# Revenue Calculator Test Automation - Detailed Flow Document

## 1. Objective

The objective of this test script is to automate the functional validation of the Revenue Calculator page on the "FitPeo" website. This includes:  
- Navigating to the Revenue Calculator page.  
- Adjusting a slider to set patient count.  
- Verifying the patient count reflects correctly across related UI elements.  
- Selecting multiple CPT options and calculating the total reimbursement based on selected options and patient count.

## 2. Pre-requisites

1. Environment Setup:  
 - Ensure Java and Maven are installed.  
 - Selenium WebDriver and WebDriverManager dependencies are configured in the project.  
 - Add required libraries: selenium-java  
  
2. Browser Compatibility:  
 - The script is developed for Chrome. Verify the latest version of ChromeDriver is installed using WebDriverManager.  
  
3. Test Data:  
 - URL: https://www.fitpeo.com/  
 - Default patient count for slider: 820 (as per requirements).  
 - Test CPT codes: CPT-99091, CPT-99453,CPT-99454, CPT-99474.

## 3. Test Flow

### 3.1 Initialization

- Set up WebDriver using WebDriverManager for ChromeDriver.  
- Configure browser settings:  
 - Maximize browser window.  
 - Define implicit wait (10 seconds) and page load timeout (120 seconds).

### 3.2 Navigation

1. Launch the FitPeo website (https://www.fitpeo.com/).  
2. Click the Revenue Calculator page link.

### 3.3 Revenue Calculator Validation

Step 1: Adjust Patient Count Slider  
- Locate the slider element using XPath: //input[@type='range' and contains(@style,'border')].  
- Read the current slider value using the value attribute.  
- Adjust the slider to the target value (e.g., 820) using the Robot class:  
 - Press the RIGHT arrow key iteratively until the desired value is achieved.  
  
Step 2: Verify Slider and Patient Count Sync  
- Locate the patient count text box using XPath: //input[contains(@class,'MuiInputBase') and @type='number'].  
- Read the value from the patient count text box and verify it matches the slider value using Assert.  
  
Step 3: Update Patient Count Manually  
- Clear the text box and enter a new value (e.g., 560).  
- Verify the slider updates to reflect the new patient count value.

### 3.4 CPT Options Validation

Step 1: Select CPT Options  
- Locate each CPT checkbox using dynamic XPath (e.g., //p[text()='CPT-99091']//following-sibling::label//input).  
- Scroll the checkbox into view using JavaScriptExecutor.  
- Click the checkbox using JavaScriptExecutor.  
  
Step 2: Extract CPT Price  
- Locate the price label associated with the selected CPT option using XPath (e.g., //p[text()='CPT-99091']//following-sibling::label//span[contains(@class,'MuiTypography')]).  
- Extract the numerical value from the price label.  
  
Step 3: Repeat for All CPT Options  
- Perform the same actions for CPT-99453, CPT-99454 and CPT-99474.

### 3.5 Total Reimbursement Validation

1. Calculate the expected total reimbursement:  
 - Formula:   
 Total = (CPT-99091 Price + CPT-99454 Price + CPT-99474 Price) \* Patient Count  
2. Locate the total reimbursement element using XPath:  
 - //p[contains(text(),'Total Recurring Reimbursement')]//child::p.  
3. Extract the displayed total reimbursement value.  
4. Compare the calculated total with the displayed total using Assert.

## 4. Test Scenarios

Scenario 1: Successful Revenue Calculation  
- Precondition: The website is accessible, and all elements load correctly.  
- Steps:  
 1. Adjust the slider to a target value.  
 2. Verify slider and patient count synchronization.  
 3. Select CPT options.  
 4. Validate the total reimbursement amount.  
- Expected Result:  
 - The calculated and displayed reimbursement amounts match.  
  
Scenario 2: Handling Invalid Inputs  
- Precondition: User manually enters a non-numeric or negative value in the patient count field.  
- Steps:  
 1. Enter invalid values (e.g., -100, abc) in the text box.  
 2. Observe system behavior.  
- Expected Result:  
 - System restricts invalid inputs or resets to the previous valid state.  
  
Scenario 3: Element Interaction Issues  
- Precondition: Checkbox or slider interactions fail due to overlay or obstruction.  
- Steps:  
 1. Attempt to click an obstructed element.  
 2. Use JavaScriptExecutor to resolve the issue.  
- Expected Result:  
 - The element is successfully interacted with.

## 5. Tools & Technologies

- Language: Java  
- Automation Tool: Selenium WebDriver  
- Browser: Chrome  
- Additional Libraries:  
 - WebDriverManager (Driver setup).  
 - Robot Class (Keyboard interaction).  
 - JavaScriptExecutor (Advanced DOM manipulation).

## 6. Expected Output

- The script validates the Revenue Calculator functionality, ensuring accurate total reimbursement calculation and seamless user interactions.