TELL ME ABOUT YOURSELF

CYDEO

Tell me about yourself

- *Appreciate the interviewer's* ***time***
- *Appreciate the* ***opportunity*** *given to you*
- *Tell the person your* ***name****, or any* ***nick name*** *you prefer*
- *Provide IT* ***experience time box*** *(6month? 2 yrs? 4 yrs?)*
- *Mention one of the* ***title*** *you like - SDET? Automation engineer?*
- *Could add* ***industry*** *category -> bank, insurance? CRM? Communication?*
- ***Application types*** *-> web-based ? Mobile?*
- *Skill set on manual or automation ? -> both*
- *Programming* ***language*** *-> Java*
- ***Job experience*** *[application layers and corresponding testing tools]*
- *Other responsible* ***testing types***
- *Work environment -> implemented* ***Agile-Scrum***
- *Personality (optional)*

CYDEO

Sample tell me about yourself answer:

Thank your for giving me this opportunity , and I appreciate your time !

As you already know, my name is __ . (You can call me __)

I have been working in IT industry more than ___(number of months or years) as an SDET / Automation engineer.

During these years, I have worked on communication and finance industry.

Through out my experience, I have been successfully tested web-based applications both manually and automatically.

My main programming language is Java. [could mention java version]

*I have automated **frontend** with Maven, Selenium, Junit and TestNG tools ;*

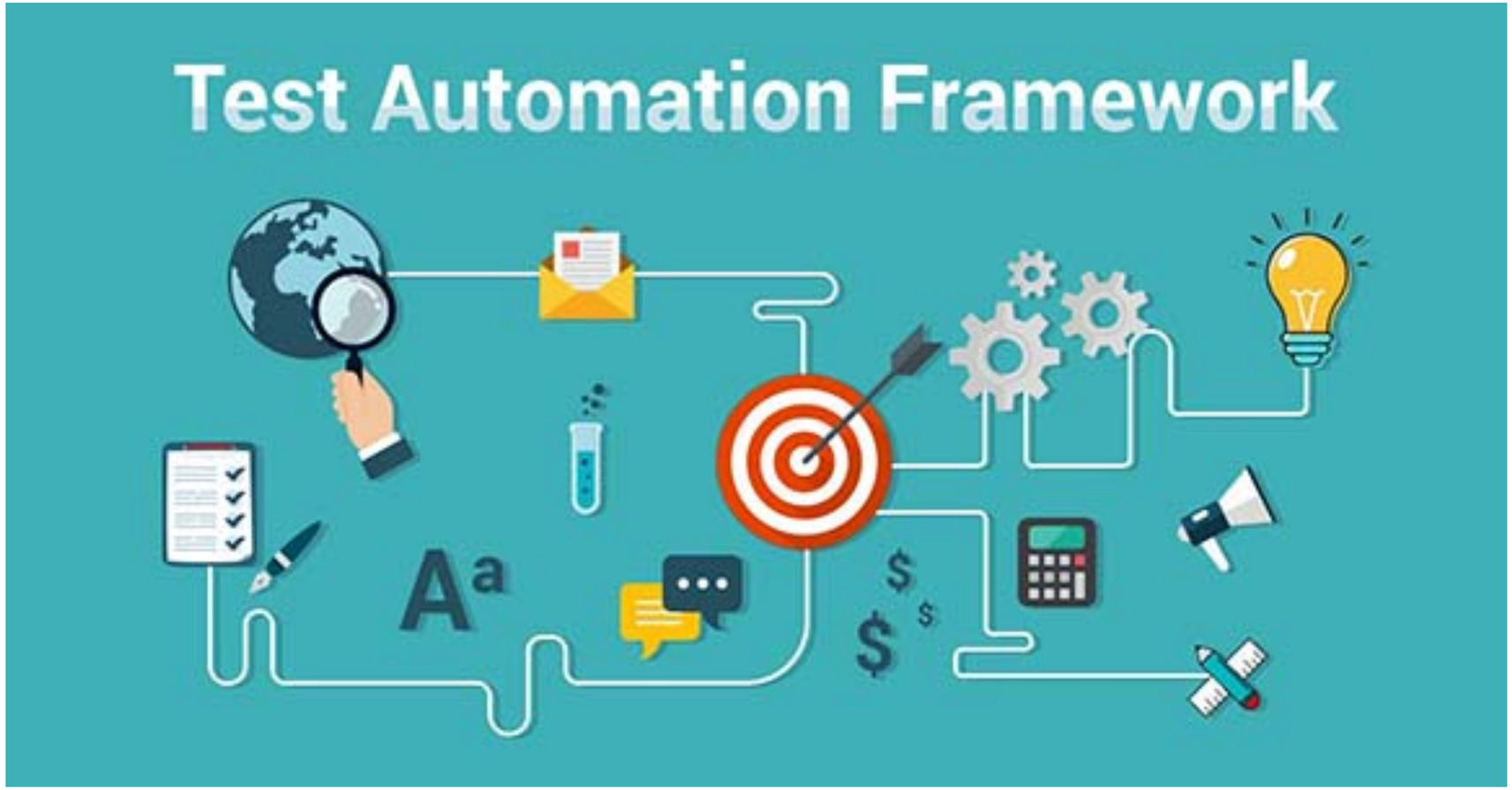
*I have used JDBC tool for **Database** testing;*

RestAssured library / tool for automating the API.

*I also performed **Smoke**, **regression** testings in a regular basis.*

*I worked in **Agile-Scrum environmen**t in past ___(months/years), so I am very confident in all Scrum meetings.*

Personally, I really like to learn new things and share my knowledge with others. I am also a detail oriented person who sticks to the deadline.

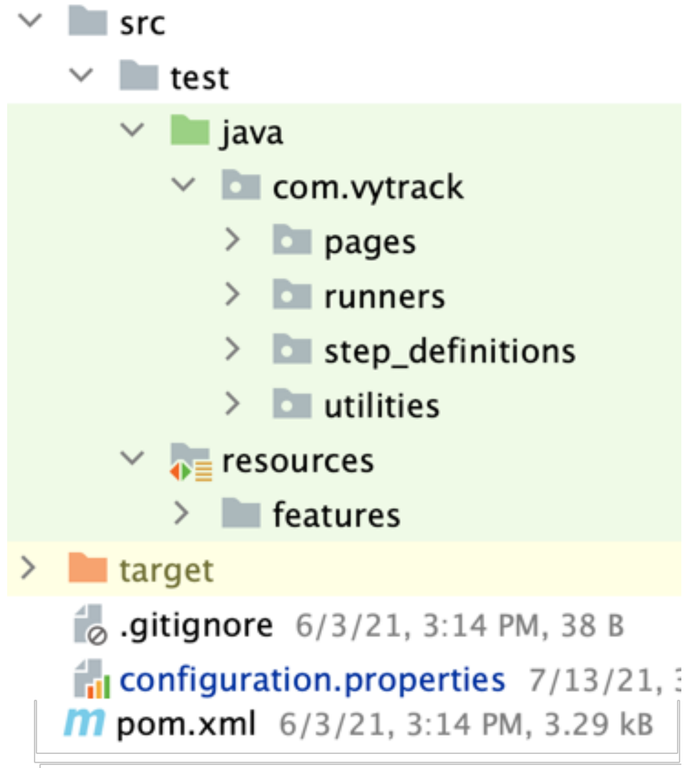


CYDEO

Tell me about your framework ?

When I assign you a user story, tell me how do you automate it in your framework?

- Framework type
- Main language
- Build tool
- Packages + files
- Design patterns
- Test data / resource locations
- Capture test reports & screenshots
- Version control used



CYDEO

- My Framework is: Behavior Driven Development (**BDD**) **Cucumber Framework**
- I used **Java** as my programming **language** in my framework.
- I used **maven** as my **built** automation **tool** WHICH HAS **pom.xml** file that contains info about my project and allows me to manage my dependencies easily.

- I implement BDD approach to simplify reading and understanding my framework for the non-technical teammates in my team.
    - **Resources** directory - **Features package**:

  [I write my test cases as if they are **scenarios** from the end users perspective in **Gherkin language** in my **feature files** under the **features package**.]

  [Features folder is used to store feature files. Every feature file contains Scenarios or/and Scenario Outlines. Every scenario consist of test steps. Every test step has code implementation - step definition method, that turns the phrase into an action.]

- **Step - Definition** package

[I implemented the actual coding logic inside of my **step_definitions package** and it their own respective/related classes.]

[In step definition classes, we instantiate page classes, so we **create page objects** that allows use to interact with page elements. Every element can be wrapped into web element object. We are making all web elements and locators private to separate page logic from the test. Thus ,we create another level of abstraction. Step definitions are not responsible for web element issues. Everything should fixed in page classes.]

- **Hooks** class file under the step-def package

[Hooks class where we implement some cucumber annotations such as Before, After, beforestep, afterstep to create outline for my scenarios.]

- **Runners** package

[I trigger my framework from my runner class.

Cucumber runner class must contain **@RunWith** And **@CucumberOptions** annotations If you are using cucumber along with junit4 .

Runner class allows me to run different types of testing suites that I created with **my tags**, such as smoke, regression, mini-regression. I have different **types of reports**. But mainly I use "maven-cucumber-reporting" which is a very detailed reporting tool that has pie-charts, matrixes on which tests are passing and failing.] In step definition classes, we instantiate page classes, so we create page objects that allows use to interact with page elements. Every element can be wrapped into web element object. We are making all web elements and locators private to separate page logic from the test. Thus ,we create another level of abstraction. Step definitions are no responsible for web element issues. Everything should fixed in page classes.

- **Utility** package

[Utilities package is used to **store reusable code**. We have utility classes based on categories: BrowserUtilities, Excel, DataBase, etc…

Example: There is a file that reads config.properties files called : **configReader**

- opening the file and passing the path of the file into FileInputStream
- loading the file to properties class object.

The **configuration.PROPERTIES,** it is a type of file where I keep the important test data about my framework. I keep Test data that can change the running flow of the whole framework, such as:

- browser
- username/password
- url: to change and run on different environments
- DB connection , API base URL

- Design Patterns
    - **Singleton** Design patterns

[I Created Singleton Design Pattern to allow my framework to pass the same instance of my webdriver in one session.]

Driver class example:

- **Page Object Model** design pattern

POM: As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

- Base Page
- PageFactory
- @FindBy

Test Data: All the historical **test data** will be kept in an excel sheet (*controller.xlsx*) that we store under the resource directory, we pass test data and handle data-driven testing. We use **Apache POI** to handle excel sheets.

- Dummy data - Generate from google, java faker
- Developers provide / DB
- Any data team provide

My BDD cucumber Framework is a java/maven project, that we are using for UI (or both UI & Backend, or BackEnd) test automation.

Selenium Webdriver is used to interact with a browser.

JDBC is used to connect with DB and get data result.

RestAssured is used to perform API testing

It's a **hybrid test automation framework** because we're using data driven and behavior driven frameworks at the same time. For storing locators we are following page object model. Basically, we create corresponding page classes for every page of our application. To develop test scenarios we use Cucumber BDD and every single test is written from end-user perspective. We have a base page class that is used as a super class for all page classes. It contains all common locators and initializes page factory. Also ,we have a utilities package were we keep all reusable code. To run tests, we just simply trigger cucumber runner class. For regression suite and smoke test we have a dedicated maven profiles. As a reporting tool we use maven- cucumber reporting plugin. For remote test execution we use Selenium Grid and couple of virtual machines. We use git as a version control system and GitHub as a remote repository.

Data Driven Framework:

A Data Driven framework is a **technique** of separating the “data set” from the actual “test case” (code).

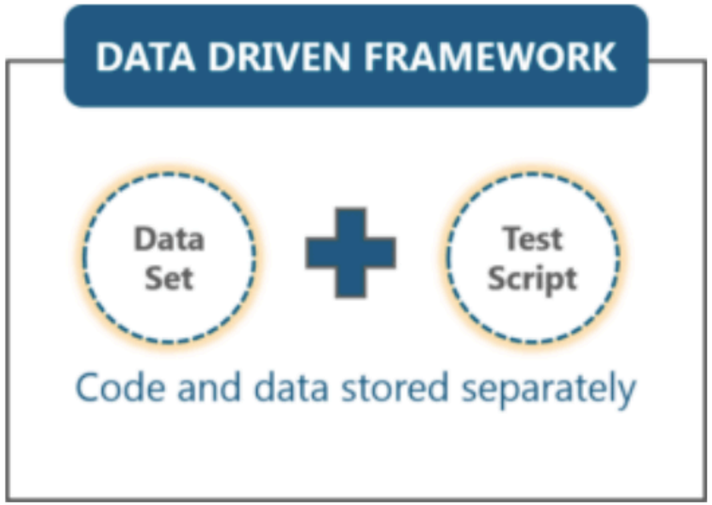
This framework completely depends on the input test data.

The test data is fed from external sources such as:

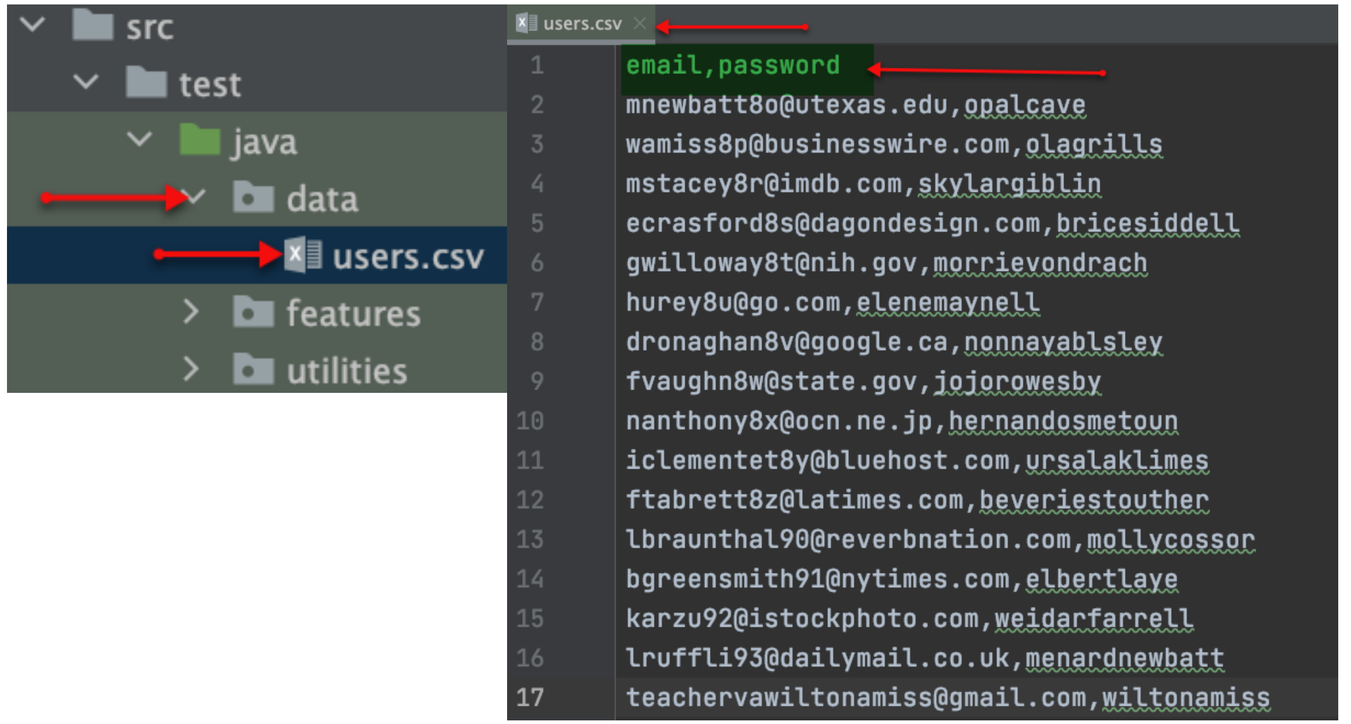
- Excel file
- Example table in the Scenario outline
- Data from the database

We can achieve Data-driven framework using :

- TestNG's data provider
- Excel apache poi
- Cucumber scenario outlines
- DataBase testing



Achieving the Data driven framework: Save the data set into a CSV file under the Dara folder example:



Achieving the Data driven framework: Save the data set as example table

```
@scenarioOutline
Scenario Outline: Wikipedia search header verification
  Given User is on Wikipedia home page
  When User types "<searchValue>" in the wiki search box
  And User clicks wiki search button
  Then User sees "<expectedMainHeader>" in the main header
  Then User sees "<expectedTitle>" is in the wiki title

  Examples: search terms we are going to search in wikipedia
    | searchValue   | expectedMainHeader | expectedTitle |
    | Steve Jobs    | Steve Jobs         | Steve Jobs    |
    | Kuzzat Altay  | Kuzzat Altay       | Kuzzat Altay  |
    | Kobe Bryant   | Kobe Bryant        | Kobe Bryant   |
    | Elon Musk     | Elon Musk          | Elon Musk     |
    | Bill Gates    | Bill Gates         | Bill Gates    |
    | Chuck Norris  | Chuck Norris       | Chuck Norris  |
    | Marie Curie   | Marie Curie        | Marie Curie   |
```

CYDEO