WIPRO JAVA SELENIUM – BATCH-10, TEAM-06

TEST PLAN FOR

<<ACKO INSURANCE APP>>

Change Log

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made

1 INTRODUCTION 3

1.1 SCOPE 3

1.1.1 In Scope 3

1.1.2 Out of Scope 3

1.2 QUALITY OBJECTIVE 4

1.3 ROLES AND RESPONSIBILITIES 4

2 TEST METHODOLOGY 5

2.1 OVERVIEW 5

2.2 TEST LEVELS 5

2.3 BUG TRIAGE 5

2.4 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS 6

2.5 TEST COMPLETENESS 6

3 TEST DELIVERABLES 7

4 RESOURCE & ENVIRONMENT NEEDS 8

4.1 TESTING TOOLS 8

4.2 TEST ENVIRONMENT 9

5 TERMS/ACRONYMS 11

1 Introduction

The purpose of this document is to define the test strategy for validating the functionality, reliability, usability, and performance of the Acko Insurance website. Acko is a fully digital insurance provider, and its web platform acts as a core business system to allow users to browse, compare, purchase, and manage insurance products. Given the nature of the business, thorough testing is critical to ensure compliance, accuracy of calculations, seamless user experience, and secure handling of sensitive data.

The testing process will follow Agile methodology with iterations and sprints. Both manual and automated testing methods will be applied to ensure early detection of defects, continuous feedback to developers, and timely releases. Testing will be aligned with the SDLC lifecycle and integrated into CI/CD pipelines for faster delivery and high-quality outputs.

This strategy defines the framework for all testing activities including functional, non-functional, integration, UI/UX validation, API testing, regression, smoke, and user acceptance testing (UAT).

1.1 Scope

1.1.1 In Scope

The following areas are covered as part of the testing activities for Acko:

- User registration and login functionalities
- Insurance product listing (Health, Car, Bike, etc.)
- Quote generation and comparison tools
- Policy purchase flow
- Policy renewal and cancellation
- Claims initiation and status tracking
- Payment gateway integration
- Customer dashboard and account management
- Responsive design testing (mobile, tablet, desktop)
- API testing for internal and third-party services
- Performance and security validation

1.1.2 Out of Scope

- Backend infrastructure testing (unless it directly impacts functionality)
- Load testing beyond agreed user volume
- Third-party systems integration outside Acko’s control
- Localization/Internationalization testing (Acko is India-focused)
- Detailed cross-browser testing on obsolete browsers

1.2 Quality Objectives

The primary objectives of testing the Acko platform are:

- To ensure the **Application Under Test (AUT)** complies with defined functional and non-functional requirements.
- To validate that the AUT meets **client expectations** for usability, speed, and reliability.
- To identify and resolve critical defects **prior to production deployment**.
- To ensure **regression stability** of core modules with every release cycle.
- To support **continuous integration/continuous delivery (CI/CD)** practices with automated regression suites.

1.3 Roles and Responsibilities

Role	**Responsibilities**
QA Analyst	- Prepare and execute test cases - Report defects - Participate in reviews
Test Manager	- Define test strategy and plan - Manage test resources - Track progress
Configuration Manager	- Maintain version control - Manage test environments and baselines
Developers	- Fix reported bugs - Participate in unit and integration testing
Installation Team	- Deploy builds to QA/Prod environments - Support environment-related issues
Automation Testers	- Develop and maintain automation test scripts - Integrate with CI tools
Business Analyst	- Clarify requirements - Support validation of business flows
Product Owner	- Prioritize features and bug fixes

	- Approve test sign-offs

2 Test Methodology

2.1 Overview – Reason for Adopting a Test Methodology

The selection of a test methodology depends on multiple factors including the project size, complexity, timeline, stakeholder involvement, and risk tolerance. Common methodologies include:

- Waterfall: A linear, sequential approach where testing is conducted after complete development. Suitable for projects with well-understood requirements and minimal expected changes.
- Iterative: Development and testing happen in repeated cycles, allowing incremental improvements and regular feedback.
- Agile: Emphasizes flexibility, continuous integration, and collaboration. Testing is integrated within short sprint cycles to enable rapid delivery and early defect detection.
- Extreme Programming (XP): A form of Agile with strong focus on technical excellence, continuous testing, and customer feedback.

For this project, Agile methodology is adopted due to the dynamic nature of insurance market requirements, frequent feature updates, and the need for rapid feedback cycles. Agile allows prioritizing features based on business value, continuous integration of new changes, and early risk detection. It supports automation and iterative testing, which are critical for sustaining quality in continuous delivery environments.

2.2 Test Levels

Test levels define the stages and types of testing to ensure comprehensive quality checks at various points of the software delivery cycle. The selection and scope of test levels depend on factors such as project scope, time, resources, and budget.

Typical test levels for this project:

- Unit Testing: Performed by developers to validate individual components or functions.
- Integration Testing: Verifies the interaction between modules and third-party integrations (e.g., payment gateways, APIs).
- System Testing: End-to-end testing of the complete integrated system to validate functional and non-functional requirements.
- User Acceptance Testing (UAT): Final validation by business stakeholders or actual users to confirm the system meets business needs.

Depending on constraints, exploratory and regression testing are integrated at relevant stages to ensure ongoing product stability and discovery of edge-case defects.

2.3 Bug Triage

The bug triage process ensures efficient handling of defects discovered during testing by:

- Defining Bug Resolution Types: Each bug is analyzed and assigned a resolution, such as Fixed, Won't Fix, Deferred, or Duplicate.
- Prioritization: Bugs are categorized by severity (Critical, High, Medium, Low) and impact on business functions.
- Scheduling Fixes: Establish timelines for fixing bugs, aligning fixes with sprint releases or hotfix priorities, and deciding which bugs can be deferred.

Goal: To maintain a focused and prioritized defect backlog that ensures critical bugs are fixed timely, preventing delays or risks to release schedules.

2.4 Suspension Criteria and Resumption Requirements

- Suspension Criteria: Define situations where testing must be halted temporarily, such as:
  - Critical environment outages or instability.
  - Major application crashes or blocker bugs preventing further testing.
  - Unavailability of test data or necessary resources.
  - Incomplete or unstable application build.
- Resumption Requirements: Conditions that must be met for testing to continue include:
  - Resolution or workaround of blocking issues.
  - Restoration of testing environments and availability of test data.
  - Approval from QA Lead and Project Manager after verifying readiness.

This ensures testing resources focus on productive testing and helps avoid time waste due to external, unresolved issues.

2.5 Test Completeness Criteria

Testing is considered complete when predefined criteria are satisfied, such as:

- 100% Test Coverage: All identified test cases aligned with requirements and user stories have been executed.
- Execution of All Manual and Automated Tests: Both manual exploratory and automated regression suites have run with acceptable results.
- Bug Resolution: All open critical and high-severity defects are fixed, retested, and closed. Remaining low-priority defects are documented with decisions to fix later.
- Sign-offs: Formal approvals from product owners and QA stakeholders confirming that the system meets agreed quality standards and is ready for release.
- Non-Functional Checks: Performance, security, and usability tests have met the acceptance criteria without major issues.

3 Test Deliverables

This document outlines the detailed test deliverables that will be provided throughout the testing lifecycle for the insurance application developed for www.acko.com. These deliverables ensure structured planning, quality execution, defect tracking, and final approval of the tested product.

Deliverable	Description	Relevance to www.acko.com
Test Plan	Defines the scope, resources, schedule, and approach of testing.	Ensures clear objectives for testing modules like policy purchase, claim initiation, and renewals.
Test Cases	Step-by-step scenarios to validate application behavior.	Covers user flows like quote generation, registration, policy payment, and support ticketing.
Requirement Traceability Matrix (RTM)	Maps requirements to test cases to ensure coverage.	Ensures all product requirements (UI, logic, APIs) are fully validated.
Bug Reports	Logs of issues identified during testing with severity and status.	Tracks issues such as payment gateway failures, form errors, and integration bugs.
Test Strategy	High-level approach including tools, environment, and types of testing.	Specifies use of Selenium for UI and Postman for API validation.
Test Metrics	Quantitative data such as test coverage, defect density, and pass/fail ratio..	Measures readiness for release and highlights areas needing improvement..

Customer Sign Off	Formal approval from the client post testing.	Confirms that all major workflows of Acko's insurance platform meet expected quality standards.

4 RESOURCE & ENVIRONMENT NEEDS

4.1 TESTING TOOLS

4.1.1 Requirements Tracking Tools:

Tool	Type	Purpose	User Purpose
Jira	Agile Requirements & Issue Tracker	Track epics, user stories, requirements, tasks	Business Analyst, PO
Confluence	Documentation & Collaboration Space	Document and manage requirement specs	Business Analyst, Dev

4.1.2.Bug Tracking Tools:

Tool	Type	Purpose	User Purpose
Jira Issues	Integrated Bug & Requirement Tracker	Report and track defects, link to stories	QA Engineers, Developers
MantisBT	Standalone Bug Tracking System	Track and manage defects independently	QA Team

4.1.3 Automation Tools:

Tool	Type	Purpose	User Purpose
Selenium + TestNG	Open Source	Cross-browser UI automation for regression testing	Frontend Dev + QA
Postman + Newman	Open Source	API Testing of policy creation, claims, premium calc	Frontend Dev + QA
Cypress (Optional)	Open Source	API Testing of policy	Frontend Dev + QA

		creation, claims, premium calc	

4.2 Test Environment

Test Environment Requirements

A. Hardware

- **Servers:**
  - **Minimal**: 4 GB RAM, 1 CPU core, SSD storage (≥ 30 GB)
  - **Recommended**: 8 GB RAM, 4 CPU cores (Intel Core i5/E5-class or equivalent), fast SSD storage (≥ 100 GB)

- **Test Workstations:**
  - Modern desktop or laptop with at least 8–16 GB RAM
  - Dual-core CPU (i5 or equivalent), SSD
  - Screens with resolution 1280×1024 or higher

B. Software

- **Operating Systems:**
  - Windows 8 or newer (Windows 10/11)—for desktops and test workstations
  - Server OS (e.g. Windows Server 2012+, Linux variants) for staging environments
- **Office & Productivity Tools:**
  - Microsoft Office 2013 or above for documentation
- **Email Client:**
  - Microsoft Exchange client—for validating notification workflows
- **Browsers :**
  - Latest stable versions of Chrome, Firefox, and Edge
- **Supporting Software:**
  - Relevant runtime environments, such as .NET Framework, Java JDKs, or middleware, to mirror production configuration

C. Network & Infrastructure

- Dedicated local or virtual network setup that closely resembles production network topology, including firewalls, DNS, VPN, and load balancers
- Stable internet access; bandwidth and latency similar to expected production usage

D. Setup Process & Validation Workflow

1. **Design Environment**: Define hardware/software specs, required tools and data.
2. **Provision and Configure**: Build staging servers, install operating systems and browsers, set up Exchange clients, etc.
3. **Access Control**: Assign testers and administrators to the environment.
4. **Smoke Testing**: Execute initial sanity tests to ensure environment stability and connectivity
5. **Maintain Environment**: Regular updates, version control, issue logging, and availability monitoring

5 Terms / Acronyms

Here's a detailed **Terms/Acronyms** section tailored for your **Test Strategy Document** for Acko Insurance under the **Agile methodology**:

TERM / ACRONYM	DEFINITION
API	Application Programming Interface – A set of protocols and tools for building and interacting with software applications. Used extensively for Acko's backend integrations.
AUT	Application Under Test – Refers to the Acko Insurance website or mobile app currently being tested.
UI	User Interface – The front-end interface where users interact with the Acko platform.
UX	User Experience – Overall user satisfaction and usability of Acko's digital offerings.
IRDAI	Insurance Regulatory and Development Authority of India – The regulatory body for the Indian insurance industry.
POM	Page Object Model – A test automation design pattern used in Selenium for UI testing.
TC	Test Case – A set of steps executed to verify a specific functionality.
TDD	Test Driven Development – A software development process where tests are written before the code.
BDD	Behavior Driven Development – Development approach that encourages collaboration between QA, developers, and business stakeholders.

UAT	User Acceptance Testing – Final testing phase to ensure the application meets business requirements.
SLA	Service Level Agreement – A contract defining the level of service expected from a vendor or API.
JIRA	A project management and issue tracking tool used for Agile teams.
CI/CD	Continuous Integration / Continuous Deployment – Practice of automating code integration and delivery to ensure faster, safer releases.
RTM	Requirements Traceability Matrix – A document mapping requirements to their corresponding test cases.
SRS	Software Requirements Specification – A document that captures functional and non-functional requirements.
BRD	Business Requirements Document – A formal document that describes the business needs and requirements.
STLC	Software Testing Life Cycle – A sequence of specific activities conducted during testing.
SDLC	Software Development Life Cycle – A framework that defines tasks performed at each stage in the software development process.
AGILE	A flexible, iterative software development methodology promoting continuous collaboration and testing.
DEFECT / BUG	An issue in the system that causes it to behave unexpectedly or incorrectly.