**Quality Assurance Test Plan**

**(QATP)**

**For**

Google Translate

**1. Introduction**

**1.1 Purpose**

**1.2 Scope**

**1.3 Overview**

**1.4 Definitions, Acronyms, and Abbreviations**

**1.5 References**

**1.6 Product Overview**

**1.7 Product Risks**

**1.8 Interfacing Systems**

**1.9 Test Coverage**

**2. Test Deliverables and Schedule**

**3. Test Design**

**4. Test Automation**

**5. Test Execution**

**5.1 Approach**

**5.2 Test Organization**

**5.3 Test Methodology**

**5.4 Test Data Requirements**

**5.4 Test Data Requirements**

**5.5 Test Reporting**

**5.6 Defect Management**

**5.7 Triage Meetings**

**5.8 Guidelines for Entering QA Remarks**

**5.8.1 Remark Types**

**5.8.2 Priority and Severity Guidelines**

**5.8.3 Remark Status**

**6. Test Team**

**6.1 QA Team Organization**

**6.2 Roles and Responsibilities**

**7. Test Environment**

**7.1 Software and Hardware**

**7.2 Tools and Accelerators**

**8. Assumptions**

**1. Introduction**

**1.1 Purpose**

The purpose of this test plan is to define the testing strategy, scope, and processes required to ensure the Google Translate web application accurately translates sentences or words between languages, specifically focusing on English, Sinhala, and Tamil.

**1.2 Scope**

This test plan covers:

* Functional testing of the translation feature between English, Sinhala, and Tamil.
* User interface (UI) testing to ensure proper functioning of buttons, dropdowns, and other UI components.
* Testing character limit validation (e.g., 5,000 characters).
* Testing across different browsers (Chrome, Firefox, Edge) and devices (desktop and mobile).
* Negative testing for invalid inputs.

**1.3 Overview**

Google Translate is an online application that allows users to input words or sentences and translate them into multiple languages. This QA test plan focuses on validating its functionality between English, Sinhala, and Tamil languages. Key areas include UI, translation accuracy, and error handling.

**1.4 Definitions, Acronyms, and Abbreviations**

* **QA**: Quality Assurance
* **UI**: User Interface
* **API**: Application Programming Interface
* **Functional Testing**: Testing the application’s features for correctness.
* **Non-functional Testing**: Testing aspects like performance and usability.

**1.5 References**

1. [Google Translate Product Documentation](https://translate.google.com/)
2. Internal development specifications for Google Translate.

**1.6 Product Overview**

Google Translate is a web-based application that provides translation services for text between multiple languages. It offers additional functionalities, such as detecting the input language, swapping translation directions, and limiting input text to 5,000 characters.

**1.7 Product Risks**

* Translation inaccuracies between certain language pairs.
* Browser compatibility issues affecting the user interface.
* Potential delays or failure in providing translation due to high server load or poor network connection.

**1.8 Interfacing Systems**

* **Google Translate API**: The system interacts with Google's API to retrieve translation results.
* **Browsers**: Chrome, Firefox, Edge (cross-browser testing needed).

**1.9 Test Coverage**

Testing will cover the following areas:

* Functional: Ensuring translations between English, Sinhala, and Tamil work as expected.
* UI: Verifying proper display and functioning of dropdowns, buttons, and input fields.
* Boundary Testing: Verifying character limits (5,000-character limit).
* Error Handling: Testing how the system responds to invalid input.

**1.10 Functional Requirements**

* The system should allow users to input text for translation up to 5,000 characters.
* The system should accurately translate between English, Sinhala, and Tamil.
* The "Detect Language" feature should correctly identify the input language.
* The "Swap Languages" feature should switch the source and target languages seamlessly.

**1.11 Non-functional Requirements**

* **Performance**: The system should return translations within 3 seconds under normal conditions.
* **Compatibility**: The system should function correctly across all supported browsers and devices.
* **Usability**: The user interface should be simple and intuitive.

**1.12 Application Security Requirements**

* Data entered into the system should not be stored or cached on the client-side.
* Input sanitization should be enforced to prevent injection attacks or security vulnerabilities.

**1.13 Out of Scope**

* Testing translation accuracy between languages other than English, Sinhala, and Tamil.
* Performance testing under high server loads.

**2. Test Deliverables and Schedule**

* **Test Plan Document**: Outlining the scope, approach, and deliverables.
* **Test Cases**: Specific steps and expected results for each test scenario.
* **Test Summary Report**: Post-execution summary of test results.
* **Bug Reports**: Documentation of any defects found during testing.
* **Test Completion Report**: Final report summarizing testing activities.

**Schedule**:

* **Day 1**: Test planning and design.
* **Day 2**: Test execution.
* **Day 3**: Documentation of results and bug reporting.

**3. Test Design**

* Functional test cases will focus on verifying translation accuracy, character limits, and UI element functionality.
* Boundary test cases will check the system’s response when the character limit is exceeded.
* Negative test cases will focus on invalid inputs such as leaving fields empty.

**4. Test Automation**

* **Automation Tools**: Automation scripts will be developed using Selenium for regression testing of the UI.
* **Scope**: Core functionalities such as translation accuracy, dropdown selection, and character limits will be automated for frequent testing.

**5. Test Execution**

**5.1 Approach**

* **Manual Testing**: For translation accuracy and boundary cases.
* **Automated Testing**: For repetitive UI checks and regression testing.

**5.2 Test Organization**

* **QA Lead**: Responsible for planning and overseeing testing activities.
* **QA Engineers**: Responsible for executing test cases and reporting defects.

**5.3 Test Methodology**

* **Functional Testing**: Verifying that translation and related features work correctly.
* **UI Testing**: Ensuring that all visual components are functioning as expected.
* **Cross-browser Testing**: Testing on Chrome, Firefox, and Edge.

**5.4 Test Data Requirements**

* Sentences and words in English, Sinhala, and Tamil for translation tests.
* Data for boundary testing (e.g., inputs exceeding 5,000 characters).

**5.5 Test Reporting**

* Test case execution results.
* Daily status reports on testing progress.
* Defect reports with details on severity, priority, and steps to reproduce.

**5.6 Defect Management**

* Defects will be logged in the tracking system with severity and priority.
* Regular triage meetings will be held to prioritize and resolve defects.

**5.7 Triage Meetings**

* Weekly meetings to review, prioritize, and assign defect fixes.

**5.8 Guidelines for Entering QA Remarks**

**5.8.1 Remark Types**

* **Functional Defects**: Errors in the translation logic or other features.
* **UI Defects**: Issues related to layout or interface elements.

**5.8.2 Priority and Severity Guidelines**

* **Critical**: Prevents the application from functioning.
* **Major**: Affects functionality but has workarounds.
* **Minor**: Cosmetic issues that do not affect functionality.

**5.8.3 Remark Status**

* **Open**: Defect is reported but not yet addressed.
* **In Progress**: Defect is being worked on.
* **Closed**: Defect has been resolved.

**6. Test Team**

**6.1 QA Team Organization**

* **Test Lead**: Responsible for coordinating the testing efforts.
* **Test Engineers**: Execute test cases and report defects.

**6.2 Roles and Responsibilities**

* **QA Lead**: Plans and organizes the test effort.
* **Test Engineers**: Execute the tests and provide feedback.

**7. Test Environment**

**7.1 Software and Hardware**

* **Browsers**: Chrome, Firefox, Edge.
* **Devices**: Desktop, mobile devices.
* **Network**: Stable internet connection.

**7.2 Tools and Accelerators**

* **Selenium**: For automating repetitive test cases.
* **Jira**: For defect tracking and reporting.

**8. Assumptions**

* It is assumed that the product is stable and functional for testing.
* The QA team has access to necessary resources like test environments and automation tools.