**WIPRO JAVA SELENIUM - BATCH -10, TEAM-06**

**TEST STATERGY FOR ACKO INSURANCE APP**

**Table of Contents**

[**1. Scope 3**](#_41jwjofdirc6)

[**2. Test Approach 4**](#_90dl9x4nrbca)

[**3. Test Environment 5**](#_x5ldyu6mzqpy)

[**4. Testing Tools 6**](#_n6hg68a23scy)

[**5. Release Control 9**](#_7tl6dncujxke)

[**6. Risk Analysis 11**](#_3f3rbb34jxcj)

[**7. Review and Approvals 14**](#_isl3nwhxgm4v)

# Scope

## Document Review & Approval:

Reviews are conducted by Business, QA, or Dev teams early in the sprint.

The Product Owner or Scrum Master gives final approval based on defined criteria and completeness of test executions per sprint.

|  |  |  |
| --- | --- | --- |
| **Role** | **Team/Person** | **Responsibility** |
| Document Author | Project / QA Team | Drafts initial version |
| Reviewers | Business Analysts, Test Leads, Dev Engineers | Provide feedback during sprint refinement |
| Approver(s) | Product Owner & Scrum Master | Final sign-off before sprint or release |

**Testing Activities & Timelines**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Sprint Timing** | **Responsible Role** |
| Document Draft & Stakeholder Review | Sprint Planning (Day1–2) | Author & Reviewers |
| Test Case Design | Early sprint (Day 2–3 ) | QA Team |
| Test Execution & Validation | Mid sprint (Day 5–10) | QA Engineers |
| Defect Triage & Closure | Ongoing (Day 5–12) | QA + Development |
| Final Review & Approval | End of sprint (Day 12–14) | Product Owner, Scrum Master |

# Test Approach

The Test Approach outlines the process of testing, levels, techniques, tools, and environments used. For AKKO Insurance:

* Multi-level Testing:  
  Conduct unit, integration, system, and user acceptance testing, each ensuring separately that code, features, and business processes work as intended.
* Agile Process:  
  Testing closely follows development in sprints, with test case design, execution, and defect triage happening in parallel and in short cycles. Regular reviews and approvals are built into each sprint.
* Waterfall Process:  
  Testing is a distinct, planned phase after development, with all test artifacts prepared and reviewed before execution, and strict phase gates for each activity (design, execution, triage).
* Test Automation:  
  Leverage tools like Selenium/TestNG for regression, Postman for API tests, JMeter for performance, and security tools (OWASP ZAP/Burp Suite) for vulnerability scanning.
* Environment Coverage:  
  Test on multiple operating systems (Windows, macOS), browsers, and devices to ensure broad compatibility.
* Data Management:  
  Use dummy and production-like data securely; backup daily and have restore strategies.
* Collaboration & Review:  
  Regular planning and review meetings with Dev, QA, Product, and DevOps teams ensure that test coverage matches business priorities, including for regulatory compliance and partner integrations.
* Reporting & Approval:  
  Use tools (e.g., Jira, Xray, Allure Reports) for test management, execution tracking, and transparent reporting. Final sign-off is given after all critical flows are validated and defects addressed.

This approach ensures the insurance site is robust, secure, compliant, and user-friendly across rapid feature changes or phased releases, with clear ownership and risk mitigation at each step.

# Test Environment

A test environment is a setup of software, hardware, and tools used to run tests and simulate real-world conditions.

**Operating Systems**

Testing will be conducted on Windows 10/11 and macOS to ensure compatibility across platforms.

**Browsers:**

The web app will be tested on Chrome, Firefox, Edge, and Safari to validate consistent performance.

**Devices:**

Testing will be done on desktop and laptop devices. If mobile access is supported, responsive testing will be performed on Android and iOS devices.

**Test Tools:**

Selenium WebDriver will be used for UI automation. Postman will handle API testing. Test execution/reporting will utilize TestNG or JUnit. Jira will manage bug tracking and test documentation.

**Test Data:**

Dummy insurance user accounts, policy details, and claim data will be used during testing. Data will be backed up daily and handled securely.

**Backup & Restore Strategy:**

Daily database snapshots will be created to allow rollback in case of test data loss or corruption.

Staging Environment:

A dedicated staging server that mirrors production will be used to avoid affecting the live environment.

**Internet Speed**

Testing will be conducted on connections ranging from 2 Mbps to 100 Mbps to simulate different user network conditions.

# Testing Tool

The following testing tools are proposed and planned for use in the testing lifecycle of the Acko Insurance Web Application to ensure quality, maintainability, scalability, and performance

## 4.1 Test Management Tools

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Type** | **Purpose** | **User Support** |
| Jira + Xray | Commercial | Test case management, execution tracking, defect logging, traceability | Up to 15 QA + Dev users |
| TestRail (Alt.) | Commercial | Requirement mapping, test suites, and run tracking | Based on license |

## Usage Activity:

## - Manage user stories, epics, and test scenarios for modules like Auto Insurance, Health Claims, etc. -Defect tracking linked to development sprints and real-time dashboards for test coverage.

## 4.2 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 creation, claims, premium calc | Frontend Dev + QA |

**Usage Activity:**

- Automate high-priority flows such as login, policy generation, claims submission, and renewal journeys.  
 - Postman tests for quote generation, pricing logic, customer details, and payment APIs.  
 - Continuous Integration with Jenkins to run automated suites nightly.

## 4.3 Performance & Load Testing Tools

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Type** | **Purpose** | **User Support** |
| JMeter | Open Source | Load testing for API/Backend endpoints | 1-2 Performance QAs |
| BlazeMeter (Alt) | Commercial | Advanced simulation with cloud users | Optional |

**Usage Activity:**

-Simulate 1000+ concurrent users for quote engine and payment endpoints.  
 - Identify slow API responses under load and optimize backend microservices.

## 4.4 Security Testing Tools

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Type** | **Purpose** | **User Support** |
| OWASP ZAP | Open Source | Vulnerability scanning, XSS, SQLi detection | 1-2 Security Testers |
| Burp Suite | Commercial | Vulnerability scanning, XSS, SQLi detection | Security Team |

**Usage Activity:**

-Regular scans of the staging environment before every production release.  
 - Test for broken access controls, token manipulation, data exposure.

**4.5 CI/CD & Reporting Tools**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Type** | **Purpose** | **User Purpose** |
| Jenkins | Open Source | CI pipeline integration, test automation runner | DevOps + QA Integration |
| Allure Reports | Open Source | Test result reporting and visual dashboards | Automation Team |
| Slack + Webhooks | Open Source | Notification of test failures | Entire Team |

**Usage Activity:**

-Jenkins triggers UI/API tests after every code commit to staging.   
 -Reports sent automatically via Slack to relevant QA and Product channels.  
 - Dashboard monitoring for daily test execution trends.

# Release Control

### 1. Release Planning

* Conduct a planning meeting with Dev, QA, Product, and DevOps teams.
* Finalize features, enhancements, and bug fixes for the release.
* Assign roles: Release Manager, QA Lead, DevOps Engineer, UAT Coordinator.
* Scheduled employment window (preferably during off-peak hours).

### 2. Pre-Release Activities

* Ensure **Code Freeze** post-QA sign-off.
* Generate a Release Candidate build from the tagged version in version control (e.g., v1.0.0).
* Perform a successful deployment on the UAT environment.
* Obtain UAT sign-off from stakeholders.
* Prepare a backup of the production database and environment.
* Verify that the rollback plan is in place and has been tested.

### 3. Release Execution

* Deploy to production using a CI/CD pipeline or controlled manual deployment.
* Perform **smoke testing** on the live environment post-deployment.
* Monitor logs and real-time alerts immediately after deployment.
* Update version number and metadata on the website (e.g., footer/versioning API).
* Mark the release status as “Live” after smoke test approval.

### 4. Post-Release Monitoring

* Monitor system health using tools like Prometheus, Grafana, or ELK.
* Track critical business flows (policy purchase, claim initiation).

* Watch for performance degradation or user-reported issues.
* Maintain a 24–48 hour stabilization window with on-call support.

### 5. Rollback Control

* Rollback if:
* Production smoke tests fail
* High-priority bugs or downtime occur
* Compliance/security concerns are detected

* Rollback Steps:
* Trigger CI/CD rollback to last stable release (e.g., v0.9.9).
* Restore production DB from backup.
* Validate core flows.
* Communicate rollback status to stakeholders and log Root Cause Analysis (RCA).

### 6. Release Documentation (Mandatory)

* Release Notes
* Deployment Checklist
* UAT Sign-off
* Backup Confirmation
* Rollback Plan
* Post-release Summary

# Risk Analysis

## Risk Analysis for Acko Insurance Website

### 1. Regulatory Compliance & Legal Risks

Recent findings show that Acko was fined ₹1 crore by the IRDAI due to using unregistered intermediaries in policy sales. This introduces a risk that parts of the insurance sales flow (e.g. partner integrations, API-based offers) may not comply with regulatory expectations.

**Mitigation**: Validate that key flows—policy purchase, quoting, agent workflows—comply with IRDAI rules. Coordinate with legal/compliance teams to test and confirm correct behavior under policy-related transactions.

### 2. Trust & Security Concerns

Users may be reluctant to share personal and sensitive data (e.g. health, vehicle information), especially given concerns about AI visibility in underwriting and policy issuance

**Mitigation**: Perform rigorous security testing (OWASP ZAP or Burp). Focus on encryption of PII, secure session handling, input validation, and information disclosure. Also ensure clear privacy disclosures and minimal data retention.

### 3. High Dependency on AI/ML-driven Services

Acko’s platform leverages AI/ML—for example, personalized pricing, chatbots, and fraud detection—potentially introducing nondeterministic behavior in the UI flows.

**Mitigation**: Include test scenarios with dynamic input variations. Validate deterministic outputs for given inputs. Use regression tests on AI-dependent features to detect unexpected back-end changes.

### 4. Complexity of Multi-Policy and Group Offerings

Acko offers group medical coverage, multiple-policy bundling, employer-based GMC, and add-on endorsements. This increases business rule complexity and potential edge-case gaps.

**Mitigation**: Develop detailed business-rule test scenarios, cross-policy flows, and endorsement/cancellation handling. Ensure coverage of combined policies, premium calculations, and composite workflows.

### 5. Frequent Mobile and UI App Updates

Given past reengineering efforts with performance improvements (e.g. migrating from React Native to Flutter), UI and layout changes may result in flaky automation or visual mismatch.

**Mitigation**: Maintain robust locators in automation, use Page Object Model, and include visual regression testing (e.g. screenshot comparisons), especially for policy purchase journeys and dashboards.

### 6. Performance & Scalability under Load

Being India’s direct digital insurer, Acko services large volumes and TPAs generate high concurrent load. The platform must scale reliably under usage spikes.

**Mitigation**: Conduct performance tests (load, stress, spike) on key flows (quote generation, checkout, claim status, policy retrieval). Monitor response times and error rates under simulated traffic patterns.

### 7. Customer Experience & Usability Risks

Trust and simplicity are critical in insurtech. Reports note mixed TrustPilot reviews citing poor claim-handling experiences and hidden premium charges.

**Mitigation**: Conduct usability testing across flows such as claim filing, policy renewal, and quote cancellation. Include exploratory testing with real users to detect friction in UI, policy language clarity, or navigation transparency.

### 8. Third-party and Partner Integrations

Acko integrates with Amazon, Ola, partner agents, labs, pharmacies, telemedicine providers, and HRMS platforms. Any integration failure or misconfiguration may break workflows.  
 **Mitigation**: Create mock endpoints for partner services. Test failure modes: timeouts, invalid data, missing responses. Validate backward compatibility and graceful fallback behavior.

### Top 5 Critical Risks Summary

|  |  |  |
| --- | --- | --- |
| **Risk** | **Potential Impact** | **Mitigation Strategy** |
| Regulatory Non‑Compliance | Legal penalties, application rollbacks | Compliance checkpoints; legal-approved test scenarios |
| AI-driven Decision Variability | Inconsistent quotes, miscommunication | Control inputs; regression on AI-dependent flows |
| Complex Multi-Policy Logic | Policy mispricing, incorrect coverage bundles | Business-rule-driven testing, scenario coverage matrix |
| UI Automation Breakage | Frequent false failures, maintenance overhead | Robust locators, visual regression testing |
| Partner Integration Failures | Broken flows via Amazon, labs, chatbots | Mock services, failure scenario testing |

### Agile-Specific Mitigations

* Prioritize risk-based testing: address critical business flows in each sprint.
* Maintain automation coverage for key policy and purchase flows using risk-based prioritization tools (e.g. QA Touch). Schedule exploratory test charters focused on claim, renewal, and onboarding.
* Collaborate with Dev, Compliance, and Product teams to map test scenarios around dynamic AI-based decisioning and integrations.

# Review and Approvals

### 1. Document Details

* **Project**: Acko General Insurance Web Application
* **Document Title**: Test Strategy Document
* **Version**: 1.0
* **Date**: [Insert Date]
* **Prepared By**: [QA Lead / Test Manager Name]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Role** | **Department** | **Review Status** | **Comments** |
| [Dev Lead Name] | Development Lead | Engineering | ✔ Approved | Reviewed technical design |
| [QA Lead Name] | Quality Assurance Lead | QA | ✔ Approved | Validated test coverage |
| [Product Owner Name] | Product Owner | Product | ✔ Approved | Business scope verified |
| [Security Analyst] | Security and Compliance | Security/Legal | ✔ Approved | Reviewed risk mitigation |
| [UAT Coordinator] | UAT Manager | Product/QA | ✔ Approved | Verified UAT completion |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Designation** | **Signature** | **Date** | **Comments** |
| [Test Manager Name] | QA/Test Manager | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [DD-MM-YYYY] | Approved for release |
| [Engineering Manager] | Engineering Head | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [DD-MM-YYYY] | Architecture verified |
| [Product Manager] | Product Head | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [DD-MM-YYYY] | Feature complete |
| [Compliance Officer] | Legal & Compliance | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | [DD-MM-YYYY] | Meets IRDAI norms |