Test Strategy Document for Insurance Aggregator Application

The test strategy for the Insurance Aggregator project outlines the approach, scope, and methodologies for ensuring the quality, reliability, and functionality of the platform. This document provides an overview of the testing process, including the key components to be tested, testing techniques, tools, and resources required.

**1.Scope and Overview:**

The web-based insurance aggregator application will cover a range of insurance types, including but not limited to health insurance, auto insurance, home insurance, and travel insurance. It will facilitate the comparison of policy features, premiums, and terms from multiple insurance providers. The application aims to provide users with a centralized platform to compare, choose, and purchase insurance policies from various providers.

Insurance aggregator application has the following types of testing,

Functional testing:

Functional testing checks whether each feature’s functionality is working according to the customer requirements. Functional testing checks whether the insurance aggregator behaves as per the user requirements and checks the all the functionality of the features are working fine with no issues.

Smoke testing will be the first stage for the testing to check all the main features and functionalities are working. The insurance aggregator follows various functional testing types like Unit testing, Integration testing, System testing and Regression testing.

Non-functional testing:

Non-functional testing is a type of software testing that verifies non-functional aspects of the product, such as performance, stability, and usability. Whereas functional testing verifies whether the product does what it is supposed to, non-functional testing verifies how well the product performs.

**2.Testing Approach:**

The test approaches are the test strategy implementations of the insurance aggregator which follows the combination of manual and automation testing.

**2.1 Functional testing:**

Functional testing in used to check the functionality of the insurance aggregator. It checks all the features are working as per their functionality. Functional testing process is as follows,

To test whether the user can register and login to the insurance aggregator-> navigate to the home page -> and they can use the dropdowns in the dashboard -> user can search for the policies or they can directly navigate the specific insurance policy(like auto, bike or car insurance) from the dashboard -> they can compare and choose the policies from the policy providers -> user can purchase policy using the payment gateway -> checks the reports are generated for the policies-> checks the user can access the reports-> checks the user can get the notification and alerts for their policies.

**2.2 Non-Functional testing:**

**2.2.1 Performance testing:**

Performance testing is a testing measure that evaluates the speed, responsiveness, and stability of an insurance aggregator under a workload. This testing can help to identify potential bottlenecks and issues that can impact the user experience of the insurance aggregator application. It should also test the response time and scalability of the application.

**2.2.2 Security testing:**

Security Testing is the process of verifying the security of an insurance aggregator application. This can include testing for vulnerabilities as well as verifying the security of the payment process. This testing is done for securing data and data storage of the application.

**2.2.3 Compliance testing:**

Compliance testing, also known as conformance testing, is a type of testing to determine whether an insurance aggregator meets a defined set of internal or external standards before it's released into production. Compliance testing is to check the regulatory requirements of the insurance aggregator application.

**2.2.4 Usability:**

Usability is the testing which is used to test the intuitive interface for the user and user can access the features as their needs.Usability testing is the process of testing how easy it is for users to compare and choose the policies and purchase policies. This includes testing the usability of the website and the purchasing policies. The goal of usability testing is to make sure that users can easily and successfully complete their policy purchase in the insurance aggregator application.

Mobile responsiveness testing is also done in usability to Ensure a seamless experience on various devices, including smartphones and tablets.

**3.Test Scenarios:**

**3.1 User Registration and Authentication:**

* Verify successful registration with valid user details.
* Validate login with correct credentials.
* Test authentication failures with incorrect credentials.
* OTP verification for User Authentication.
* User can navigate to home page after logging in.

**3.2 Insurance Policy Search and Comparison:**

* Verify the search functionality in the home page.
* Ensure accurate search results based on the user inputs.
* Validate comparison of policies based on coverage, premiums.
  1. **Policy Details and Information**:
* Ensure accurate display of policy details.
* Verify completeness and correctness of information.

**3.4 Reviews and Ratings:**

* Test display of reviews and ratings for the policies.
* Ensure accuracy of displayed ratings for the policies.

**3.5 Quote Generation and Purchase:**

* Validate the accurate generation of insurance quotes.
* Test smooth transition from quote to the purchase policy.

**3.6 User Dashboard and Profile Management:**

* Ensure proper display of user dashboard with relevant information.
* Verify functionality of profile editing and updating and the display of the different insurances.

**3.7 Policy Management:**

* Test addition, modification, and deletion of policies for the different insurance policies.
* Validate policy renewal processes.
  1. **Claim Submission:**
* Test submission of insurance claims for the different policies.
* Verify documentation upload and processing for the policies.

**3.9 Notification and Alerts:**

* Ensure users receive notifications for relevant policies.
* Test alerts for the policies that user purchase.

**3.10 Admin Panel and Modules:**

* Validate admin login and add, modify, and update policies.
* Test functionalities for policy management, user management.

**3.11 Reports:**

* Verify generation and accuracy of reports for the policies like user on boarding reports, policy renewal report, claim processing reports, coverage details and benefits.

**4. Test execution and Monitoring:**

* Execute all test cases according to the test plan for the application.
* Monitor test progress, identify bottlenecks, and adapt strategies if necessary for the insurance aggregator application.

**5. Test environment and tools:**

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.

* 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

**6.Release control:**

To ensure proper and successful test execution, release management plans should be created thoroughly. Furthermore, by setting up the build management process, one can get information on when, where, and how the new build should be made available and deployed. The idea is to create and release a product that is viable – something usable that can be released within the mandated deadline and meets end-user needs to a substantial extent. Invariably, the software will require changes over time, and end-users are aware of this.

Awareness of the coverage of your test cases also makes it easy to predict post-release issues. Inadequately tested features are almost always guaranteed to cause problems.

90% of the Test Cases should be covered in the Stable every high priority defect should be fixed, and it should not affect the user to use the website.

It is wise to expect that some kind of issue will crop up once the software is being accessed by the end-users. Have a mechanism in place that can address points of malfunction quickly and effectively. Without doing so, issue resolution becomes a long-drawn, effort-intensive process that wastes developers’ time and fosters unsatisfactory user experience.

However, the need for documentation does not end with product release. User feedback needs to be collected so that the product can be improved accordingly. The process of implementing every change needs to be documented so that any flaws in execution can be identified and addressed quickly.

Ensure that the support team is aware of any features that may raise questions. Set them up for success by giving them as much relevant information as possible.

**7. Risk analysis:**

Risk analysis is the process of identifying and assessing potential risks and uncertainties associated with software. It involves evaluating the likelihood and impact of various risks to determine how they might affect the insurance aggregator application’s quality, schedule, and budget. The primary objectives of risk analysis in testing are to prioritize risks, develop mitigation strategies, and make informed decisions to ensure the successful delivery of a high-quality application. The risks that can be prevented in the application are,

**7.1 Functional risks:**

Functional risks can lead to a poor user experience and can cause potential users to abandon your site. Thorough testing can help identify any potential functional errors so they can be fixed before launch.

**7.2 Performance risks:**

Performance risks occur due to the high traffic volumes, causes lack of performance, slow response time and downtime of the application. It should involve proper scalability planning of the application to overcome the error.

**7.3 Scalability risks:**

Scalability risks occurs when the application has poor performance under the peak loads. It is prevented by scalability testing to assess performance bottlenecks, and capacity planning for the application to work under the increased user demand during peak periods.

**7.4 Security risks:**

The application deals with the sensitive user information such as personal details and payment data, making it vulnerable to cyberattacks. It is prevented by following proper encryption techniques, regular security audits, and compliance with data protection regulations.

**7.5 Compatibility risks:**

Compatibility issue occurs when the application is not supported for the different browsers and devices The insurance aggregator needs to be compatible with all major browsers and devices to prevent risks. Testing can help identify any potential compatibility issues so that they can be fixed before launch.

**8. Reviews and approvals:**

When all these activities are defined in the test strategy Test plan, they need to be reviewed for sign-off by all entities involved in project management, business team, development team, and system administration team.

A summary of the review changes should be tracked at the beginning of the document along with the approver’s name, date, and comment. Also, it’s a living document meaning this should be continuously reviewed and updated with testing process enhancement.

Approver:

Date:

Comment: