Website Application - Testing assignment(Ankit Tiwari)

JUNE 2024

# 

# Introduction & Background The purpose of this document is to outline the strategy, scope, and tools for automating the testing of the "Add Person," "Contact Edit," and "List Creation" features within the Contacts module of the Kelp application. The goal is to ensure the quality and reliability of these functionalities through efficient and comprehensive automated testing.

#### **Module Overview**

The Contacts module consists of three primary functionalities:

1. **Add Person**: Allows users to add new contacts.
2. **Contact Edit**: Enables users to edit existing contact details.
3. **List Creation**: Facilitates the creation of lists of contacts, which can be shared internally.

# 

# Test Objectives

* **Validate Functionality:** Ensure that all features within the Contacts module work as intended, including form submissions, data persistence, UI interactions, and notifications.
* **Improve Efficiency:** Reduce manual testing effort and accelerate the feedback loop for developers.
* **Enhance Reliability:** Increase test coverage and consistency by automating repetitive and time-consuming test cases.
* **Ensure Regression:** Safeguard against introducing new defects while fixing existing ones or adding new features.

# 

# Automation Scope

# Form Validation:

# Ensure mandatory fields are correctly validated.

# Verify optional fields can be left empty without causing errors.

# Test validation messages for both successful and unsuccessful submissions.

# Check for correct behavior when fields exceed character limits or contain invalid data.

# UI Elements:

# Verify the presence and functionality of UI elements like buttons, dropdowns, radio buttons, and multi-select fields.

# Test the visibility and upload functionality of profile pictures.

# Confirm the display of initials when no profile picture is available.

# Validate the expand/collapse functionality of the "MORE FIELDS" section.

# Data Handling:

# Ensure data persistence for both adding and editing contacts.

# Verify the pre-filling of existing contact information during the edit process.

# Test data retrieval for address fields using the Google API and internal address search.

# User Interactions:

# Confirm correct behavior when selecting and setting primary contact methods (email/phone).

# Validate the approval/rejection flow for new contacts by admins.

# Check notification triggers for shared lists.

# Notifications and Toast Messages:

# Verify that toast messages appear correctly upon successful or failed actions.

# Confirm that notifications are sent to users when a list is shared.

# Permissions and Access Control:

# Ensure only authorized users can approve/reject contacts.

# Validate access restrictions for editing contacts and list creation.

# 

# 

# Tool Selection for Automation

Given the requirements and scope of the Contacts module, the following tools are recommended for automation testing:

**Selenium WebDriver with BDD Cucumber Framework**:

1.**Scope**: Selenium is suitable for end-to-end testing of web applications. It can automate form interactions, UI validations, and user workflows.The best part about using the Cucumber BDD framework are:

* Tests are first documented before being implemented.
* Tests are easy to understand for a user who doesn’t even know the functionality.
* It efficiently combines the automated tests having a living documentation and specifications that can be executed.
  + **Pros**:
    - Supports multiple programming languages (Java, C#, Python, etc.).
    - Integrates well with test frameworks like Cucumber, TestNG and JUnit.
    - Provides impeccable reporting system(Allure, Extent Report & Cucumber -JVM Reports)
    - Robust community support and extensive documentation.
  + **Cons**:
    - Requires coding knowledge.
    - May face challenges with handling dynamic elements and pop-ups.

1. **Postman (for API Testing)**:
   * **Scope**: Postman can be used for testing APIs, such as the Google API used for address fields.
   * **Pros**:
     + Easy to set up and use.
     + Excellent for testing RESTful APIs.
     + Can be integrated with CI/CD pipelines.
   * **Cons**:
     + Limited to API testing, not suitable for UI testing.
2. **Cypress**:
   * **Scope**: Cypress is a modern end-to-end testing framework that can handle complex user interactions.
   * **Pros**:
     + Real-Time Reloads- Cypress automatically reloads tests whenever you make changes to your test files, providing immediate feedback and improving development speed.
     + Time Travel- Cypress takes snapshots as your tests run, allowing you to see the state of your application at different points in time. This makes it easier to debug issues by reviewing the steps leading to a failure.
     + Automatic Waiting- Cypress automatically waits for elements to appear and for assertions to pass, reducing the occurrence of flaky tests due to timing issues.
     + Easy Setup and Configuration- Setting up Cypress is straightforward and typically requires minimal configuration, allowing developers to start writing tests quickly.
     + Real Browser Testing - Tests run directly in the browser, providing accurate and reliable end-to-end testing that mimics real user interactions.
   * **Cons**:
     + Limited to JavaScript(mostly uses built in JQuery functions).
     + Relatively new, so fewer integrations and plugins compared to Selenium.
3. **Version Controlling System (Git & GitHub)**:
   * **Scope**: Git enables version control, allowing teams to track changes and collaborate effectively on code. GitHub provides a platform to host Git repositories, facilitating collaboration, code review, and project management for software development.
4. **Jenkins (for CI/CD integration)**:
   * **Scope**: Jenkins can automate the running of test cases and integration into the development pipeline.
   * **Pros**:
     + Widely used CI/CD tool.
     + Supports a wide range of plugins.
     + Strong community support.
   * **Cons**:
     + Requires setup and configuration.
     + Can be complex for beginners.

# 

## Risks and Mitigation

* **Flaky Tests:** Use robust element locators and explicit waits in Selenium scripts.
* **Environment Instability:** Maintain a stable test environment with regular backups and monitoring.
* **Test Data Management Overhead:** Automate data generation and cleanup processes as much as possible.
* **Skillset Limitations:** Provide adequate training and support for the team to effectively use the chosen tools.

# 

## Conclusion

This automation test plan outlines a comprehensive approach to automate the testing of the Contacts module. By leveraging the right tools and frameworks, we can achieve greater efficiency, reliability, and confidence in the quality of the software.