# Project Planning & Management

## Overview

This document outlines the testing scope, roles, identified risks, and key performance indicators (KPIs) that will guide the testing process for the [Testing site](https://the-internet.herokuapp.com/). This plan is designed to ensure thorough test coverage across all specified features/pages, assign clear responsibilities, and monitor the testing process with defined success metrics.

## Testing scope

* Login/Authentication: Verify correct behavior when users provide valid and invalid credentials.
* Form Submission: Test various user input fields, including required validations.
* Dropdowns & Checkboxes: Confirm that the selection mechanisms operate as expected.
* File Upload/Download: Validate file handling operations to ensure files are correctly uploaded and accessible.
* API Endpoints: Validate functionality, response, and error handling of designated API endpoints.
* Navigation & Links: Ensure all links and page navigational elements lead to their intended destinations.
* Browser Responsiveness: Test major functionalities across different browsers and devices for compatibility.

## Testing roles

Manual Testing

* Test Analyst/Tester: Responsible for executing test cases manually, performing exploratory tests, and documenting defects.
* User Acceptance Testing (UAT) Manager: Oversees UAT sessions with end users to gather feedback on usability and overall experience.

API Testing

* API Tester: Will verify RESTful/GraphQL API endpoints for correct response codes, data integrity, and performance.
* Integration Tester (if applicable): Focus on how the API integrates with the frontend and other systems.  
    
  Automation Testing
* Test Automation Engineer: Develops, maintains, and executes automated test scripts using frameworks (e.g., Selenium, Cypress, Postman for API automation).
* Continuous Integration (CI) Specialist: Integrates automated tests into the CI/CD pipeline to ensure tests run with every build.

## Risk Identification

1. Browser Compatibility Issues: Certain features may not function uniformly across all browsers and devices.
2. Environment Limitations: Limited test environments might lead to discrepancies when comparing results between staging and production.
3. API Flakiness: Unexpected downtime or inconsistent responses from API endpoints could delay testing.
4. Resource Constraints: Limited personnel and infrastructure (hardware, software tools) could impact timely test execution.
5. Accidental Data Overwrites: During manual testing, there is potential to inadvertently modify or delete key data.
6. Communication Gaps: In case of distributed teams, there might be delays or misunderstandings without proper coordination channels.

For each risk, a mitigation plan will be developed:

* Browser compatibility: Use cross-browser testing tools and dedicate specific time slots to test on multiple configurations.
* Environment limitations: Ensure that the test environment is regularly refreshed and mirrors the production setup.
* API flakiness: Schedule API tests during off-peak hours and implement retry mechanisms.
* Resource constraints: Prioritize critical test cases and employ automation where possible.
* Data management: Use dummy/test data and maintain regular backups.
* Communication: Establish regular status updates and dedicated communication channels.

## KPI

1. Bug Detection Rate: The number of bugs identified per testing cycle. This metric will help determine the effectiveness of the test cases and the quality of each release.
2. Test Coverage Percentage: The proportion of features and pages covered by test cases (both manual and automated). This helps in identifying untested areas.
3. Response Time for API Requests: Average time taken for API responses during testing to ensure performance meets specified thresholds.