Candidate Name:MD ATIQUL ISLAM.

Email- atik.cmttiu1001@gmail.com

Qustions of 1st Assessment:

1. Write 10 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:** Here are ten test cases covering different modules, including CRUD (Create, Read, Update, Delete) operations. These test cases are written in a generic format and can be adapted to specific applications:

*1. User Registration:*

- Verify that a new user can successfully register with valid credentials.
- Ensure that the user is redirected to the login page after successful registration.

*2. Login Functionality:*

- Verify that a registered user can log in with correct credentials.
- Check that the system displays an error message for invalid login attempts.

*3. Forgotten Password:*

- Test the "Forgot Password" functionality.
- Verify that a user can reset their password via a password reset link sent to their email.

*4. Create a New Record: (CRUD - Create)*

- Create a new record in a database or application.
- Confirm that the new record appears in the list or database table.

*5. Update an Existing Record: (CRUD - Update)*

- Edit the details of an existing record.
- Ensure that the changes are saved correctly and reflected in the updated record.

*6. View Record Details: (CRUD - Read)*

- Select a record from the list or database.
- Verify that the details of the selected record are displayed accurately.

*7. Delete a Record: (CRUD - Delete)*

- Delete a record from the system.
- Confirm that the record is no longer present in the list or database.

*8. Search Functionality:*

- Test the search functionality by entering a valid search query.
- Verify that the search results match the query and are displayed correctly.

*9. User Profile Update:*

- Test the ability to update user profile information (e.g., name, email, profile picture).
- Ensure that changes are saved and displayed in the user's profile.

*10. Data Export/Import:*

- Export data (e.g., a CSV file) from the system.
- Verify that the exported data contains the expected records.
- Import the exported data back into the system.
- Confirm that the imported data is correctly processed and added to the system.

These test cases cover various aspects of functionality, including user management, data manipulation, and system behavior. Make sure to adapt them to the specific requirements and features of your application or system.

Example of CRUD Opration: Here's a simple Java program that demonstrates CRUD (Create, Read, Update, Delete) operations on a list of objects. In this example, we'll create a program to manage a list of "Person" objects.

java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Person {
    private int id;
    private String name;

    public Person(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Person [ID: " + id + ", Name: " + name + "]";
    }
}

public class CRUDExample {
    public static void main(String[] args) {
        List<Person> personList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("Choose an operation:");
            System.out.println("1. Create");
            System.out.println("2. Read");
            System.out.println("3. Update");
            System.out.println("4. Delete");
            System.out.println("5. Exit");
            int choice = scanner.nextInt();
```

```
switch (choice) {
        case 1:
                System.out.print("Enter ID: ");
                int id = scanner.nextInt();
                System.out.print("Enter Name: ");
                scanner.nextLine(); // Consume newline
                String name = scanner.nextLine();
                Person person = new Person(id, name);
                personList.add(person);
                System.out.println("Person created: " + person);
                break;
        case 2:
                System.out.println("List of Persons:");
                for (Person p : personList) {
                        System.out.println(p);
                }
                break;
        case 3:
                System.out.print("Enter ID to update: ");
                int updateId = scanner.nextInt();
                System.out.print("Enter new Name: ");
                scanner.nextLine(); // Consume newline
                String newName = scanner.nextLine();
                for (Person p : personList) {
                        if (p.getId() == updateId) {
```

```
                p.setName(newName);
                System.out.println("Person updated: " + p);
                break;
            }
        }
        break;
    case 4:
        System.out.print("Enter ID to delete: ");
        int deleteId = scanner.nextInt();
        Person deletePerson = null;
        for (Person p : personList) {
            if (p.getId() == deleteId) {
                deletePerson = p;
                break;
            }
        }
        if (deletePerson != null) {
            personList.remove(deletePerson);
            System.out.println("Person deleted: " + deletePerson);
        } else {
            System.out.println("Person not found.");
        }
        break;
    case 5:
        System.out.println("Exiting program.");
        scanner.close();
```

```
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Try again.");
                    break;
            }
        }
    }
}
```

**2.Answer:**

Below is a basic example of a web automation program using Selenium in Python. In this example, automate the process of opening a website, searching for a keyword on Google, and capturing the search results.

```
python

# Import the necessary Selenium libraries
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# Create an instance of the WebDriver (in this case, using Chrome)
driver = webdriver.Chrome(executable_path="path_to_chromedriver")

# Navigate to a website (in this case, Google)
driver.get("https://www.google.com")

# Find the search input element by its name attribute (usually "q" for Google)
search_box = driver.find_element_by_name("q")
```

```
# Enter a search query
search_box.send_keys("Web automation with Selenium")

# Simulate pressing Enter key to perform the search
search_box.send_keys(Keys.RETURN)

# Wait for a moment to ensure search results load
driver.implicitly_wait(10)

# Capture the search results (e.g., print the titles of the top 5 results)
search_results = driver.find_elements_by_css_selector(".tF2Cxc")
for index, result in enumerate(search_results[:5], start=1):
    print(f"Result {index}: {result.text}")

# Close the browser window
driver.quit()
```

**3.Answer:** I can provide you with a simple example of using the Page Object Model (POM) and Behavior-Driven Development (BDD) framework for automated web testing in Java with Selenium and Cucumber. Here's a basic structure:

*1. Project Structure:*

- src/
  - main/
  - test/
    - java/
      - com/

- yourcompany/
  - pages/
    - LoginPage.java
  - stepdefinitions/
    - LoginSteps.java
  - runners/
    - TestRunner.java
- resources/
  - features/
    - login.feature

*2. LoginPage.java (Page Object):*

java

```
package com.yourcompany.pages;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class LoginPage {
    private WebDriver driver;

    private By usernameInput = By.id("username");
    private By passwordInput = By.id("password");
    private By loginButton = By.id("login-button");

    public LoginPage(WebDriver driver) {
```

```
        this.driver = driver;
    }

    public void enterUsername(String username) {
        driver.findElement(usernameInput).sendKeys(username);
    }

    public void enterPassword(String password) {
        driver.findElement(passwordInput).sendKeys(password);
    }

    public void clickLoginButton() {
        driver.findElement(loginButton).click();
    }
}
```

*3. login.feature (BDD Feature File):*

```
gherkin
Feature: Login Functionality

  Scenario: Successful Login
    Given I am on the login page
    When I enter username "myusername" and password "mypassword"
    And I click the login button
```

```
    Then I should be logged in
```

*4. LoginSteps.java (Step Definitions):*

```
java
package com.yourcompany.stepdefinitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;
import com.yourcompany.pages.LoginPage;

public class LoginSteps {
    private LoginPage loginPage;

    public LoginSteps() {
        // Initialize the WebDriver here (not shown in this example)
        // Initialize the loginPage using the WebDriver
    }

    @Given("I am on the login page")
    public void i_am_on_the_login_page() {
        // Navigate to the login page
    }
```

```
    @When("I enter username {string} and password {string}")
    public void i_enter_username_and_password(String username, String password) {
        loginPage.enterUsername(username);
        loginPage.enterPassword(password);
    }

    @When("I click the login button")
    public void i_click_the_login_button() {
        loginPage.clickLoginButton();
    }

    @Then("I should be logged in")
    public void i_should_be_logged_in() {
        // Add assertions to verify successful login
    }
}
```

*5. TestRunner.java (Cucumber Test Runner):*

```
java
package com.yourcompany.runners;

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

```
@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = "com.yourcompany.stepdefinitions"
)
public class TestRunner {
}
```

**4.Answer:** To generate an Allure report for automated tests using the Page Object Model (POM) and Behavior-Driven Development (BDD) framework, you'll need to integrate Allure with your test project and configure it properly. Here are the steps to generate an Allure report:

*1. Add Allure Dependencies:*

In your project's build file (e.g., pom.xml for Maven), add the Allure dependencies:

xml

```
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-cucumber5-jvm</artifactId>
    <version>2.13.7</version>
    <scope>test</scope>
</dependency>
```

*2. Configure Allure in Cucumber Options:*

In `TestRunner.java` class (or whichever class you use to run your Cucumber tests), configure Allure by adding `@CucumberOptions` annotations:

java

```
import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

    features = "src/test/resources/features",

    glue = "com.yourcompany.stepdefinitions",

    plugin = {"io.qameta.allure.cucumber5jvm.AllureCucumber5Jvm"}

)

public class TestRunner {

}
```

*3. Run Your Tests:*

Execute Cucumber tests as you normally would, either using your IDE or a build tool (e.g., Maven). Ensure that the Allure plugin is included in your test execution.

*4. Generate Allure Report:*

After running your tests, generate the Allure report by running the following command from your project directory:

shell

```
allure generate --clean
```

This command will analyze the test results and generate an HTML report in the "allure-report" directory.

*5. View the Allure Report:*

To view the Allure report, open it in a web browser by running the following command:

shell

```
allure serve
```

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

You should now have a nicely formatted Allure report that provides detailed information about your test runs, including test steps, results, and any attachments you may have added in your step definitions.