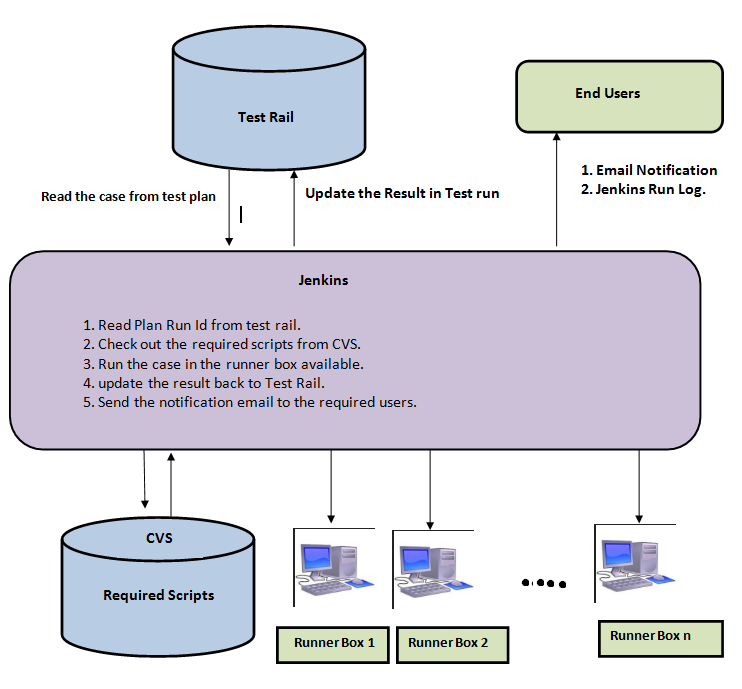
High Level Automation Design Framework:

Below image describes the High Level architecture of our selenium automation framework, which I implemented:

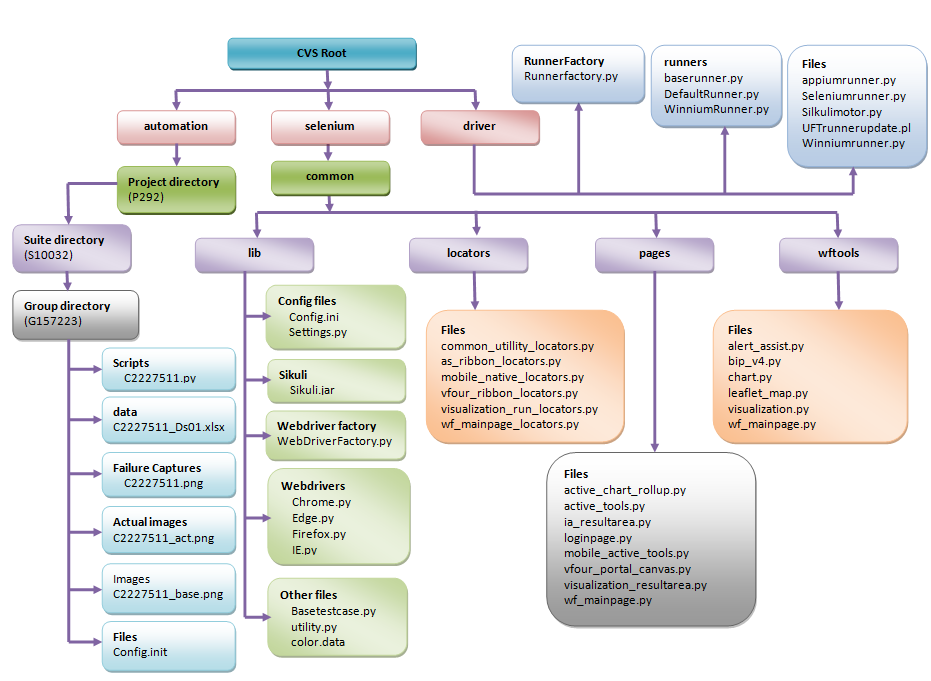
****

File Level Automation Design Framework:

**We follow Page Object Model for the following reasons:**

1. Design simplicity.
2. Easy mantainance. If a change is needed globally, then we can change in one place and the affect will be globally.
3. Easily scalable for frequent change in product behavior,

**Please refer bellow image**:



CVS Root: our automation codebase belongs to /qa directory in CVS. This contains two base folders: selenium and automation.

Followings are the short description:

**A. automation:**

1. Here the base level directory is the project directory. A project may contain several suites.
2. The child level to project directory is suite directory. A suite may contain several groups.
3. The child level to suite directory is group directory. This group directory contains the following directories and files:
   1. scripts: This contains all the test scripts.
   2. data: This contains all the excel sheets or any text files or csv files for data verification.
   3. failure\_captures: This contains the screen shots captured during script failure situation.
   4. actual\_images: This contains all the Actual images captured during run time for image verification.
   5. images: This contains all the base images stored during development for image verification.
   6. Config.init:- This contains the set up info, browser info, etc.

**B. selenium:**

1. Common: All common activities for all the suites.
   1. Lib: This contains all the library files which will be used across all the product areas for scripting. One such example is “**utility.py**”, which contains all the common reusable functions and can be used for all areas of scripting.
   2. Locators: Here we separated the static web element description based on their areas.
   3. Pages: This contains the entire function library for different areas. For visualization we have visualization\_metadata.py, visualization\_resultarea.py, visualization\_run.py…etc.
   4. Wftools: Here we have segregated the entire functions based on the tools, for e.g.. for IA charts – the functions for all the functionalities in charts has been covered in chart.py

**C. driver:**

This contains the configuration related files which are used to run the script or the suite and post the result to the test rail API.