**Automation Test Strategy Document**

**Project Overview**

The objective of this project is to perform comprehensive automated testing consists of 6 user stories defined for the Working-Class Heroes application. The testing suite covers API, UI, and database validations to verify the end-to-end functionality of the application. Playwright (UI), Rest Assured (API) with Java is used as the primary Functional automation framework (Hybrid), leveraging the Page Object Model (POM) with Behavior-Driven Development (BDD) for structured, readable, and reusable test components. The automation suite includes high-priority test scenarios across different functional areas of the application, ensuring that core requirements are validated with a focus on user stories.

**Testing Approach**

The automated test strategy is designed to achieve complete test coverage for core functionalities across different layers of the application. The main testing types are structured as follows:

1. **End-to-End (E2E) Testing**:
   * **Combination of UI, API, and Database Validations**: E2E testing will cover workflows involving interactions across multiple layers to ensure data consistency and functionality. This includes API requests, UI element validations, and database integrity checks.
   * **BDD (Behavior-Driven Development) Tests**: Select scenarios are structured as BDD scenarios for clarity and stakeholder communication.
2. **Automated Testing Scope**

The following testcases derived from user stories and covered by automated tests:

* **User Story 1: Create Working Heroes in Database via API**
  + Testcase: Single Working-Class Hero Creation
  + Testcase: Check with invalid natid where it contains alphanumeric characters
  + Testcase: Check with invalid natid where it does not prefix with natid
  + Testcase: Check with invalid natid where the number is not inclusive
  + Testcase: Check with invalid name where the it includes numeric value
  + Testcase: Check with invalid name where the it exceeds than allowed limit
  + Testcase: Check with invalid gender
  + Testcase: Check with future death Date
  + Testcase: Check with invalid salary
  + Testcase: Check with invalid taxpaid
  + Testcase: Check with creation of duplicate Single Working-Class Hero
  + Testcase: Check created Working-Class Hero in database
* **User Story 2: Upload CSV file to Populate via UI**
  + Testcase: Validate user lands on Clerk Dashboard page
  + Testcase: Check Add Hero Button is visible.
  + Testcase: Upload a working record CSV File
  + Testcase: Upload a erroneous record CSV File
  + Testcase: Upload a CSV File with Column headers
  + Testcase: Upload a text file no CSV File
  + Testcase: Upload a empty record CSV file
  + Testcase: Add a working-class hero using Add a hero page.
* **User Story 3: Generate Tax Relief Egress File via UI**
  + Testcase: Validate Bookkeeper Dashboard page
  + Testcase: Generate Tax Relief Egress File Button is visible
  + Testcase: Generate Tax Relief Egress File Successfully
  + Testcase: Check file status in db once file process is triggered
  + Testcase: Check the total count of records, if file has contents
  + Testcase: Check the total count, if file has empty records
* **User Story 4: Provide API for Creating Hero with Vouchers**
  + Testcase: Creation of Single Working-Class Hero with Vouchers
  + Testcase: Check created vouchers added in database
  + Testcase: Single Working-Class Hero creation with empty Vouchers.
  + Testcase: Check vouchers added in db, which created with invalid details.
* **User Story 5: Working class Owes money via External API**
  + Testcase: Check owe-money api with valid natid
  + Testcase: Check owe-money api with invalid natid
  + Testcase: Check owe-money api with valid natid and validate its response format
  + Testcase: Check owe-money api with valid natid and validating its response schema structure
* **User Story 6: Provide API insight into number of vouchers**
  + Testcase: Number of vouchers each customer has each voucher category

**Testing Architecture and Tools**

1. **Test Framework**: Playwright(UI), Rest Assured(API) with Java
2. **Design Pattern**: Page Object Model (POM)
3. **Testing Framework**: Hybrid Framework (BDD) using Playwright’s Cucumber-style syntax.
4. **Continuous Integration**: Jenkins/GitHub Actions (Can Integrate)

**Automation Test Structure**

1. **Page Object Model (POM)**: Each UI component and page flow are encapsulated in a separate class, reducing duplication and ensuring reusable and maintainable code. This structure enhances readability and scalability of test cases.
2. **BDD Implementation**:
   * Select scenarios are implemented with BDD syntax, enhancing communication with non-technical stakeholders.
3. **Test Data Management**:
   * Dynamic and static test data configurations are included to support test scenarios, using external files and environment-specific configurations.
4. **Reporting**:
   * Detailed test execution reports with screenshots and logs for failed steps are generated, aiding in faster debugging and resolution.

**Further Test Types for Automated Testing Pipeline**

To enhance test coverage, additional testing types are recommended:

1. **Performance Testing**: Simulate load on API endpoints, including data retrieval, updates, and file generation processes. This will help ensure the application performs efficiently under anticipated traffic.
2. **Security Testing**: Incorporate security checks, especially around the API endpoints (e.g., SQL injection, XSS). Tools such as OWASP ZAP or Burp Suite could be integrated to identify vulnerabilities.
3. **Accessibility Testing**: Validate compliance with accessibility standards, particularly for UI components in the Clerk and Bookkeeper dashboards.
4. **Regression Testing**: Build a suite of critical regression tests to validate core functionalities after each new deployment, ensuring that existing features remain unaffected.
5. **Compatibility Testing**: Run the UI tests across multiple browsers and devices to verify cross-browser compatibility, as the application is used in various environments.