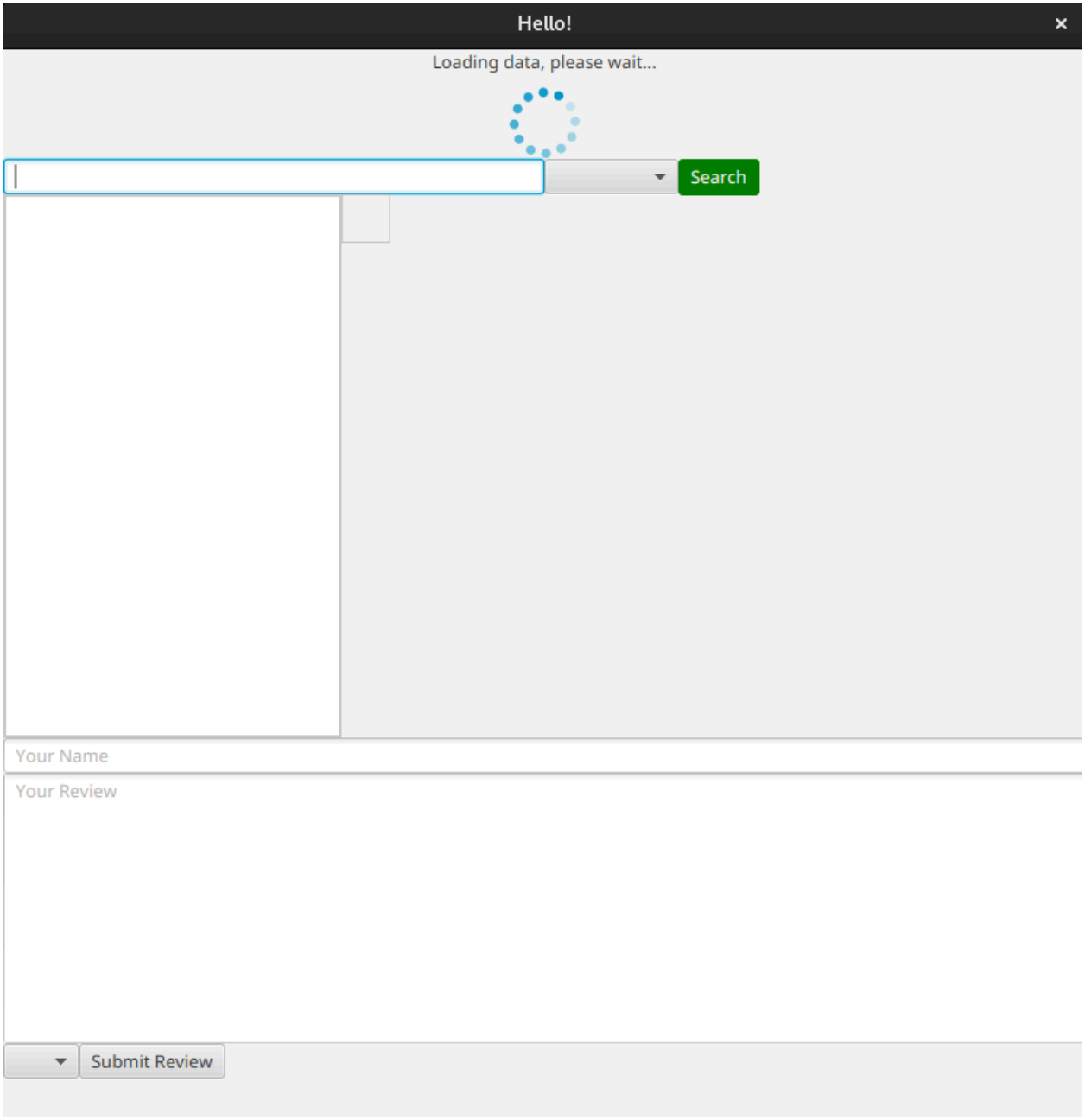
Testing Manual for Farmers Market



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.

Test Environment

- **Operating System**: Windows 10, macOS, or Linux
- **Java Version**: JDK 11 or later
- **JavaFX Version**: 11 or later

Prerequisites:

- Java Development Kit (JDK) installed on your system.
- JavaFX SDK installed.

Configuring Database Settings:

1. **Navigate to the root directory**
2. **Open dbconfig.properties using any text editor. Contents should be similar to this:**

```
# dbconfig.properties
jdbc.url=jdbc:mysql://128.113.195.120:3306/user003?allowPublicKeyRetrieval=true&useSSL=false
jdbc.user=user003
jdbc.password=H*FAGIbugbfsdiFG
```

3. **Edit the values to your choosing.**

Running the Program:

1. **Navigate to the Source Directory**.
2. **Compile the Java files**: Use the `javac` command to compile all Java files in the specified package directory.
3. **Run the JavaFX Application**: Use the `java` command to run the main application class. Ensure that you specify the fully qualified class name, including the package.

Notes:

JavaFX Modules: Since JavaFX is not included in the JDK by default, ensure you have the JavaFX SDK installed and properly configured. If you are using Java 11 or later, you might need to add the JavaFX modules explicitly:

```
java --module-path /path/to/javafx-sdk/lib --add-modules
javafx.controls,javafx.fxml
edu.rpi.cs.csci4963.u24.wangn4.hw05.farmers_market.farmers_market.java
```

Summary:

To start the program:

1. Ensure you have Java and JavaFX installed on your system.
2. Configure the database config.
3. Compile and run the `FarmersMarketApplication`class.

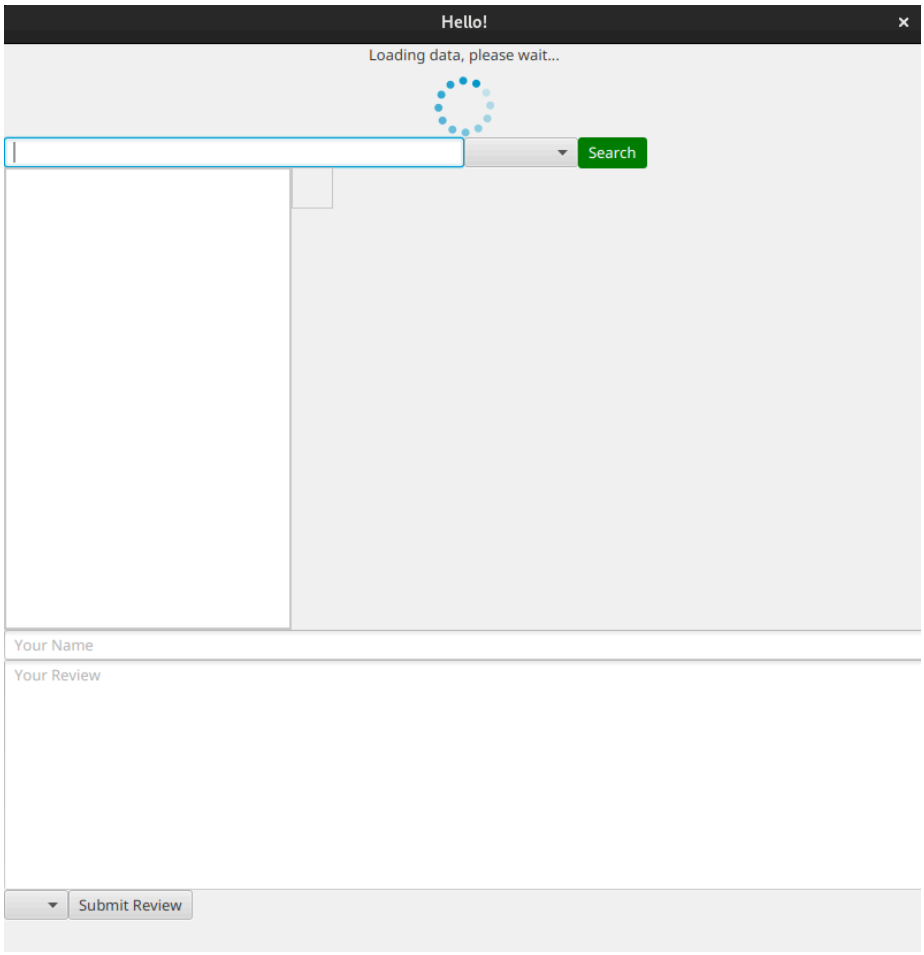
Test Cases:

Functional Tests:

Application Launch:

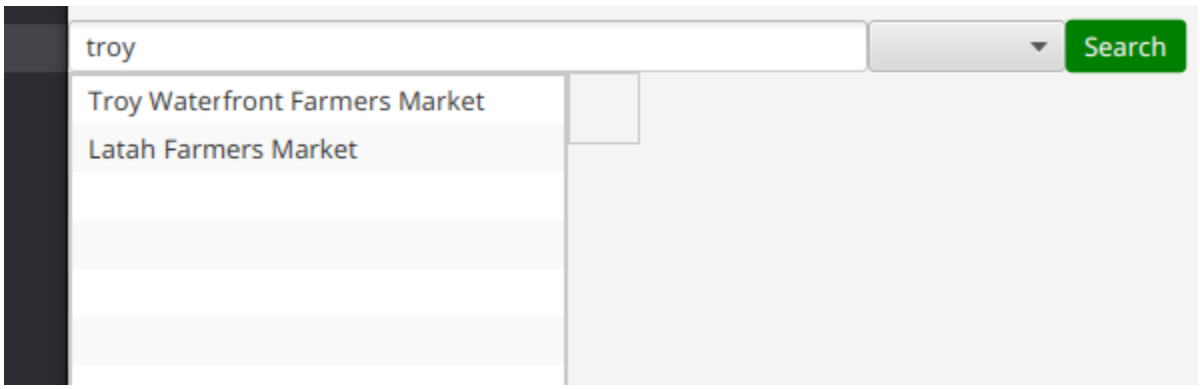
- **Test Case**: Launch the FarmersMarketApplication.
- **Steps**:
    1. Open a terminal or command prompt.
    2. Navigate to the directory containing the compiled classes.
    3. Run the application using the command provided in the setup instructions.
- **Expected Result**: The application window should open without errors.

  Main window should look like so:



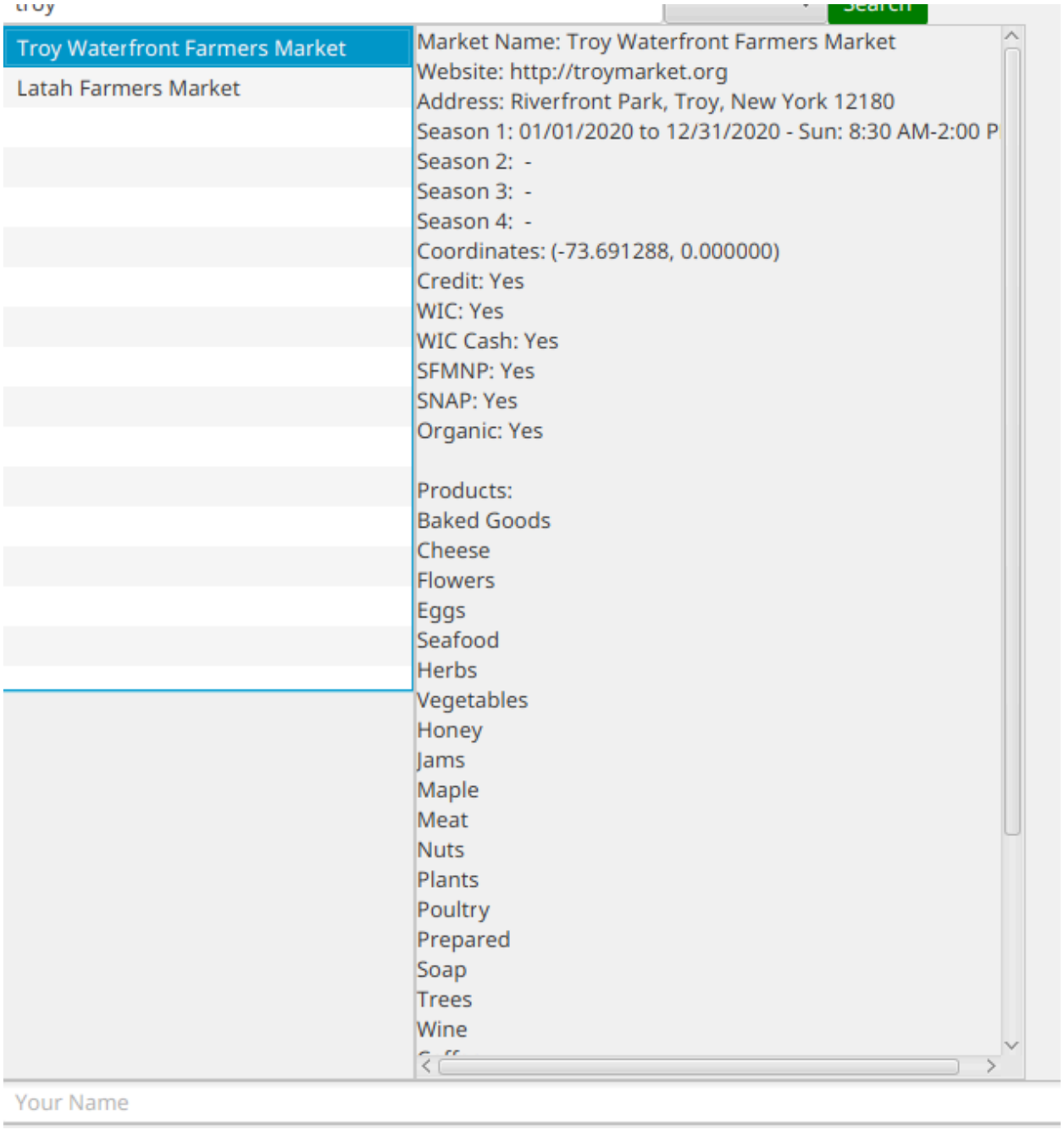
Search Funtion:

- **Test Case**: Search for a market
- **Steps**:
    1. Enter Troy into the text field and pick 10 miles from the drop down.
    2. Click the Search button
- **Expected Result**: The search results list should update like so:



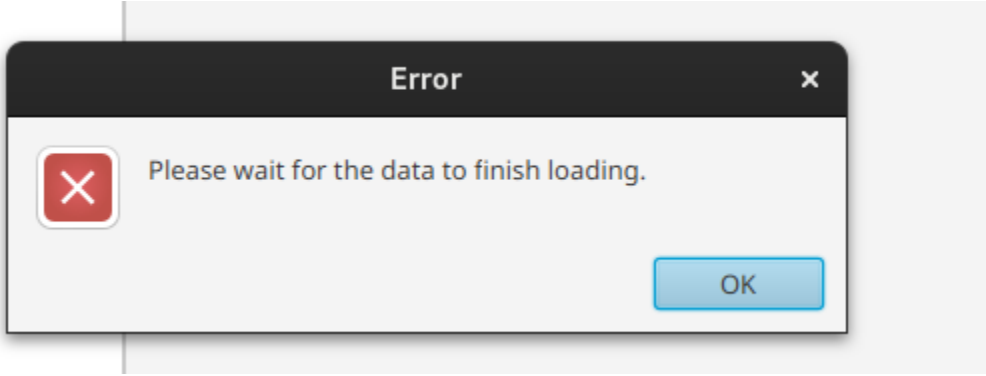
Select Market Function:

- **Test Case**: Select a market from the search results list
- **Steps**:
    1. Click on a market from the search results list
- **Expected Result**: A description of the selected market should appear on the right of the search results list.

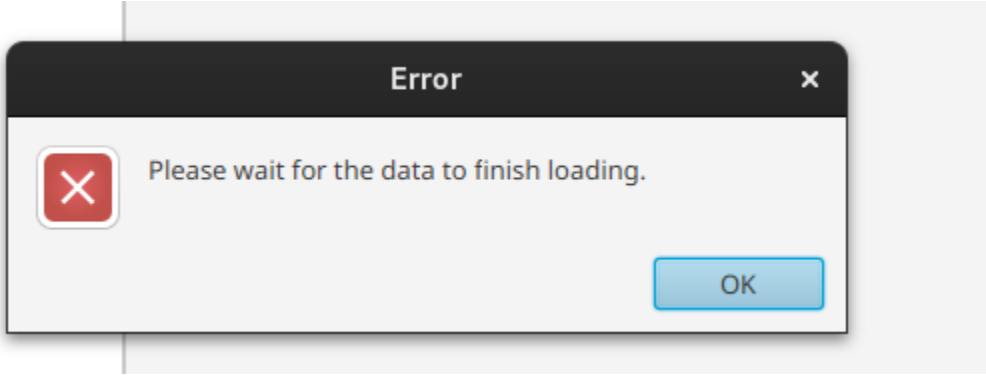


Testing Warnings:

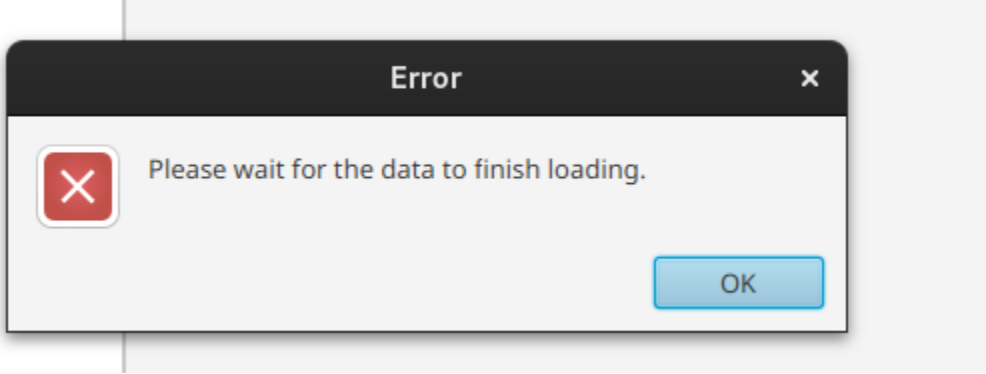
Clicking on the search button before the data is loaded will result in the following popup:



Clicking on the submit review button before the data is loaded will result in the following popup:



Clicking on the Search Results Box before the data is loaded will result in the following popup:



Testing Suite Description:

The testing suite for the Farmers Market application is designed to rigorously assess and validate each aspect of the system, similar to the approach used in the epidemic simulation project. This suite includes the following tests:

Integration Tests:

- **Database Interaction**: Test the seamless integration between the JavaFX GUI and the database. Verify that searches by city, state, and ZIP code return accurate results, and that the system correctly handles edge cases like nonexistent locations or markets.
- **Review Submission**: Assess the full cycle of the review submission process, ensuring that reviews are correctly written to the database, properly associated with the corresponding market, and accurately displayed in the detailed view.

Performance Tests:

- **Data Handling**: Evaluate the application's performance with large datasets, particularly focusing on load times and the responsiveness of the search functionality. Stress tests will be conducted to simulate high usage scenarios.
- **Search Efficiency**: Test the efficiency of search operations, including the handling of complex queries with multiple filters (e.g., location and distance). Measure the query execution time and the impact on the user experience.

Security Tests:

- **SQL Injection Protection**: Validate the application's defense against SQL injection attacks by attempting to insert malicious inputs. Ensure that all database queries are parameterized, preventing unauthorized access or data corruption.
- **Data Integrity**: Ensure that all user inputs, particularly those involving the review submission, are correctly sanitized and stored, maintaining the integrity and accuracy of the data.

End-to-End Tests:

- **User Journeys**: Simulate typical user interactions, from viewing the market list to searching for specific locations and submitting reviews. Ensure that each step works as intended and that the overall experience is smooth and intuitive.
- **Cross-Browser/Platform Testing**: Test the application across different operating systems, browsers, and devices to confirm consistent performance and appearance.

Regression Tests:

- **Post-Update Validation**: After any new features or changes are implemented, run a full suite of tests to confirm that existing functionality remains unaffected. This includes re-testing critical paths and previously identified edge cases.