**Test Yantra Software Solutions India (P) Ltd**



**DOCSAPP**

**TEST AUTOMATION**

**About DocsApp TeleMER and its challenges:**

DocsApp TeleMER is a web application for insurance verification. It has 3 logins:

1. Ops – Through ops login applicants insurance cases are created and assigned to a doctor for diagnosis
2. Doctor – Doctors login to application and diagnose the cases assigned to them.
3. Vendor – Vendors are the insurance companies and these vendors will be having applicants applying for insurance. Once the applicant registers with a vendor. The vendors send their details to ops team for diagnosis and verification.

**Challenges:**

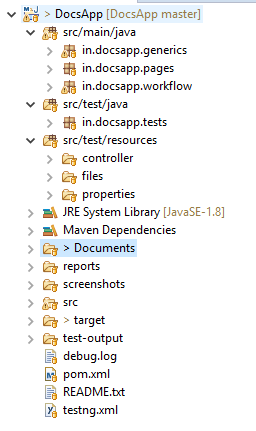
1. As the cases are to be created based on the applicants information. It needs more data to be entered into the application. And every time the data is dynamic because of different applicant
2. Cases are to be diagnosed by the doctors and as there are different vendors, so the diagnose questions will be different based on vendors
3. As there are 3 different login and each case information should be same throughout all three login

**Framework:**

Based on the application, its workflow and challenges. We implemented the hybrid framework which is combination of:

1. Data driven – As it has too much dynamic data to be entered into the application.
2. Modular driven – As it has 3 login portals. So, if we can go for modular we can easily manage the modules. If there are any enhancements or modification of extra features. It can be easily managed and maintained

**Design**:



It has four packages:

1. **Generics** – It has all the generic classes like GenericMethods, Assertion, ExcelLibrary, ExtentReports
2. **Pages** – It is an element repository and has classes based on the modules
3. **Workflow** – It has the ServiceLib class, which has the application based generic methods. Which performs integration tasks.
4. **Tests** – It has test cases scripts

**Folders**:

1. **Controller** – It has an excel file from which we can control the complete automation execution.
2. **Files** – It has all the supporting files for the execution
3. **Properties** – It has the config properties file which has path of other files
4. **Reports** –It has execution status in ExtentReport format
5. **Screenshots** – It has the failed test screenshots
6. **Target** – It has the surefire reports
7. **Test-output** – It has TestNG reports

**Advanced Implementation**:

1. **Browserstack** – Test scripts can be executed on different platforms on cloud
2. **Jenkins** – It is integrated with Jenkins for ease of execution.

**Execution Process**:

The execution can be controlled through an execution\_controller.xlsx file. By changing the data in it. And once the required data is changed in this excel. The scripts can be executed on local system or on cloud i.e, browserstack and it can be triggered from either batch file or from Jenkins.

These 4 automation test cases: [DocsApp - Automation Test Cases.xlsx](DocsApp%20-%20Automation%20Test%20Cases.xlsx) are scripted in this framework for demo.

**Future plugins and implementations**:

1. Automation of triggering complete execution status reports after execution
2. Integration of API