**Git:**

* Clone the git url [**https://github.com/vamsilathab/AvaamoProject.git**](https://github.com/vamsilathab/AvaamoProject.git)
* Import from master branch

**Framework:**

* All driver related common methods are defined under package – **driverFactory**
* All data providers property file and excel file readers are defined under package – **dataProvider**
  + **ConfigDataProvider** – Will help in retrieving info from properties file
  + **ExcelDataProvider** – Will help in retrieving info from excel file and returns in two-dimensional String array format.
  + **ExcelDataSetter** – This class will pick the info from ExcelDataProvider and using POJO methods, it sets the data to HashMap which is used in test methods in fetching input and expected output data without hardcoding. It will help in achieving encapsulation mechanism to set data dynamically.
* For common functionality which are required all the tests are maintained under **Utility** package.
  + **AppUtils**.java - For common methods used by selenium
  + **ApiAppUtils**.java – For API methods
  + **Helper**.java – CapturingTestScreenshot

**Folder Structure:**



**POM: Execution**

1. **Initiate tests from testNG.xml -> IRAAIAgentProject.xml**
2. All the tests are maintained under **Test Class**
   * /AvaamoProject/src/test/java/testcases/IRAAIAgentTest.java
3. Respective page related functional methods are maintained under **Page class**
   * /AvaamoProject/src/test/java/pages/IRAAIAgentPage.java
4. And all the **locators** are maintained as constants in an interface
   * /AvaamoProject/src/main/java/utility/LocatorUtils.java
5. All **urls** or configs are maintained under property file
   * /AvaamoProject/Config/Config.properties
6. **Testdata** in
   * /AvaamoProject/test-data/Testdata.xlsx
7. For **reporting** we have used extent reports, it is generated as a html file.
   * /AvaamoProject/Reports/IRAReport.html
8. Captured screenshots for failed testcases will be attached to the report, can also found under
   * /AvaamoProject/Screenshots

**Sample Failure Report:**





**Assignment Status:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Automation Tests** | **Test Method** | **Status** |
| 1 | Welcome Screen - Click on Notification, get that text and get into the bot. | verifyWelcomeMsg  verifyNavigationToBot | **Done** |
| 2 | Type any greetings message and fetch the response message | verifyBotResponse | **Done** |
| 3 | Menu Options click on the “Start Over” button > Click on Download Motor Policy and Click on Download link | verifyDownloadMotorPolicy | **Done** |
| 4 | Type "Test Bot" > Fill the Form and Submit | validateFormSubmission | **Done** |
| 5 | Type "New Test" > Click on Links “Google” and close the web view. Click on call and Close the call popup | validateGoogleCallLinks | **Done** |
| 6 | Please try at least 5 different questions in the Automation script – | verifyBotResponse | **Done** |
| 7 | verify renew policy  2) Use policy number, vehicle registration number and phone number | verifyRenewPolicy | **Done** |

|  |  |  |
| --- | --- | --- |
| S.No. | **Other Tasks** | **Status** |
| 1 | Demonstrate Automation Framework Building Capabilities in the assignment | **Done (refer page 1 for details)** |
| 2 | Capture report and screenshots for failures | **Done** |
| 3 | Report should capture:  - Couple of messages sent and received by Bot (Which is helpful for debugging) | **Partially Done (Optimization required)** |
| 4 | Report should capture:  Number of passed and number of failed test cases with proper screenshots and reasons for failure. | **Done** |

**REQURIED OPTIMIZATIONS:**

* Tests can be run in parallel instead of serial execution, without being dependent on every test.
* Can enable dynamic browser inputs using mvn config (as of now hardcoded to chrome for testing purpose)
* Negative test scripts for each scenario by passing invalid inputs / null values
* Fine tuning test data sheet