**Project Overview and Structure**

This document provides an overview of the project’s structure, tools, and workflows to ensure clarity and consistency across all teams.

**Project Components**

**1. Desktop Version**

* **Description:** Node.js app with python integration. Windows, Mac, and Linux support.
* **Testing:**
  + Automated UI testing with SikuliX.
  + API testing integrated via Postman monitors.
  + Unit testing using Jest and pytest.
  + Manual testing for edge cases and gaps.

**2. Android Version**

* **Description:** Native Android app developed in Android Studio using Kotlin.
* **Testing:**
  + Unit tests for business logic and functionality.
  + UI tests using Espresso.
  + Manual testing for device-specific issues.

**3. iOS Version**

* **Description:** Native iOS app developed in Xcode using Swift.
* **Testing:**
  + Unit tests for app logic.
  + UI tests using XCTest or equivalent.
  + Manual testing for device-specific issues.

**4. API**

* **Description:** Backend services supporting the web, Android, and iOS platforms.
* **Testing:**
  + Automated testing using Postman for API endpoints.
  + Monitored test suites running at scheduled intervals.
  + Manual testing for scenarios not covered by automated tests.

**Testing Workflow**

**Goals**

* Achieve full testing automation wherever feasible.
* Leverage manual testing to fill gaps and verify edge cases.
* Utilize AI for test creation, result review, and gap identification.

**Automation Tools**

* **Desktop:** SikuliX for end-to-end tests, Jest and pytest for unit tests.
* **Android:** Espresso for UI tests, JUnit for unit tests.
* **iOS:** XCTest for UI and unit tests.
* **API:** Postman for endpoint testing and monitoring.

**AI Integration**

* **Test Creation:** AI will generate and suggest tests based on existing requirements and code.
* **Result Review:** AI will analyze test results and highlight potential issues.
* **Discord Integration:** AI will notify results and request assistance via Discord when required.

**Documentation**

**Guideline Documents**

* **Description:** All guideline documents (e.g., Jira, testing, push workflows) are stored in a centralized location.
* **Location:**
  + GitHub repository under /docs/guidelines/.
  + Accessible via the project’s shared drive.

**Test Plans and Reports**

* **Description:** Detailed test plans and reports for each platform.
* **Location:**
  + GitHub repository under /docs/testing/.
  + Monitored results are logged and accessible in Postman and SikuliX dashboards.

**Task Management**

* **Tool:** Jira Kanban board.

**Jira Task Structure**

* **Epics:** Represent features or major functionality areas.
* **Stories, Tasks, and Bugs:** Grouped under their related epic to maintain clear organization. This ensures that if a feature is removed, all associated work is also removed.
* **Labels:** Use platform-specific and test-type labels (e.g., desktop, android, ios, unit-test, regression-test) to categorize tasks effectively.
* **Workflow:**
  + Create tasks following the guidelines in the Jira Task Creation and Completion Guidelines document.
  + Ensure all tasks link back to their respective epic.

**Continuous Integration and Deployment (CI/CD)**

**Service: GitHub Actions**

* **Reason:** GitHub Actions is seamlessly integrated with GitHub, supports multiple platforms, and provides a robust environment for automating workflows.

**Workflow:**

1. **Trigger:** Automatically triggered on every commit or pull request to the repository.
2. **Steps:**
   * Checkout the code.
   * Set up the environment for the platform (e.g., Java for Android, Xcode for iOS).
   * Run automated tests, including unit tests, UI tests, and API tests.
   * Generate and store reports (logs, screenshots, and test coverage).
3. **Notification:** Notify the team via Discord or email about test results and build status.
4. **Deployment:**
   * Successful builds can trigger deployment pipelines for staging or production environments.

**Benefits:**

* Early detection of issues through automated testing.
* Consistent testing across all platforms.
* Faster feedback loops for developers.

**Communication and Notifications**

* **Channels:**
  + Developer chat for reporting progress and asking questions.
  + Test chat for QA team and AI test notifications and assistance requests.
* **Responsibilities:**
  + Notify team members of testing results and issues.
  + Use Jira for task updates and tracking.