AuQA-ReadMe

Table Of Contents

[Section A: Installations And Cloning the Repo: 2](#_Toc84441464)

[Windows: 2](#_Toc84441465)

[Ubuntu Machine: 2](#_Toc84441466)

[Section B: Test Execution: 3](#_Toc84441467)

[Ubuntu Machine: 3](#_Toc84441468)

[Windows Machine: 3](#_Toc84441469)

[Section C: Framework Folder Structure: 3](#_Toc84441470)

[Section D: GitHub Process: 4](#_Toc84441471)

[Git configuration and Key generation: 4](#_Toc84441472)

[Configure Git 4](#_Toc84441473)

[Generating a new SSH key 5](#_Toc84441474)

[Adding key to Github account: 5](#_Toc84441475)

[Checkout a project from a Git repository using pycharm: 7](#_Toc84441476)

[Process to create the branch and push the branch to github 7](#_Toc84441477)

[Section E: Standards to be followed while coding: 8](#_Toc84441478)

[Section F: Code Review Process and Merging to Master: 9](#_Toc84441479)

[Create Pull Request: 9](#_Toc84441480)

[Merge Pull Request: 9](#_Toc84441481)

[Section G: Looking for Results and Analysis: 9](#_Toc84441482)

[Setting up Apache in Ubuntu machine to view the reports: 9](#_Toc84441483)

[Section H: Stopping the google chrome auto update: 10](#_Toc84441484)

[windows machine: 10](#_Toc84441485)

[Section I: Installation of Google chrome and chrome driver on Ubuntu 10](#_Toc84441486)

[Section J: Updating the Inputs from excel: 11](#_Toc84441487)

[To update the values just for single instance: 11](#_Toc84441488)

[To change the values and save it 11](#_Toc84441489)

[Scripting Note: 12](#_Toc84441490)

[Section K: Scheduling the cronjob: 12](#_Toc84441491)

[Section L: References: 12](#_Toc84441492)

# Section A: Installations And Cloning the Repo:

## Windows:

1. Install python from <https://www.python.org/> and check the installation using python --version

Set the path variable for python

1. Install Pychram community version from <https://www.jetbrains.com/pycharm/download/#section=windows>
2. Clone the project from github

git clone https://github.com/ananiac/AuQA.git

1. Install the required libraries for robotframework:

Navigate to AuQA/Libraries/ and execute installLibrariesWinX.sh using command

bash installLibrariesWinX.sh

1. Install robot framework selenium library for pychram

Navigate to settings-> project interpreter search for robot framework selenium library and install package

1. Install the plugin “intellibot-seleniumlibrary-patched” - > adds smart editing feature to support Robot Framework

Click install

1. Navigate to [https://github.com/lte2000/intellibot and download intellibot.jar](https://github.com/lte2000/intellibot%20and%20download%20%20intellibot.jar) and add it as plugin . settings->plugin->install plugin from disk-> select the file and install
2. Set Up Webdriver : Check the version of the chrome using chrome://version

Download the driver for the version of the chrome from <https://chromedriver.chromium.org/downloads>

Place the unzipped exe file in the location where python is downloaded …../python/script

1. Supporting Plugins for element locator check:

css and xpath checker to chrome

selenium IDE for firefox

1. Install pycharm plugin JS GraphQL, Pylint

## Ubuntu Machine:

Prerequisite:

Python, pip and Git should be installed on the ubuntu machine

Google chrome and respective version of chromedriver should be installed

Automation directory should be created under /home/fc

1. Login to 10.252.9.35 with fc user
2. Navigate to automation folder and clone the repo from github

git clone https://github.com/ananiac/AuQA.git

cd AuQA

1. Install the required libraries for robot framework

Navigate to AuQA/Libraries/ and execute installLibrariesWinX.sh using command

sh installLibrariesUbuntu.sh

# Section B: Test Execution:

## Ubuntu Machine:

cd automation/AuQA

* To execute the specific testcases using pabot

sudo pabot --pabotlib --processes 2 --name "Guard1" --reporttitle "BasicHotAbsoluteGuard" --outputdir Reports --output basichotGuard.xml --variable environment:config37 -v groupname: General-test Testcases/basicHotAbsoluteGuardTest.robot Testcases/staleStatePrevention.robot

where basicHotAbsoluteGuardTest.robot is the file for testcase Guard1 and staleStatePrevention.robot is for stalestate prevention

* To execute the entire suite on machine 37 use tuesdayexecutesuite.sh and on 118 use thursdayexecutesuite.sh

sudo chmod 755 thursdayexecutesuite.sh  
sudo sh thursdayexecutesuite.sh

* Note: Make sure the latest code pulled on the master

git pull

For details refer to the Test Execution .docx in the AuQA repo

## Windows Machine:

* To execute the specific testcases using pabot

pabot --processes 2 --outputdir Reports --variable environment:config37 -v groupname:General-test /home/fc/automation/AuQA/Testcases/basicHotAbsoluteGuardTest.robot /home/fc/automation/AuQA/Testcases/staleStatePrevention.robot

* To execute the entire suite on machine 37 execute executesuiteon37.sh

bash executesuiteon37.sh

# Section C: Framework Folder Structure:

Configuration: contains py file with all the configuration like url, browser specific to the t

ExternalKeywords: contains the python files for any external keywords created using python

Inputs: contains mutation and queries for GraphQL and the inputs required for the each testcase

JsonPath: contains json path

Libraries: contains the sh file to install the required libraries on windows and ubuntu

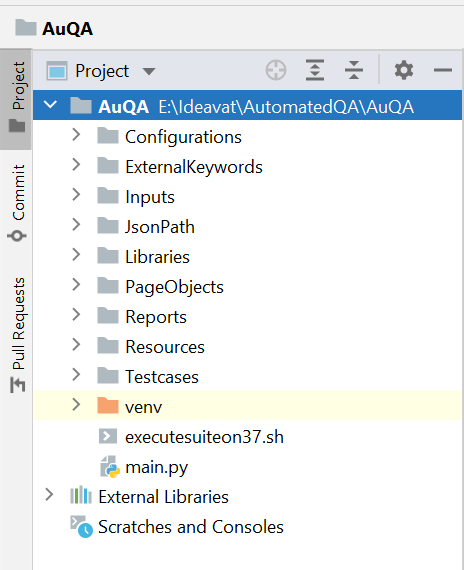
PageObject: contains the pagewise element locator

Reports: contains snapshots in the snapshot folder, log file, output files for robot and the report html file

Resources: contains module wise folder for the internal keywords for the testcases

Testscases: contains all modulewise test suite files

executesuiteon37.sh: contains commands for execute the suite on 37 that be used by the scheduled cronjob on ubuntu machine (10.252.9.35)



# Section D: GitHub Process:

## Git configuration and Key generation:

### Configure Git

* git config --global user.name YOUR\_NAME\_HERE

Ex: git config --global user.name "ananiac"

* git config --global user.email YOU\_EMAIL\_HERE

git config --global user.email "ananiac@ideavat.com"

* git config --global push.default upstream
* git config --global color.ui auto

### [Generating a new SSH key](https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent#generating-a-new-ssh-key)

* Run below

ssh-keygen

Enter file in which to save the key (/c/Users/<username>/.ssh/id\_rsa):[press enter]

Enter passphrase (empty for no passphrase):[Press Enter]

Enter same passphrase again: [Press Enter]

* Ensure the agent is running

eval "$(ssh-agent -s)"

Agent pid <number> will be displayed

* Navigate to the location where your public key has been saved in c:/Users/<username>/.ssh/id\_rsa.pub copy the file content

### Adding key to Github account:

* Login to Github account and navigate to Profile -> setting-> SSH and GPG key
* Click New SSH key
* Add Tittle paste the copied key

Graphical user interface, text, application

Description automatically generated

* Add SSH Key – password prompt will appear – enter the password

Graphical user interface, text, application, email

Description automatically generated

* Should be good to clone from the repo from

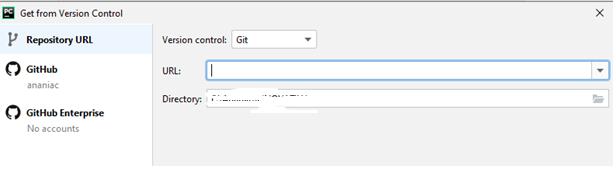
Ex: git clone https://github.com/ananiac/AuQA.git

Text

Description automatically generated

## Checkout a project from a Git repository using pycharm:

* Pycharm- VCS -Enable version control Integration- select Git
* Click on the Always Add option
* From plugins- Version concontrol - github - add account
* From plugins- Version concontrol - git - add the path of git.exe
* To Clone the project from Github:
  + Git -> Clone -> Specify the url of the repository and the directory where the project has to be cloned



* To create the repo in GitHub:

From Git option -Github - share the project on GItHub

## Process to create the branch and push the branch to github

Prerequiste: Repo should be cloned and user must be on the master branch

* create a branch in git

git branch <branchname>

* After working on the branch check all the modified files using

git status

* Add the modified and added files

git add file1 file2

* commit the files

git commit -m “message”

* checkout the master branch

git checkout master

* pull the latest master

git pull

* checkout the branch that you worked on

git checkout <branch name>

* Rebase the branch with the master

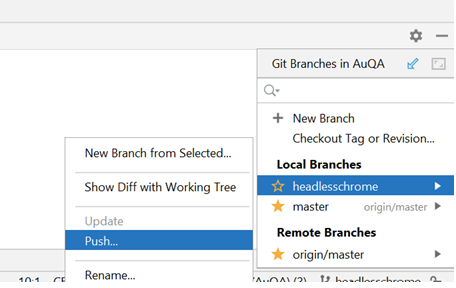
git rebase -i master

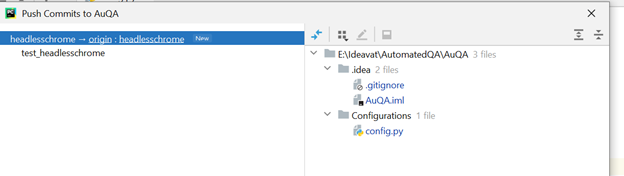
This will open a file , select f (fixup) for the new branch commit, save and close (Shift +Z+Z)

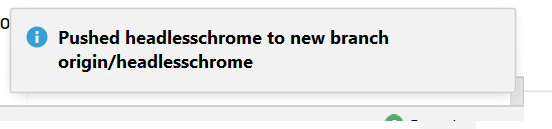
Push the branch to master

* git push --force

or do it from pycharm







# Section E: Standards to be followed while coding:

Naming convention:

* Pascal Case: Ex: ThisIsPascalCase
* Camel case: EX: thisISCamelCase
* Snake case: EX: this\_is\_snake\_case

Naming Rules:

* Folder/directory name will be pascal case
* Files name will be in camel case
* variable name will be in snake case
* All the keywords will be using camel case
* All the test cases will be using pascal case
* All the default robot framework keywords will be used in lowercase ex: *input text*

Data Flow:

* All the data/variables from Configuration/ExternalKeywords/PageObjectgets will be called in the files in Resources using the *Variables and Library under* the settings.
* All the Keywords from the files in Resources will be called in the test files using the *and Resource under* the settings.

Syntax Format:

Two lines space above and below Settings, Variables, Testcases and Keywords

One line space between Keywords

Keywords and comment to start with a Tab

Tab between the keyword and the variable

Two spaces in the keywords

# Section F: Code Review Process and Merging to Master:

## Create Pull Request:

Once the branch is pushed to GitHub follow the below steps

1. Open branch on GitHub
2. Create pull request by clicking **New Pull Request**
3. Enter details about the pull request and click Create pull request.

## Merge Pull Request:

1. Click on Pull Request tab and the branch is good to merge , click Merge Pull Request
2. On successful merge delete the branch by clicking the branch link and click Delete against the branch that is merged.

# Section G: Looking for Results and Analysis:

Results will be generated as the html file and placed under Reports folder as report.html

Standard output/terminal output will be placed under the Reports/pabot\_result /<process index 0 or 1>/robot\_stdout.out

Last Run Reports will be found in the location automation/testReports

And can be accessed using  <http://10.252.9.35/testReports>

## Setting up Apache in Ubuntu machine to view the reports:

log into 10.252.9.35 as fc

ls -l /var/www …. shouldn’t exist if Apache is not installed

ls -l /var/log/apach\* …. shouldn’t exist if Apache is not installed

sudo apt update

sudo apt install apache2

display 10.252.9.35 in a browser

ls /var/www/html … should have index.html (which is displayed in the browser above)

cd /home/fc/automation

mkdir test Reports

cd test Reports

mkdir dummyReport1

mkdir dummyReport2

cd /var/www/html

sudo ln -s /home/fc/automation/testReports/ testReports … make link to testReports directory

# Section H: Stopping the google chrome auto update:

## windows machine:

Go to C:\Program Files (x86)\Google\Update and delete GoogleUpdate.exe

Check by navigating to chrome://settings/help

Graphical user interface, text, application

Description automatically generatedUbuntu machine:

Add the following line to /etc/default/google-chrome:  
repo\_add\_once=false

cd /etc/default

Sudo vim google-chrome

Add **repo\_add\_once=false** to the file

Esc: wq!

# Section I: Installation of Google chrome and chrome driver on Ubuntu

**Download the latest google chrome:**

wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb

**Install chrome:**

sudo apt install ./google-chrome-stable\_current\_amd64.deb

**Installing chrome driver:**

sudo apt update -y && sudo apt-get install -y libxss1 libappindicator1 libindicator7 xvfb unzip

wget https://chromedriver.storage.googleapis.com/87.0.4280.20/chromedriver\_linux64.zip

sudo wget https://chromedriver.storage.googleapis.com/91.0.4472.101/chromedriver\_linux64.zip

https://chromedriver.storage.googleapis.com/91.0.4472.101/chromedriver\_linux64.zip

sudo unzip chromedriver\_linux64.zip

sudo chmod +x chromedriver

**Move chromedriver executable and create symlinks:**

sudo mv -f chromedriver /usr/local/share/chromedriver

sudo ln -s /usr/local/share/chromedriver /usr/local/bin/chromedriver

sudo ln -s /usr/local/share/chromedriver /usr/bin/chromedriver

**Check using:**

cd /usr/local/share/

Ls -la

**setting the chromedriver path in .bashrc :**

cd

sudo vim. bashrc

add the the below code - save and exit

PATH = "/usr/local/bin/chromedriver:${PATH}"

export PATH

**verify the .profile file has the code to call .bashrc file:**

if [ -n "$BASH\_VERSION" ]; then

# include .bashrc if it exists

if [ -f "$HOME/.bashrc" ]; then

. "$HOME/.bashrc"

fi

fi

# Section J: Updating the Inputs from excel:

All the input for the testcases are passed from the excel sheet testInputs.xlsx (AuQA / Inputs/testInputs.xlsx)

Column wise key value pair is associated with the each testcase. Ex: column A and B are associated with Guard1i.e basicHotAbsoluteGuardTest identified with header

|  |  |
| --- | --- |
| basicHotAbsoluteGuardInputs | values |

## To update the values just for single instance:

Fetch the latest master and open the file Inputs/testInputs.xlsx, update the changes, save and run

## To change the values and save it

Create a branch, Update the values in Inputs/testInputs.xlsx

Follow the steps under section:

Process to create the branch and push the branch to github

Code Review Process and Merging to Master

Take the master run with changes

## Scripting Note:

readingInputsFromExcel is the robot keyword used to read the inputs from excel as dictionary ${test\_input}.Keyword takes sheet name as first argument and the column name like A, B for the key and value.

readingInputsFromExcel need to be called in the setup of each testcases

ex: testInputs.readingInputsFromExcel guardTest A B called in basicHotAbsoluteGuardTestResources.robot and access the values like

${test\_input} [ num\_guard\_units]

# Section K: Scheduling the cronjob:

To schedule the run using the cronjob use cron.d

cd /etc/cron.d

update the file with the schedule using

sudo vim run-auqa-tests

ex: add below to schedule the Wednesdaysuite at 7:00 am pdt

#Add comments

0 7 \* \* 3 root sudo /home/fc/automation/AuQA/wednesdayexecutesuite.sh >> /home/fc/automation/AuQA/wed.txt 2>&1

# Section L: References:

<http://robotframework.org/robotframework/#user-guide>