Dhar,Gopal Chandra

Project Manager Setup

Contents

[Introduction 2](#_Toc19827347)

[**C:\Users\Tamoghna\_Choudhury\_210125\_FSEFinalAssessment\FSE\_Final\_210125** 2](#_Toc19827348)

[Software Requirements and Technology details 2](#_Toc19827349)

[Required Software 2](#_Toc19827350)

[Technology Details 2](#_Toc19827351)

[Setting up the application 2](#_Toc19827352)

[Taking checkout 2](#_Toc19827353)

[Folder Structure 3](#_Toc19827354)

[02\_Front\_End\_Layer 3](#_Toc19827355)

[03\_Middle\_Tier\_Layer 4](#_Toc19827356)

[01\_Database\_Scripts 4](#_Toc19827357)

[Running the application 4](#_Toc19827358)

[10. Referred the video related to Code coverage with OpenCover Console and Report Generator. 16](#_Toc19827359)

# Introduction

This document describes the setup of the application “**Project Manager**” in the local machine for development and debugging purpose.

Source code for the “**Project Manager**” final assignment is kept under in the VM.

C:\FSE\FSE\_Final\_424618\_Git\FSE\_Final

# Software Requirements and Technology details

## Required Software

These are the list of the software installed in the system to debug/develop and run the application:

1. Visual Studio 2017
2. Visual Studio Code (preferable latest version)
3. SQL Server Management Studio 2017
4. Google Chrome
5. Node JS v 8.12.0
6. GIT BASH and GIT UI (for connecting to the repository)

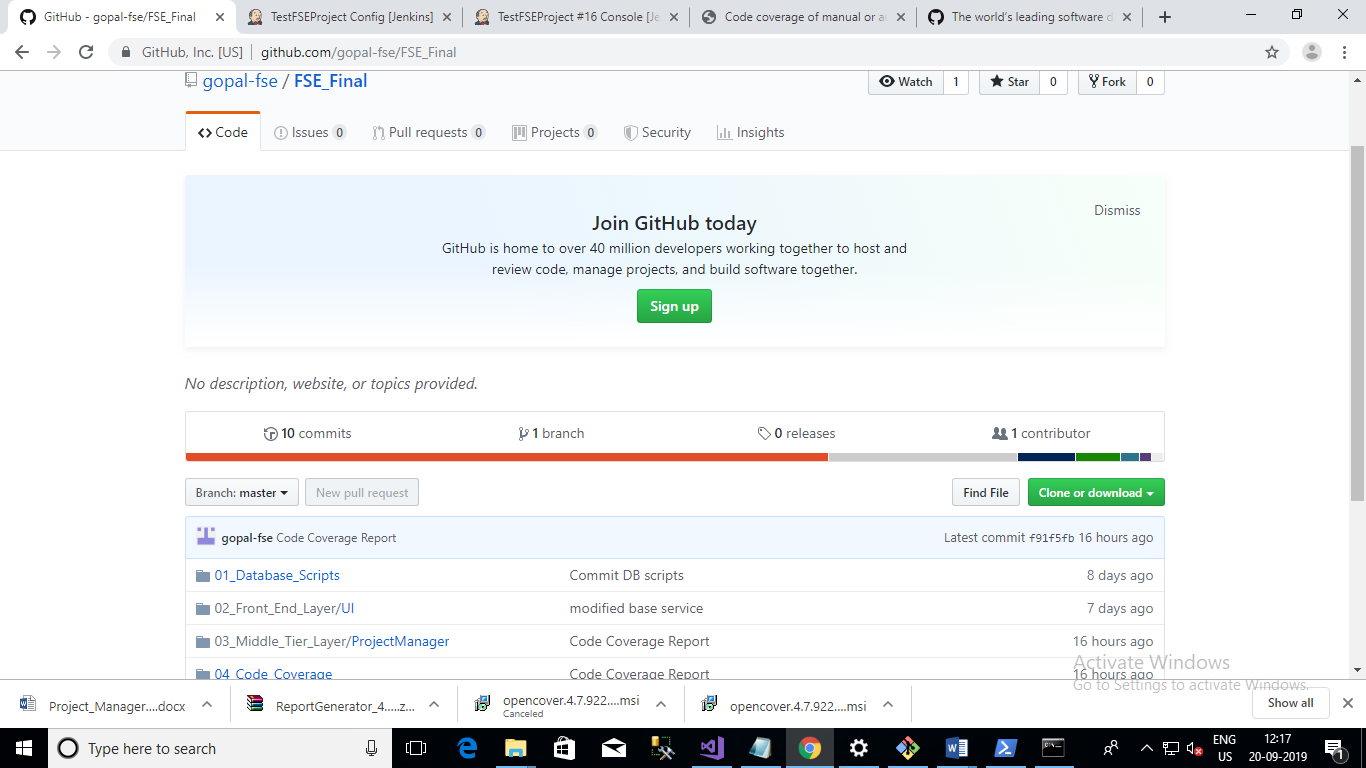
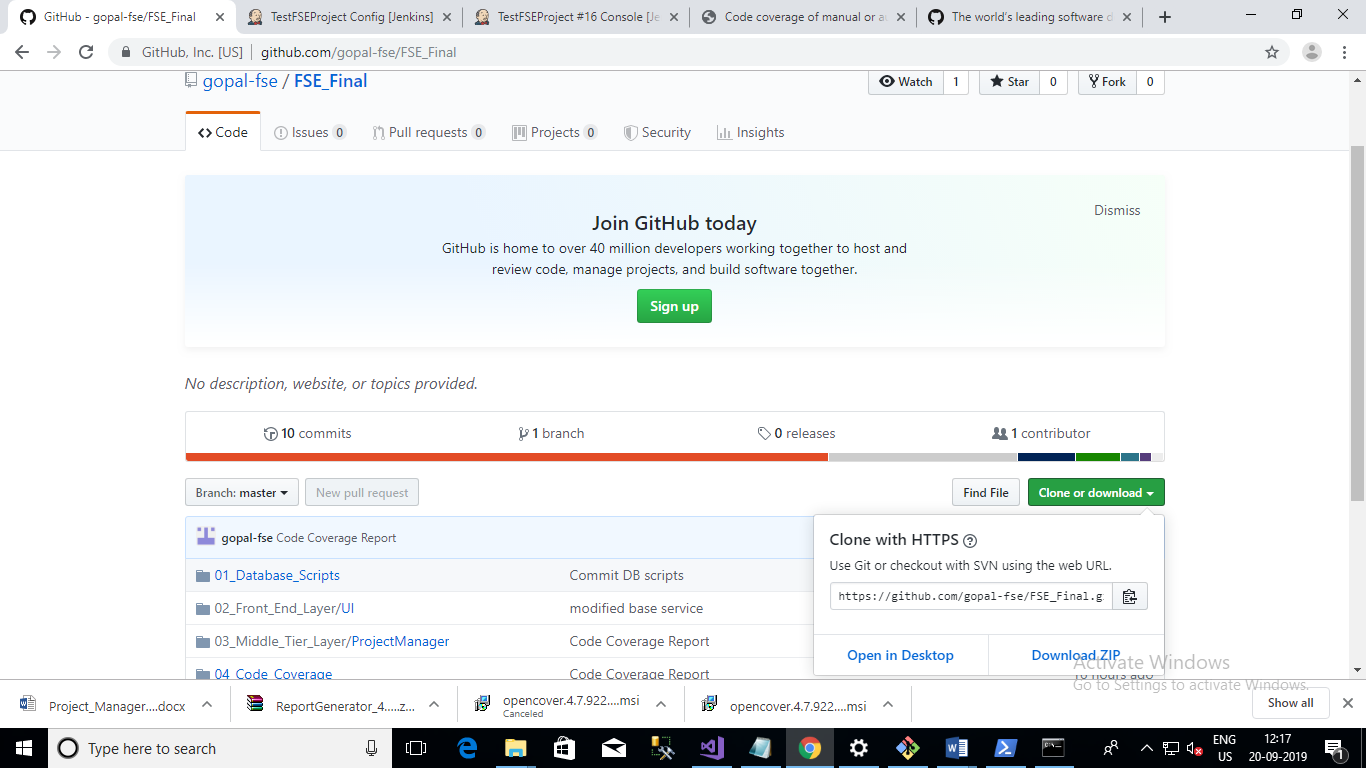
## Technology Details

These are the list of the technologies used in the application:

1. Angular 4 (UI)
2. HTML (UI)
3. CSS3 and BOOTSTRAP (UI)
4. .NET Framework
5. Web API 2.0 (C#)
6. Entity Framework (C#)
7. SQL (C#)

# Setting up the application

## Taking checkout

1. Visit the URL [https://github.com/](https://github.com/Tamoghna/FSE_Final_210125)gopal-fse/FSE\_Final where the code is checked in.
2. Click on the “Clone or download” button.  
   
3. Once a small pop-up opens, click on the “Download ZIP” button.  
   
4. Open the downloaded file and extract the folder (FSE\_Final) to some path on the system.

## Folder Structure

FSE\_Final contains the following folders:

1. *01\_Database\_Scripts*: This folder contains the database scripts for creating database & tables.
2. *02\_Front\_End\_Layer*: This folder contains the angular web code.
3. *03\_Middle\_Tier\_Layer*: This folder contains .NET web api code
4. *04\_Code\_Coverage*: this folder contains ”OpenCover” code coverage report.
5. *05\_Performance\_Testing*: This folder contains NBench performance testing report
6. *06\_Unit\_Testing\_Result*: This folder contains NUnit testing report
7. *07\_Jenkins\_Report*: This folder contains Jenkins build report

### 02\_Front\_End\_Layer

1. Go to the path “*\FSE\_Final\02\_Front\_End\_Layer\UI*” where you can see the file angular.json
2. Open NodeJS command prompt
3. Copy the path from Step No. 1
4. Traverse to the path of Step 1 in the command prompt
5. Once you are in this path, run this command “*npm install –g -f @angular/cli*”
6. After this installation is done, run this command “*npm install -f*”
7. Let all the npm packages install in the project
8. Once the installation is done you will be able to see a folder “node\_modules” in your system
9. Don’t close the command window yet
10. Do an *npm start*

### 03\_Middle\_Tier\_Layer

1. Open the folder “\FSE\_Final\03\_Middle\_Tier\_Layer\ ProjectManager”
2. Open the file “*ProjectManager.sln*” in Visual Studio 2017
3. Build the application & run the application
4. Don’t close the Visual Studio 2017 yet

### 01\_Database\_Scripts

1. Open the SQL Server Management Studio
2. Run the script “*ProjectManagerDB\_CreateDatabase\_Script.sql*”
3. Run the script “*ProjectManagerDB\_CreateTables\_Script.sql*”

# Running the application

Once the build is succeeded:

1. Open the command prompt and run the command “npm start”
2. Open the visual studio 2017 and select the “ProjectManager” project as startup project and press “Start” to run the application
3. Once the node modules are built after the step 1, open Google Chrome and enter the URL “localhost:4200”

Enabling the Cors for the Web Api project:

1. Installed the **Microsoft.AspNet.WebApi.Cors** package, which is available on NuGet.

Install-Package Microsoft.AspNet.WebApi.Cors -pre -project WebService

2. Done the highlighted Change in **C:\FSE\FSE\_Final\_424618\_Git\FSE\_Final\03\_Middle\_Tier\_Layer \ProjectManager\ProjectManager\App\_Start\WebApiConfig.cs** file:

**Code Snippet :**

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

config.MapHttpAttributeRoutes();

config.IncludeErrorDetailPolicy = IncludeErrorDetailPolicy.Never;

config.Filters.Add(new ProjectManagerLogFilter());

config.Filters.Add(new ProjectManagerExceptionFilter());

var cors = new EnableCorsAttribute("\*", "\*", "\*");

config.EnableCors(cors);

var jsonFormatter = config.Formatters.OfType<JsonMediaTypeFormatter>().First();

config.Formatters.Remove(config.Formatters.XmlFormatter);

jsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

}

**Reference Article** : <https://www.hexacta.com/2014/09/15/How-to-enable-CORS-on-your-Web-API>

Adding the Web Api service reference in the UI Project:

1. Added the Webapi url in the base.service.ts file C:\FSE\FSE\_Final\_424618\_Git\FSE\_Final\02\_Front\_End\_Layer \UI\src\app\services\base.service.ts file

**Code Snippet:**

import { Response } from '@angular/http';

import { Observable } from 'rxjs';

import { ServiceError } from '../models/serviceerror';

export class BaseService {

constructor() { }

protected extractData(res: Response) {

const body = res.json();

if (body.status === 'success') {

return body.data;

} else if (body.status === 'fail') {

throw new ServiceError(body.message, body.data, 'fail');

} else if (body.status === 'error') {

throw new ServiceError(body.message, body.data);

} else {

throw new ServiceError('Invalid JSend Response Status [' + body.status + ']');

}

}

public baseurl(): string {

return 'http://localhost:50830/';

}

protected handleError(error: any) {

if (error instanceof ServiceError) {

return Observable.throw(error);

} else {

const errMsg = (error.message) ? error.message : error.status ? `${error.status} - ${error.statusText}` : 'Server error';

return Observable.throw(new ServiceError(errMsg));

}

}

}

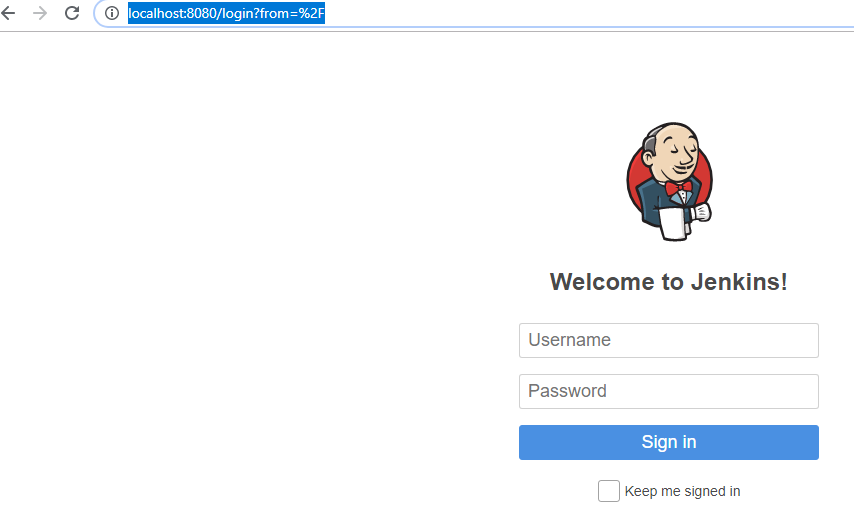
Jenkins Setup:

1.Installed Jenkins 2.129 version in the Virtual macghine

After installing Jenkins LogIn screen should appear

**Reference URL for Jenkins Installations:**

<https://www.guru99.com/download-install-jenkins.html>



2.Created a free style project in Jenkins.

**Project Name**: TestFSEProject

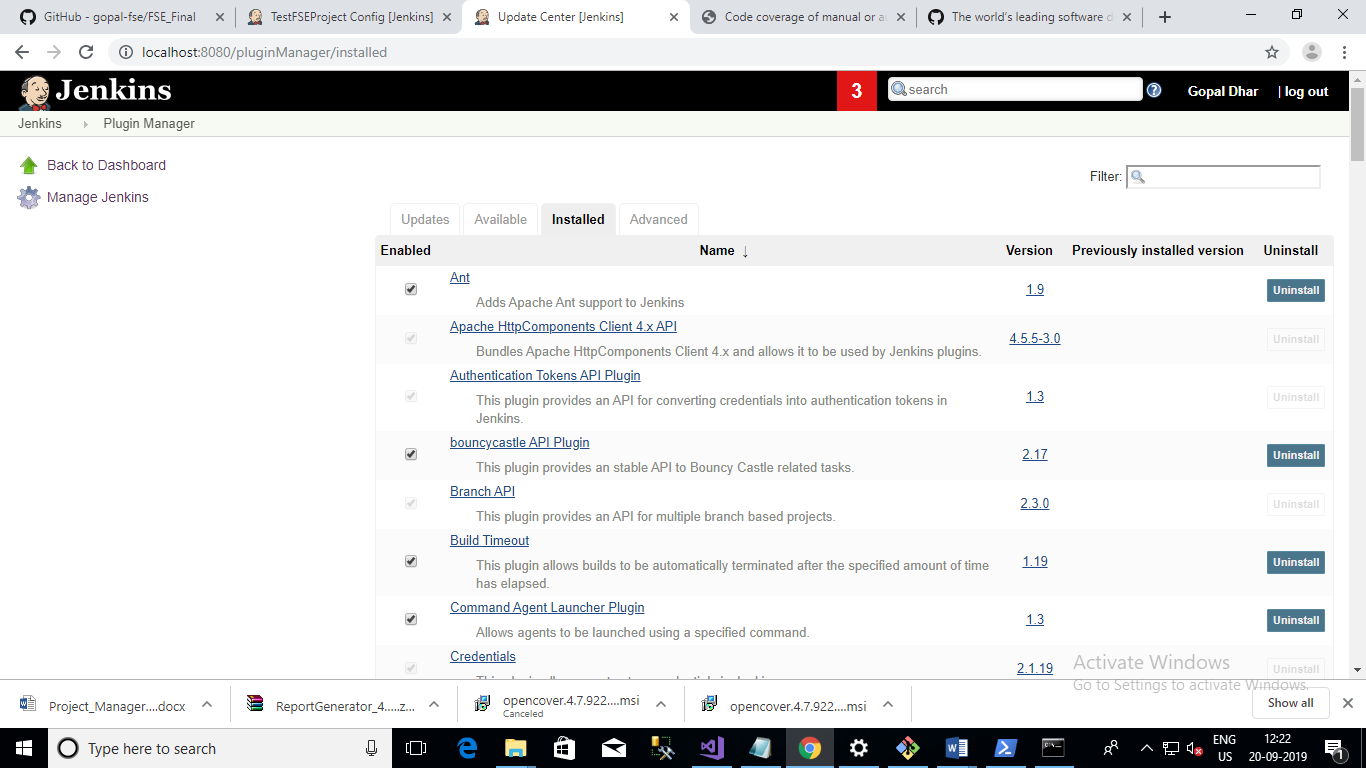
**Reference URL:** <https://www.guru99.com/create-builds-jenkins-freestyle-project.html>

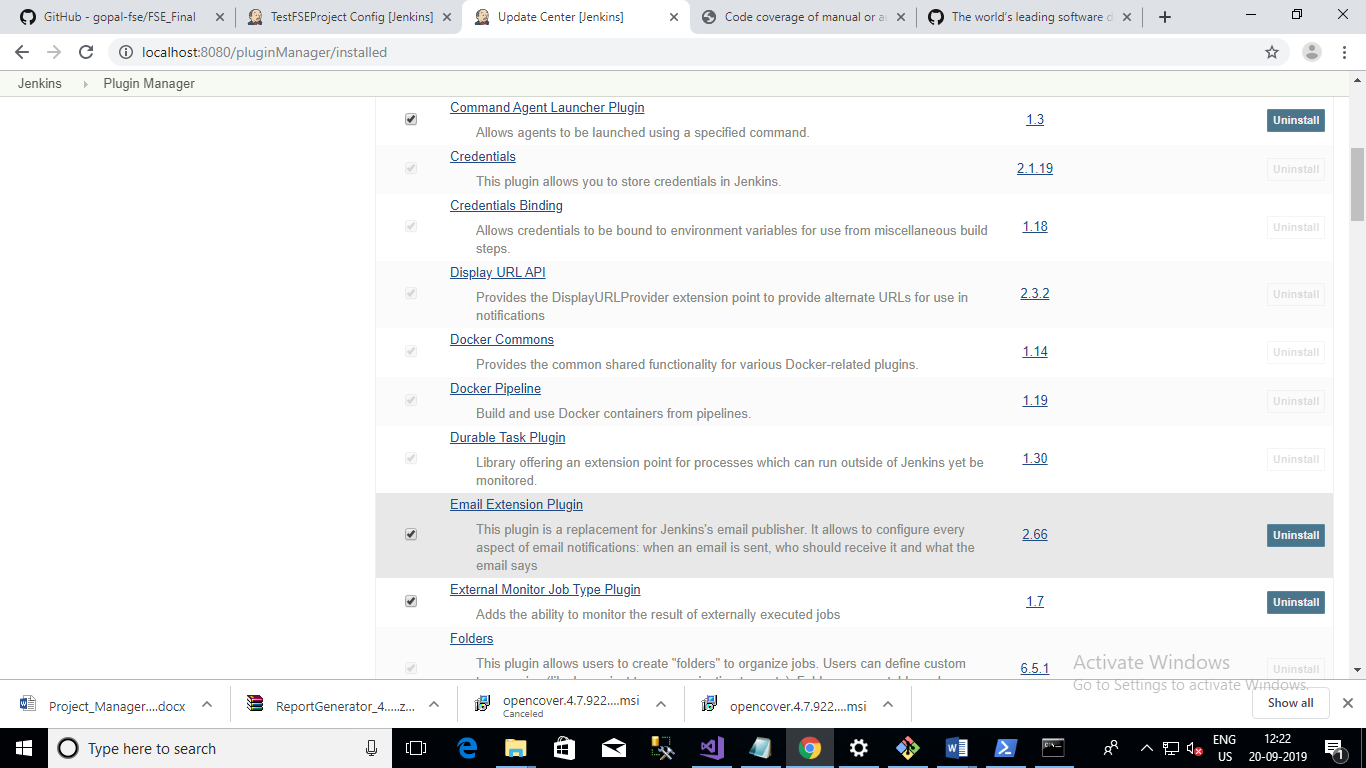
3.Navigate to the Jenkins ->Plugin Manager->Manage Jenkins

Following plugins should be installed/Preinstalled:

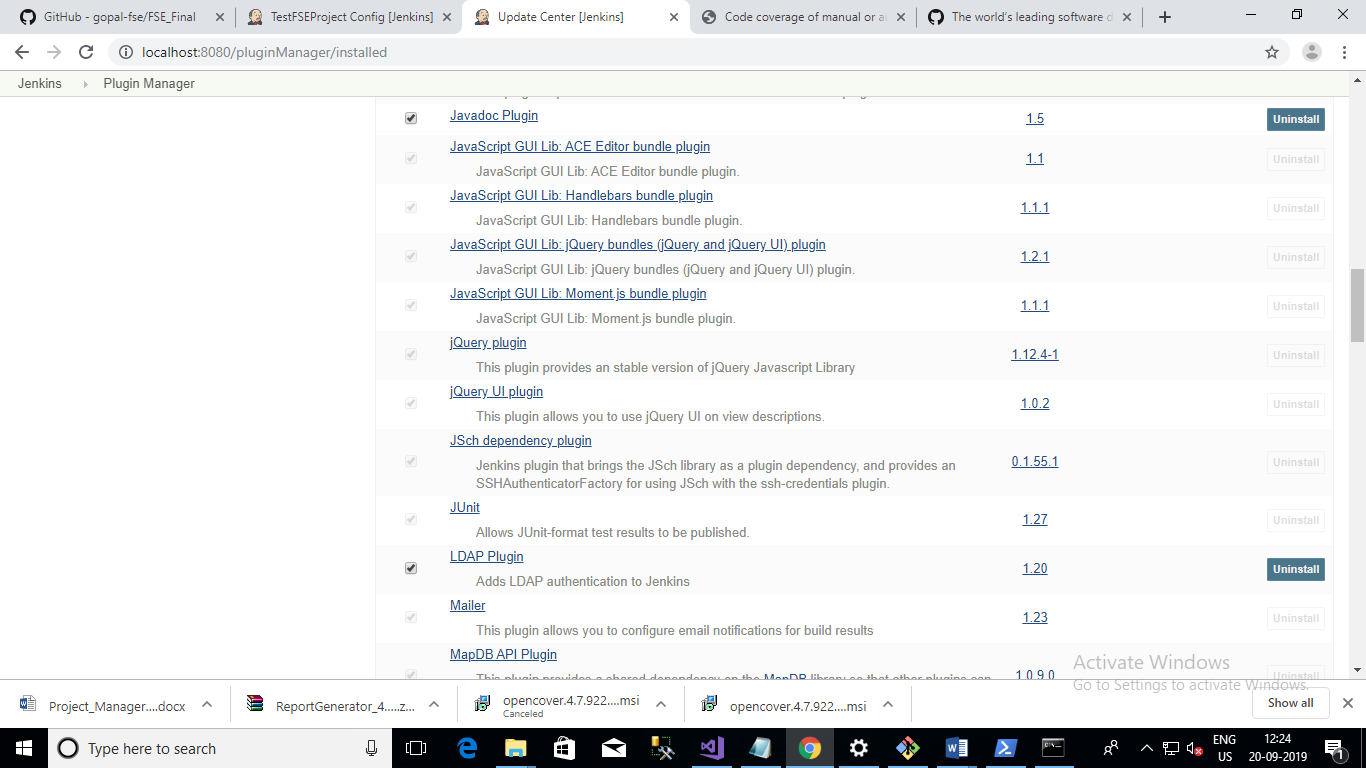
**Important plugins :**

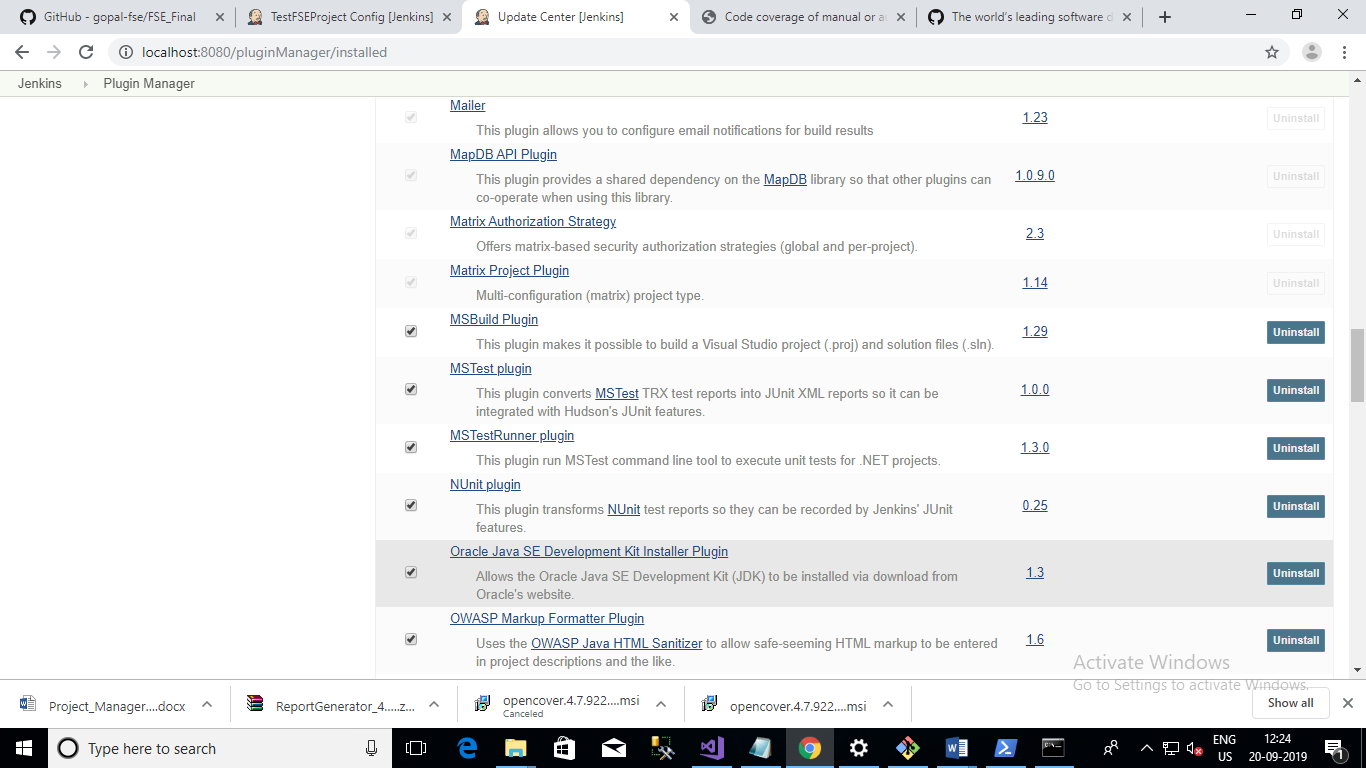
Msbuild,mstest,mstestrunner,nunit.







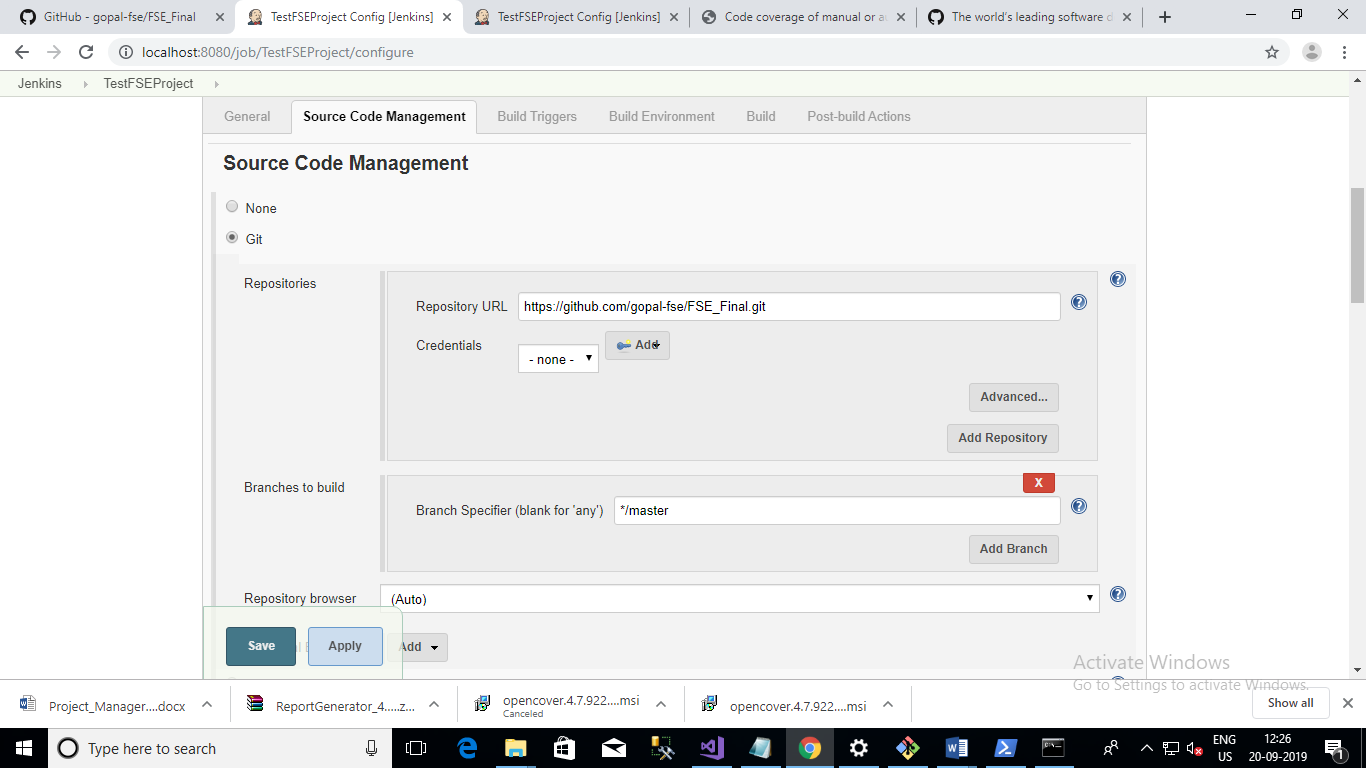




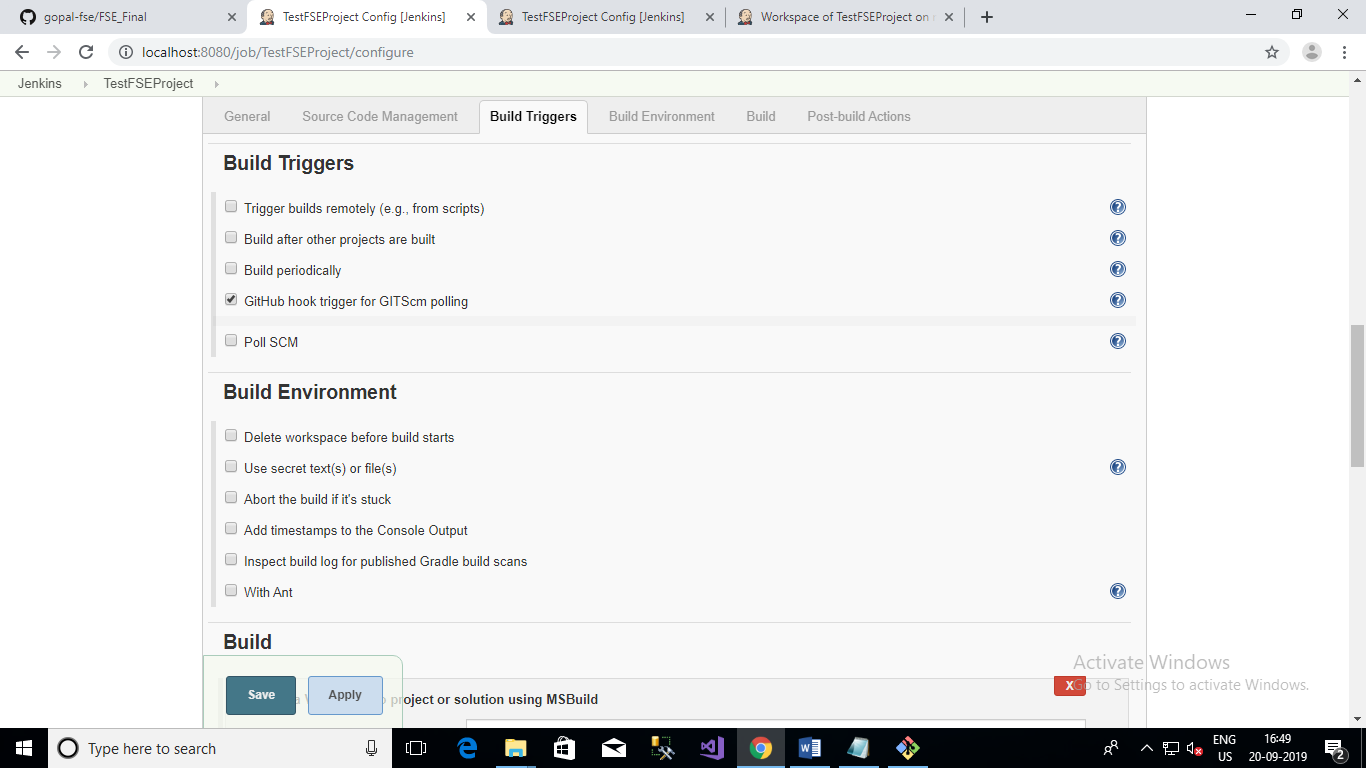
4.Navigate to Jenkins -> TestFSEProject ->Configure

Git URL: https://github.com/gopal-fse/FSE\_Final.git

**Source Code Management**



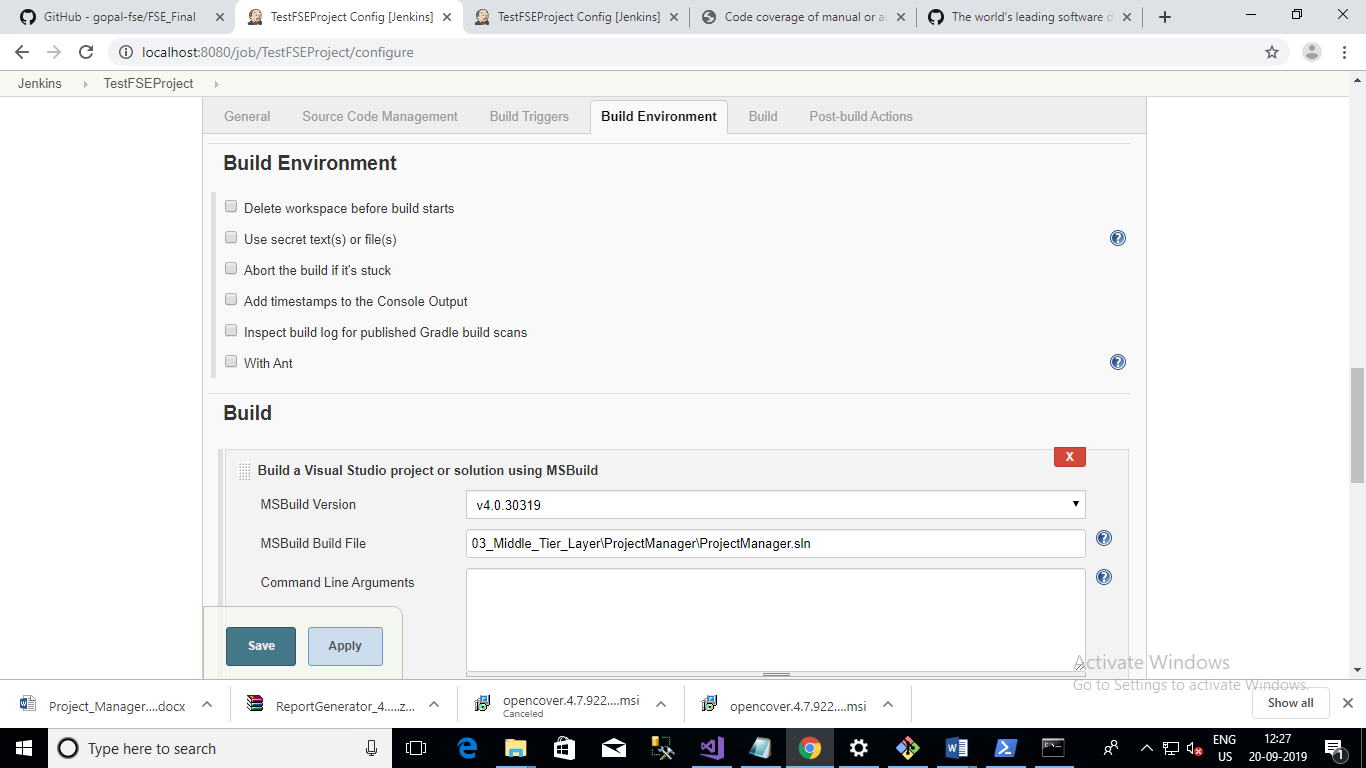
**Build Trigger Screenshot:**



**Build Environment :**

**MSBuild Build File**: Solution File Name of the ProjectManager

MS Build File: 03\_Middle\_Tier\_Layer\ProjectManager\ProjectManager.sln



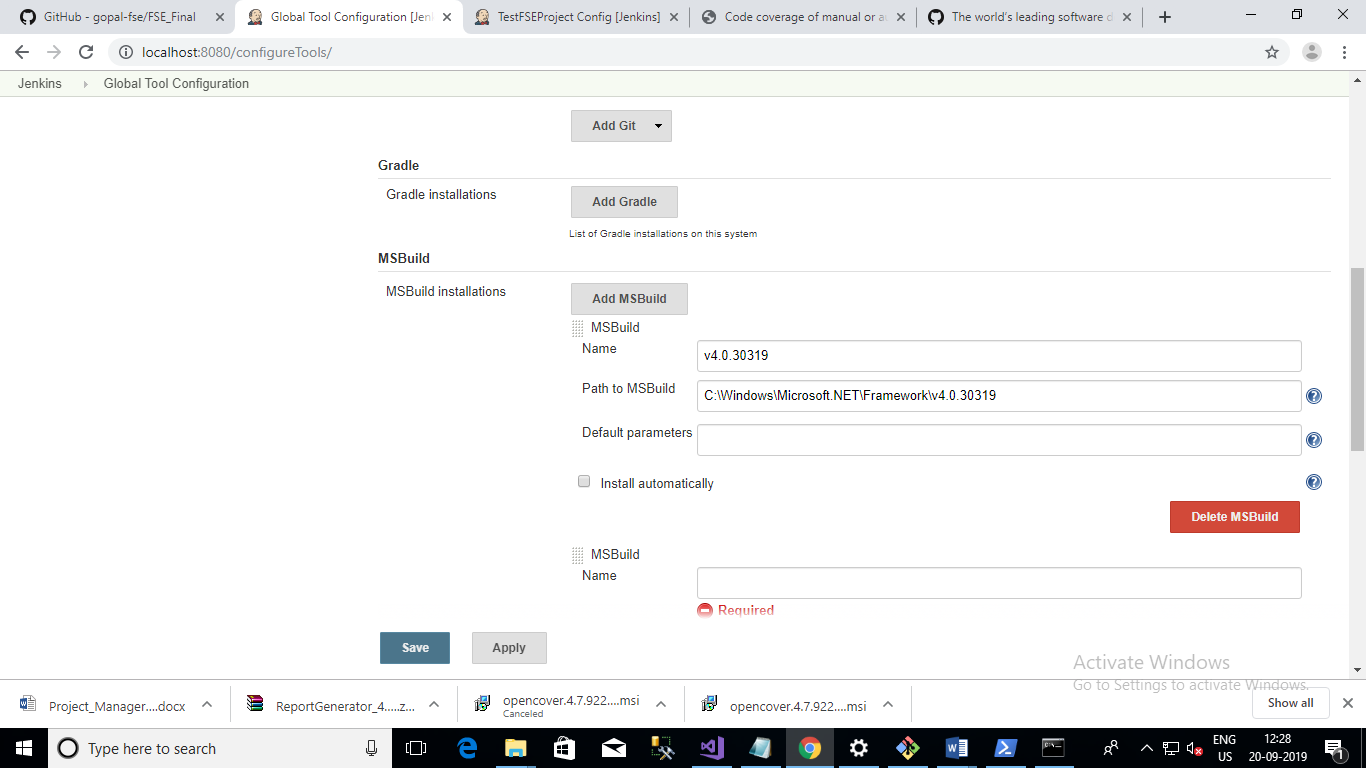
**Build Screenshot:**

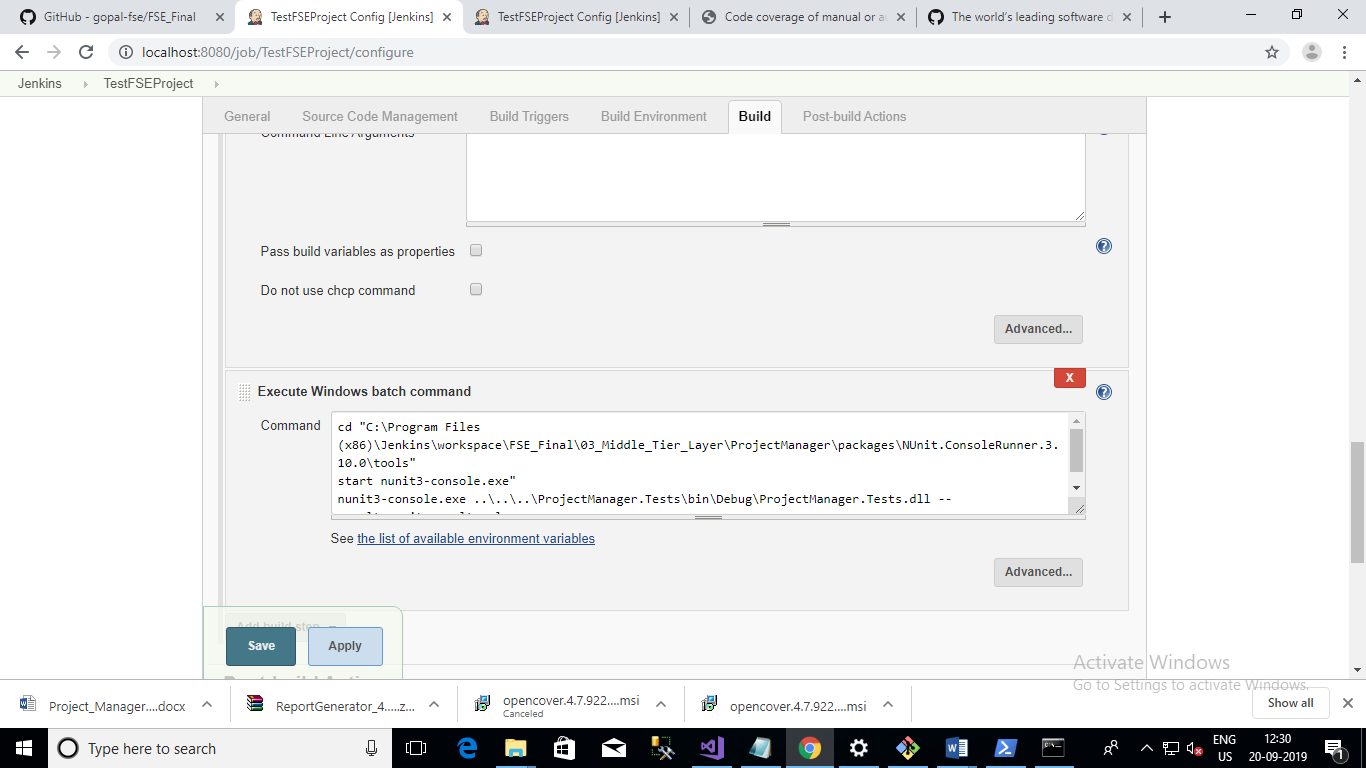
Execute Batch Command :

Before doing the Jenkins add the MSbuild

Jenkins->Global Tool Configuration

**MS Build Path** : C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe





**Command :**

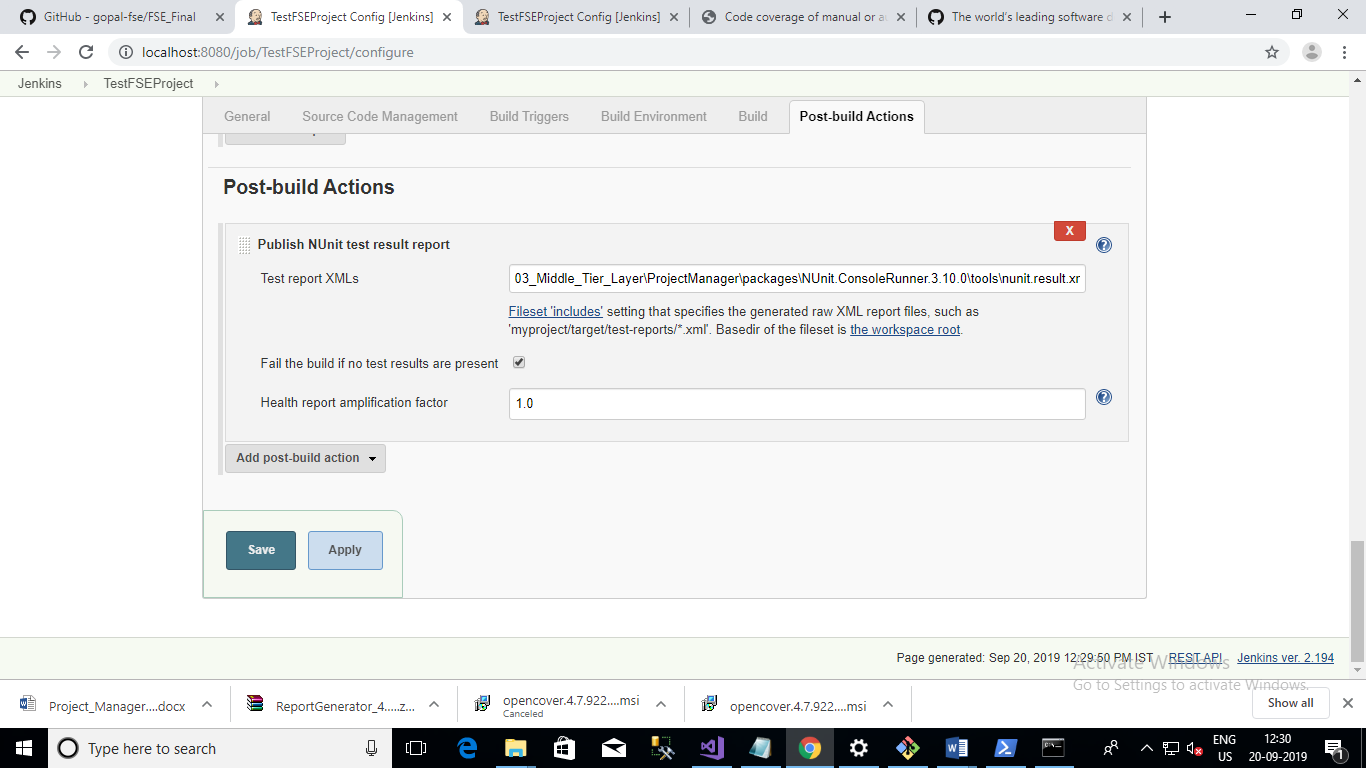
cd "C:\Program Files (x86)\Jenkins\workspace\FSE\_Final\03\_Middle\_Tier\_Layer\ProjectManager\packages\NUnit.ConsoleRunner.3.10.0\tools"

start nunit3-console.exe"

nunit3-console.exe ..\..\..\ProjectManager.Tests\bin\Debug\ProjectManager.Tests.dll --result=nunit.result.xml

**Post Build Actions Screenshot :**

**Test Report XML :**

03\_Middle\_Tier\_Layer\ProjectManager\packages\NUnit.ConsoleRunner.3.10.0\tools\nunit.result.xml 

**Configuring the web hook in the Jenkins Pipeline :**

**1. Reference article:**<https://dzone.com/articles/adding-a-github-webhook-in-your-jenkins-pipeline>

2.Download the ngrok from the below link and follow the instructions:

<https://ngrok.com/download>

3.Finally run ngrok http 8080 for Jenkins pipeline

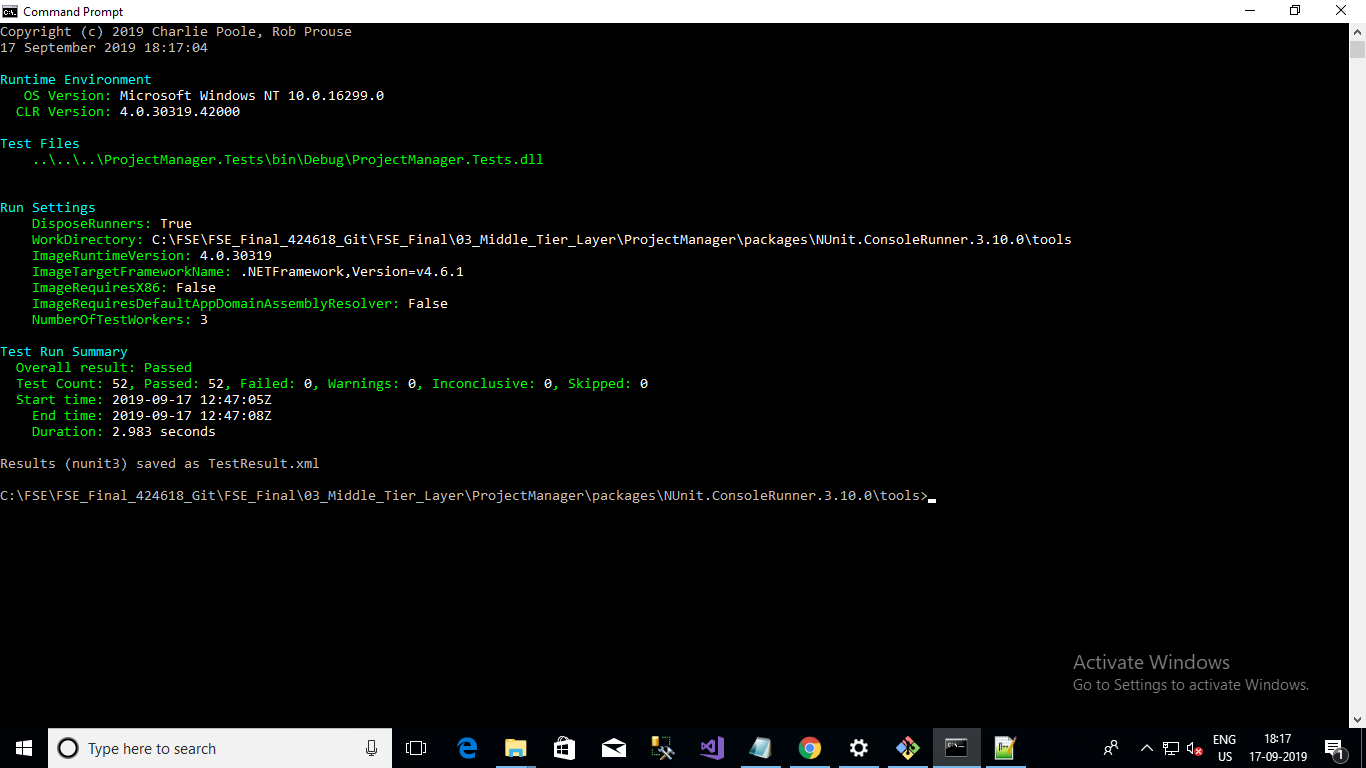
4.Navigate to dashboard.ngrok.com

<https://dashboard.ngrok.com/>

Nunit Test result :

Nunit.ConsoleRunner.3.10.0 should be present in the package

Launch the Command prompt and execute the command mentioned in the below screen shot:

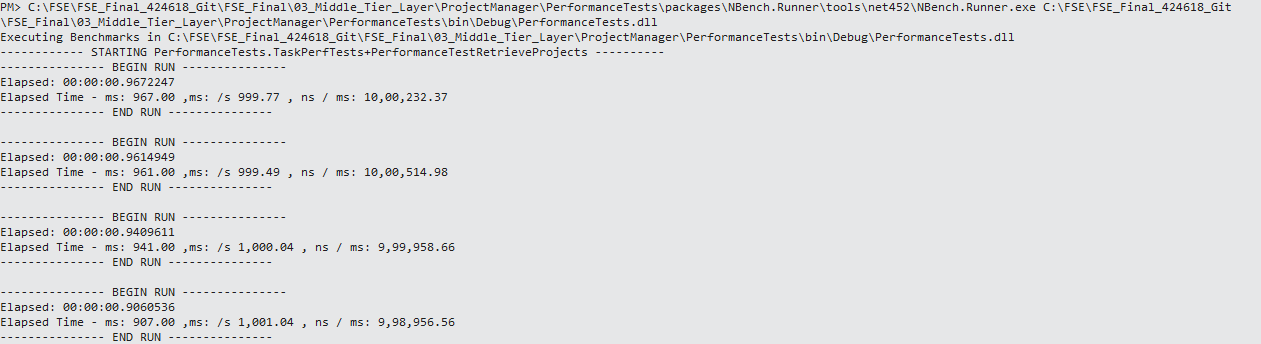


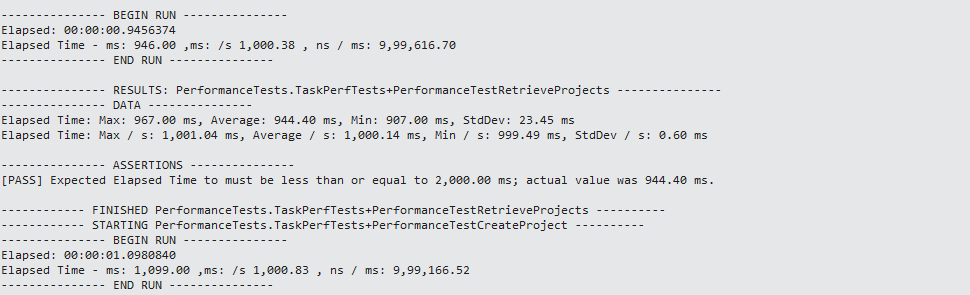
Performance Testing:

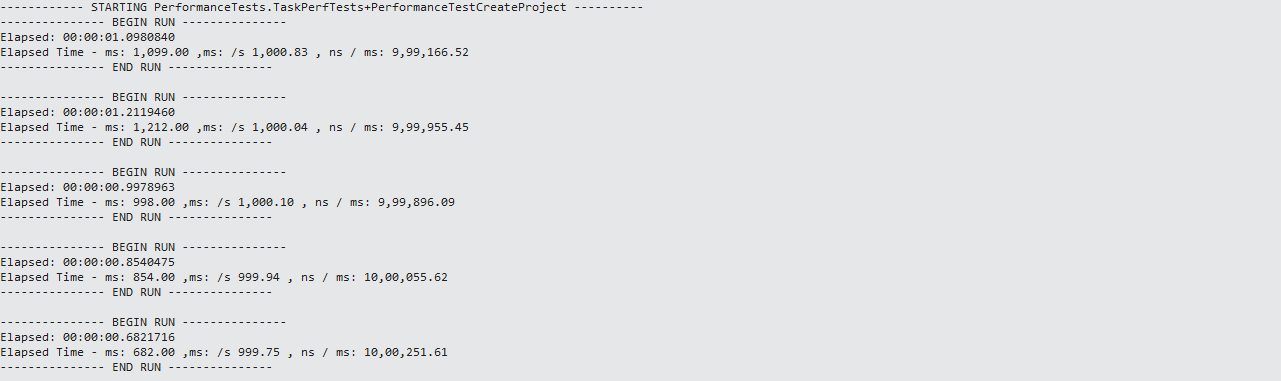
1. Performance testing is done using NBench

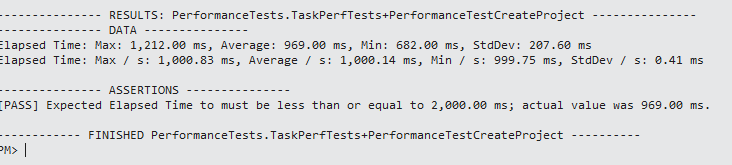
2. Install NBench from the nugget package from the Web Api project

3.Then execute the below mentioned command :









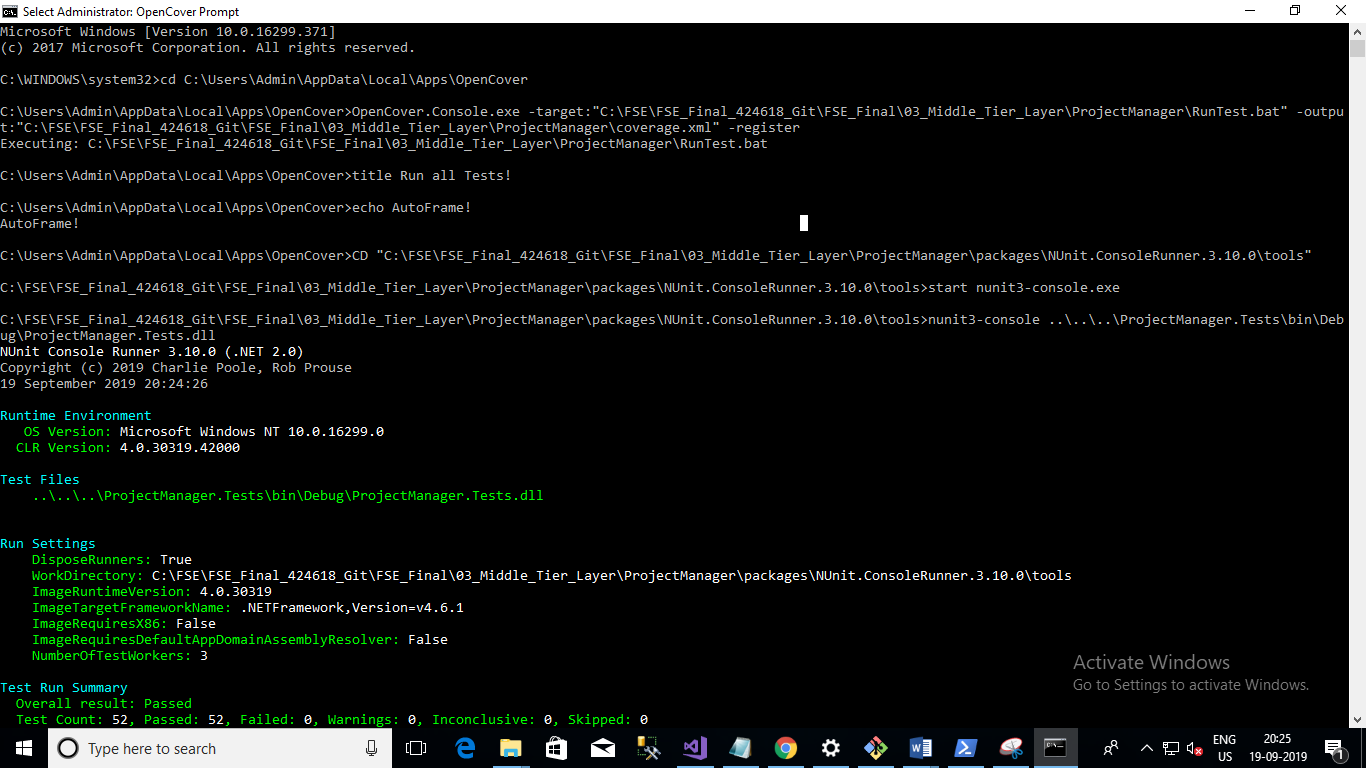
Code Coverage Report :

1.Code coverage xml has been generated using the OpenCover msi

2. Prerequisite: .NETFramework 4.7.2

3.Open cover installations Path : C:\Users\Admin\AppData\Local\Apps

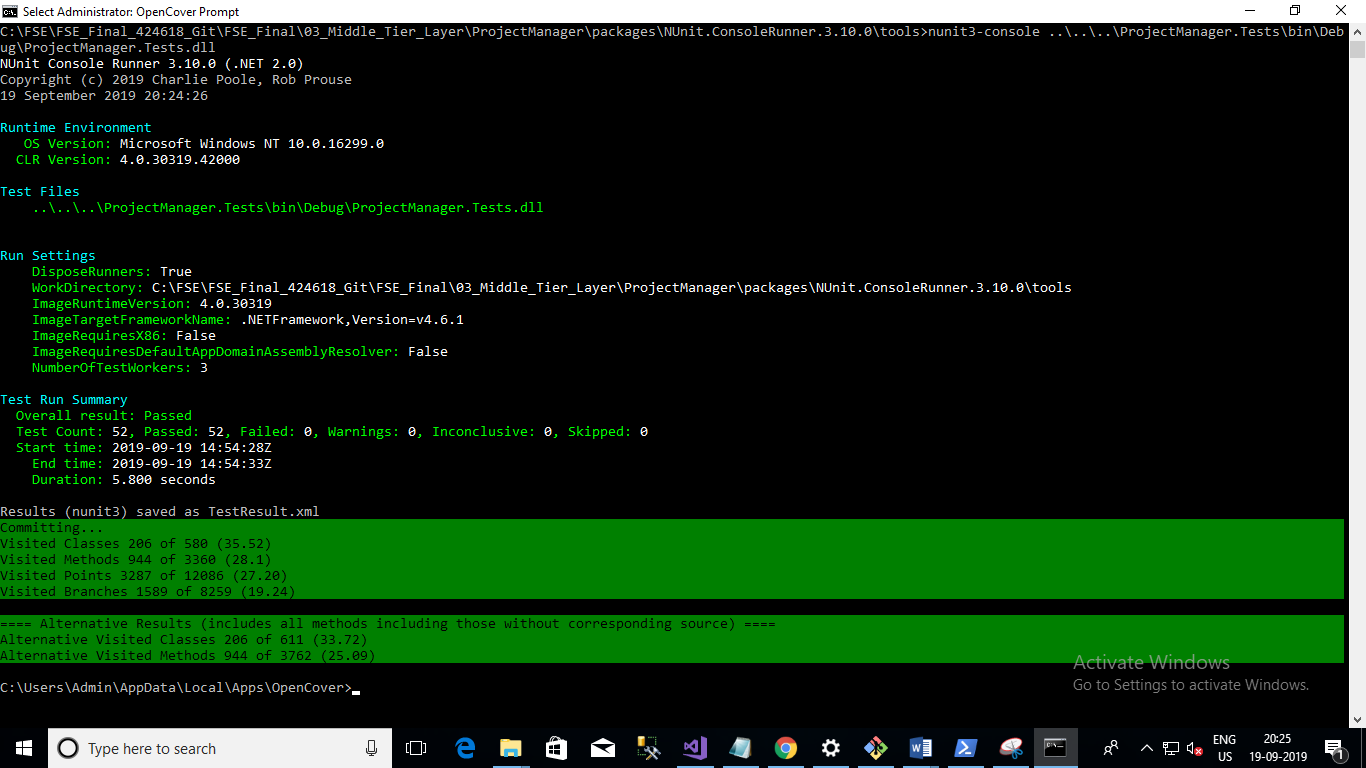
4. Launch the Open cover command prompt in the Admin Mode and launch the below mentioned command to generate the coverage.xml.



5.After generating the coverage.xml deleted the files section

6.Downaload the ReportGenarator4.2.11

7.Launched the command prompt in the Admin Mode and executed the below command :



8. Reference URL: <https://automationrhapsody.com/code-coverage-manual-automated-tests-opencover-net-applications/>

9.Navigate to C:\FSE\FSE\_Final\_424618\_Git\FSE\_Final\04\_Code\_Coverage\report\index.htm file to check the code coverage.

# 10. Referred the video related to Code coverage with OpenCover Console and Report Generator.