**Full Stack Automation Project**

[Short Video - Demonstration](https://drive.google.com/file/d/1ddsjZ5Gh9PACZDX_-DlhT223e2kquy_j/view?usp=sharing)

***This project created to demonstrate my knowledge and skills in Automation Testing.***

***About***

The project demonstrates a smart automation infrastructure that allows us to test different kinds of applications and is easily maintained! This project is based on selenium, TestNG, maven, allure report, and java version 8. Can execute our automated tests inside a docker container using Jenkins, GitHub, DockerHub, and Selenium Grid.

**Requirements**

To run test on Firefox or Chrome browsers with docker you need to install:

* docker
* docker-compose
* selenium grid

**Quick run test**

provide 3 parameters

* HUB\_HOST = IP address of selenium grid
* Browser = Chrome/Firefox
* Module = support 4 modules

\* commerce-web-module.xml

\* commWeb-regression-module.xml

\* map-api-module.xml

\* search-duck-module.xml

docker run -e HUB\_HOST=localhost -e BROWSER=chrome -e MODULE=commerce-web-module.xml darkklin/finalproject

**Project Structure**

Automation-Project/

├─ Configuration/

│ ├─ DataConfig.xml

├─ DDTFiles/

│ ├─ product.csv

| ├─ keyword.csv

├─ Drivers/

│ ├─ electrondriver.exe

├─ ImageRepository/

│ ├─ logo01.png

├─ src/

│ ├─ main/

│ │ ├─ java/

│ │ │ ├─ extensions/

│ │ │ | ├─ ApiAction.java

│ │ │ │ ├─ DataBaseAction.java

│ │ │ │ ├─ LoadAPI.java.java

│ │ │ │ ├─ MobileAction.java

│ │ │ │ ├─ UIActions.java

│ │ │ │ ├─ Verifications.java

│ │ │ ├─ pageObjects/

│ │ │ │ ├─ calculator/

│ │ │ │ │ ├─ MainPage.java

│ │ │ │ ├─ duckduckgo/

│ │ │ │ │ ├─ DuckduckgoPage.java

│ │ │ │ ├─ mortgage/

│ │ │ │ │ ├─ MainPage.java

│ │ │ │ ├─ toDo/

│ │ │ │ │ ├─ MainPage.java

│ │ │ │ ├─ nopCommerce/

│ │ │ │ │ ├─ CheckoutPage.java

│ │ │ │ │ ├─ HeaderPage.java

│ │ │ │ │ ├─ LoginPage.java

│ │ │ │ │ ├─ MainPage.java

│ │ │ │ │ ├─ ProductPage.java

│ │ │ │ │ ├─ RegisterPage.java

│ │ │ │ │ ├─ RegisterPage.java

│ │ │ │ │ ├─ MyaccountPage.java

│ │ │ │ │ ├─ ShoppingCartPage.java

│ │ │ ├─ utilities/

│ │ │ │ ├─ Base.java

│ │ │ │ ├─ CommonOps.java

│ │ │ │ ├─ Listeners.java

│ │ │ │ ├─ ManagePages.java

│ │ │ │ ├─ ManageDDT.java

│ │ │ │ ├─ MonteScreenRecorder.java

│ │ │ ├─ utilities/

│ │ │ | ├─ actions/

│ │ │ │ │ ├─ WebFlowActions.java.java

│ │ │ | ├─ DuckduckgoFlow.java

│ │ │ | ├─ MapApiFlows.java

│ │ │ | ├─ DuckduckgoFlow.java

│ │ │ | ├─ MapApiFlows.java

│ │ │ | ├─ ToDoFlows.java

│ │ │ | ├─ WebFlows.java

│ │ │ | ├─ WindowsCalculateFlows.java

│ ├─ test/

│ │ ├─ java/

│ │ │ ├─ sanity/

│ │ │ │ ├─ CalculaterWindows.java

│ │ │ │ ├─ CommerceWeb.java

│ │ │ │ ├─ MapAPI.java

│ │ │ │ ├─ MortgageMobile.java

│ │ │ │ ├─ ToDoListElectron.java

│ │ │ │ ├─ DuckduckgoWeb.java

│ │ │ ├─ regression/

│ │ │ │ ├─ CVVTest.java

├─ .gitignore

├─ Dockerfile

├─ Jenkinsfile

├─ healthcheck.sh

├─ pom.xml

├─ README.md

├─ commerce-web-module.xml

├─ commWeb-regression-module.xml

├─ electron-module.xml

├─ map-api-module.xml

├─ search-duck-module.xml

├─ windows-module.xml

***Project Overview***

The project is an example of infrastructure for automation testing of different kinds of applications:

* Web based application
* Mobile application
* Web API
* Electron application
* Desktop application

***Infrastructure project includes using of:***

* Page Object Design Pattern
* Project Layers(Extensions/Work Flows/Test Cases...)
* Using Lambda & Streams Functional Style Programming
* Support of Different Clients/Browsers
* Failure Mechanism
* Common Functionality
* External Files Support
* Reporting System (including screenshots)
* Visual Testing
* DB support
* CI support

***List of applications were used in this project:***

* NopCommerce store - eCommerce platform
* DuckDuckGo - web search engine
* Mortgage calculator - Mobile application
* DemoGoogleMap API - Web API
* Electron application
* Windows calculator - Desktop application

***Tools & Frameworks used in the project:***

* TestNG - Testing Framework
* Maven - project management
* Listeners - interface used to generate logs and customize the TestNG reports
* [Jenkins](https://github.com/darkklin/Automation-Project/blob/268537637089b8259eb37648d989adf2fd800ca3/docker-image)- for tests execution
* Selenium grid - run tests in parallel on multiple machines
* REST Assured - for API testing
* [Allure](http://allure.qatools.ru/) Reports - as the main reporting system

**Tests Execution:**

Each of the applications has a few tests for demonstration purpose. These tests can be developed in a very simple way, due to a lot of work with the infrastructure. [[Sanity Tests]](https://github.com/sashka999/FullStackFinalProject/tree/master/src/test/java/SanityTests)

***Known Issues:***

Test Body section is missing from Allure report while using with Jenkins. Possible solution - [allure-framework/allure2#1041](https://github.com/allure-framework/allure2/issues/1041)