QA Engineer Technical Test

# Part I - UI : Make an analysis of the Ornikar insurance website: https://www.ornikar.com/assurance-auto

### Analysis

This part of the website is dedicated to auto insurance services. Ornikar offers car insurance solutions tailored to young drivers and those buying their first car.

On this page, Ornikar details various insurance plans, from extensive coverage to basic third-party liability with additional options based on the driver's needs. The information highlights the flexibility and financial accessibility of these plans, allowing users to seelct coverage that fits their budget and needs. The page also highlights the ease of obtaining a quote online and managing the insurance through a digital platform.

Additionally, Ornikar integrates its insurance services with its driving school offerings, making it a complete solution for new drivers who are learning to drive and need insurance for their first car.

### Strategy

#### The tests I would perform first:

##### Manual Testing

**Main Functionality:** I would test the primary function of the page which is to allow users to obtain an insurance quote. Initial manual tests should ensure that the form accurately collects all information provided by the user and generates a precise quote after being processed.

**Form Validation:** Form validation is crucial, particularly for fields related to vehicle information. Automated tests should cover various scenarios, including as examples : incorrect data formats, required but empty fields, invalid license formats, and missing start dates for coverage.

**API Testing:** Ensure the accuracy of quote calculations and retrieval processes by testing the APIs. Verify that there are no API failures and that the responses are correct.

**Dynamic Elements:** Verify that all dynamic elements (e.g., links, buttons) redirect users to the appropriate pages. This ensures stability and helps prevent future development from breaking these functionalities.

**Performance Testing:** Perform tests to measure page load times and responsiveness, especially during quote generation. Monitoring these metrics ensures the page remains performant under various conditions.

##### Automation Testing

**Regression Testing:** First automation tests to be performed should be regression tests on all aspects of the form and user interface to ensure that new changes do not introduce bugs or regressions.

**End-to-End Testing:** Implementation of an end-to)end use case that will verify a user journey, from entering the page and filling out the form to generating and retrieving a quote. This verifies that all integrated components work together as expected.

**Security & Compliance Testing:** Test data protection measures, such as GDPR compliance, to ensure user information is securely handled and meets regulatory requirements.

#### The points I would address in a 3-amigos meeting

Our main goal is ensure that the page is functional, user-friendly, and secure, we should focus on the following key points:

* **Main Functionality:** Focus on testing the insurance quote flow to verify that the form works correctly, generates accurate quotes, and validates inputs effectively.
* **User Experience:** Test responsiveness across various devices and assess performance to guarantee a smooth user experience
* **Integration:** Verify robust error handling and effective backend communication to ensure that all components work together as expected
* **Automation:** Prioritizing automated regression testing and end-to-end scenarios to continuously validate that the page functions as expected and integrates well.
* **Data Protection:** Ensure that personal data entered by users is securely handled and protected.
* **GDPR Compliance:** Ensure the site complies with GDPR regulations, especially regarding user consent and data handling, Test for proper cookie consent mechanisms, data access requests, and clear opt-ins.

### Testing/Automation Strategy

#### Objectives

Goals and purposes of testing:

* **Validate Main Functionality:** Verify that the page’s main functionality, including form submission and quote generation, works correctly and meets user requirements.
* **Validate User Experience:** Confirm that the page provides a smooth, intuitive user experience across various devices and browsers.
* **Assess Performance:** Ensure that the page loads quickly and performs well under expected and peak loads.
* **Verify Security:** Test data protection measures and compliance with relevant regulations to safeguard user information.
* **Validate Integration:** Validate that the page integrates as expected with backend systems.

#### Scope

What will be tested**:**

* **Insurance Quote Form:** Main functionality, the validation of the form and processing of user input data.
* **User Interface:** Responsiveness and usability across different devices and browsers.
* **Performance:** Page load times and the responsiveness of the page in different situations (pour internet connection, different devices).
* **Backend Integration:** API calls, error handling..

#### Approach

General approach and different types of testing**:**

* **Functional Testing:** Test main functionalities such as form inputs, quote generation, and result.
* **User Acceptance Testing**: Validate that the page meets the expectations and requirements set by the product team.
* **Performance Testing:** Evaluate page load times and performance.
* **Usability Testing:** Verify the user experience - responsiveness, usability, and accessibility.
* **Compatibility Testing**: Ensuring the software works across different devices, operating systems and browsers.
* **Integration Testing:** Verify that the page integrates correctly with backend APIs and handles data as expected.
* **Security Testing:** Test for vulnerabilities and ensure compliance with data protection standards.
* **Automated Regression Testing:** Write and run automated test cases using chosen tools to ensure that new changes don’t break existing functionality.
* **Continuous Integration Testing:** Automated tests run to be part of a CI/CD pipeline to provide quick feedback on code changes.
* **End-to-EndTesting:** Automated testing simulating a real user scenario to ensure all components work together as expected.

#### Tools

* **Test Plan**: Xray/TestRail/Zephyr/Document
* **Bug Management Tool:** Jira
* **API Testing:** Postman
* **Automation Technical Stack**: Nightwatch, Typescript

#### Resources

* **Test Environments:** Environment used for testing purposes (staging, UAT).
* **Test Data:** Pre-defined sets of test data covering various scenarios (e.g. for the form inputs).
* **Team:** Test engineers, developers, and project managers to execute testing activities.

#### Roles and responsibilities

* **Test Engineers:** Design, execute, and maintain automated and manual test cases, writing test plan, report and track defects.
* **Developers:** Address and fix issues identified during testing and if applicable collaborate on test case design.
* **Project Managers:** Oversee the testing process, ensure alignment with project goals, and manage schedules and resources.

#### Risk management

The time it takes for testing to be reassessed depends on the defects or any edge cases found during testing that were not overseen before the test campaign started.

* Ensure detailed test planning and coverage to address all critical functionalities and scenarios; Testing duration to be reassessed depending on the defects or any edge cases found during testing that were not overseen before the test campaign started.

##### 

#### Metrics and reporting

##### Measuring progress and results:

* **Test Coverage:** Track the percentage of features and functionalities tested.
* **Defect Metrics:** Monitor the number, severity and priority of defects found, as well as the time taken to resolve them.
* **Performance Metrics:** Measure page load times and create a dedicated space to track any changes.
* **Test Execution:** Report on the number of test cases executed, passed, and failed - usually found in a test plan created before the test campaign started.

##### Reporting:

* **Test Reports:** Provide detailed reports on test results, including pass/fail status and defect logs if applicable.
* **Dashboards:** Use dashboards to check key metrics and progress.
* **Updates:** Regularly update manager/PO on testing progress, issues, and risks that may be discovered before or during the test campaign.

##### Automation Strategy

###### Regression testing:

### Objective: Ensure that new changes don’t break existing functionality .

### Actions: Develop a suite of automated regression tests covering critical functions like form submissions, quote generation, and data processing.

### Automation: Integrate these tests into the CI pipeline to verify for regressions early in the process.

###### End-to-End testing:

### Objective: Validate the entire user flow from accessing the page to receiving a quote.

### Actions: Simulate and automate a real user scenario, including form completion, quote result, and handling different input conditions.

### Automation: Automate these scenarios to ensure continuous validation of user flows.

### 

# Part II - API : Make an analysis of the Ornikar insurance API

### Analysis

An API is a tool that allows different systems to communicate and share information.

For example, our API endpoint *https://insurance-api.ornikar.com/api/v2/vehicles/license-plate/<plate>* allows an user to automatically retrieve his vehicle details just by entering their plate number into our form. The API sends a request to our database or external sources, taking the relevant vehicle details, returning them and filling automatically required inputs into the form.

Not requiring the user to manually input his data we speed up the process and reduce the risk of errors.

### Strategy

#### Manual Testing

* **Response Validation**: Verify that the API correctly retrieves and returns the expected data based on each endpoint (e.g entire list of vehicle brands/models; the expected model, year , vehicle information based on a license plate).
* **Input Validation**: When the API endpoint requires an input validate with different entries such as empty or invalid input, wrong license plate.
* **Different Response Codes**: Validate that the API endpoints return the expected HTTP status codes (e.g 200 for success, 404 for not found..)
* **Response Time**: Measure and monitor the response times to requests to ensure they are in the expected limit range.
* **External System Integration**: Validate that the API interactions with external sources is accurate (e.g government/police databases when retrieving vehicles details based on license plates)
* **Error Messages**: Validate that in case of an invalid input (license plate, empty input) the user receives a message that explains the exact issue.

##### Automation

* Automate the functional test cases to ensure no updates are breaking the existing endpoints (Response Validation, Input Validation, Response Codes, Error Messages)
* Integrate automated tests into CI/CD pipelines and setup automated monitoring and alerts for the team

#### 3. Testing/Automation Strategy

##### Objectives

The goal of this testing and automation strategy is to ensure an accurate, reliable and performant API. Validate the functional requirements, the response status and the integration with external and internal sources.

The focus on automation is to create automated core test cases to ensure good quality and enable efficient regression testing.

##### Scope

What will be tested**:**

* **Functional Testing:** Response Validation, Input Validation, Response Codes, Error Messages.
* **Non Functional Testing:** Performance and load testing of the API endpoints.
* **Integration:** Integration of the API with different external and internal sources.
* **Automation** : Automate key functional test cases

##### Approach

General approach and different types of testing**:**

* **Functional Testing:**
  + Perform manual testing to ensure the accuracy of the API - vali, invalid, empty inputs.
  + Automation: Automate key functional cases and ensure an automated monitoring and report of the tests.
* **Performance Testing**:
  + Load testingby simulating a high volume of requests.
  + Response time- monitor the response time of API to ensure performance is in the expected range.
* **Integration Testing**: Test the integration with different external and internal systems.
* Continuous Integration: Integrate the automated tests into the pipeline CI/CD and set up a monitoring and alert to notify the team.

##### Tools

* **Test Plan**: Xray/TestRail/Zephyr/Document
* **Bug Management Tool:** Jira
* **API Testing:** Postman
* **Automation Technical Stack**: Nightwatch, Typescript

##### Resources

* **Test Environments:** Environment used for testing purposes (staging, UAT).
* **Test Data:** Pre-defined sets of test data covering various scenarios (e.g. for the form inputs).
* **Team:** Test engineers, developers, and project managers to execute testing activities.

##### Roles and responsibilities

* **Test Engineers:** Design, execute, and maintain automated and manual test cases, writing test plan, report and track defects.
* **Developers:** Address and fix issues identified during testing and if applicable collaborate on test case design.
* **Project Managers:** Oversee the testing process, ensure alignment with project goals, and manage schedules and resources.

##### Risk management

The time it takes for testing to be reassessed depends on the defects or any edge cases found during testing that were not overseen before the test campaign started.

* Ensure detailed test planning and coverage to address all critical functionalities and scenarios; Testing duration to be reassessed depending on the defects or any edge cases found during testing that were not overseen before the test campaign started.

##### 

##### Metrics and reporting

###### Measuring progress and results:

* **Test Coverage:** Track the percentage of features and functionalities tested.
* **Defect Metrics:** Monitor the number, severity and priority of defects found, as well as the time taken to resolve them.
* **Performance Metrics:** Measure page load times and create a dedicated space to track any changes.
* **Test Execution:** Report on the number of test cases executed, passed, and failed - usually found in a test plan created before the test campaign started.

###### Reporting:

* **Test Reports:** Provide detailed reports on test results, including pass/fail status and defect logs if applicable.
* **Dashboards:** Use dashboards to check key metrics and progress.
* **Updates:** Regularly update manager/PO on testing progress, issues, and risks that may be discovered before or during the test campaign.

##### Automation Strategy

###### Regression testing:

### Objective: Ensure that new changes don’t break existing functionality .

### Actions: Develop a suite of automated regression tests covering critical functions like form submissions, quote generation, and data processing.

### Automation: Integrate these tests into the CI pipeline to verify for regressions early in the process.

###### End-to-End testing:

### Objective: Validate the entire user flow from accessing the page to receiving a quote.

### Actions: Simulate and automate a real user scenario, including form completion, quote result, and handling different input conditions.

### Automation: Automate these scenarios to ensure continuous validation of user flows.

# NOTES:

* I used Nightwatch.js with TypeScript for automation testing. Although this was my first experience with Nightwatch, I found the setup and configuration process to be straightforward and user-friendly. However, implementing the Page Object Model (POM) introduced some complexity. Despite this, I have made an effort to adhere to best practices within the limited time available.
* While writing automated test cases, I encountered challenges in selecting the most reliable selectors for various page elements, particularly in the absence of unique IDs. In some instances, I resorted to using text-based selectors, which I am aware is not considered best practice.