Tell me about yourself

- **Me:**

  Been in the IT industry for about 5 years. Started as a manual tester and now doing automation for about 3 years.

- **Domain:**

  I have experience in different domains: education, fleet management, cloud storage.

- **Certificates:**

  I have Scrum Master and Oracle Java Programmer certificates.

- **Language:**

  My main language is **JAVA**, designed and developed many automation test scripts by using **Data Driven, Behavior Driven,** and **Hybrid Frameworks.**

- **Automation Tools:**

  I used **Selenium WebDriver**, **Maven**, **Cucumber**, **JUnit**, **TestNG**, **Jenkins** and some other tools for test automation.

- **Database and API:**
  - Manual: **SQL Developer** to produce SQL queries.
    Automation: We use **JDBC** in our framework for automation.
  - Manual: I used Postman for manual API testing
    Automation: Used RestAssured Library for deserialization and serialization.

- **Personality:**
  - I'm a team player and do really value teamwork. I spend special effort to have better relationships with my team members as I believe this will improve the quality of work and productivity. That is why I'm not interested in any short term jobs. It takes time to build a team spirit.
  - I'm really detail oriented, you will see the shirts in my closet are color coded :)
  - I'm open to learning new tools and skills that is why I want to start a new chapter in your company as there are many opportunities.

Current Company/Project:

- ➔ **Company:** I work for MeetSky which is an internally cloud-based document management and collaboration platform that provides productivity solutions for businesses focused on additional security, privacy and controls.
- ➔ **Current Module:** Currently I'm working on the File Module which allows users to store and share files, collaborate on the same documents, leave feedback or comments, direct messages to co-workers or create groups, web meetings, etc.
- ➔ **New Feature:** Recently, we added the feature that now users can leave video messages as well.
- ➔ **Size:** The size of the company is about 60 people.
- ➔ **Role:** I work as an SDET on a testing team focused on continuous testing of the features, upgrades, and bug fixes in product development.
- ➔ **Client:** Companies that are implementing new document management and collaboration solutions for their teams. Companies like accounting firms, looking for more security and privacy.
- ➔ **API**: Sending test data, creating checklists and verifying them, adding announcement boards and verifying them, etc.
- ➔ **SQL**: Checking files in the database if they are created or updated.
- ➔ **Achievement**:
    - ◆ I helped automate 80 % percent of the application including the smoke/ regression test cases.
    - ◆ I used to send all test data (vehicles) via API in my previous company, so suggested my team create our own test data and send it with API. We kept all data in an excel doc and started to create all users from there.

Tell me about your Framework!

- ➔ It's a Cucumber-JUnit framework built by Maven.
- ➔ It is written in Java, Hybrid: Data Driven & BDD framework.
- ➔ For **UI testing,** we use **SELENIUM WebDriver**; for **API** testing we use **RestAssured**, and for Database Testing we use the **JDBC** libraries.
- ➔ Generates **HTML REPORTS** with all the test steps, **screenshots** and the test metrics.
- ➔ It is structured based on **PAGE OBJECT MODEL** and uses the **SINGLETON** webdriver.
- ➔ We use the **GIT** version control system in my company. I store the code for the automation framework in a certain git repository.
- ➔ **JENKINS** for Continuous Integration to run the tests in **AWS** EC2 Machines.
- ➔ **Hybrid Framework:** Data Driven and Behavior Driven
- ➔ **Maven:** It was a maven project which helped to manage the dependencies, run the test from the command line, and generate reports.
- ➔ **Java:** I used Java as the programming language - working frontend, backend(api), and database I use Java Collection framework to store data and compare.
- ➔ **Properties File:** I used a properties file to store sensitive and reusable data like URL(environment), username & password, browser, etc.
- ➔ **JUnit:** I use JUnit testing tool to control flow and assert data.
- ➔ **Singleton Pattern:** I created a Driver class in a utility package that uses the singleton pattern to create and use only one universal webdriver.

Front End:

- I used a Cucumber-JUnit framework with Selenium WebDriver.
- To increase code reusability and have a more readable framework, we used **Page Object Model** as my design pattern to create page objects; store webElements and their methods/actions.
- We used **Page Factory Design Pattern** to instantiate the webElement using **@FindBy** annotation which makes it easier and convenient.
- We also had a **Utility** package where we kept all global and most frequently used codes such as browser utils.

Back End (API):

- RESTASSURED in order for the process of **Deserialization** and **serialization** to occur that way you store **Json response** into a java collection data structure(I produce high level Pojos and map objects) and assert the data with the expected value (also stored in java data structure)
- I also have a api utility class method where one line creates a Pojo
- I use postman for manual testing first BEFORE I INVOKE MY FRAMEWORK

DataBase Testing

- Manual: **SQL Developer** to produce SQL queries.
- Automation: I use **JDBC** library to integrate java by getting a **CONNECTION** from oracle database then creating **STATEMENTS** using SQL queries and then storing the data into a **RESULTSET** object.
- I use java **data structures** to store data inside and compare them. Since I'm using **DATA DRIVEN** and **CUCUMBER BDD** framework, all of these tests are stored inside feature files.

DDT

- If I'm working with small amounts of test data, I work with **scenario outlines**, where I create examples and store data using a pipeline.
- If there is a large amount of test data it's usually in an external file (**excel**) so I use **Apache POI** to **INVOKE DDT EXCEL AUTOMATION** and read from excel file and store the data into java data structure.
- I use **Rerun.txt code** in cucumber ***"rerun:target/rerun.txt"*** generated by the cucumber library that will store my **failed** cucumber feature files. I also have **failedScenario** runner class which has the location of failed scenarios (rerun.txt).
- I create failedScenario xml file so whenever I have failed feature files I use the mvn command ; **mvn -Drunner=failedScenarios**. xml file to run my failed tests
- **Reporting** - I used **html report** that's located in target folder which is called **cucumber-reports** -***"html:target/cucumber-report"***
- **Parallel testing** - used cucumber **jvm-parallel plugin** to generate runners and maven fail-safe plugin to run the tests
- For continuous integration (**jenkins**) but our devops team takes care of all of the configuration but I provide the github path. The tool is invoked by a maven command **- mvn verify -drunner=smoketest.xml** that is provided by the testers - xml file
- For reports each build will have a cucumber report that give graphical information of test and screenshot

Project Details

- Team Size: 9 people:
  - 4 developers (Brian, Yasin, Zeynep, Vasyl)
  - 3 testers (Ozzy, Aaron*(manual)*, and myself)
  - 1 Project Owner: Mike
  - 1 Scrum Master: Sam
- Sprint Cycle & Current Sprint Number:
  - 2 Weeks for each sprint
  - Sprint # 49 (each year 23 sprints)
- Point System:
  - Fibonacci: 1, 2, 3, 5, 8, 13, 21…
  - 1 point = 1 day
  - Team Capacity: 56
    7 people x 8 days in two weeks (as one day is for meetings etc.) = 56
  - Velocity: changes every sprint depending on if there is anyone on PTO, sick, etc. If one member is off for 3 days and the other for 2 days which is a total 5 days, we take it out from the team capacity 56 - 5 = 51. This is the velocity for that specific sprint and this is the point we assign.
- Test Plan Document:
  - Team lead is responsible
  - Test Strategy is in the Test Plan in our company
- Smoke Test:
  - Around 80 test cases
  - 6-7 user stories
  - Runs at 5am and takes about 15 mins
- Regression:
  - 160 feature files
  - 700+ scenarios/test cases (350 written by myself)
  - 2 years ; 2 testers worked 1 year, third worked 2 years

Framework Snapshot

What is Page Object Model

1. Reduces code redundancy and organizes code
2. Helps identify elements and store it as a page object variable
3. You can link it to where it was stored
4. Added PageFactory design pattern

Selenium WebDriver as my automation tool

1. **Manual test it first by:**
    a. Front end
        i. Functional testing
    b. Back-end
        i. Database- SQL Developer IDE
        ii. API - Postman
2. **Integrate Selenium with:**
    a. Maven Project:
        i. Test package
        ii. Utility package: Classes: - UI - DB - API
        iii. Configuration file
            1. Properties
        iv. Driver class
            1. Singleton Design Pattern *(Have a private constructor)*
3. **Cucumber BDD**
    a. Facilitate the collaboration during the BDD process
    b. Enables explaining the story and the acceptance criteria in easy language.
4. **Git/GitHub** - Version control / remote repository
5. **Jenkins**: Continuous Integration
6. **Java**
    a. Collections Framework
    b. Apache POI
    c. JDBC
    d. Rest-Assured

Behavior Driven Development

- Developing with the client to ensure it meets the right standards for the customer

Data Driven Development

- Executing same test case against different sets of data
- Test flow should not change based on data

Cucumber reporting

- Target folder
- Jenkins

How many Test cases (in your regression suite) do you usually complete in a week?

10 small test cases, 7-8 medium, 2-3 larges

OR It depends on the project. In COOLSIS we have 2000 test cases. In 4Stay, we have around 700 test cases.

CHALLENGES

1. Old frameworkI inherited a framework that was not efficient, no page object model, no reporting. I had to refactor the framework and change many packages/ classes. Introduced page object model, html reporting.
2. Dynamic ids, locators we had cross training exercise where devs tried to do some selenium testing. This is when they understood the importance of ids in selenium.
3. StaleELementExceptionuse page object model: it reloads the web element when we call it. I had to use explicit wait in many places.
4. Screen size issue in VMdue to screen size different in the VM where we run smoke tests, pages were loading differently than in a normal screen. I used Actions moveToElement to move to the pop up and close it
5. Tests taking too long.
    a. Parallelization
    b. Headless browsers
6. Sign up with Facebook functionality and need a new Facebook account every time. Used the Facebook api to create new accounts.
7. Devs not accepting the bugs when I had a situation where I had disagreement with devs on whether something is a bug or not, I documented the steps to recreate the bug and communicate the issue with the team. I discuss the issue with BA who is more familiar with the requirements and our BA will say if it is big or not. In general to avoid this kind of confusion, we discuss all the test cases/requirements in the backlog grooming sessions. And also in my tests I have detailed steps and screenshots for all the bugs, failures which I can use when I want to show a bug.
8. When I have little time to test several stories. There are a couple we can do
    a. I always make sure my team / scrum master is aware of the situation.
    b. I prioritize the tests which need to be completed first based on the priority. and in the sprint retrospective meeting we discuss this and talk about how to avoid this again.
9. Convince the team to use new toolsI prepare a demo of the new tool and present it to the team. I list all the advantages that it can bring. I try to show the business value of implementing the new team.