TEST PLAN  
ESTATELAND  
INDIVIDUAL TRACK

November 2, 2021

*Location:*

Eindhoven, Netherlands

*Author:*

Ana Maria Nitu - [a.nitu@student.fontys.nl](mailto:a.nitu@student.fontys.nl)

*Project Title:*

Estate Land

Contents

1. Software Testing…………………….………………..………….2
2. Types of Software Testing…………………….…………………2
3. Test Tools…………………………………………………...……3
4. Test Scope………………….…………………………...………..3
5. User Acceptance Tests/ Test Cases...…………………………….5

# software testing

Software testing is an investigation conducted in order to provide stakeholders with information about the quality of the product and to determine whether the actual software product meets the expected requirements and ensuring that the software product is free of defects. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

When working on a software development project, the developers need to be aware that errors might occur at any stage of the process. In relation to the real requirements, software testing's purpose is to detect mistakes, gaps, bugs or missing requirements.

Software testing is critical because it allows any flaws or errors in the software to be found early and corrected before the final product is delivered. A well-tested software product provides dependability, security, and excellent performance, which saves time, money, and improves customer satisfaction.

# TYPES OF SOFTWARE TESTING

For this project, I will be using 2 types of testing: Unit Tests and Integration Tests.

1. ***Unit tests***

Unit testing is a kind of software testing that examines individual software units or components, the smallest unit of code in a system that can be logically isolated. The goal is to ensure that each unit of software code works as intended. Unit testing is carried out by software developers throughout the coding phase of an application.

As far as this project is concerned, I will create unit tests in order to tackle small parts of the code and ensure the quality of the features, taking into consideration the scope, pre-conditions, steps, expected result and the actual result that needs to match the expectations.

1. ***Integration tests***

Following unit testing, integration testing is a more complicated step of software testing in which different software components are merged and tested as a group. Integration testing is used to assess a system's or component's compliance with defined functional requirements. Integration testing takes unit-tested modules as input, collects them into bigger aggregates, applies tests described in an integration test plan to those aggregates, and delivers the integrated system ready for system testing as an output.

For the project, I will start the integration tests after I make sure that all the unit tests are complete and they pass.

# TEST TOOLS

AS a testing tool for my code, I will use JUnit framework.

JUnit is a Java unit testing framework that's one of the best test methods for regression testing. It is an open-source framework, used to write and run repeatable tests. After running the unit test, JUnit will mark them and it is considered passed if all the tests pass successfully.

# test scope

The test scope refers to the software components to be tested.

|  |  |  |
| --- | --- | --- |
| *Test item* | *Type* | *Description* |
| UserDao | Interface | Extends the JpaRepository interface, communicates with the MySql database, and manages the User entity's CRUD (create, read, update, delete) operations. |
| PropertyDao | Interface | Extends the JpaRepository interface, communicates with the MySql database, and manages the Property entity's CRUD (create, read, update, delete) operations. |
| RoleDao | Interface | Extends the JpaRepository interface, communicates with the MySql database, and manages the Role entity's CRUD (create, read, update, delete) operations. |
| AddressDao | Interface | Extends the JpaRepository interface, communicates with the MySql database, and manages the Address entity's CRUD (create, read, update, delete) operations. |
| NewsDao | Interface | Extends the JpaRepository interface, communicates with the MySql database, and manages the News entity's CRUD (create, read, update, delete) operations. |
| IUserDal | Interface | Declares the methods related to the User (getUserById, addUser, getAllUsers etc.), that are going to be fully implemented in UserService class. |
| IPropertyDal | Interface | Declares the methods related to the Property (getPropertyById, addProperty, getAllPropertiies etc.), that are going to be fully implemented in PropertyService class. |
| IAddressDal | Interface | Declares the methods related to the Address (get AddressById, add Address, getAll Addresses etc.), that are going to be fully implemented in AddressService class. |
| INewsDal | Interface | Declares the methods related to the News (get NewsById, add News, getAllNews etc.), that are going to be fully implemented in NewsService class. |
| UserDalJPA | Class | Implements the IUserDal interface, making use of the UserDao interface, which directly communicates with the database. |
| PropertyDalJPA | Class | Implements the IPropertyDal interface, making use of the PropertyDao interface, which directly communicates with the database. |
| AddressDalJPA | Class | Implements the IAddressDal interface, making use of the AddressDao interface, which directly communicates with the database. |
| NewsDalJPA | Class | Implements the INewsDal interface, making use of the NewsDao interface, which directly communicates with the database. |
| IPropertyService | Interface | Declares the methods related to the Property, that are going to be used in PropertyService class. |
| IAddressService | Interface | Declares the methods related to the Address, that are going to be used in AddressService class. |
| IUserService | Interface | Declares the methods related to the User, that are going to be used in UserService class. |
| INewsService | Interface | Declares the methods related to the News, that are going to be used in NewsService class. |
| UserService | Service Class | Implements the IUserService interface. Uses a IUserDal variable. |
| PropertyService | Service Class | Implements the IPropertyService interface. Uses a IPropertyDal variable. |
| AddressService | Service Class | Implements the IAddressService interface. Uses a IAddressDal variable. |
| NewsService | Service Class | Implements the INewsService interface. Uses a INewsDal variable. |
| UserController | Controller Class | All incoming REST API calls for the User object are processed. |
| PropertyController | Controller Class | All incoming REST API calls for the Property object are processed. |
| AddressController | Controller Class | All incoming REST API calls for the Address object are processed. |
| NewsController | Controller Class | All incoming REST API calls for the News object are processed. |

# user acceptance tests/ test cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Pre-condition | Steps | Test data | Expected result | Actual result |
| TC- 01 | Create an account as an user | The frontend and the backend need to be running. The user navigates to the sign-up page. | 1.Complete all the required fields.  2.Set a password.  3.Press on the „Register” button. | First name: Ana Maria  Last name: Nitu  Username: anamarianitu  Password:  Password123! | The user will receive a „Successfully registered ” message. The user is now able to log in. |  |
| TC-02 | Log in as a user | The frontend and the backend need to be running. The user needs to have an account. The user navigates to the login page. | 1. Enter the username or email.  2.Enter the password.  3.Press on the “Log In” button. | Username: anamarianitu  Password:  Password123! | The user will receive a „Successfully logged in ” message. The user has now access to her/him profile page appeared on the navbar. |  |
| TC-03 | Log in as an admin | The frontend and the backend need to be running. The admin needs to have an account with correct credentials. The admin navigates to the login page dedicated to the admin login. | 1. Enter the username or email.  2.Enter the password.  3.Press on the “Log In” button. | Username: anamarianitu\_admin  Password:  Password123! | The admin will receive a „Successfully logged in ” message. The admin has now access to the system. |  |
| TC-04 | Log in as an agent | The frontend and the backend need to be running. The agent needs to have an account with correct credentials. The agent navigates to the login page dedicated to the agent login. | 1. Enter the username or email.  2.Enter the password.  3.Press on the “Log In” button. | Username: anamarianitu\_agent  Password:  Password123! | The agent will receive a „Successfully logged in ” message. The agent has now access to the system. |  |
| TC-05 | Log in with incorrect credentials | The frontend and the backend need to be running. The user needs to have an account. The user navigates to the login page. | 1. Enter the username or email.  2.Enter the password.  3.Press on the “Log In” button. | Actual username: anamarianitu  Username used to log in:  anamaria  Password:  Password123!  (or any other combination of wrong credentials) | The user will receive a red warning message „Wrong credentials. Try again! ”. The user has the possibility to try to log in again. |  |
| TC-06 | Search properties based on the name | The frontend and the backend need to be running. The user can have access to the feature with/without an account. The user navigates to the „Properties” page. | 1.Navigate to the properties page.  2.Start typing (the title/part of the title) on the Search bar. | Example of search:  Strationsweg Apart | The system automatically filters the properties in order to show the user only the needed one/s. |  |
| TC-07 | View more information about a certain property | The frontend and the backend need to be running. The user can have access to the feature with/without an account. The user will find properties displayed on both the home page or the „Properties” page. | 1.In both cases, the user needs to press the “View more” button from that certain property. | Selecting the first property on the „Properties” page and clicking „View more” for that one | The user will be redirected to the page where there are all the details about the property. There, the user can see a full overview of the information regarding the selected property. |  |
| TC-08 | Submit a form to schedule a viewing for a property | The frontend and the backend need to be running. The user can have access to the feature with/without an account. The user will find properties displayed on both the home page or the „Properties” page. | 1.Enter the name  2.Enter the e-mail  3.Enter the message  4.Enter a desired date for the viewing  5.Enter a preferred moment of the day for the viewing(morning)  6.Enter a phone number | Name:  Bianca Onea  E-mail:  [anamaria@gmail.com](mailto:anamaria@gmail.com)  Message:  “I would like to schedule a viewing for this apartment. It s perfect for my family”  Date:  14.11.2021  Time:  Afternoon  Phone:  234567890 | The user receives a message after completing the form (Successfully sent!), the data is then sent to the agent that manages the property in order to contact the clients. |  |
| TC – 09 | Create a new property | The frontend and the backend need to be running. The admin needs to be successfully logged in. The admin navigates to the properties page and clicks on the “Create” button. | 1.Press “Create new property” button.  2.Complete all the required fields (title, description, price, rent/sale, address, agent in charge, etc.).  3.Press on the “Create” button after the admin inputs the data. | Title:  Spacious apartment in Eindhoven Center  Description:  This apartment is perfect for a family, great location etc.  Type:  Rent  Price:  1500$/month  (all the needed fields and photos of the property) | The admin receives a “Successfully created” message, and it can be visible on the list of properties and, if available, it will be displayed on the website for the users. |  |
| TC - 10 | Update profile details as a user | The frontend and the backend need to be running. The user needs to be successfully logged in to have access to this feature. The user navigates to the “My profile” page. | 1.Navigate to “My profile” page.  2.Click “Edit profile”.  3.Complete the necessary fields (name, city, country, property type interest etc.)  4.Click “Edit” after completing the fields. | Current name:  Jane Doe  New name:  Jane Smith  Current interest:  Apartment  New interest:  House  Current city:  Eindhoven  New city:  Rotterdam  Etc. | After completing the fields and clicking “Edit”, the user will receive a “Successfully edited” message and now, the user is able to view the updated data on his/her profile page. |  |
| TC - 11 | Update an existing property | The frontend and the backend need to be running. The admin needs to be successfully logged in. The admin navigates to the properties page where there is a list of all the properties. Selecting one, the admin will press on the “Update” button. | 1.Press “Update property” button.  2.Change the needed fields (title, description, price, rent/sale, address, agent in charge, etc.).  3.Press on the “Update” button after the admin inputs the new data. | Current title:  Spacious apartment in Eindhoven Center  New title:  Luxurious apartment Strationsweg  Current price:  1500$/month  New price:  1400$/month | The admin receives a “Successfully updated” message, and it can be visible on the list of properties and, if available, it will be displayed on the website for the users. |  |
| TC - 12 | Delete an existing property | The frontend and the backend need to be running. The admin needs to be successfully logged in. The admin navigates to the properties page where there is a list of all the properties. The admin can select a specific one and press the “Delete” button. | 1.Press “Delete property” button.  2.The admin receives a pop-up to confirm the action.  3.Press “Confirm”. | - | The admin receives a “Successfully deleted” message, and it can no longer be visible on the list of properties and on the website. |  |
| TC - 13 | Send a request to modify a property (as an agent to an admin) | The frontend and the backend need to be running. The agent needs to be successfully logged in. The agent navigates to the “Requests” page and presses the “Create request” button. The agent needs to complete the form in order to be sent to the admin to change it. | 1.Press “Create new Request” button.  1.Complete the form with the (username, property to be changed, action (update, delete), message etc.)  3.After completing the form, press “Send request” button. | Username:  John\_agent  Property:  Select a property from the select bar  Action:  Update  Message:  The property is rented, it needs to be modified as not available. | After successfully completing the form, the request is sent to the admin, that needs to take the specified action. The admin has access to all the requests from the agents on the “Requests ” page. |  |
| TC-14 | Filter the properties as a user | The frontend and the backend need to be running. The user can have access to the feature with/without an account. The user navigates to the „Properties” page. | 1.Navigate to the properties page.  2.Check the specific checkboxes for filters. | Checkbox “Rent”  Checkbox price between 1000-2000$  Checkbox “Apartment” | After checking the needed fields, the user is able to view only the properties which meet the requirements. |  |
| TC-15 | View all the registered users from the website | The frontend and the backend need to be running. The admin needs to be successfully logged in. The admin can navigate to the “Users” page. | 1.Navigate to the “Users” page. | - | The admin is able to see an overview of all the registered users on the website. |  |
| TC-16 | Create an agent | The frontend and the backend need to be running. The admin needs to be successfully logged in. The admin can navigate to the “Agents” page. The admin needs to press on the “Create new agent” button. | 1.Press “Create new agent” button.  2.Complete all the required fields (name, username, job position, salary, city etc.).  3.Press on the “Create” button after the admin inputs the data. | First name:  Anne  Last name:  Hathaway  Email:  Anne.h@gmail.com  Phone:  456789  Username:  Annehathaway\_agent  City:  Einfhoven  Job Position:  senior | After completing all the required fields and pressing “Create”, the admin receives a “Successfully created” message and he/she is able to view the new agent on the list of all the agents. |  |