**iOS Test Plan**

This document outlines the test plan for the iOS application, focusing on comprehensive automated testing with recommended tools and libraries, supported by manual testing to ensure quality and reliability.

**Goals**

1. **Ensure Functional and UI Integrity**
   * Validate all features and UI components behave as expected across iOS devices.
2. **Achieve High Automation Coverage**
   * Use unit tests, UI tests, and API tests to minimize manual effort.
3. **Verify Cross-Device Compatibility**
   * Test the app on various screen sizes, resolutions, and iOS versions.
4. **Maintain High Performance Standards**
   * Assess app responsiveness and resource usage.
5. **Provide Reliable Test Evidence**
   * Document all test results with logs, screenshots, and reports.

**Testing Types and Workflow**

**1. Unit Tests**

* **Description:** Test individual components of the app’s logic, ensuring they work as expected.
* **Tool:** XCTest
* **Workflow:**
  1. Write tests for core functionality and business logic.
  2. Mock dependencies using libraries like Cuckoo or similar.
  3. Run tests locally or as part of the CI/CD pipeline.
  4. Review coverage reports to identify untested code.
* **Types of Tests:**
  1. **Functionality Tests:** Validate methods and functions.
  2. **Boundary Tests:** Ensure edge cases are handled correctly.

**2. UI Tests**

* **Description:** Validate the behavior and appearance of the user interface.
* **Tool:** XCTest (UI Testing module)
* **Workflow:**
  1. Identify critical user flows (e.g., login, navigation, form submission).
  2. Create XCTest UI test cases for these flows.
  3. Execute tests on multiple iOS simulators and physical devices.
  4. Capture screenshots and logs during test execution.
* **Types of Tests:**
  1. **Smoke Tests:** Verify basic app functionality.
  2. **Regression Tests:** Ensure changes do not affect existing functionality.
  3. **End-to-End Tests:** Test complete workflows.

**3. Postman API Tests**

* **Description:** Automate backend API testing to ensure consistent responses and behavior.
* **Workflow:**
  1. Define test cases for all API endpoints used by the app.
  2. Create and automate Postman collections.
  3. Integrate API tests into the CI/CD pipeline.
  4. Monitor API performance and error handling.
* **Types of Tests:**
  1. **Functional Tests:** Validate API responses.
  2. **Load Tests:** Test API performance under heavy usage.
  3. **Error Tests:** Ensure appropriate error responses.

**4. Manual Tests**

* **Description:** Identify issues that are difficult to automate, such as usability and exploratory testing.
* **Workflow:**
  1. Define test scenarios for edge cases and exploratory testing.
  2. Execute tests on various devices and iOS versions.
  3. Document test results and observations in a shared template.
* **Types of Tests:**
  1. **Exploratory Testing:** Discover unexpected behaviors.
  2. **Compatibility Testing:** Validate app functionality on different devices and iOS versions.
  3. **Usability Testing:** Assess ease of use and adherence to design guidelines.

**Reporting and Documentation**

* **Test Evidence:**
  + Unit Tests: Coverage reports and logs.
  + UI Tests: Screenshots and execution logs.
  + API Tests: Automated reports from Postman.
  + Manual Tests: Logs and documented observations.
* **Storage:**
  + All test plans, scripts, and results will be stored in the GitHub repository under /docs/testing/ios/.
  + Reports will also be available in the shared drive for easy access.

**Test Frequency**

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