NewLine

Real Estate Management Web Application

Anshul Sharma

Test Report

Table of Contents

Test Report (NewLine)

Introduction

NewLine is the name of the web application design using Django Framework. It is an Online real estate management web application through which a user can access information about the various properties and listings provided by the company which is handling the real estate business. The project is oriented for both the admin (in this case, the head of a company or whoever manages the website) and a user who would search for information about the listing. The admin can manage all the adding, updating, and deleting of the assets. The project effectively advertises the listings which are available and provides options to contact the realtor of the listing. The admin can update the information regarding the listing or the realtors who are present for the service. The objectives that were kept in mind during development were-

- The Admin maintains the property and the realtors kept on the behalf. Admin (or superuser) of the website can add, delete, and update the properties and listings available alongside the information of the users of the website, realtors. The website is equipped with a login system.
- The user can browse through the listings advertised and can register to submit information regarding his/her interest in the listing.

The application, after being designed and developed, is subjected to automation testing, which is discussed in this report.

The document gives details about the testing procedure of the application and contains information regarding the test requirements, environment and test cases.

Requirements

The following requirements were stated to create testing scenarios-

- The application should have a navigation bar, to allow the users to navigate through the following sections- Home, About, Featured Listings, Registration, and Login Page.
- Users should be able to register an account (if not registered already), by providing information in a form, such as FirstName, LastName, Username, Email and Password.
- Users should be able to access the listings and information related to them.

- The user must be able to send his/her information to the admin based on interest in a specific listing.
- Registration Page must not let invalid values pass through it.
- Validations must be present on the Login Page as well.
- The Validations must be a combination of server-side and client-side responses. It enables the login and registration modules to be more dynamic and responsive.
- Admin must have a separate panel dedicated to the admin's view of the application. A normal user must not be able to access it.

Scope

The scope of the testing phase covers the following testing processes/levels-

- Smoke Test – To check whether the application runs on the development server (runserver – dev/wsgi-based server), we need to test whether the application is executed successfully.
- Functional Testing- Validates the application's functionality with the stated requirements.
- Unit Testing- Some of the testing is based on a single module (For instance-Some of the Register module's input validations are handled through client-side scripting, and testing these modules is part of the unit testing.
- Integration Testing- The function **redirect** and **render** utilized in the views.py files of each module. This is done to integrate all these essential components/modules of the application, which comprises the Integration testing.
- System Testing

Test Approach

The following points were considered while designing the test approach-

- The Test cases should correspond to the requirements and scenarios.
- Expected Outcome should be compared with the actual outcome of the test case execution.

- The tester has the access to the source code.
- Test Scenarios and Test cases should consider the System Under Test and modules of the application that are being subjected to be automated.
- Manual Testing is also a feasible option in this case, but the test would be automated to demonstrate the working of TestNG.
- The steps of the test process would be-
  - Smoke Test
  - Creating Test scenarios and cases
  - Configuring the Test Environment
  - Developing Automation Code
  - Executing and recording the results of the test

Test Environment

Operating System- For the test execution, Microsoft Windows Operating System (Windows 10) is utilized.

Web Browser – As chromedriver (chromedriver.exe is included in the directory) is utilized, the testing is primarily done on the Google Chrome web browser.

Development Server- The server where the web application launches itself. It is configured to be the 'localhost' in the application code. On automation, the browser is expected to launch the development server, which comprises the Smoke Test.

Eclipse IDE – The integrated Development Environment used to develop the automation code and maintain the project directory.

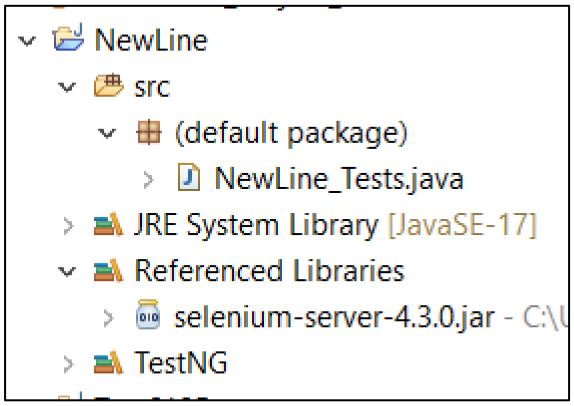
Java, JDK and JRE – The system must be capable of running .java files, and must be accompanied by Java Development Kit (JDK) and JRE (Java Runtime Environment), with their respective paths configured in the system variables.

Selenium Webdriver – Web framework for automation and cross-browser test execution.

TestNG – The testing framework used to make the test execution more readable and structured. For instance, a user can simply execute the code and check on the console screen to verify the execution and outcome of test cases. It is possible due to the readability functionality provided by the TestNG.

Test Project structure and Build Paths

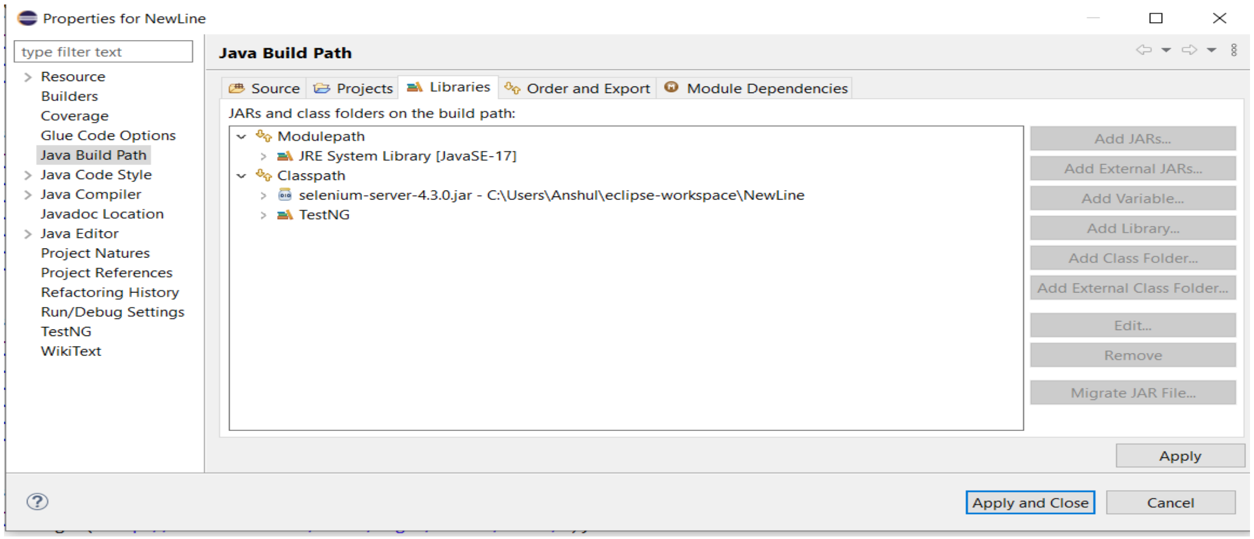
The project directory is structured as-



```
NewLine
  src
    (default package)
      NewLine_Tests.java
  JRE System Library [JavaSE-17]
  Referenced Libraries
    selenium-server-4.3.0.jar - C:\
  TestNG
```

The 'src' folder has the source code to the Java test file, as shown in the above figure. Referenced libraries has all the internal and external libraries that the project is integrated with, which in this case is the selenium server '.jar file'. TestNG is added as a library to the path of the project.

Right-clicking on Newline>Build Path, the IDE displays the properties of the project.



Properties for NewLine

Java Build Path

JARs and class folders on the build path:

```
Modulepath
  JRE System Library [JavaSE-17]
Classpath
  selenium-server-4.3.0.jar - C:\Users\Anshul\eclipse-workspace\NewLine
  TestNG
```

In this window, we can add relevant .jar and libraries to the classpath or modulepath of the project.

Test Criteria

Test Entry Criteria –

1) Environment is configured and ready to use.

2) The project directory is built, and all the required JARs and libraries are integrated with the build path of the project.

3) Test script and test data are designed and ready to be executed.

Test Exit Criteria-

1) All the planned test cases have been executed.

2) If the required level of test coverage has been met.

3) All the test outcomes have been recorded.

Smoke Test

Smoke Test is conducted by launching the application. The application should be opened under the URL- 127.0.0.1:8000 (which is the default destination for the framework's development environment.

Smoke Test Status- Successful (The application runs at the URL 127.0.0.1:8000).

Note- the port might differ on different systems, depending on which port the PostgreSQL server listens to/has been configured, and the default port for the localhost.

Test Scenarios and Test Cases

The test case classes defined in the test code are named after their respective Test Case IDs.

As input fields are involved in the project (modules such as admin,

Corresponding to the test cases, the tests can be defined using '@Test' in the source code.

The test cases are divided as 'valid' and 'invalid' test cases. An invalid test case is the one in which the inputs given to the system are not supposed to be, and given to observe the systems's response.

The following table shows the Test scenarios along with its related test case and data. The test results are also recorded in 'Actual Outcome' and 'Test Status'.

For Valid Test cases-

Test Case ID	**Corresponding Testing Scenario**	**Test Case Description**	**Expected Outcome**	**Actual Outcome**	**Test Status**
TC_001_VALID	Check the Navigation by visiting the Home, About and Featured Listings sections. Also, check whether Login and Registration forms are accessible.	System – Development Server Browser- Google Chrome Test Level – Integration Testing Test Method- Functional Testing Function- driver.get("URL of the page") Test Steps- • Visit- 127.0.0.1:8000 • Then- 127.0.0.1:8000/about. • 3)127.0.0.1:8000/listings. • 127.0.0.1:8000/accounts/register. • 5) 127.0.0.1:8000/accounts/login.	All the pages are visited with the mentioned URLs.	All the pages are visited with the mentioned URLs.	Pass
TC_002_VALID	Verify the functionality of the registration page.	System – Development Server Browser- Google Chrome Test Level – Integration Testing Test Method- Functional Testing Function- Register Test Steps- • Visit- 127.0.0.1:8000/accounts/register • Fill out FirstName, LastName, username, Email and Password. • Click on 'Register'. • Log in with the same credentials.	Registration Successful.	Registration Successful.	Pass

		Test Data – ‘First Name’: Anshul ‘Last Name’: Sharma ‘Username’:anshul10 ‘email’:******@gmail.com ‘Password’: ******			
TC_003_VALID	Verify if the user can access information about the listings.	System – Development Server Browser- Google Chrome Test Level – Integration Testing Test Method- Functional Testing Function- More Info (Listings) Test Steps- • Visit- 127.0.0.1:8000/listings/. • Select ‘More Info’.	User can see the following details- Asking Price, Bedrooms , Bathrooms, Garage, Square Feet, Lot size, Listing date, Realtor.	User can see the following details- Asking Price, Bedrooms , Bathrooms, Garage, Square Feet, Lot size, Listing date, Realtor.	Pass
TC_004_VALID	Check the ‘Make An inquiry’.	System – Development Server Browser- Google Chrome Test Level – Integration Testing Test Method- Functional Testing Function- Make An Inquiry (Listings) Test Steps- • Visit- 127.0.0.1:8000/listings/. • Select ‘More Info’. • Click on ‘Make An Inquiry’ • Fill the information- Name, Email, Phone, Message. Test Data- ‘Name’: ‘Anshul’ ‘email’: ‘abc@gmail.com’ ‘Phone’:111111110 ‘Message’: ‘Inquiry’	User can fill out the information and send the information to the admin.	User can fill out the information and send the information to the admin.	Pass
TC_005_VALID	Verify the login with	Test Level –Integration Testing	Server Side	Server Side	Pass

	registered credentials.	Test Method- Functional Testing Function- Login Test Steps- • Visit- 127.0.0.1:8000/accounts/login. • Fill out username and Password. • Click on 'Login'. • Log in with the same credentials. Test Data – 'Username': 'anshul10' 'Password': *******	Prompt- "You are Logged In"	Prompt- "You are Logged In	
TC_006_VALID	Verify the 'Admin' Login.	Test Level –Integration testing. Test Method- Functional Testing Function- Admin Login. Test Steps- • Visit- 127.0.0.1:8000/admin • Fill out username and Password for the superuser. • Click on 'Login'. • Log in with the same credentials. Test Data – 'Username': 'anshul' 'Password': *****	The admin Panel opens.	The admin Panel opens.	Pass

For Invalid test cases-

TC_001_INVALID	Verify the validation on the register page.	System – Development Server Browser- Google Chrome Test Level – Unit Testing Test Method- Functional Testing Function- Registration	Client-Side Prompt- 'Please Fill out this field'	Client-Side Prompt- 'Please Fill out this field'	Pass

		Test Steps- • Visit- 127.0.0.1:8000/accounts/register • Fill out FirstName,LastName, username, Email, Password. • Click on 'Register'. • Log in with the same credentials. Test Data – 'First Name': ' ' 'Last Name': ' ' 'Username':' ' 'Email':' ' 'Password': ' '			
TC_002_INVALID	Verify the validation on the register page, with an invalid value of email.	Test Level – Unit Testing Test Method- Functional Testing Function- Registration Test Steps- • Visit- 127.0.0.1:8000/accounts/register • Fill out FirstName,LastName, username, Email, Password. • Click on 'Register'. • Log in with the same credentials. Test Data – 'First Name': ' a' 'Last Name': ' b' 'Username':' c' 'Email':' @gmail.com 'Password': ******	Client-Side Prompt- 'Please enter a part, followed by@'. Or Validations are present.	Client-Side Prompt- 'Please enter a part, followed by@'. Or Validations are present.	Pass
TC_003_INVALID	Verify the validation on the register page, with invalid value of	Test Level – Integration Testing Test Method- Functional Testing Function- Register	Client-Side Prompt- 'Please include an @'. Or	Client-Side Prompt- 'Please include an @'. Or	Pass

	email (without @).	Test Steps- • Visit- 127.0.0.1:8000/accounts/register • Fill out FirstName,LastName, username, Email, Password. • Click on 'Register'. • Log in with the same credentials. Test Data – 'First Name': ' a' 'Last Name': ' b' 'Username':' c' 'Email':' gmail.com 'Password': ******	Validations are present.	Validations are present.	
TC_004_INVALID	Type passwords that don't match.	Test Level – Integration Testing Test Method- Functional Testing Function- Register Test Steps- • Visit- 127.0.0.1:8000/accounts/register • Fill out FirstName,LastName, username, Email, Password. • Click on 'Register'. • Log in with the same credentials. Test Data – 'First Name': ' a' 'Last Name': ' b' 'Username':' c' 'Email':' abc@gmail.com 'Password': * 'Confirm Password':**	Server Side Prompt- Passwords do not match.	Server Side Prompt- Passwords do not match.	Pass
TC_005_INVALID	Verify the validations in the Login Page.	Test Level –Unit Testing Test Method- Functional, Testing Function- Login	Client-Side Prompt- 'Please Fill out this form'	Client-Side Prompt- 'Please Fill out this form'	Pass

		Test Steps- • Visit- 127.0.0.1:8000/accounts/login. • Fill out username and Password. • Click on 'Login'. • Log in with the same credentials. Test Data – 'Username':' ' 'Password': ' '			
TC_006_INVALID	Entering Unregistered credentials in Login page and observing server-side response	Test Level – Integration testing Test Method- Functional Testing Function- Login Test Steps- • Visit- 127.0.0.1:8000/accounts/login. • Fill out username and Password. • Click on 'Login'. • Log in with the same credentials. Test Data – 'Username':'abc ' (Unregistered) 'Password': ******* (Unregistered)	Server Side prompt- 'Invalid Credentials'.	Server Side prompt- 'Invalid Credentials'.	Pass
TC_007_INVALID	Other user tries to access admin dashboard.	Test Level –Unit Testing Test Method- Functional, Testing Function- Admin Login Test Steps- • Visit 127.0.0.1:8000/login. • Login with user credentials. • Visit- 127.0.0.1:8000/admin • Fill out username and	Prompt- "You are authenticated as 'an', but are not authorized to access this page, Would you like to login to a different account".	Prompt- "You are authenticated as 'an', but are not authorized to access this page, Would you like to login to a different account".	Pass

		Password. • Click on 'Login'. • Log in with the same credentials. Test Data – 'Username':' anshul10' 'Password': *******			

Test Verdict

The following points can be noted after the text execution and recording the observations-

- All the stated requirements are met, and all the test cases have the status of 'pass'.
- The test was automated and the script used TestNG assertions to log the successful test cases in the console.
- The test cases and requirements were based on the development of the actual project. The actual web application is still subject to more potential additions and features. For instance, work is being done on implementing more complex regular expressions in the input fields of the register and login modules. This would result in more requirements and test cases. Updated versions of the application and test cases/scripts would be available in the same GitHub repository.
- The tester, while updating the test cases, should keep in mind that a user might be logged In while performing some of the test cases. Not taking consideration might result in false positives.