Insurance Aggregator

Web application project

Table of Contents

[1 Introduction 3](#_Toc515524393)

[1.1 Purpose 3](#_Toc515524394)

[1.2 Project Overview 3](#_Toc515524395)

[2 Scope 3](#_Toc515524396)

[2.1 In-Scope 3](#_Toc515524397)

[2.2 Out-of-Scope 3](#_Toc515524398)

[3 Testing Strategy 3](#_Toc515524399)

[3.1 Test Objectives 3](#_Toc515524400)

[3.2 Test Assumptions 3](#_Toc515524401)

[3.3 Data Approach 3](#_Toc515524402)

[3.4 Level of Testing 3](#_Toc515524403)

[3.5 Unit Testing 4](#_Toc515524404)

[3.6 Functional Testing 4](#_Toc515524405)

[3.7 User Acceptance Testing 4](#_Toc515524406)

[3.8 Regression Testing 5](#_Toc515524407)

[4 Execution Strategy 5](#_Toc515524408)

[4.1 Entry Criteria 5](#_Toc515524409)

[4.2 Exit criteria 6](#_Toc515524410)

[4.3 Validation and Defect Management 6](#_Toc515524411)

[5 Environment Requirements 7](#_Toc515524412)

[5.1 Test Environments 7](#_Toc515524413)

[6 Significantly Impacted Division/College/Department 7](#_Toc515524414)

[7 Dependencies 7](#_Toc515524415)

# Introduction

## Purpose

The Purpose of this test plan is to ensure that the Insurance aggregator web application meets the user specifications and requirements. By conducting testing across various dimensions, we aim to identify and rectify any issues or defects, thereby enhancing the overall user experience and promoting user satisfaction. This test plan is to ensure that reliability, functionality, security, and performance of the insurance aggregator web application. The organization can identify and report issues, minimize risks for delivering a high-quality application to the user by following the test plan.

## Project Overview

The Insurance aggregator aims to provide users with a centralized platform to compare, choose, and purchase insurance policies from various providers. This application will cover a range of insurance types, including health insurance, auto insurance, home insurance, and travel insurance. It will facilitate the comparison of policy features, premiums, and terms from multiple insurance providers. It provides a user-friendly platform for comparing insurance policies and enable users to make informed decisions by presenting detailed policy information.

# Scope

## In-Scope

1. User registration and authentication.

2. Insurance policy search and comparison.

* + Search functionality
  + Comparison tools.

3. Policy details and information.

* + Policy description
  + Reviews and ratings

4. Quote generation and purchase

5. User dashboard

* + Policy management
  + Claim submission

6.Notifications and alerts

7.Admin Panel

8.Policy management

9.Admin modules

10.Reports

## 2.2 Out-of-Scope

# Testing Strategy

## Test Objectives

To ensure that the insurance aggregator web application meets the specified requirements by providing a centralized platform to compare, choose, and purchase insurance policies from various providers by validating the reliability, functionality, security, and performance. Identify and rectify any defects or issues present in the application for the defect management. It aims to establish deliverables, identify test tasks and responsibilities, outline the test environment and configuration, and define the test schedule to ensure efficient and effective testing.

## Test Assumptions

**Availability of test resources and tools:**

The test resources and tools for the testing are assumed to be functional and available.

**Availability of Test Environment and test data:**

The test environment includes the hardware, software and the web browsers are assumed to be available and functional and also the test data availability for the successful completion of the testing.

## Data Approach

Test data are provided for the positive and negative test cases(valid data, invalid data) for each test scenarios and ensure that the all functionalities and features which require test data are added in the test case document.

## Level of Testing

*List the types of testing to be performed*.

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Responsible Parties** |
| **Smoke testing** | Smoke testing, also called build verification testing or confidence testing, is a software testing method that is used to determine if a new software build is ready for the next testing phase.Its primary purpose is to quickly determine whether the most critical functionalities of an application are working as expected after a new build or a release. It helps the testing team for deciding whether the further testing can be done or not. **This testing method determines if the most crucial functions of a prograve into finer details.** | **QA** |
| Functional testing | Unit testing is a software testing technique where individual units or components of a software application are tested independently in isolation from the rest of the system.  Integration testing is a software testing technique where multiple units or components of a software application are combined and tested as a group to evaluate their interactions and interfaces.  System testing is a software testing technique where the entire software application is tested as a complete and integrated system to verify its compliance with specified requirements and functionality.  Regression testing is performed to find out whether the updates or changes had caused new defects in the existing functions. It is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made.  Usability testing points to operating a product (software or hardware) and services by testing them on the customer’s/end-users/consumers’ side. The primary aim of this testing is to check that the product becomes easy to use for the customers.  User Acceptance Testing (UAT) is a technique where the application is tested by end users or stakeholders to determine whether it meets their needs and requirements. |  |
| Non functional testing | Non-Functional Testing is a type of testing used to evaluate a software application’s performance, usability, dependability, and other non-functional characteristics. Non-Functional testing is essential for confirming the software’s reliability and functionality. The Software Requirements Specification (SRS) serves as the basis for this software testing method, which enables quality assurance teams to check if the system complies with user requirements.  Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload. Organizations will run performance tests to identify performance-related bottlenecks. JMeter tool is used for performance testing.  The system’s loading capability is tested during load testing. The system can handle increasing simultaneous users because of its loading capacity and there are another types of non-performance testing like security, reliability, etc. |  |

## Smoke Testing

Smoke testing is done of the most critical functionalities of an application are working as expected after a new build or a release. It comprise the testing of,

The user should be able to login to the home page, dashboard should be visible and quote should be to the user, search option should be available, user should be able to search and get the policies details and description, should be available,

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| MMMounika M |  | Test MaQAQA |
|  |  | Test Lead |
|  |  | Test Analyst |

## Functional Testing

Features to be tested in functional testing,

1. User registration and authentication.

2. Insurance policy search and comparison.

* + Search functionality
  + Comparison tools.

3. Policy details and information.

* + Policy description
  + Reviews and ratings

4. Quote generation and purchase

5. User dashboard

* + Policy management
  + Claim submission

6.Notifications and alerts

7.Admin Panel

8.Policy management

9.Admin modules

10.Reports

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| MMounika M |  | Test QA |
|  |  | Test Lead |
|  |  | Test Analyst |

## User Acceptance Testing

# Execution Strategy

## Entry Criteria

* The entry criteria refer to the desirable conditions to start test execution.
* Entry criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions, and provide a recommendation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Test Team** | **Technical Team** | **Notes** |
| *Test environment(s) is available* |  |  |  |
| *Test data is available* |  |  |  |
| *Code has been merged successfully* |  |  |  |
| *Development has completed unit testing* |  |  |  |
| *Test scripts are completed, reviewed and approved by the Project Team* |  |  |  |

## Exit criteria

* The exit criteria are the desirable conditions that need to be met in order proceed with the implementation.
* Exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Exit Criteria** | **Test Team** | **Technical Team** | **Notes** |
| *100% Test Scripts executed* |  |  |  |
| *90% pass rate of Test Scripts* |  |  |  |
| *No open Critical and High severity defects* |  |  |  |
| *All remaining defects are either cancelled or documented as Change Requests for a future release* |  |  |  |
| *All expected and actual results are captured and documented with the test script* |  |  |  |
| *All test metrics collected based on reports from daily and Weekly Status reports* |  |  |  |
| *All defects logged in Defect Tracker/Spreadsheet* |  |  |  |
| *Test environment cleanup completed and a new back up of the environment* |  |  |  |

## Validation and Defect Management

* Specify how test cases/test scenarios should be validated
* Specify how defect should be managed
  + It is expected that the testers execute all the scripts in each of the cycles described above.
  + The defects will be tracked through Defect Tracker or Spreadsheet.
  + It is the responsibility of the tester to open the defects, retest and close the defect.

Defects found during the Testing should be categorized as below:

|  |  |
| --- | --- |
| **Severity** | **Impact** |
| 1 (Critical) | * Functionality is blocked and no testing can proceed * Application/program/feature is unusable in the current state |
| 2 (High) | * Functionality is not usable and there is no workaround but testing can proceed |
| 3 (Medium) | * Functionality issues but there is workaround for achieving the desired functionality |
| 4 (Low) | * Unclear error message or cosmetic error which has minimum impact on product use. |

# Environment Requirements

## Test Environments

* Manual testing - Web browsers (Chrome, Edge, Firefox) .
* Automation testing:
* Functional testing using Selenium
* Performance testing would be conducted using JMeter
* API Testing using Postman/SOAPUI
* Test management and defect tracking- Jira