**Desktop Test Plan**

This document outlines the test plan for the desktop application, which supports Windows, macOS, and Linux platforms. The focus is on achieving robust, automated testing through SikuliX and Postman, complemented by manual testing to cover edge cases and unanticipated scenarios.

**Goals**

1. **Ensure Cross-Platform Compatibility**
   * Validate the application’s functionality across Windows, macOS, and Linux.
2. **Automate Repetitive Tests**
   * Utilize SikuliX for UI-based tests and Postman for API validation.
3. **Identify and Resolve Critical Issues Early**
   * Catch and address platform-specific and cross-platform bugs before release.
4. **Maintain High Performance Standards**
   * Verify application responsiveness and resource usage on different platforms.
5. **Provide Reliable Test Evidence**
   * Document all test results with screenshots, logs, and reports.

**Testing Types and Workflow**

**1. SikuliX UI Tests**

* **Description:** SikuliX is a visual automation tool that uses image recognition to interact with UI elements. It is ideal for testing graphical interfaces.
* **Workflow:**
  1. Identify critical UI workflows (e.g., login, navigation, file operations).
  2. Create image-based scripts for interacting with the application.
  3. Run SikuliX scripts across Windows, macOS, and Linux.
  4. Capture screenshots of test execution and results.
* **Types of Tests:**
  1. **Smoke Tests:** Verify basic functionality like launching the app and navigating the main menu.
  2. **Regression Tests:** Ensure recent changes do not break existing features.
  3. **End-to-End Tests:** Validate complete workflows (e.g., user login, data processing, and export).
* **Advantages:**
  1. Platform-independent test scripts.
  2. Effective for visually verifying UI consistency.

**2. Unit Tests**

* **Description:** Validate the correctness of individual code components and logic in the application.
* **Tools:**
  + **TypeScript Code:** Use Jest, a JavaScript testing framework, for writing and running unit tests.
  + **Python Code:** Use pytest, a Python testing framework, for robust and efficient testing of Python modules.
* **Workflow:**
  1. Write test cases for individual functions, methods, and modules.
  2. Mock dependencies to isolate the code being tested.
  3. Execute tests locally and integrate them into the CI/CD pipeline.
  4. Analyze coverage reports to identify untested code areas.
* **Types of Tests:**
  + **Functionality Tests:** Verify that functions and methods produce expected outputs.
  + **Boundary Tests:** Ensure edge cases are handled correctly.
  + **Error Handling Tests:** Confirm that errors are appropriately managed.

**2. Manual Tests**

* **Description:** Manual testing complements automated tests by identifying issues not covered by scripts.
* **Workflow:**
  1. Define test scenarios not easily automated (e.g., exploratory testing).
  2. Execute tests across platforms, documenting results and anomalies.
  3. Focus on edge cases and usability aspects.
* **Types of Tests:**
  1. **Exploratory Testing:** Identify unexpected behaviors through unscripted testing.
  2. **Compatibility Testing:** Verify behavior on various hardware and OS versions.
  3. **Usability Testing:** Assess ease of use and adherence to design standards.

**3. Postman API Tests**

* **Description:** Postman will be used to automate API testing, ensuring backend functionality supports the desktop application.
* **Workflow:**
  1. Define test cases for all API endpoints.
  2. Create Postman collections and automate execution.
  3. Integrate tests into CI/CD pipeline for continuous monitoring.
  4. Review API responses for accuracy, performance, and error handling.
* **Types of Tests:**
  1. **Functional Tests:** Validate API endpoints behave as expected.
  2. **Load Tests:** Simulate high usage to assess performance.
  3. **Error Handling Tests:** Ensure appropriate responses for invalid inputs.

**Reporting and Documentation**

* **Test Evidence:**
  + SikuliX: Screenshots of execution steps and results.
  + Manual Tests: Logs and observations documented in shared templates.
  + Postman: Automated reports from test collections.
* **Storage:**
  + All test plans, scripts, and results will be stored in the GitHub repository under /docs/testing/desktop/.
  + Reports will also be available in the shared drive for quick reference.

**Test Frequency**

* **Automated Tests:**
  + Run on every build and at scheduled intervals to ensure ongoing stability.
* **Manual Tests:**
  + Conducted during major releases and feature additions to address gaps in automation.