Candidate Name:MD ATIQUL ISLAM.

Email- atik.cmttiu1001@gmail.com

Qustions of the Assessment:

2nd Assessment:

#Web Automation:

Application URL: https://hishabee.business/

1. Write 20 Test Cases from different modules (including CRUD operation).

2. For web Automation you can use the selenium/cypress library.

3. Try to use POM/BDD framework.

4. Generate Allure report.

1.Answer: There are 20 test cases for a web application like "https://hishabee.business/" involves covering various functionalities and scenarios across different modules. Here's a list of 20 test cases that span different aspects of web automation:

Module: User Registration and Authentication

1. Verify that a new user can successfully register an account.

2. Validate that registration fails if an invalid email format is used.

3. Check that registration fails if the password doesn't meet complexity requirements.

4. Confirm that a registered user can log in with valid credentials.

5. Ensure login fails with incorrect login credentials.

6. Test the "Forgot Password" functionality, ensuring a password reset email is sent.

Module: Product Management

7. Create a new product listing with valid details.

8. Verify that a product cannot be created with missing mandatory fields.

9. Edit an existing product's information and save the changes.

10. Delete a product and confirm it's no longer listed.

Module: Shopping Cart

11. Add a product to the shopping cart and validate its presence.

12. Increase the quantity of a product in the cart and verify the total price.

13. Remove a product from the cart and ensure it's no longer present.

Module: Search Functionality

14. Search for a product by its name and verify the search results.

15. Test the search filter options (e.g., price range, category) and ensure they work as expected.

Module: Order Placement and Payment

16. Place an order for one or more products and proceed to checkout.

17. Validate that the payment gateway redirects to the correct page.

18. Test different payment methods (e.g., credit card, PayPal) for successful payments.

Module: User Profile and Settings

19. Edit the user's profile information (e.g., name, address) and save changes.

20. Change account settings (e.g., email notifications, password) and confirm the changes take effect.

2.Answer: We can use either Selenium or Cypress for web automation on the website "https://hishabee.business/." Both libraries are capable of automating web interactions, but they have different approaches and features. Here's a brief overview of how you can use each of them:

1. *Selenium*:

- Selenium is a versatile framework that supports various programming languages (e.g., Python, Java, JavaScript).

- You can write test scripts to interact with web elements and perform actions such as clicking buttons, filling forms, and navigating through pages.

- Selenium allows for cross-browser testing, meaning you can test your application on multiple web browsers.

- You'll need to set up a WebDriver for your chosen browser and language, and then you can write test scripts to automate interactions.

2. *Cypress*:

- Cypress is a JavaScript-based end-to-end testing framework.

- It's known for its simplicity and fast test execution, making it a great choice for web applications.

- Cypress provides a built-in test runner and a rich set of commands for interacting with elements on the web page.

- It's more focused on testing single-page applications and modern web applications.

Your choice between Selenium and Cypress depends on your specific project requirements and your familiarity with the tools. If you have a preference for JavaScript and want a straightforward setup, Cypress might be a good choice for testing the given website. However, if you prefer using another programming language or need extensive cross-browser testing, Selenium is a solid option.

3.Answer: We use either Selenium or Cypress for web automation on the website "https://hishabee.business/." Both libraries are capable of automating web interactions, but they have different approaches and features. Here's a brief overview of how you can use each of them:

1. *Selenium*:

- Selenium is a versatile framework that supports various programming languages (e.g., Python, Java, JavaScript).

- You can write test scripts to interact with web elements and perform actions such as clicking buttons, filling forms, and navigating through pages.

- Selenium allows for cross-browser testing, meaning you can test your application on multiple web browsers.

- You'll need to set up a WebDriver for your chosen browser and language, and then you can write

test scripts to automate interactions.

2. *Cypress*:

- Cypress is a JavaScript-based end-to-end testing framework.

- It's known for its simplicity and fast test execution, making it a great choice for web applications.

- Cypress provides a built-in test runner and a rich set of commands for interacting with elements on the web page.

- It's more focused on testing single-page applications and modern web applications.

Your choice between Selenium and Cypress depends on your specific project requirements and your familiarity with the tools. If you have a preference for JavaScript and want a straightforward setup, Cypress might be a good choice for testing the given website. However, if you prefer using another programming language or need extensive cross-browser testing, Selenium is a solid option.

If you have any specific questions or need assistance with writing automation scripts for this website, feel free to ask!

[1:13 am, 28/09/2023] A.i.Atik: Certainly, you can create a web automation framework using Page Object Model (POM) and Behavior-Driven Development (BDD) principles. Here's a high-level overview of how you can approach this:

1. Set up the Environment:

- Choose a programming language (e.g., Java, Python) and a test automation framework (e.g., Selenium).

- Install necessary libraries and tools such as Selenium WebDriver, Cucumber (for BDD), and a build tool like Maven or Gradle.

2. Project Structure:

- Create a project directory structure. For example:

  - src

- main

- test

  - java

    - pages

    - stepdefinitions

  - resources

    - features

3. Define Page Objects:

- Create a class for each web page you want to automate. These classes should encapsulate the page's elements and actions.

- Use locators (XPath, CSS selectors, etc.) to identify and interact with elements on the pages.

4. Write Feature Files (BDD):

- In the "features" directory, create feature files using Gherkin syntax. These files describe the behavior of your application in a human-readable format.

- Define scenarios and steps for each feature.

5. Implement Step Definitions:

- Create step definition classes in the "stepdefinitions" directory.

- Map each step in the feature files to a step definition method.

- In these methods, use Selenium WebDriver to perform actions on the page and make assertions.

6. Set up Test Runner:

- Configure a test runner (e.g., JUnit or TestNG) to run your Cucumber tests.

- Specify the path to your feature files and the package where your step definitions are located.

7. Write Test Cases:

- Create test cases that utilize the step definitions to execute the test scenarios defined in your feature files.

8. Execute Tests:

- Use the test runner to execute your tests against the target application URL (https://hishabee.business/).

9. Reporting and Logging:

- Implement reporting and logging mechanisms to capture test results and any relevant information.

10. Maintain and Expand:

- Regularly maintain and update your automation framework as the application evolves.

- Add more features, scenarios, and tests as need.

4.Answer: To generate an Allure report for your Selenium WebDriver tests on the "https://hishabee.business/" website, you'll need to follow these steps:

1. *Add Allure Dependencies:*

- Ensure you have Allure dependencies added to your project. You can typically add these dependencies to your `pom.xml` file if you're using Maven or your build.gradle file if you're using Gradle. Here's an example for Maven:

```
xml

<dependencies>

    <!-- ... other dependencies ... -->

    <dependency>

        <groupId>io.qameta.allure</groupId>

        <artifactId>allure-cucumber6-jvm</artifactId>
```

```
            <version>2.15.0</version> <!-- Check for the latest version on the Allure website -->
            <scope>test</scope>
        </dependency>
    </dependencies>
```

2. *Configure Allure with Cucumber:*

- Configure Allure to work with Cucumber. You might need to specify the location of your Allure results in your test runner class. Here's an example for JUnit:

```
java
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"pretty", "io.qameta.allure.cucumber6jvm.AllureCucumber6Jvm"},
        features = "src/test/resources/features",
        glue = "stepdefinitions"
)
public class TestRunner {
}
```

3. *Run Your Tests:*

- Execute your Selenium WebDriver tests using the test runner you configured. This will generate Allure-compatible XML files.

4. *Generate the Allure Report:*

- After running your tests, you can generate the Allure report using the following command in your project's root directory (assuming you have Allure CLI installed):

allure generate allure-results -o allure-report

This command will create an "allure-report" directory containing the HTML report.

5. *View the Report:*

- To view the report, you can serve it using Allure CLI:

allure serve allure-report

This will start a local web server and open the Allure report in your default web browser.

Now, you should have an Allure report that displays the results of your Selenium WebDriver tests on the "https://hishabee.business/" website. You can explore the report to view test results, logs, and more. Make sure to customize your test runner and configuration based on your project's specific setup.