 Load Testing

Test Case: Verify that the application can handle a large number of simultaneous users.

- **Steps:**
    - Use a tool like Apache JMeter to simulate multiple users accessing the homepage and performing searches.
    - Gradually increase the number of users and observe the application's performance.
- **Expected Result:**
    - The application should remain responsive under load.
    - There should be no significant degradation in performance.

3.2 Stress Testing

Test Case: Test the application’s behavior under extreme conditions.

- **Steps:**
    - Simulate extreme conditions by overloading the server with requests.
    - Observe the application’s behavior when resources are exhausted.
- **Expected Result:**
    - The application should handle extreme conditions without crashing.
    - It should provide meaningful error messages when it cannot process requests.

---

4. Usability Testing

Test Case: Verify that the user interface is intuitive and easy to use.

- **Steps:**
    - Perform tasks such as searching for markets, navigating to market details, and paginating through results.
    - Gather feedback on the ease of use, layout, and navigation.
- **Expected Result:**
    - The application should be easy to navigate with clear labels and instructions.
    - Users should be able to complete tasks with minimal difficulty.

---

5. Security Testing

Test Case: Verify that the application is secure against common vulnerabilities.

- **Steps:**
  - Test for SQL injection by entering malicious inputs in the search field.
  - Test for XSS vulnerabilities by inputting JavaScript code in the search field.
  - Ensure that sensitive data is not exposed in URLs or error messages.
- **Expected Result:**
  - The application should sanitize inputs and prevent SQL injection and XSS attacks.
  - Sensitive data should be protected and not exposed.