DOCUMENTATION

Project Name: UI/API testing project.

Programming Language Used: JAVA

Automation Tools: Selenium, Rest Assured

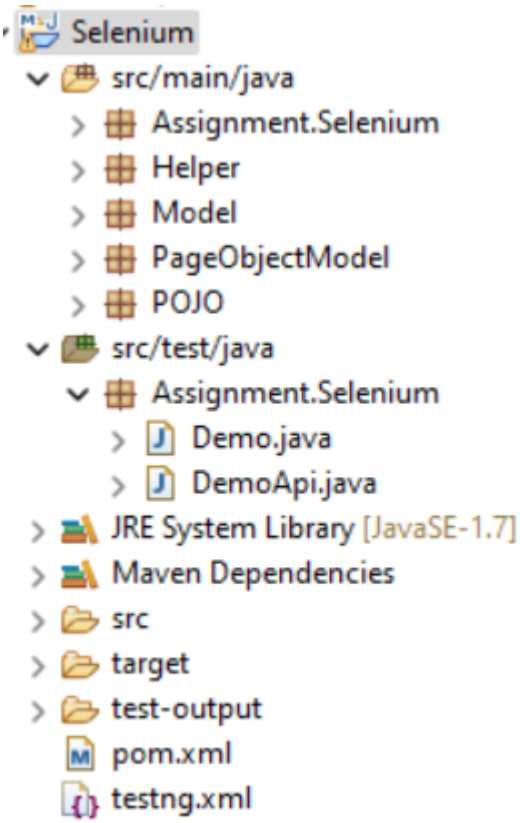
Build Tool: Maven

Unit Testing Framework: TestNG

Reporting: TestNG

DOCUMENTATION

Project Structure:



Scr/main/java: This folder contains all the helper, models, pojo and page object model required for the framework.

Scr/test/java: This folder contains test classes for both UI and API tests.

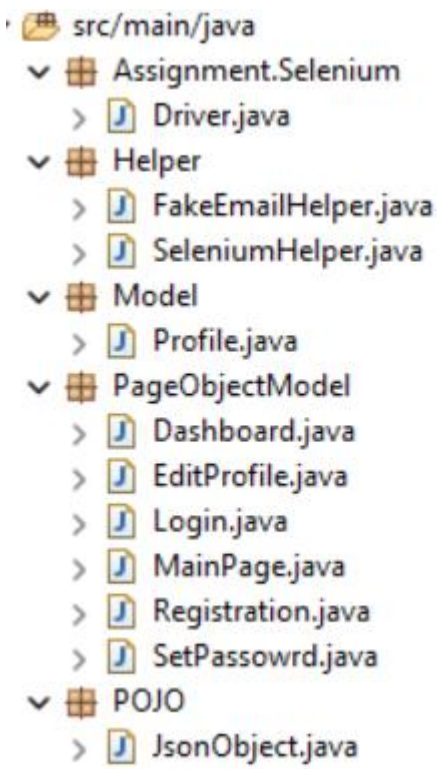
Test-output: This is the folder in which testNG reports are stored.

Pom.xml: All the dependencies for the project are set here**.**

Testing.xml: This is the file that runs the tests in bulk**.**

DOCUMENTATION

Scr/main/java (Packages):



Assignment.Selenium: This package contains Driver class. This class is used to initialize the browser. I am using chrome browser for this project.

Helper: This package contains helpers for the project for random strings, fake email for registration. The fake email helper uses one more chrome driver instance to open fake email in browser to get fake email id.

```
public class FakeEmailHelper {

    static WebDriver secondarydriver;

    public static void invokeDriver()
    {
        WebDriverManager.chromedriver().setup();
        secondarydriver = new ChromeDriver();
    }

    public static String GetFakeEmail()
    {
        invokeDriver();
        secondarydriver.get("https://emailfake.com/");
        String username =secondarydriver.findElement(By.id("userName")).getAttribute("value");
        String domain = secondarydriver.findElement(By.id("domainName2")).getAttribute("value");
        return username + "@" + domain;
    }

    public static String getVerifyAccountLink()
    {
        secondarydriver.navigate().refresh();
        String url =  secondarydriver.findElement(By.xpath(".//a[text()='verify your account now']")).getAttribute("href");
        secondarydriver.close();
        return url;
    }

}
```

DOCUMENTATION

Model: This package contains model for edit profile test case. It contains getters and setters for all properties.

```
public class Profile {
    private String firstName;
    private String lastName;
    private String jobTitle;
    private String nmlsId;
    private String directPhNo;
    private String mobileNo;
    private String address;

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getJobTitle() {
        return jobTitle;
    }
    public void setJobTitle(String jobTitle) {
        this.jobTitle = jobTitle;
    }
    public String getNmlsId() {
        return nmlsId;
    }
    public void setNmlsId(String string) {
```

Page Object Model: This package contains selectors and methods for various pages on the web application. This contains methods for clicking buttons or typing into textboxes.

```
package PageObjectModel;

import org.openqa.selenium.By;

public class EditProfile extends Driver {

    public void fillFirstName(String firstname) {
        WebDriverWait wait = new WebDriverWait(driver, 90);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("firstName")));
        WebElement firstName = driver.findElement(By.id("firstName"));
        firstName.clear();
        firstName.sendKeys(firstname);
    }

    public void fillLastName(String lastname) {
        WebElement lastName = driver.findElement(By.id("lastName"));
        lastName.clear();
        lastName.sendKeys(lastname);
    }

    public void fillMobilePhone(String mobile) {
        driver.findElement(By.id("mobilePhone")).sendKeys(mobile);
    }

    public void fillJobTitle(String jobTitleName) {
        driver.findElement(By.id("jobTitle")).sendKeys(jobTitleName);
    }

    public void fillnmlsId(String nmls) {
        driver.findElement(By.id("nmlsId")).sendKeys(nmls);
    }

    public void filldirectPhone(String directPhoneNo) {
```

DOCUMENTATION

POJO Class: This package is used for API testing with Rest Assured. Getters and Setters for various properties for the API are created in this class. The advantage of this pojo class is that there is no need to deserialize API response with JsonPath.

```
1  package POJO;
2
3  public class JsonObject {
4      private String name;
5      private int age;
6      private long count;
7      public String getName() {
8          return name;
9      }
10     public void setName(String name) {
11         this.name = name;
12     }
13     public int getAge() {
14         return age;
15     }
16     public void setAge(int age) {
17         this.age = age;
18     }
19     public long getCount() {
20         return count;
21     }
22     public void setCount(long count) {
23         this.count = count;
24     }
25
26 }
27
```

DOCUMENTATION

Src/test/java:

Demo.java: This is the UI test class. The tests are based on selenium with testNG. The @beforeMethod annotation method runs before every test method to initialize the browser and the @afterMethod annotation method runs after every test method to close the driver instance. All the methods with @Test annotation are test methods, priority attribute specifies the order of tests in the test class.

```
public class Demo {
    WebDriver chromeDriver;
    MainPage mainPage = new MainPage();
    Registration registrationPage = new Registration();
    SetPassowrd setPasswordPage = new SetPassowrd();
    Dashboard dashboardPage = new Dashboard();
    Login loginPage = new Login();
    EditProfile editProfile = new EditProfile();
    String emailId;
    String passsword;

    @BeforeMethod
    public void setup()
    {
        Driver driver = new Driver();
        driver.intializeDriver();
        chromeDriver = Driver.driver;
    }

    @AfterMethod
    public void tearDown()
    {
        chromeDriver.quit();
    }

    @Test(priority=1)
    Run | Debug
    public void testRegistration() {
        emailId =  FakeEmailHelper.GetFakeEmail();
        chromeDriver.get("https://floify.com/");
        mainPage.clickTryItOutButton();
        SeleniumHelper.switchToLastWindow(chromeDriver);
        registrationPage.FillFirstName(SeleniumHelper.getRandomString());
```

DOCUMENTATION

DemoAPI.java: This is API test class. I am using rest assured API test tool for API tests. API test class also uses testNG for running and debugging tests. TestNG assertions are used to validate the response from the API. Rest Assured assertion is using to verify the status code.

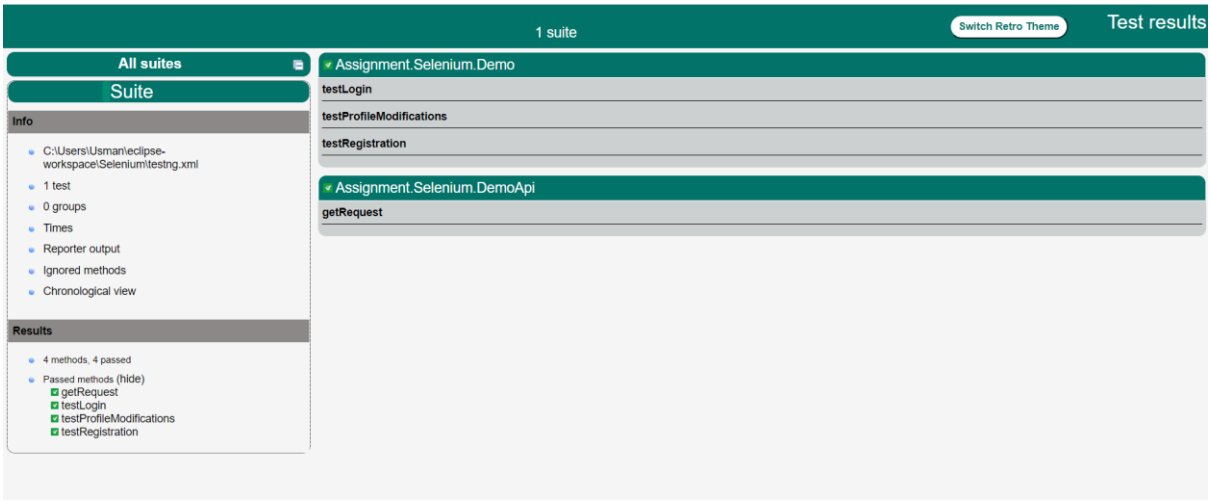
```
un All
ublic class DemoApi {

    @Test
    Run | Debug
    public void getRequest()
    {
        RestAssured.baseURI ="https://api.agify.io";
        JsonObject response = given().queryParam("name", "bella")
        .header("Content-Type","application/json")
        .when()
        .get()
        .then()
        .assertThat().statusCode(200)
        .extract().body().as(JsonObject.class);
        assertEquals(response.getAge() , 35 , "incorrect age");
        assertEquals(response.getName() , "bella" , "incorrect name");
        assertEquals(response.getCount() , 40138 , "incorrect count");

    }
```

DOCUMENTATION

Test-output: This folder is reporting folder, it is testNG reporting. The index.html file contains the reports. It displays all passed and failed tests along with the reason for failure.



testng.xml: This file is used to run the tests in bulk. Parallel tests can be run if thread count specified in the file, parameters for test classes can also be set in this method.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3  <suite name="Suite">
4    <test thread-count="5" name="Test">
5      <classes>
6        <class name="Assignment.Selenium.Demo"/>
7        <class name="Assignment.Selenium.DemoApi"/>
8      </classes>
9    </test> <!-- Test -->
10 </suite> <!-- Suite -->
11
```

Test Execution Steps

1. Right Click on testng.xml file.
2. Hover on Run as and select Testng Suite. Test execution will start.
3. After test execution refresh the project.
4. Navigate to test-output folder and open index.html file to view report of tests run.