 **Fitpeo Automation Documentation**

**1.Project Overview**

The purpose of this automation project is to test and validate the functionality of the **FitPeo Revenue Calculator** using Selenium WebDriver. The tests ensure that key features, such as slider adjustments, text field updates, CPT code selections, and total reimbursement validation, are working as expected. The project is built on a **hybrid framework** with **TestNG**, incorporating **data-driven testing**, **cross-browser execution**, and **report generation** using **Apache Extent Reports**.

**2. Technologies used**

 **Programming Language:** Java

 **Automation Tool:** Selenium WebDriver

 **Test Framework:** TestNG

 **IDE:** Eclipse

 **Reporting:** Apache Extent Reports

 **Data Storage:** Properties File

 **Framework Type:** Hybrid Framework (Keyword-Driven and Data-Driven)

 **Cross-Browser Support:** Configured through XML

 **Error Handling:** Screenshots captured for failed test cases.

**3. Test Cases**

Test cases are documented in FitpeoTestcases.xlsx excel file.

**4. Framework Features**

1. **Hybrid Framework with TestNG:**
   * Combines Keyword-Driven and Data-Driven approaches for flexibility.
   * XML-based test configuration for modular execution.
2. **Cross-Browser Testing:**
   * Configurable through TestNG XML to run tests on multiple browsers like Chrome, Firefox,and Edge.
3. **Data-Driven Testing:**
   * Uses a properties file for input data such as URLs, slider values, and expected results.
4. **Reporting with Extent Reports:**
   * Generates comprehensive HTML reports with test execution details.
5. **Error Handling:**
   * Captures and stores screenshots of failed test cases for debugging purposes.

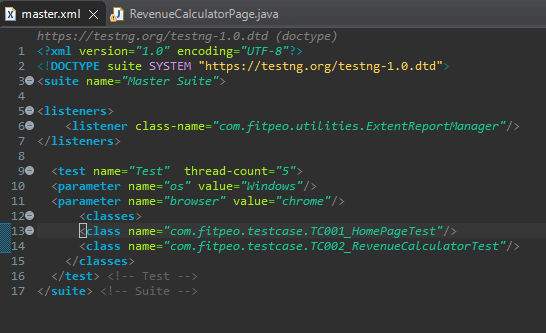
**5.Setup and Execution Instructions**

1. **Environment Setup:**
   * Install Java JDK (8 or later) and set the system environment variables.
   * Install Eclipse IDE for Java development.
   * Download the Selenium WebDriver JAR files and add them to the project's build path.
   * Add dependencies for TestNG and Extent Reports (if using Maven, include them in pom.xml).
2. **Framework Configuration:**
   * Place the chromedriver and other browser drivers in the system PATH or provide their paths in the properties file.
   * Configure the TestNG XML file for:
     + Test suites and test cases.
     + Cross-browser execution.
3. **Run Tests:**
   * To run a specific test, execute the TestNG XML file via Eclipse or from the command line:

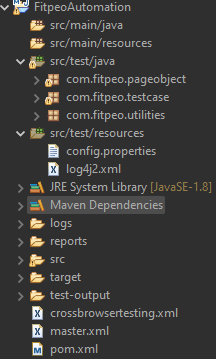
mvn test -DsuiteXmlFile=testng.xml

1. **Report Generation:**
   * The HTML report will be generated automatically in the specified directory (configured in the Extent Reports setup).

**6. Sample TestNG XML Configuration**



1. **Folder Structure of Project**

****

**8. Highlights of the Automation**

* **Dynamic Element Handling:**
  + Used explicit waits to manage dynamic elements (e.g., slider and text field updates).
* **Error Recovery:**
  + Implemented robust exception handling for cases like NoSuchElementException and ElementNotInteractableException.
* **Logging and Debugging:**
  + Integrated logs into Extent Reports for detailed tracking of each test case execution.
* **Flexibility:**
  + Modular design allows for easy addition of new test cases or changes in the test flow.

**9. Additional Information**

* **Assumptions:**
  + The test environment has stable internet connectivity.
  + Browser drivers (e.g., ChromeDriver) are up to date.
* **Known Issues:**
  + Dynamic loading delays in the Revenue Calculator page may require optimized wait times.

**10. Sample Output**

 **Extent Report:**

* Provides detailed logs of passed and failed test cases.
* Includes screenshots for failures.

 **Screenshot Example:**

* Stored in the /screenshots folder within the project directory.