**Test Plan Document for BillWise Pro**

**Made By:**

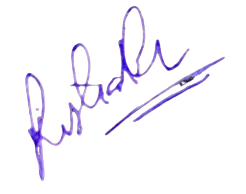
**Dev Varshney (2200290129009) 7C**

**Shreshthi Tayal (2100290120160) 7C**

**Vanshika Jain (2100290120186) 7C**

**Viyapi Tyagi (2100290120195) 7C**

**Test Manager:**

****

**Mr. Rishabh Chakraborty**

***Change Log***

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Change Date** | **By** | **Description** |
| 001 | 04.12.2024 | Shreshthi Tayal | Initial Draft |
| 002 | 11.12.2024 | Shreshthi Tayal | Complete Draft |
|  |  |  |  |

1 Introduction 1

1.1 Scope 4

1.2 Quality Objective 5

1.3 Roles and Responsibilities 6

2 Test Methodology 9-18

2.1 Overview 10

2.2 Test Levels 10

2.3 Suspension Criteria and Resumption Requirements 13

2.4 Test Completeness 13

3 Test Deliverables 18-22

4 Resource & Environment Needs 18-22

4.1 Testing Tools 18

4.2 Test Environment 20

5 Terms/Acronyms 22-23

1. **Introduction**

This document outlines the test plan for the BillWise Pro web application, a billing and inventory management system designed for small business owners. The purpose of this test plan is to ensure the functionality, usability, and performance of the application meet the desired quality standards. Testing will be conducted using Selenium for automation to validate core functionalities across multiple browsers.

* 1. **Scope**
     1. **In Scope**

**Features of Project:**

1. **User Authentication: Sign-up/Sign-in**

* Testing the **Sign-up** process with OTP validation.
* Verifying **Sign-in** functionality using valid and invalid credentials.
* Ensuring proper error messages for invalid or missing inputs (e.g., invalid OTP or phone number).

1. **Inventory Management: Add/Show Item**

* Verifying the ability to **add items** with valid details (e.g., item name, price, quantity).
* Ensuring all added items are correctly displayed.
* Testing edge cases like duplicate items, invalid data (e.g., negative price or quantity), and maximum character limits.
* Ensuring inventory updates correctly after sales and purchases.

1. **Customer Management: Add/Show Party**

* Testing the **Add Clients** functionality to allow adding customers with valid information (e.g., name, email, phone number).
* Verifying that all parties are listed correctly.
* Testing scenarios like invalid customer details, missing fields, or long inputs.

1. **Sales Management: Add Sales and Invoice Generation**

* Validating the ability to **create a sale** by selecting items, quantities, and clients.
* Ensuring **invoice generation** functionality works correctly (e.g., displays itemized list, total amount, due amount).
* Testing that inventory is updated appropriately after a sale is made.
* Verifying error handling for cases like insufficient stock or missing required fields.

1. **Purchase Management: Add Purchases**

* Testing the functionality to **add purchases** by selecting items and specifying purchase details.
* Ensuring inventory increases appropriately after adding a purchase.
* Validating error handling for invalid or incomplete inputs (e.g., missing quantity or price).

1. **Dashboard Metrics**

* Verifying that the **dashboard displays accurate metrics**, including:
* Total sales
* Amount due
* Current inventory status
* Ensuring metrics update in real-time after sales and purchases.

1. **User Interface Testing**

* Ensuring all pages have a consistent and user-friendly design.
* Verifying the proper alignment and functionality of buttons, input fields, and other UI components.
* Ensuring error messages and confirmations are displayed in a readable and consistent manner.

1. **Usability Testing**

* Evaluating ease of navigation through different modules.
* Testing the application's overall intuitiveness for end users.

1. **Performance Testing**

* Verifying that all pages load within an acceptable timeframe (e.g., under 2 seconds for key pages like Dashboard).
* Ensuring the application remains responsive under normal and high user loads.
  + 1. **Out Scope**
* Testing on mobile devices.
* API testing beyond basic integration.
* Database-level testing and performance optimizations.
  1. **Quality Objective**
* Ensure that all critical modules of the BillWise Pro application work as per the functional and non-functional requirements.
* Identify and resolve defects before deployment.
* Confirm that the system performs reliably under expected loads.
* Achieve at least a 90% test case pass rate for critical features.
* Ensure the application is compatible with the supported web browsers.
  1. **Roles And Responsibilities**
* **QA Analyst**: Shreshthi Tayal is responsible for preparing and executing test cases, identifying bugs, and ensuring all functionalities meet the requirements. She also conducts regression and cross-browser testing using Selenium and coordinates with developers to resolve defects.
* **Test Manager**: Mr. Rishabh Chakraborty oversees the entire testing process, ensuring quality and adherence to timelines. He defines the test plan, reviews deliverables, and manages the QA team’s progress.
* **Developers**: The development team including Dev Varshney, Shreshthi Tayal, Vanshika Jain, and Viyapi Tyagi, implements the project features and fixes defects identified during testing. They ensure seamless module integration and provide technical support during the testing process.
* **Configuration Manager**: Dev Varshney manages the configuration and setup of the testing environment. He ensures all necessary tools, libraries, and environments are ready for testing.
* **Installation Team**: Mr. Akash Goel, Dev Varshney, Shreshthi Tayal, Vanshika Jain, and Viyapi Tyagi handles the deployment of the BillWisePro application. He ensures the application is installed and functions correctly in the production environment.

1. **Test Methodology**
   1. **Overview**

The testing methodology for BillWise Pro follows an iterative approach. The process begins with unit testing for individual components, progresses to integration testing for connected modules, and culminates in system testing. Automated regression testing using Selenium ensures consistency and repeatability of test cases after updates.

* 1. **Test Levels**

1. **Unit Testing:**

* Focused on individual modules such as Sign-up/Sign-in and Add Item.
* Conducted by developers using mocks and stubs as required.

1. **Integration Testing:**

* Verifies interactions between modules (e.g., inventory updates after adding a sale).
* Ensures that dependent modules work as a cohesive system.

1. **System Testing:**

* Conducted on the entire application to validate functional and non-functional requirements.
* Focuses on end-to-end workflows, such as sales creation affecting inventory and dashboard updates.

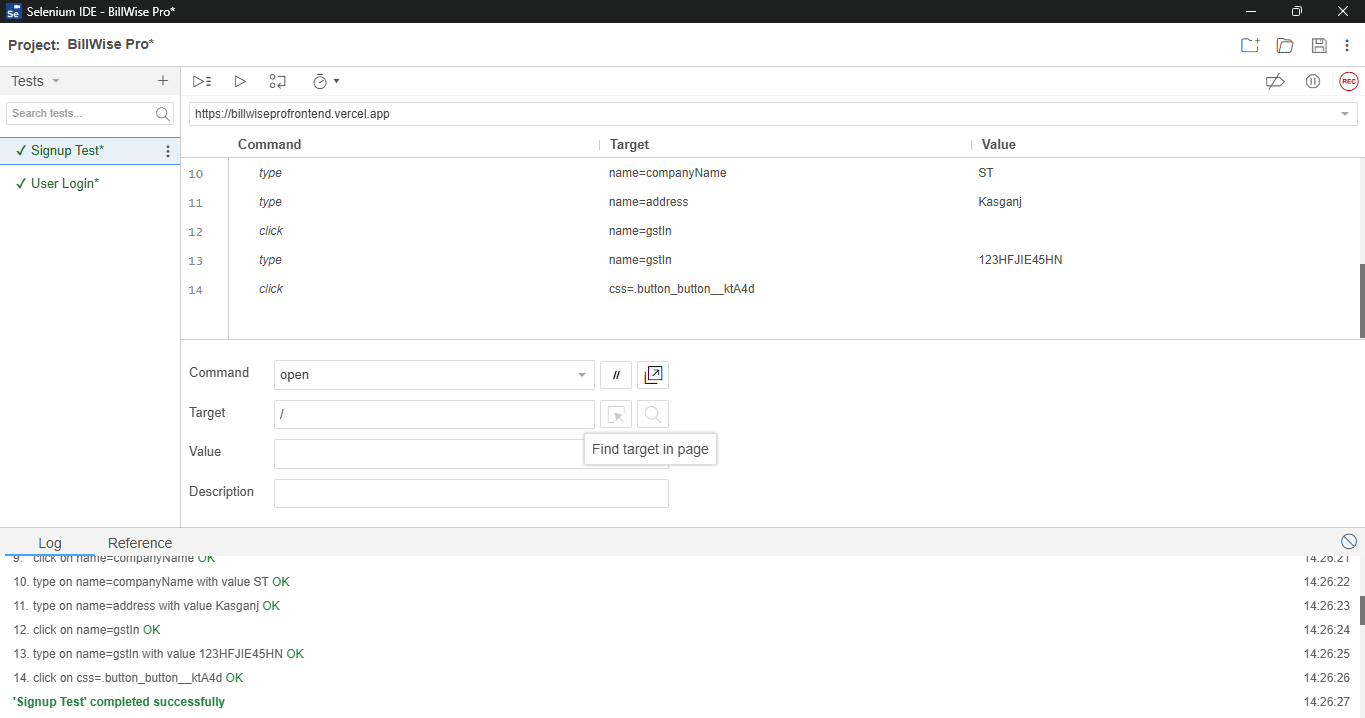
1. **Regression Testing**:

* Ensures new changes do not negatively impact existing functionalities.
* Automated using Selenium for efficiency and consistency.
  1. **Suspension Criteria and Resumption Requirements**
* **Suspension Criteria**:
* Testing will be paused if critical blockers are encountered (e.g., the application crashes or key modules are inaccessible).
* **Resumption Requirements**:
* Testing will resume after the blockers are resolved and the fixes are verified.
  1. **Test Completeness**

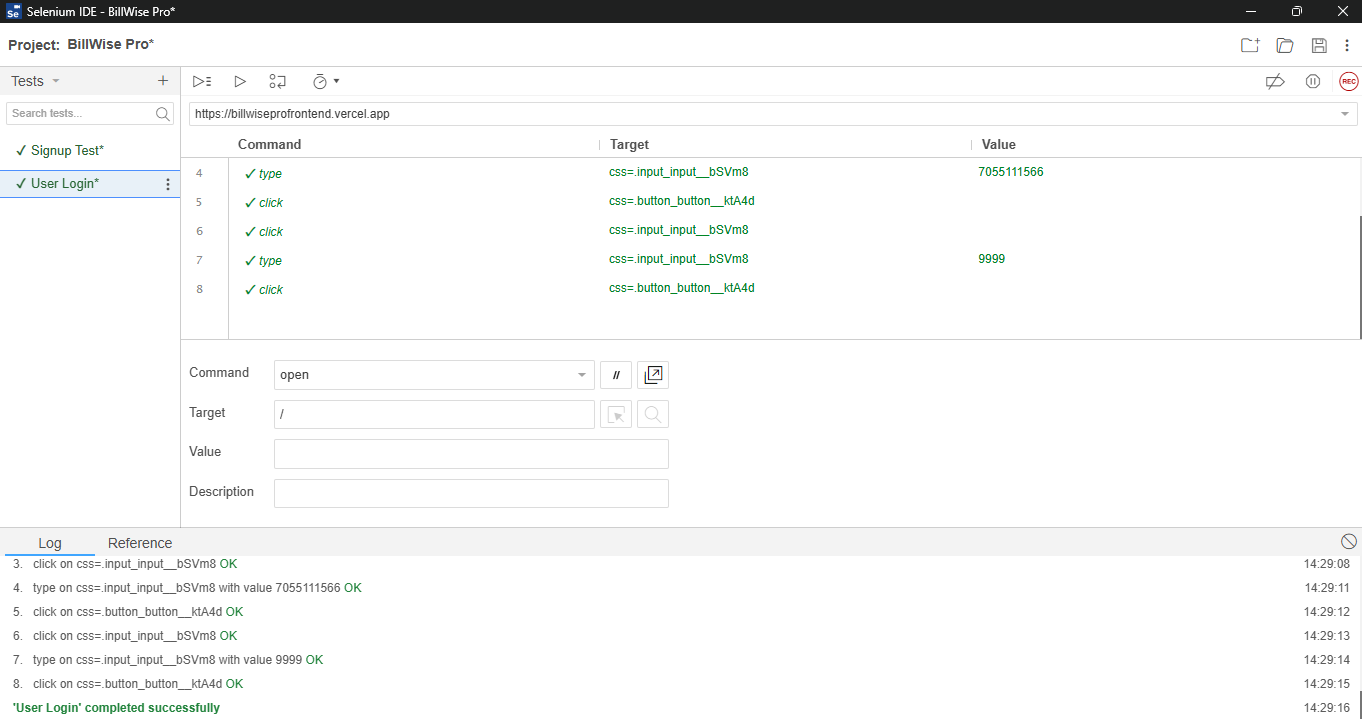
Testing will be considered complete when:

* All critical test cases pass.
* Major defects are fixed or documented for future resolution.
* Regression tests show no failures.
* Test coverage reaches at least 95% for critical modules

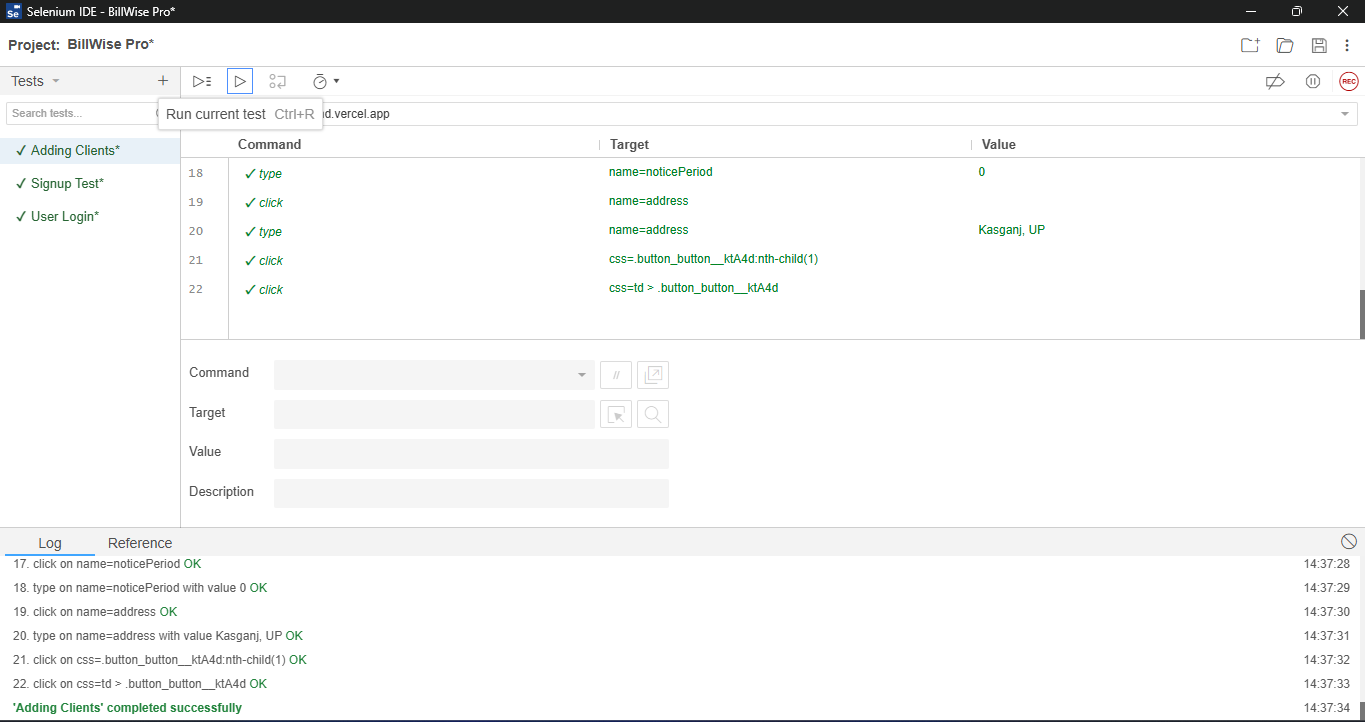
1. **Test Deliverables**
   1. **Automation Testing using Selenium**
      1. **Test Case Name: User Signup**

****

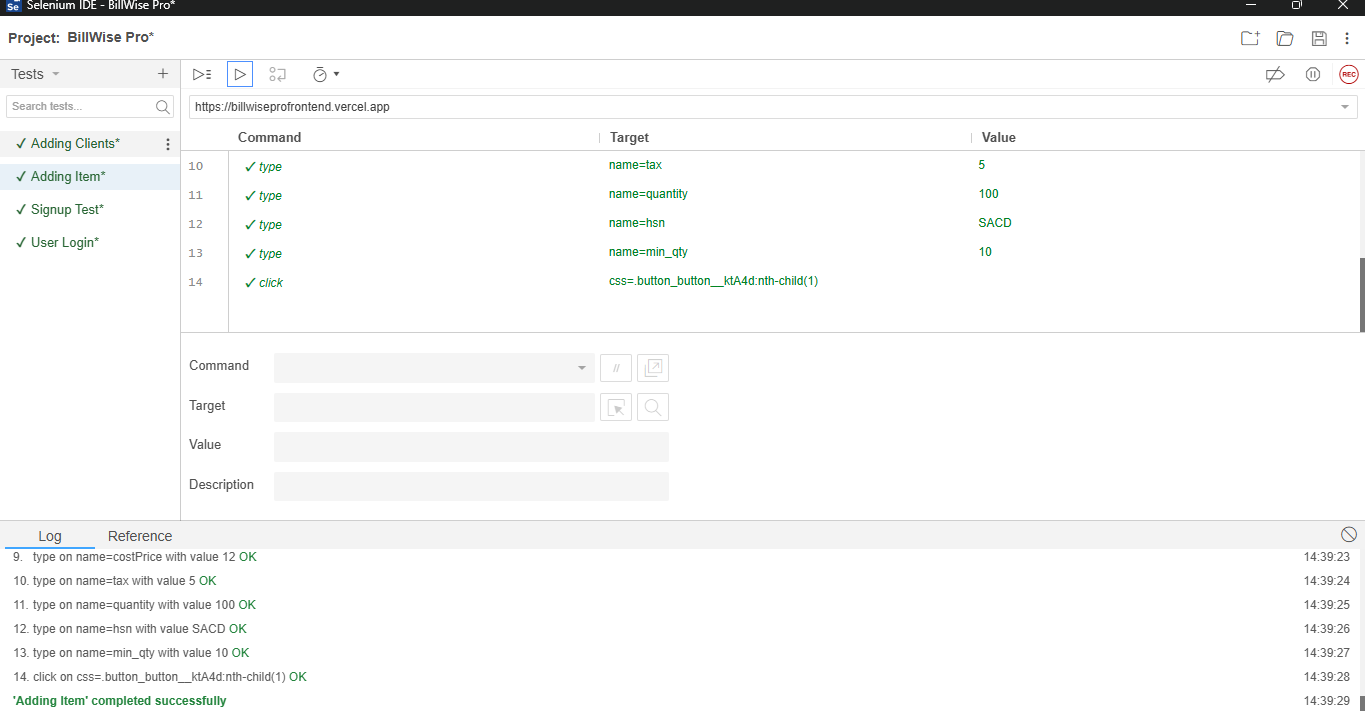
* + 1. **Test Case Name: User Login**

****

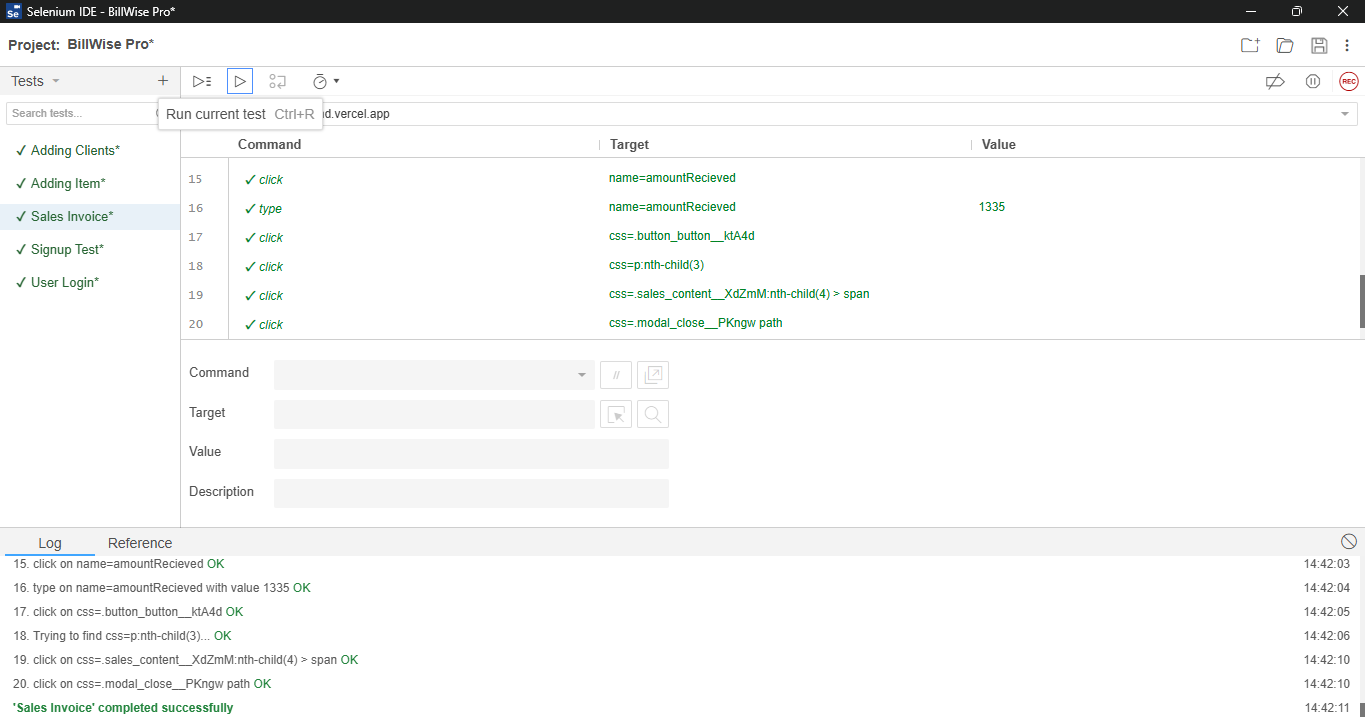
* + 1. **Test Case Name: Adding Clients**

****

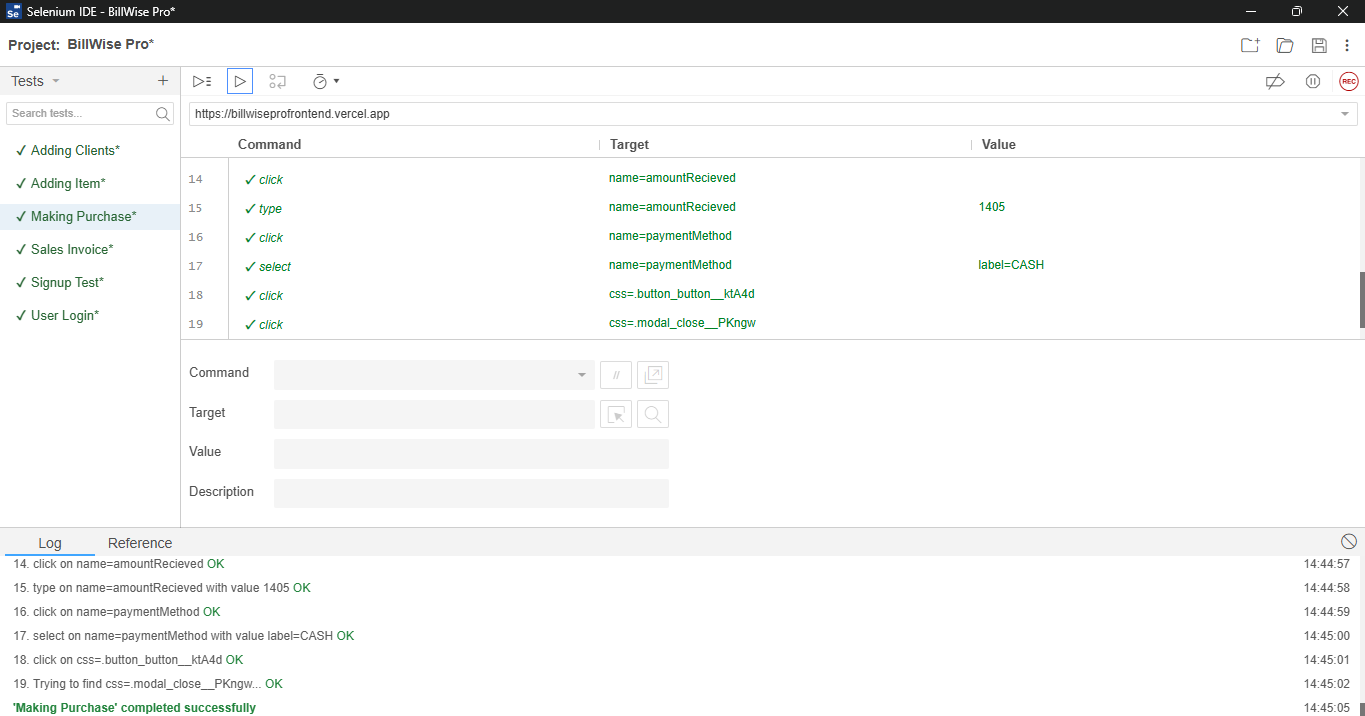
* + 1. **Test Case Name: Adding item**

****

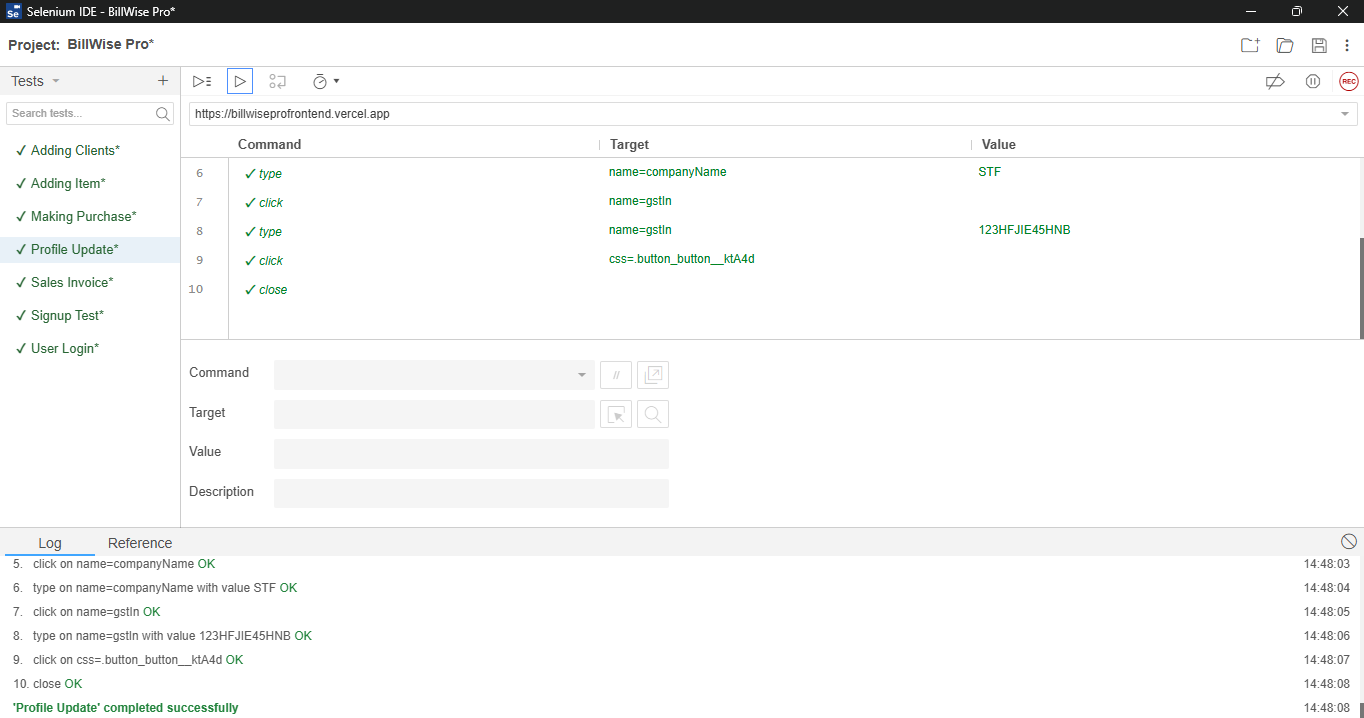
* + 1. **Test Case Name: Sales Invoice**

****

* + 1. **Test Case Name: Making Purchase**

****

* + 1. **Test Case Name: Profile Update**

****

* 1. **Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Steps to Execute** | **Expected Result** |
| TC01 | Validate User Sign-up functionality | 1. Navigate to the Sign-up page.  2. Enter valid phone number, name, email id, etc.  3. Click "Sign Up." | User account is created successfully. |
| TC02 | Validate User Login with valid credentials | 1. Navigate to the Login page.  2. Enter valid phone number and OTP.  3. Click "Log In." | User is logged in and redirected to the dashboard. |
| TC03 | Validate User Login with invalid credentials | 1. Navigate to the Login page.  2. Enter valid phone number but invalid OTP.  3. Click "Log In." | Error message displayed: "Invalid OTP." |
| TC04 | Verify adding a new item in inventory | 1. Navigate to "Inventory" page.  2. Enter valid item details (name, price, quantity).  3. Click "Add." | Item is added successfully and visible in the "Inventory" list. |
| TC05 | Verify adding a new customer | 1. Navigate to "Clients" page.  2. Enter valid client details (name, email, phone).  3. Click "Add." | Customer is added successfully and listed in "Clients." |
| TC06 | Validate duplicate customer cannot be added | 1. Add a customer with existing details.  2. Attempt to save. | Error message: "Customer already exists." |
| TC07 | Validate sales and invoice generation functionality | 1. Navigate to "Sales" page.  2. Select items and quantities.  3. Click "Save" | Invoice is generated successfully, and inventory is updated. |
| TC08 | Validate inventory updates after sales | 1. Check item quantity before sales.  2. Make a sale.  3. Check item quantity after sales. | Inventory count decreases appropriately. |
| TC09 | Verify purchase updates inventory | 1. Navigate to "Add Purchase" page.  2. Select items and quantities.  3. Click "Add Purchase." | Inventory count increases appropriately. |
| TC10 | Validate dashboard displays accurate metrics | 1. Navigate to Dashboard.  2. Check total sales, amount due, and inventory metrics. | Dashboard displays correct and real-time data. |

1. **Resouce and Environment Needs**
   1. **Testing Tools**

* **Selenium WebDriver**: For automating web-based testing.
* **ChromeDriver, GeckoDriver**: For browser-specific testing.
* **TestNG**: To manage test execution and generate reports.
* **Eclipse or IntelliJ IDEA**: IDEs for writing and managing Selenium scripts.
  1. **Test Environment**
* **Hardware Requirements**:
* Operating System: Windows 10 or higher.
* RAM: Minimum 8GB.
* Processor: Intel Core i5 or higher.
* **Software Requirements**:
* Google Chrome, Mozilla Firefox, and Microsoft Edge (latest versions).
* Stable internet connection.
* **Test URL**:
* [BillWisePro](https://billwiseprofrontend.vercel.app)

1. **Terms/ Acronyms**

|  |  |
| --- | --- |
| **Term/Acronym** | **Definition** |
| AUT | Application Under Test |
| QA | Quality Assurance |
| Selenium WebDriver | A tool for automating web browser interactions |
| TestNG | A testing framework for Java applications |
| ChromeDriver | A driver for automating tests on Chrome |
| GeckoDriver | A driver for automating tests on Firefox |
| Regression Testing | Testing to ensure changes do not break existing functionalities |