Insurance Aggregator

Web application project

Table of Contents

[1 Introduction 3](#_Toc160561160)

[1.1 Purpose 3](#_Toc160561161)

[1.2 Project Overview 3](#_Toc160561162)

[2 Scope 3](#_Toc160561163)

[2.1 In-Scope 3](#_Toc160561164)

[2.2 Out-of-Scope 3](#_Toc160561165)

[3 Testing Strategy 4](#_Toc160561166)

[3.1 Test Objectives 4](#_Toc160561167)

[3.2 Test Assumptions 4](#_Toc160561168)

[3.3 Data Approach 4](#_Toc160561169)

[3.4 Level of Testing 4](#_Toc160561170)

[3.5 Smoke Testing 6](#_Toc160561171)

[3.6 Functional Testing 6](#_Toc160561172)

[3.6.1 Integration Testing 7](#_Toc160561173)

[3.6.2 System Testing 7](#_Toc160561174)

[3.6.3 Regression Testing 7](#_Toc160561175)

[3.6.4 User acceptance Testing 8](#_Toc160561176)

[4 Execution Strategy 8](#_Toc160561177)

[4.1 Entry Criteria 8](#_Toc160561178)

[4.2 Exit criteria 9](#_Toc160561179)

[4.3 Validation and Defect Management 9](#_Toc160561180)

[5 Environment Requirements 10](#_Toc160561181)

[5.1 Test Environments 10](#_Toc160561182)

# 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

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

* + Reviews and ratings

# 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 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 [e.g., proper email format, valid date range], invalid data exceeding length limits, empty or missing inputs) for each test scenarios and ensure that all functionalities and features which require test data are added in the test case document.

## Level of Testing

The types of testing performed as follows,

|  |  |  |
| --- | --- | --- |
| **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.**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. To Check whether the website is stable for testing Checking the main functionality of the website like,

The user should be able to login to the home page, dashboard should be visible, quote should be displayable to the user, search option should be available, user should be able to search and get the policies details and description and should verify all the features should be displayable and functionable in the UI.

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| Mounika M | As a User | Test QA Tester |

## Functional Testing

Functional testing checks whether each feature’s functionality is working according to the customer requirements. 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** |
| Mounika M | As a User | Test QA tester |

## 3.6.1 Integration Testing

Integration testing is known as the second level of the software testing process, following unit testing. the interface between two software units or modules It focuses on determining the correctness of the interface The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed.

Check the integration between the login page and home page, between home page and dashboard, between user and the policy providers and between the policies and purchase of policy using payment gateway.

## 3.6.2 System Testing

System testing, also referred to as system-level testing or system integration testing, is the process in which how the various components of an application interact together in the full, integrated system or application. It verifies that an application performs tasks as designed. It checks that every kind of user input produces the intended output across the application. System testing is the third level of testing in the software development process.

Checks all the integration modules are working together as user can login and navigate to the homepage and access the dashboard features and user can search the policies from the policy holders for the policy features, terms and premiums, User can navigate to all the sections of different insurance types and choose the policy, users can generate the reports for each policies, users can navigate to the payment gateway for the specify policy purchase and check the notifications and alerts are sent to the specified users based on the policy then the reviews and ratings are available for specific policies.

## 3.6.3 Regression Testing

Regression testing is conducted after a code update to ensure that the update introduced no new bugs. This is because new code may bring in new logic that conflicts with the existing code, leading to defects. It is performed after a new build or update of a feature.

This testing should be done to check whether all the features in insurance aggregator are working after an update or a release.

This check features like user registration and login, insurance policy and search functionality, different policies in dashboard, quote generation, policy details and comparison, policy purchase using payment gateway, notification and alerts, report generation, reviews and ratings.

## 3.6.4 User acceptance Testing

The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.

User Acceptance testing is done by the end user perspective by the customers.

User checks the insurance aggregator which is used to compare, choose, and purchase insurance policies from various providers as per the requirements.

# Execution Strategy

Execution strategy provides Execution the test cases according to the defined test strategy and test plan. Utilize appropriate tools and techniques for executing manual and automated tests. Monitor test execution progress, document test results, and capture relevant metrics and performance data.

## Entry Criteria

The entry criteria refer to the desirable conditions to start test execution. The entry criteria are,

* The requirement document should be available.
* Complete understanding of the application flow is required.
* The Test Plan Document should be ready.
* The application should be developed.
* Test environment and test data should be available.

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. The exit criteria are as follows,

* Verification of all the features and test cases are tested and executed.
* Ensuring application is built and there is no defects/bugs.
* When the application is built as per the user requirements.

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 (Jira).
  + 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

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured. Test environment is configured as per the need of the Application Under Test.

Setting up a right test environment ensures software testing success. Any flaws in this process may lead to extra cost and time to the client.

## 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

## 