## Automation Tools:

**For: Test Case # 1 UI Automation following tools are used:**

1. Visual Studio
2. C# / .Net Core 3.1
3. NUnit
4. Selenium Web driver / Chrome Driver

How to Run: Visual Studio> Test > Test Explorer > Run All Tests

A screenshot of a computer

Description automatically generated

**For: Test Case # 2 API Automation following tools are used:**

1. JMeter

How to Run:

1. Run Jmeter application (Java based tool)
2. Open [API Testing.jmx](https://github.com/Momina-code/NimbusQAChallenge/blob/main/API%20Testing/API%20Testing.jmx) file in Jmeter
3. Click Run

Details:

1. Auth Request – This is POST request to Authenticate and get the UserId and Token
2. Auth Header – This is Header of Auth Request in Step 1
3. Json Extractor Token – This extracts the token from the Auth Request in Step 1
4. Json Extractor User ID – This extracts the User ID from the Auth Request in Step 1
5. HTTP Report Request – This is GET request to execute the report. Token and User ID extracted in Step 2 and Step 3 are passed in Header to this request
6. JSON Assertion – This will parse HTTP Report Request response from Step 5 and verify if Forename ‘Momina’ is present.

Graphical user interface, text, application, email

Description automatically generated

## Setup Continuous Integration

**For: Test Case # 1 UI Automation:**

**Assumption:**

Since I’m not sure how the webapp (autotest.time2work.com) is setup in the Repo therefore I have taken following assumptions:

1. It is assumed that webapp already has a pipeline
2. It is assumed that unit test project will be added existing webapp solution as a unit test project for it.

**Steps:**

1. Do some configuration changes to make the project compatible for Azure DevOps and chrome e.g. <publishchromedriver> to true in the Unit test project csproj.
2. Push your Unit Test code into a project in DevOps repo.
3. Since it is assumed that pipeline for webapp is already there so edit the build pipeline and add Task of “Visual Studio Test” and add your unit test project dll in it.

Graphical user interface, application

Description automatically generated

**Test Case # 2 API Automation**

Cloud Load/Function Testing has been deprecated in Azure DevOps. Previous we used to do it in “cloud-based apache jmeter load test” but in alternative ways, Jmeter is still there but there is no official documentation on how Jmeter file can be integrated in pipeline so people have used their own different ways to do it. One of the way is given below.

One of the way is:

1. Push .jmx file in the Repos
2. Create two sh files i.e. run.sh that launches Docker image and run the tests and create test.sh for each Jmeter test plan which will just call run.sh passing in the test plans and parameters
3. Publish the reports directory for post-run analysis

Please note that I have not tested the above method myself as the requirement in the assignment was just to explain the process but if I have to do it in actual, I can definitely able to do it and integrate jmx in pipeline.