

Annexure-IX (a)

Selenium Automation Testing with Java

AETHEREUS

A Training Report

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

Computer Science and Engineering

(Cyber Security)

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB



LOVELY
PROFESSIONAL
UNIVERSITY

From 13/01/2023 To 28/04/2023

SUBMITTED BY

Name of student: Gourav

Registration Number: 11904486

Signature of the student: *Gourav*

SUBMITTED TO

Name of the supervisor: Sakshi

Designation:

Signature of the supervisor:

Annexure-IX (b):

Student Declaration

To whom so ever it may concern

I, Gourav, 11904486, hereby declare that the work done by me on “**Automation Testing With Selenium Java**” from 13th January-2023 to 28th april-2023, under the supervision of **Epam** certified trainer **Sakshi Takkar at Lovely Professional University Phagwara,**

Punjab is a record of original work for the partial fulfilment of the requirements for the award of the BTech Computer Science and Engineering.

Name of the Student (Registration Number)

Gourav (11904486)

Gourav

Signature of the student Dated: 28/04/2023.

Declaration by the supervisors

To whom so ever it may concern

This is to certify that **Gourav 11904486** from **Lovely Professional University,**

Phagwara, Punjab, has completed his training in “**Automation Testing with Selenium Java**” from **13th January-2023** to **28th april-2023**. It is further stated that the work carried out by the student is a record of original work to the best of my knowledge for the partial-fulfillment of the requirements for the award of the BTech, Computer Science and Engineering.

Name of External Supervisor

Name of Internal Supervisor **Sakshi**

Sakshi

Assistant Professor

Designation of the Internal Supervisor

Signature of the external Supervisor

Signature of the Internal Supervisor

Dated:

Dated:

Table of Contents

S.No	Title	Page No.
1	Declaration by Supervisors	3
2	Declaration by Student	2
3	List of Tables	4
4	Chapter-1 about company	5
5	Chapter-2 INTRODUCTION OF THE PROJECT UNDERTAKEN	10
6	Chapter 2.2 Individual project	19
7	Chapter 3 TECHNOLOGIES LEARNT	35
8	Chapter-4 CONCLUSION	44

CHAPTER-1

INTRODUCTION TO COMPANY

1.1 About EPAM

Since 1993, EPAM Systems, Inc. (NYSE: EPAM) has leveraged its advanced software engineering heritage to become the foremost global digital transformation services provider leading the industry in digital and physical product development and digital platform engineering services. Through its innovative strategy; integrated advisory, consulting, and design capabilities; and unique 'Engineering DNA,' EPAM's globally deployed hybrid teams help make the future real for clients and communities around the world by powering better enterprise, education and health platforms that connect people, optimize experiences, and improve people's lives.

In 2021, EPAM was added to the S&P 500 and included among the list of Forbes Global 2000 companies. Selected by Newsweek as a 2021 and 2022 Most Loved Workplace, EPAM's global multi-disciplinary teams serve customers in more than 50 countries across six continents. As a recognized leader, EPAM is listed among the top 15 companies in Information Technology Services on the Fortune 1000 and ranked four times as the top IT services company on Fortune's 100 Fastest Growing Companies list. EPAM is also listed among Ad Age's top 25 World's Largest Agency Companies for three consecutive years, and Consulting Magazine named EPAM Continuum a top 20 Fastest Growing Firm.



Vision And Mission

EPAM Systems is a global provider of software engineering and IT consulting services. The company was.

founded in 1993 in Minsk, Belarus, and has since expanded to have a presence in over 35 countries around.

the world. EPAM Systems serves clients from a wide range of industries, including financial services, healthcare, technology, travel, and more.

Mission:

EPAM Systems' mission is to help clients transform their businesses through the power of software.

engineering and digital innovation. The company's goal is to help clients stay ahead of the curve in their

respective industries by providing cutting-edge software solutions that drive business growth and improve.

customer experience.

EPAM Systems aims to achieve its mission by focusing on three core principles: Company Core Values:

Innovation: EPAM encourages its employees to be innovative and to explore new ideas and approaches in order to develop cutting-edge solutions for its clients.

Collaboration: EPAM values teamwork and collaboration, recognizing that the best solutions are often developed through the combined efforts of a diverse group of individuals.

Excellence: EPAM strives for excellence in everything it does, from the quality of its solutions to the level of service it provides to its clients.

Customer focus: EPAM is committed to understanding its clients' needs and preferences in order to deliver solutions that meet or exceed their expectations.

Social responsibility: EPAM is committed to being a responsible corporate citizen and to making a positive impact in the communities where it operates.

Origin and growth of company

EPAM Systems was founded in 1993 in Princeton, New Jersey by Arkadin Dobkin and Leo Loaner, who were both software engineers. The company started as a small software development startup with a handful of employees, primarily serving clients in the financial services industry.

In the early years, EPAM focused on providing custom software development services to its clients. However, as the company grew, it expanded its offerings to include digital platform engineering, product development, and consulting services.

EPAM experienced significant growth throughout the 2000s and 2010s, expanding its global footprint and increasing its employee base. The company went public in 2012, trading on the New York Stock Exchange under the ticker symbol "EPAM."

Today, EPAM has over 47,000 employees across 35 countries, serving clients in various industries, including financial services, healthcare, retail and distribution, travel and hospitality, and media and entertainment. The company has received numerous awards for its work, including being recognized as a leader in the software development industry.

EPAM's growth has been driven by its focus on delivering innovative solutions to its clients and providing exceptional service. The company has also invested heavily in its employees, fostering a culture of continuous learning and development to ensure that its workforce is equipped to handle the challenges of the digital age.

Various departments and their functions

EPAM Systems has various departments that work together to deliver high-quality solutions to its clients. Here are some of the key departments and their functions:

Software Engineering: This department is responsible for developing custom software solutions for clients across various industries. The team works closely with clients to understand their business needs and develop software that meets their specific requirements.

Digital Platform Engineering: This department focuses on designing and building digital platforms that enable clients to deliver engaging and personalized experiences to their customers. The team leverages cutting-edge technologies to create scalable and secure platforms that can handle large volumes of data and traffic.

Product Development: This department is responsible for developing and launching new products for clients in various industries. The team works closely with clients to identify market opportunities and develop products that meet customer needs and preferences.

Quality Assurance and Testing: This department is responsible for ensuring that all software and digital platforms developed by EPAM meet the highest quality standards. The team uses a variety of testing methodologies to identify and resolve issues before software is released to clients.

UX/UI Design: This department is responsible for designing user interfaces and experiences that are intuitive, engaging, and visually appealing. The team works closely with clients to understand their target audience and design interfaces that meet their needs and preferences.

Data and Analytics: This department focuses on leveraging data and analytics to drive business insights and inform decision-making. The team works with clients to collect, analyze, and visualize data, providing actionable insights that can improve business performance.

Consulting: This department provides strategic guidance to clients, helping them navigate the complex landscape of digital transformation. The team works closely with clients to understand their business objectives and develop solutions that drive growth and profitability.



Arkadiy Dobkin

Chairman of the Board, CEO & President

CHAPTER2

INTRODUCTION TO PROJECT

▪ 2.1 Spotify-UI Test

For my project report, I conducted a UI test for Spotify, focusing on the login process, playlist creation, adding songs, playing, and pausing songs, and deleting both songs and playlists.

- **1. Login process:** I first tested the login process of Spotify, making sure that users can enter their username and password correctly and access their account. I also checked for any error messages or bugs that may occur during the login process.
- **2. Playlist creation:** I then proceeded to test the playlist creation feature of Spotify. I created a new playlist and ensured that the name was entered correctly. I also tested the addition of a description and cover art. I then checked that the playlist appeared in the user's library.
- **3. Adding songs:** Next, I tested the ability to add songs to a playlist. I selected a song and added it to the newly created playlist. I then checked that the song was added correctly and appeared in the playlist.
- **4. Play and pause:** I tested the play and pause feature of Spotify. I clicked on the play button to start playing the song and then clicked on the pause button to stop the song. I then checked that the play and pause buttons worked correctly and the song played and paused as expected.
- **5. Deletion of song and playlist:** Finally, I tested the ability to delete both songs and playlists. I removed a song from the playlist and checked that it was no longer in the playlist. I then deleted the playlist entirely and ensured that it was removed from the user's library.

Overall, I found that the UI of Spotify was intuitive and easy to use. The login process was straightforward, and playlist creation and management were simple. Adding songs, playing and pausing songs, and deleting songs and playlists were also easy to do. There were no significant issues or bugs that I encountered during the test.

Objectives of the work undertaken:

- ❖ **Test the performance of the website:** The objective of the testing is to test the performance of the website under different conditions, such as whether all functionalities are working properly. This will help identify any bottlenecks or issues that may affect the website's performance.
- ❖ **Evaluate the usability of the website:** The second objective of the testing is to evaluate the usability of the website. This involves testing the user interface of the website, including the layout, navigation, and ease of use.
- ❖ **Ensure that the website functions properly:** The primary objective of the testing is to ensure that the website is functioning as intended. This involves testing the basic functionality of the website, such as navigating to different pages, searching for songs, and add to playlist, remove song to playlist, create playlist, delete playlist.

Scope of the Work:

The scope of the testing was limited to the following areas:

- ❖ Developed an automation framework for the Spotify website.
- ❖ Write Test Cases for “**UI -Spotify**”
- ❖ **Test cases:**
 - Check login functionality.
 - Check creates playlist functionality.
 - Check add song to playlist functionality.
 - Check play and pause song functionality.
 - Check delete song functionality.

- Check delete playlist functionality.
- Logout and delete cookies.

❖ Framework Details:

- WebDriver manager for managing browser connectors.
- Page Factory for page abstractions
- Property files with test data
- XML suites for Test Management
- Parallel Testing to save Time.
- Use TestNg

❖ Technologies Used

For my project report, I used several technologies to perform UI testing for the Spotify application. These technologies include Maven, Java, Selenium Web-Driver, Jenkins, TestNG, and the Page Object Pattern.

- **Maven:** Maven is a build automation tool that helps in managing project dependencies, build processes, and deployment. It simplifies the development process by automating the building of projects and managing libraries and dependencies. I used Maven to build and manage the project dependencies of my UI testing framework.
- **2. Java:** Java is a popular programming language used for developing enterprise applications, web applications, and mobile applications. I used Java to write the test scripts for my UI testing framework. Java is a robust language with an extensive library of APIs and tools, making it an excellent choice for UI testing.
- **3. Selenium WebDriver:** Selenium WebDriver is a popular open-source tool used for automating web browsers. It supports different browsers, including Chrome, Firefox, and Safari, and provides a set of APIs for interacting with web elements. I used Selenium WebDriver to automate my UI testing for the Spotify application.
- **4. Jenkins:** Jenkins is an open-source automation server that helps in building, testing, and deploying software applications. It provides a wide range of plugins that enable continuous integration and continuous deployment. I used Jenkins to set up and run automated tests as part of the CI/CD pipeline.
- **5. TestNG:** TestNG is a testing framework that provides a wide range of features for automating unit tests, integration tests, and functional tests. It supports different test types,

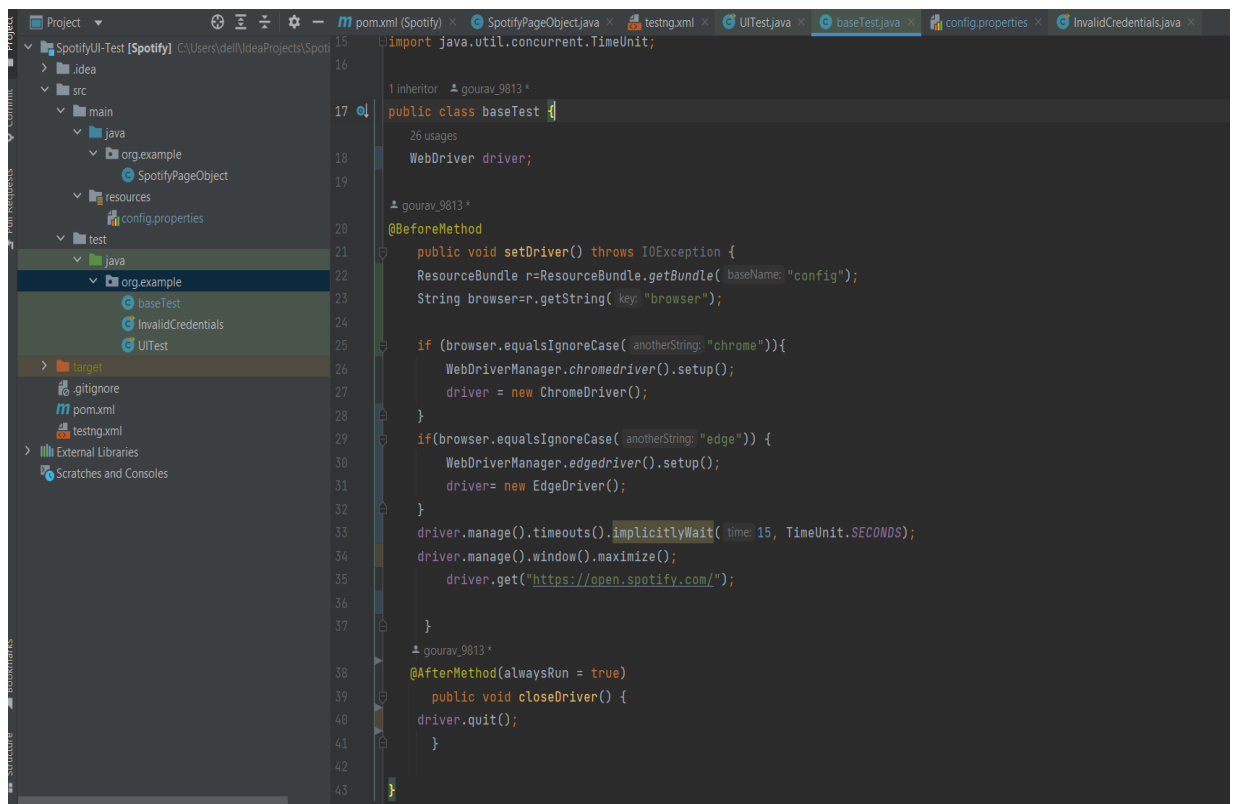
including data-driven tests, parameterized tests, and dependency tests. I used TestNG to create and execute test cases for my UI testing framework

- **6. Page Factory Pattern:** The Page factory Pattern is a design pattern used for implementing UI testing frameworks. It helps in creating a modular and maintainable code structure by separating the page objects from the test scripts. I used the Page factory Pattern to structure my UI testing framework, making it easier to maintain and update.

In summary, I used Maven, Java, Selenium WebDriver, Jenkins, TestNG, and the Page factory Pattern to create a robust and reliable UI testing framework for the Spotify application. These technologies helped me automate the testing process, reducing the time and effort required to test the application manually.

Screenshot

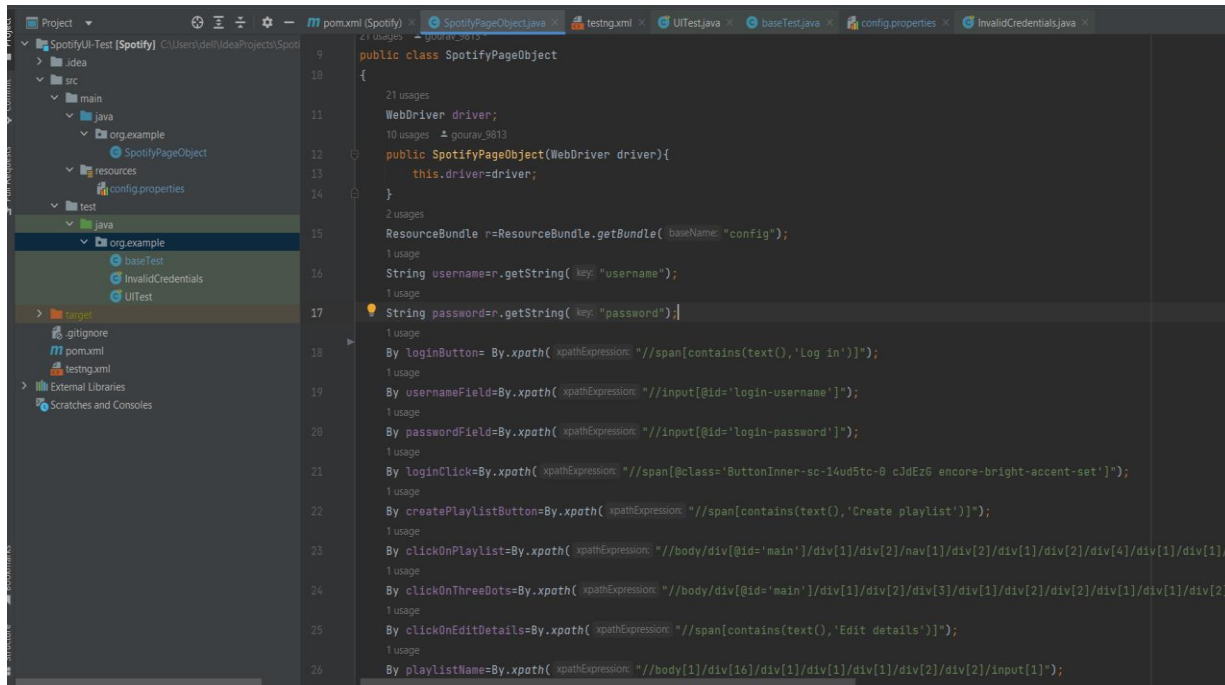
- **BaseTest class** -used to store common code for all test and run on different browser and close test after test done.



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a 'test' directory with a 'java' subdirectory containing 'baseTest', 'InvalidCredentials', and 'UITest' classes. The code editor displays the 'baseTest' class, which is a public class with a 'WebDriver driver' field. It includes a '@BeforeMethod' annotation for a 'setDriver()' method that configures the driver based on the browser type (chrome or edge) and a '@AfterMethod' annotation for a 'closeDriver()' method that quits the driver. The code is as follows:

```
15 import java.util.concurrent.TimeUnit;
16
17 public class baseTest {
18     WebDriver driver;
19
20     @BeforeMethod
21     public void setDriver() throws IOException {
22         ResourceBundle r=ResourceBundle.getBundle( "baseName: \"config\"");
23         String browser=r.getString( key: \"browser\");
24
25         if (browser.equalsIgnoreCase( anotherString: \"chrome\")){
26             WebDriverManager.chromedriver().setup();
27             driver = new ChromeDriver();
28         }
29         if(browser.equalsIgnoreCase( anotherString: \"edge\")) {
30             WebDriverManager.edgedriver().setup();
31             driver= new EdgeDriver();
32         }
33         driver.manage().timeouts().implicitlyWait( time: 15, TimeUnit.SECONDS);
34         driver.manage().window().maximize();
35         driver.get(\"https://open.spotify.com/\");
36     }
37
38     @AfterMethod(alwaysRun = true)
39     public void closeDriver() {
40         driver.quit();
41     }
42
43 }
```

Page class – used to store page elements and methods of actions



```
9 public class SpotifyPageObject
10 {
11     21 usages
12     WebDriver driver;
13     10 usages  gourav_9813
14     public SpotifyPageObject(WebDriver driver){
15         this.driver=driver;
16     }
17     2 usages
18     ResourceBundle r=ResourceBundle.getBundle( "baseName: 'config'");
19     1 usage
20     String username=r.getString( key: "username");
21     1 usage
22     String password=r.getString( key: "password");
23     1 usage
24     By loginButton= By.xpath( xpathExpression: "//span[contains(text(),'Log in')]");
25     1 usage
26     By usernameField=By.xpath( xpathExpression: "//input[@id='login-username']");
27     1 usage
28     By passwordField=By.xpath( xpathExpression: "//input[@id='login-password']");
29     1 usage
30     By loginClick=By.xpath( xpathExpression: "//span[@class='ButtonInner-sc-14ud5tc-0 cJdEzG encore-bright-accent-set']");
31     1 usage
32     By createPlaylistButton=By.xpath( xpathExpression: "//span[contains(text(),'Create playlist')]");
33     1 usage
34     By clickOnPlaylist=By.xpath( xpathExpression: "//body/div[@id='main']/div[1]/div[2]/nav[1]/div[2]/div[1]/div[2]/div[4]/div[1]/div[1]/div[1]");
35     1 usage
36     By clickOnThreeDots=By.xpath( xpathExpression: "//body/div[@id='main']/div[1]/div[2]/div[3]/div[1]/div[2]/div[2]/div[1]/div[1]/div[2]");
37     1 usage
38     By clickOnEditDetails=By.xpath( xpathExpression: "//span[contains(text(),'Edit details')]");
39     1 usage
40     By playlistName=By.xpath( xpathExpression: "//body[1]/div[16]/div[1]/div[1]/div[1]/div[2]/div[2]/input[1]");
```

```
public void login() throws InterruptedException {
    driver.findElement(loginButton).click();
    Thread.sleep( millis: 1000);
    driver.findElement(usernameField).sendKeys(username);
    driver.findElement(passwordField).sendKeys(password);
    Thread.sleep( millis: 500);
    driver.findElement(loginClick).click();
    Thread.sleep( millis: 2000);
}
9 usages  gourav_9813
public void createPlaylist() throws InterruptedException {
    driver.findElement(createPlaylistButton).click();
    Thread.sleep( millis: 2000);
}
2 usages  gourav_9813
public void editPlaylistDetails() throws InterruptedException {
    driver.findElement(clickOnPlaylist).click();
    Thread.sleep( millis: 1000);
    driver.findElement(clickOnThreeDots).click();
    Thread.sleep( millis: 1000);
    driver.findElement(clickOnEditDetails).click();
    Thread.sleep( millis: 1000);

    WebElement nameInput2 = driver.findElement(playlistName);
    nameInput2.clear();
    Thread.sleep( millis: 500);
    nameInput2.sendKeys( ...keysToSend: "Test Playlist");
    driver.findElement(clickSave).click();
    Thread.sleep( millis: 1000);
}
```

```

3 usages  📄 gourav_9813
public void addSongToPlaylist() throws InterruptedException {
    driver.findElement(addButton).click();
    Thread.sleep( millis: 1000 );
    driver.findElement(crossButton).click();
    Thread.sleep( millis: 500 );
}

3 usages  📄 gourav_9813
public void playSong() throws InterruptedException {
    driver.findElement(playButton).click();
    Thread.sleep( millis: 5000 );
}

2 usages  📄 gourav_9813
public void pauseSong() throws InterruptedException {
    driver.findElement(pauseButton).click();
    Thread.sleep( millis: 1000 );
}

2 usages  📄 gourav_9813
public void removeSongFromPlaylist() throws InterruptedException {
    driver.findElement(clickOnSong).click();
    Thread.sleep( millis: 1000 );
    driver.findElement(removeSong).click();
    Thread.sleep( millis: 1000 );
}

9 usages  📄 gourav_9813
public void deletePlaylist() throws InterruptedException {
    driver.findElement(playlistOption).click();
    Thread.sleep( millis: 1000 );
    driver.findElement(delete).click();
    Thread.sleep( millis: 1000 );
    driver.findElement(deleteConfirm).click();
    Thread.sleep( millis: 1000 );
}

```

- **Test class-** use to create diff test cases and call page class methods.

```

public class UITest extends baseTest {
    no usages
    SpotifyPageObject spo;

    📄 gourav_9813
    @Test
    public void loginTest() throws InterruptedException {
        SpotifyPageObject spo = new SpotifyPageObject(driver);
        spo.login();
        driver.manage().deleteAllCookies();
    }

    📄 gourav_9813
    @Test
    public void createPlayList() throws InterruptedException {
        SpotifyPageObject spo = new SpotifyPageObject(driver);
        spo.login();
        spo.createPlaylist();
        spo.deletePlaylist();
        driver.manage().deleteAllCookies();
    }

    📄 gourav_9813
    @Test
    public void EditPayListDetails() throws InterruptedException {
        SpotifyPageObject spo = new SpotifyPageObject(driver);
        spo.login();
        spo.createPlaylist();
        spo.editPlaylistDetails();
        spo.deletePlaylist();
        driver.manage().deleteAllCookies();
    }
}

```

```

@Test
public void SearchSong() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.searchSong("Ram Siya Ram");
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

gourav_9813
@Test
public void AddSongToPlayList() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.searchSong("Ram Siya Ram");
    spo.addSongToPlaylist();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

gourav_9813
@Test
public void PlaySong() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.searchSong("Ram Siya Ram");
    spo.addSongToPlaylist();
    spo.playSong();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

```

```

@Test
public void PauseSong() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.searchSong("Ram Siya Ram");
    spo.addSongToPlaylist();
    spo.playSong();
    spo.pauseSong();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

gourav_9813
@Test
public void RemoveSong() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.searchSong("Ram Siya Ram");
    spo.addSongToPlaylist();
    spo.removeSongFromPlaylist();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

gourav_9813
@Test
public void DeletePlayList() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

```

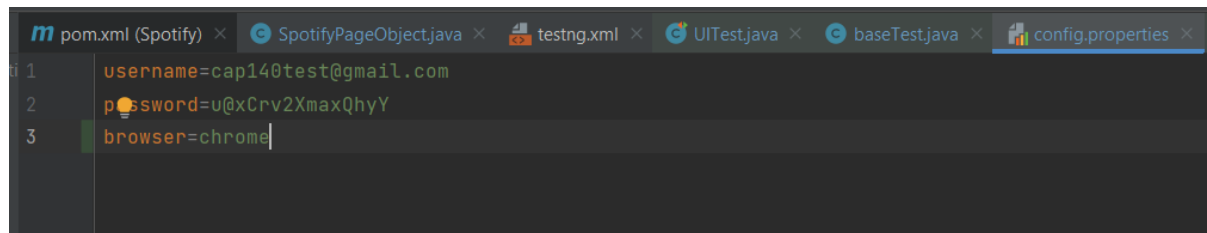


```

@Test
public void FullTest() throws InterruptedException {
    SpotifyPageObject spo = new SpotifyPageObject(driver);
    spo.login();
    spo.createPlaylist();
    spo.editPlaylistDetails();
    spo.searchSong("Ram Siya Ram");
    spo.addSongToPlaylist();
    spo.playSong();
    spo.pauseSong();
    spo.removeSongFromPlaylist();
    spo.deletePlaylist();
    driver.manage().deleteAllCookies();
}

```

- **Config properties file** –used to store test data.



The screenshot shows an IDE with several tabs open: pom.xml (Spotify), SpotifyPageObject.java, testng.xml, UITest.java, baseTest.java, and config.properties. The config.properties file is active and contains the following properties:

```

1 username=cap140test@gmail.com
2 password=u@xCrv2XmaxQhyY
3 browser=chrome

```

- **Negative test case of login-**

```

public class InvalidCredentials {
    5 usages
    ChromeDriver driver = new ChromeDriver();
    @gourav_9813
    @Test
    public void emptyCredentials() throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        driver.get("https://open.spotify.com/");
        driver.manage().window().maximize();
        driver.findElement(By.xpath(xpathExpression: "//span[contains(text(),'Log in')]")).click();
        Thread.sleep(millis: 1000);
        driver.findElement(By.xpath(xpathExpression: "//span[@class='ButtonInner-sc-14ud5tc-0 cJdEz6 encore-bright-accent-set']")).click();
        boolean isEmptyFieldsLoginSuccessful = driver.getCurrentUrl().contains("login");
        Assert.assertEquals(isEmptyFieldsLoginSuccessful, expected: true);
    }
}

```

- **Passed test cases- 11/11 passes**

The screenshot displays a list of test cases on the left and their corresponding logs on the right. All test cases passed successfully.

Test Case	Duration	Status
Invalid Credentials	25 sec 338 ms	Passed
InvalidCredentials	25 sec 338 ms	Passed
emptyCredentials	25 sec 338 ms	Passed
AllFunctionality	35 sec 126 ms	Passed
UITest	35 sec 126 ms	Passed
FullTest	30 sec 147 ms	Passed
LoginTest	10 sec 370 ms	Passed
UITest	10 sec 370 ms	Passed
loginTest	5 sec 444 ms	Passed
createPlayListTest	21 sec 333 ms	Passed
UITest	21 sec 333 ms	Passed
createPlayList	14 sec 224 ms	Passed
EditPayListDetails	28 sec 798 ms	Passed
UITest	28 sec 798 ms	Passed
EditPayListDetails	19 sec 535 ms	Passed
SearchSong	19 sec 881 ms	Passed
UITest	19 sec 881 ms	Passed
SearchSong	15 sec 358 ms	Passed
AddSongToPlaylist	23 sec 61 ms	Passed
UITest	23 sec 61 ms	Passed
AddSongToPlaylist	17 sec 89 ms	Passed
PlaySong	25 sec 959 ms	Passed
UITest	25 sec 959 ms	Passed
PlaySong	21 sec 698 ms	Passed
PauseSong	28 sec 697 ms	Passed
UITest	28 sec 697 ms	Passed
PauseSong	23 sec 975 ms	Passed
RemoveSong	25 sec 969 ms	Passed
UITest	25 sec 969 ms	Passed
RemoveSong	19 sec 211 ms	Passed
DeletePlaylist	22 sec 624 ms	Passed
UITest	22 sec 624 ms	Passed
DeletePlaylist	13 sec 428 ms	Passed

The logs on the right show the following messages:

```

ChromeDriver was started successfully.
May 02, 2023 6:58:18 PM org.openqa.selenium.devtools.CdpVersionFinder
WARNING: Unable to find an exact match for CDP version 112, so returning
Starting ChromeDriver 112.0.5615.49 (bd2a7bcb881c11e8cfe3078709382
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations
ChromeDriver was started successfully.
May 02, 2023 6:58:44 PM org.openqa.selenium.devtools.CdpVersionFinder
WARNING: Unable to find an exact match for CDP version 112, so returning
Starting ChromeDriver 112.0.5615.49 (bd2a7bcb881c11e8cfe3078709382
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations
ChromeDriver was started successfully.
May 02, 2023 6:59:13 PM org.openqa.selenium.devtools.CdpVersionFinder
WARNING: Unable to find an exact match for CDP version 112, so returning
Starting ChromeDriver 112.0.5615.49 (bd2a7bcb881c11e8cfe3078709382
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations
ChromeDriver was started successfully.
May 02, 2023 6:59:39 PM org.openqa.selenium.devtools.CdpVersionFinder
WARNING: Unable to find an exact match for CDP version 112, so returning
Starting ChromeDriver 112.0.5615.49 (bd2a7bcb881c11e8cfe3078709382
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations
ChromeDriver was started successfully.

=====
All Test Suite
Total tests run: 11, Passes: 11, Failures: 0, Skips: 0
=====

Process finished with exit code 0
  
```

- **Jenkins result- used trigger for every 5 minutes.**

The screenshot shows the Jenkins Test Results Analyzer interface. The top navigation bar includes the Jenkins logo and a search bar. The main content area displays the test results for the 'SpotifyTest' project.

Test Results Table:

Chart	Package/Class/Testmethod	Passed	Transitions	8	7	6	5	4
<input type="checkbox"/>	org.example	100% (100%)	0	PASSED	PASSED	PASSED	PASSED	PASSED

Top 10 Most Broken Tests:

There are no failing tests.

Build Status:

The graph shows the number of tests passed over time. The x-axis represents the build number (4, 5, 6) and the y-axis represents the number of tests (0, 1, 2, 3). The data points are as follows:

Build number	No of tests
4	2
5	2
6	2

Build History:

The build history shows the following builds:

- #8: 02-May-2023 1:22 pm
- #7: 02-May-2023 1:13 pm
- #6: 02-May-2023 12:13 am
- #5: 02-May-2023 12:13 am

(2.2) Individual Tasks

Objectives of the work undertaken:

- ❖ **Test the performance of the website:** The objective of the testing is to test the performance of the website under different conditions, such as all functionality are working properly . This will help identify any bottlenecks or issues that may affect the website's performance.
- ❖ **Evaluate the usability of the website:** The second objective of the testing is to evaluate the usability of the website. This involves testing the user interface of the website, including the layout, navigation, and ease of use.
- ❖ **Ensure that the website functions properly:** The primary objective of the testing is to ensure that the website is functioning as intended. This involves testing the basic functionality of the website, such as navigating to different pages, searching for services, and calculating cost for given services.

Scope of the Work:

The scope of the testing was limited to the following areas:

- ❖ Developed an automation framework for the Hurt Me Plenty task in Webdriver module.
- ❖ Write Test Cases for “**Google Cloud Platform Pricing**”
- ❖ Test cases:
 - Check monthly rent.
 - Check instance type.
 - Check local SSD.
 - Check region.
 - Check commitment term.
 - Check selected series.
 - Automate pasteBin.com.

❖ Framework Details:

- WebDriver manager for managing browser connectors.
- Page Factory for page abstractions
- Property files with test data
- XML suites for Test Management
- Parallel Testing to save Time.
- Use TestNg

❖ Jenkins Details:

- Set up build triggers so that the task is performed every 5 minutes.
- - Clone the project (<https://github.com/vitalliuss/hellocli>) –
- Launch tests from the project in Java directory with the help of mvn test goal.

Importance and Applicability

Cost-effective: Testing the website with Selenium and Java is a cost-effective way to ensure that the website is functioning as intended. This is because it helps to identify and resolve issues before they become major problems, which can be more expensive to fix.

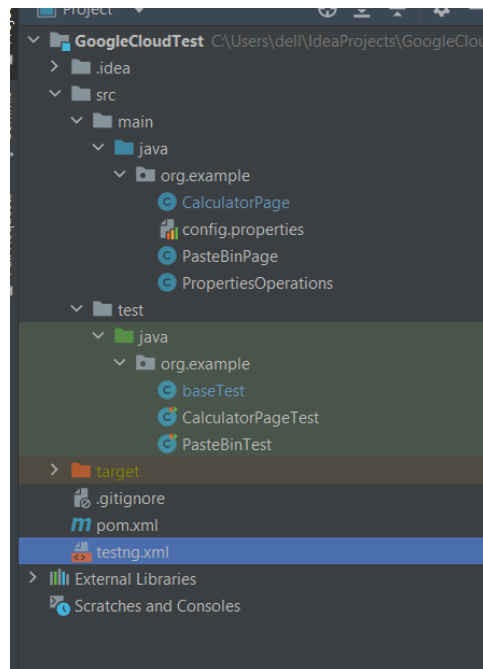
Ensures website functionality: Testing the website with Selenium and Java helps to ensure that the website functions as intended. This ensures that users can navigate the website, search for services, and calculate cost of the given services.

Identifies and resolves issues: Testing the website with Selenium and Java helps to identify any issues that may be present on the website. These issues can then be resolved, which helps to improve the website's quality and user experience.

Improves website performance: Testing the website with Selenium and Java helps to identify any performance bottlenecks that may be present on the website. These bottlenecks can then be addressed, which helps to improve the website's performance and speed.

Screenshot:

- **File Path (Structure)**



- **Property class**—used to get data from config.properties file

```
public class PropertiesOperations {

    //object of property file
    Properties prop=new Properties();

    public String getPropertyFileData(String key) throws IOException {

        //load data first

        String propFilePath=System.getProperty("user.dir")+ "/src/main/java/org/example/config.properties";
        FileInputStream fis=new FileInputStream(propFilePath);
        prop.load(fis);

        //read data
        String value=prop.get(key).toString();
        return value;
    }
}
```

- **Test Case Pass --all test cases passed**

The screenshot displays the Selenium IDE interface with a test suite named 'All Test Suite' that has completed successfully. The left sidebar shows a tree view of the test suite and its individual test cases, all marked with green checkmarks. The right pane shows the execution log, which includes warnings about CDP version 112 not being found, but no failures. The summary at the bottom indicates that all 7 tests passed.

Test Case	Duration	Status
Check commitment	5 sec 442 ms	Pass
Check region	5 sec 182 ms	Pass
Check Cost	5 sec 335 ms	Pass
Check SSD	5 sec 191 ms	Pass
Check Instance Type	5 sec 211 ms	Pass
Check VM Class	2 min 22 sec	Pass
CalculatorPageTest	2 min 17 sec	Pass
checkInformationInVmClass	1 min 39 sec	Pass
Check region	2 min 42 sec	Pass
Check SSD	51 sec 625 ms	Pass
Check Instance Type	51 sec 625 ms	Pass
CalculatorPageTest	51 sec 625 ms	Pass
checkInstanceType	2 min 40 sec	Pass
Check Cost	2 min 40 sec	Pass
CalculatorPageTest	2 min 40 sec	Pass
checkCost	2 min 44 sec	Pass
Check commitment	2 min 44 sec	Pass
CalculatorPageTest	2 min 44 sec	Pass
checkCommitment	486 ms	Pass
Check Instance Type	1 min 20 sec	Pass
Paste bin	1 min 20 sec	Pass
PasteBinTest	1 min 19 sec	Pass
paste	374 ms	Pass
Check region	1 sec 159 ms	Pass
Paste bin	361 ms	Pass
Check VM Class	474 ms	Pass
Check Cost	389 ms	Pass
Check SSD	322 ms	Pass
Check commitment		

Summary: All Test Suite
Total tests run: 7, Passes: 7, Failures: 0, Skips: 0
Process finished with exit code 0

- Run Testcase Parallely—all test cases will run at same time.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="All Test Suite" parallel="tests" thread-count="6">
  <test verbose="2" preserve-order="true" name="Check VM Class">
    <classes>
      <class name="org.example.CalculatorPageTest">
        <methods>
          <include name="checkInformationInVmClassString"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

- **Verify Cost Test case**-main test case for calculating cost for given services

```
@Test
public void checkCost() throws InterruptedException {
    calculatorPage1.openGcpPage(url);
    calculatorPage1.goToCloudCalculatorPage(searchText);
    //calculatorPage1.ManageCookie();
    calculatorPage1.NumberOfInstancesField("4");
    calculatorPage1.selectSeries();
    calculatorPage1.selectMachineType();
    calculatorPage1.AddGpusCheckBox();
    calculatorPage1.selectTypeOfGpus();
    calculatorPage1.selectNumberOfGpus();
    calculatorPage1.selectLocalSsd();
    calculatorPage1.selectDataCenterLocation();
    calculatorPage1.selectCommittedUsage();
    calculatorPage1.pushAddToEstimate();
    String cost = calculatorPage1.getCost().getText();
    Assert.assertEquals(cost, expected: "Total Estimated Cost: USD 1,081.20 per 1 month");
}
```

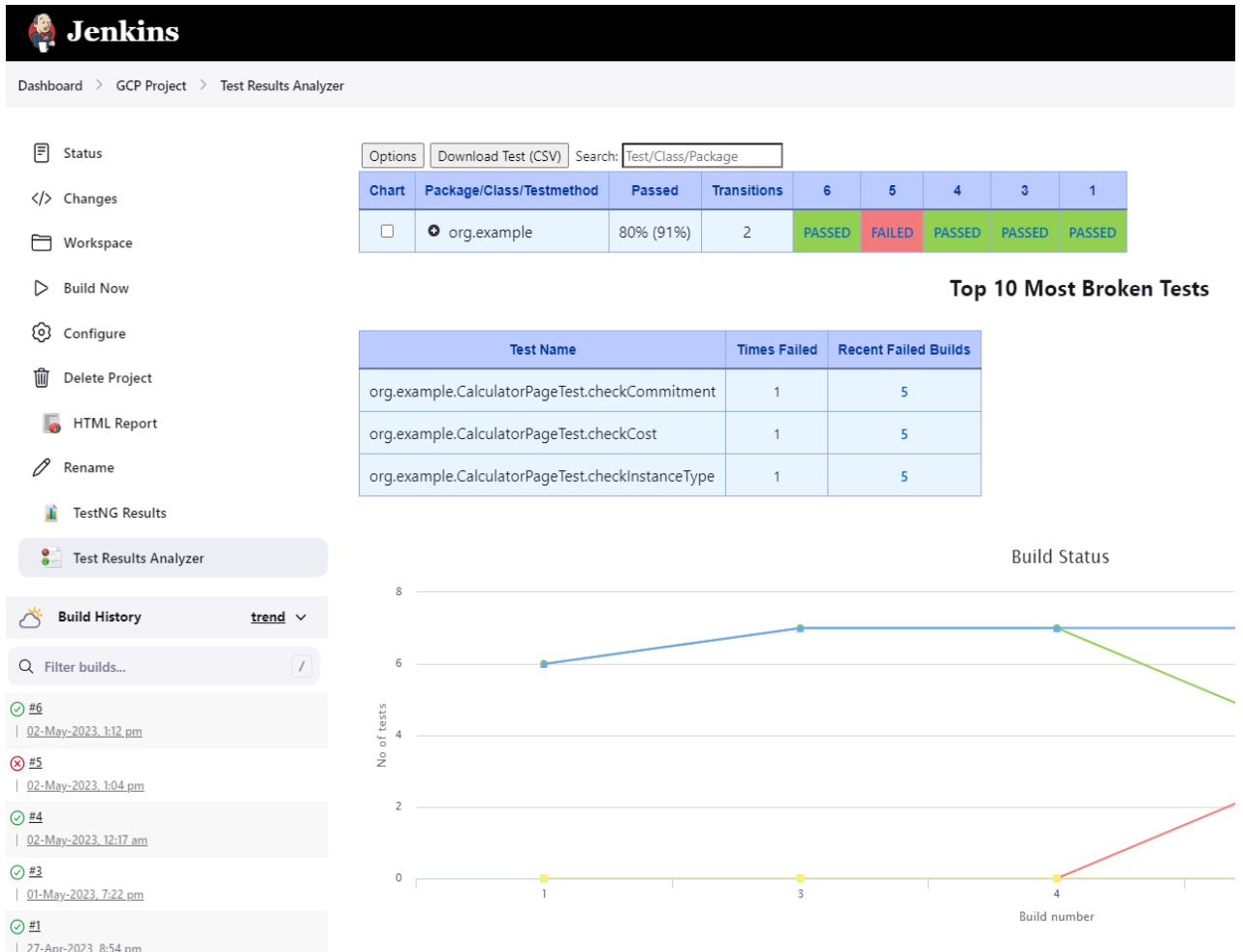
- Property file- contains data which we use in our code ----easy to update data

```
browser= chrome
url = https://cloud.google.com/
searchtext = Google Cloud Platform Pricing Calculator
```

- BaseTest Class—contains common code for all test cases. AND for diff Environment

```
public class baseTest {  
    7 usages  
    WebDriver driver;  
    79 usages  
    CalculatorPage calculatorPage1;  
    5 usages  
    PasteBinPage PasteBinPage1;  
    7 usages  
    String url;  
    7 usages  
    String searchText;  
  
    @BeforeMethod  
  
    public void setDriver() throws IOException {  
  
        PropertiesOperations p = new PropertiesOperations();  
        String browser = p.getPropertyFileData( key: "browser");  
        url=p.getPropertyFileData( key: "url");  
        searchText=p.getPropertyFileData( key: "searchtext");  
  
        if (browser.equalsIgnoreCase( anotherString: "chrome")){  
            WebDriverManager.chromedriver().setup();  
            driver = new ChromeDriver();  
        }  
        if(browser.equalsIgnoreCase( anotherString: "edge")) {  
            WebDriverManager.edgedriver().setup();  
            driver = new EdgeDriver();  
        }  
        driver.manage().timeouts().implicitlyWait( time: 5, TimeUnit.SECONDS);  
        driver.manage().window().maximize();  
        calculatorPage1 = new CalculatorPage(driver);  
    }  
}
```


- Jenkins build for Cloud project and Testng report



Jenkins Part

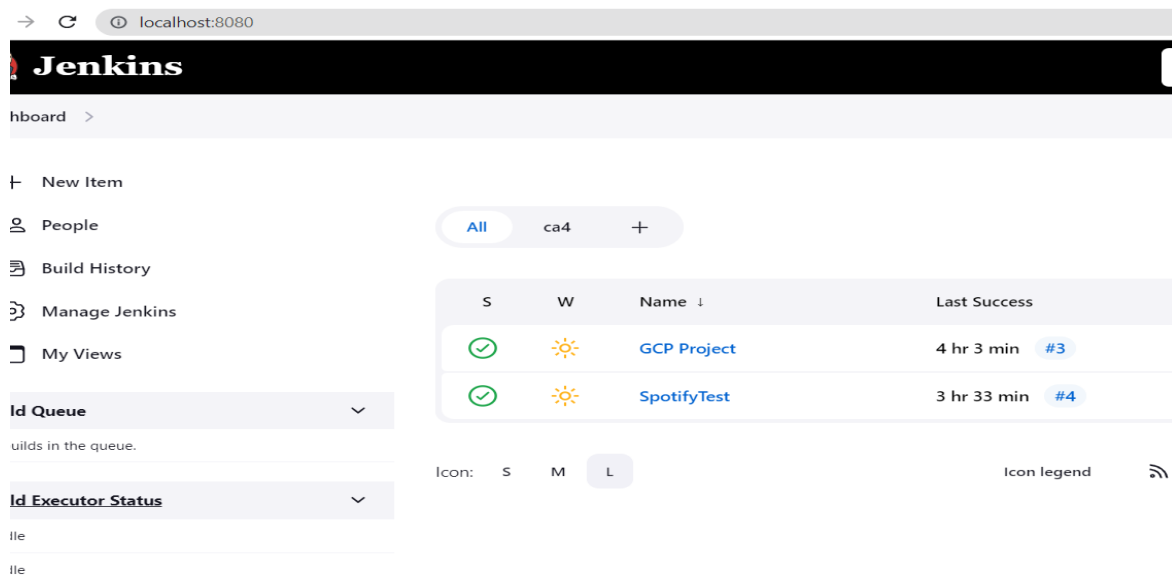
- Start Jenkins-with command `java -jar Jenkins.war --enable-future-java`

```
C:\Windows\System32\cmd.exe - java -jar jenkins.war --enable-future-java
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

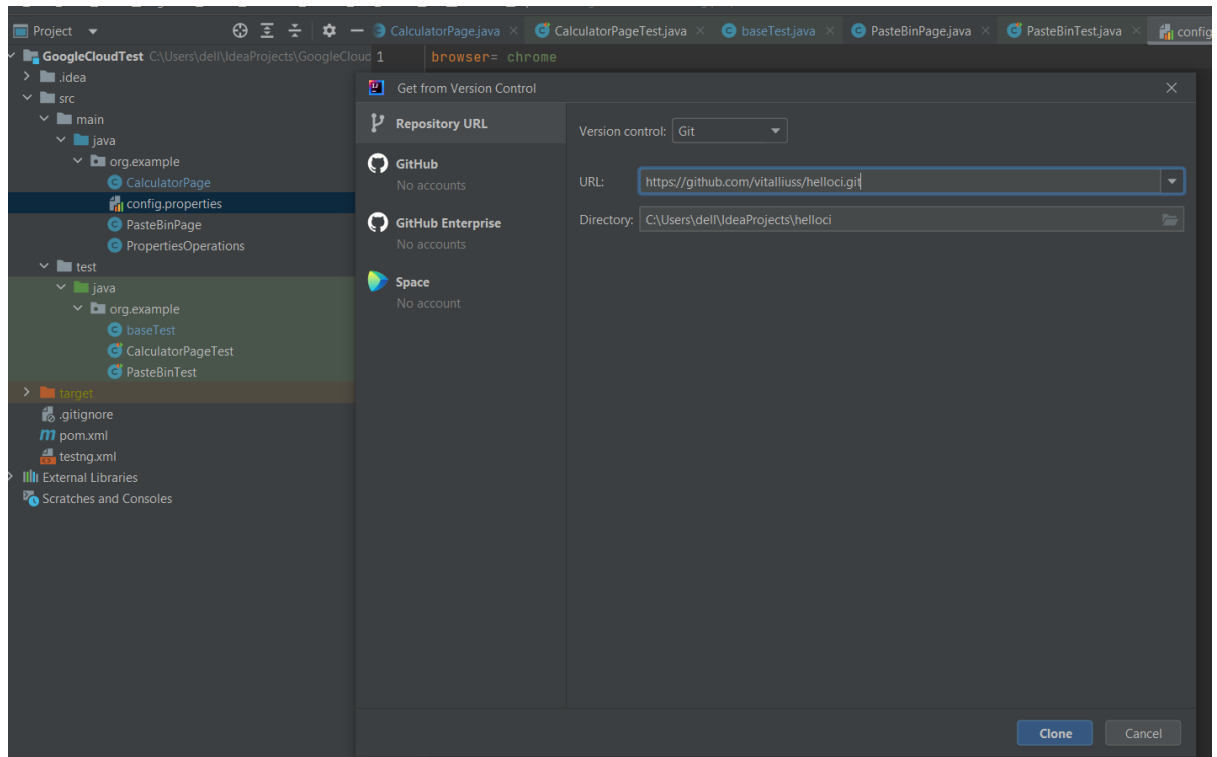
C:\tool\jenkins>java -jar jenkins.war --enable-future-java
Running with Java 19 from C:\Program Files\Java\jdk-19, which is not fully supported. Continuing because --enable-future-java is set. Supported Java versions are: [11, 17]. See https://jenkins.io/redirect/java-support/ for more information.

Running from: C:\tool\jenkins\jenkins.war
webroot: C:\Users\dell\.jenkins\war
2023-05-01 17:54:14.049+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2023-05-01 17:54:14.124+0000 [id=1] WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2023-05-01 17:54:14.251+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-10.0.13; built: 2022-12-07T20:13:20.134Z; git: 1c2636ea05c0ca8de1ffd6ca7f3a98ac084c766d; jvm 19.0.2+7-44
```

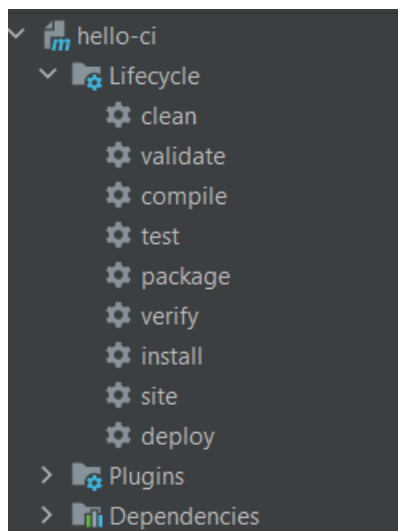
Jenkins localhost:8080- open GUI of Jenkins on port 8080



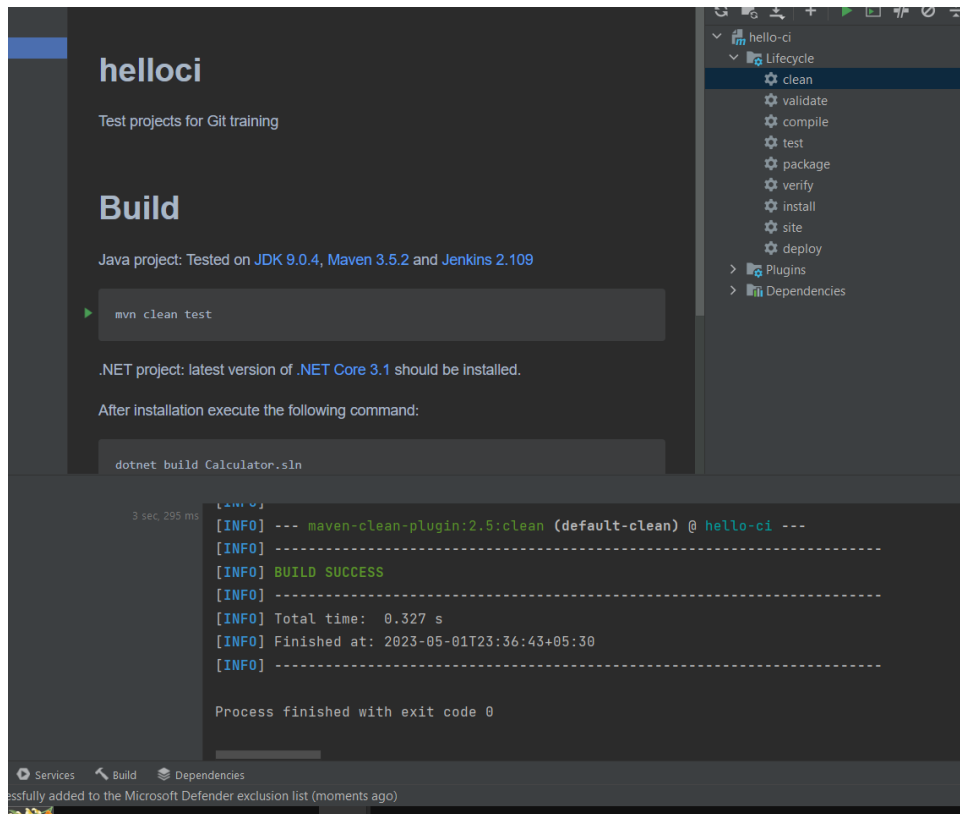
- **Clone Repository**—clone git repo in local repo with IntelliJ idea.



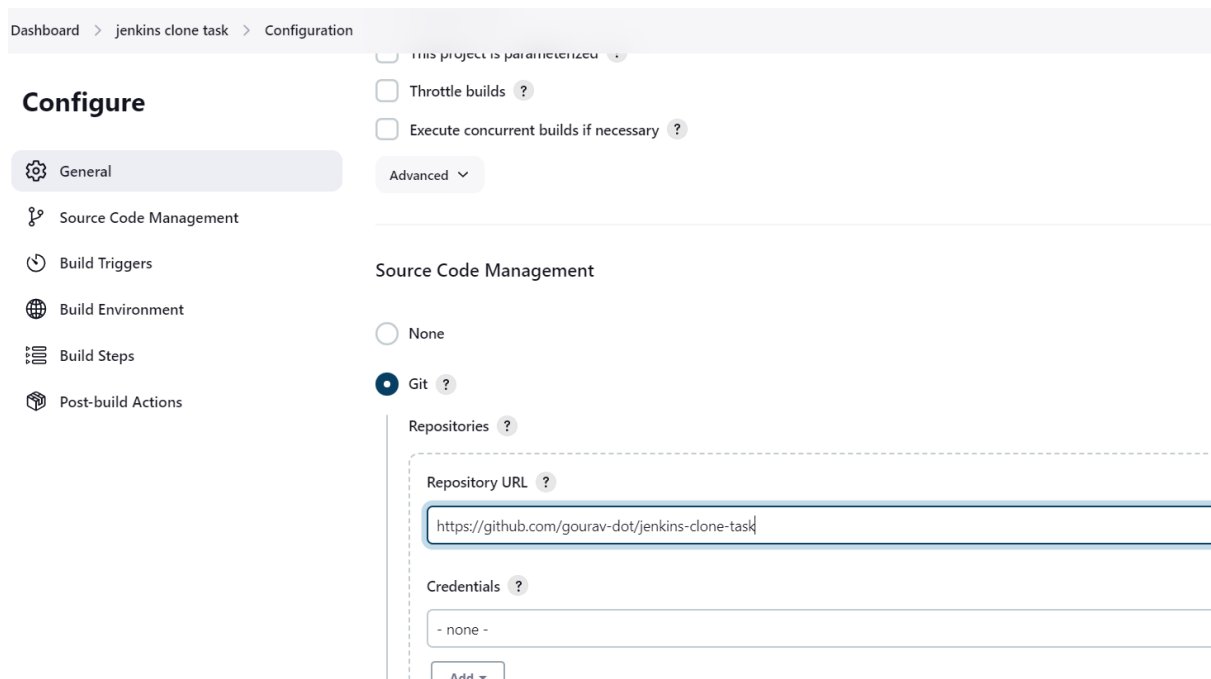
- **Perform MVN Lifecycle**—clear all the lifecycle from maven lifecycle



- **Build Success—from maven life cycle build created successfully.**



- **Push URL of clone in Jenkins Configuration**



- **Build Trigger for 5 minute.**

☒ Build periodically ?

Schedule ?

H/5 * * * * |

Would last have run at Monday, May 1, Time.

- **Maven Clean test**

Build Steps

≡ Invoke top-level Maven targets ?

Maven Version

Maven

Goals

clean test

Advanced ^ Edited

POM ?

pom.xml

- **Jenkin Build Now Successfully**

Dashboard > jenkins clone task >

Status

</> Changes

Workspace

Build Now

Configure

Delete Project

Rename

Test Results Analyzer

Project jenkins clone task

Permalinks

- Last build (#14), 42 sec ago
- Last stable build (#14), 42 sec ago
- Last successful build (#14), 42 sec ago
- Last failed build (#13), 1 min 19 sec ago
- Last unsuccessful build (#13), 1 min 19 sec ago
- Last completed build (#14), 42 sec ago

Build History trend v

Filter builds... /

#15
02-May-2023, 12:07 am

#14
02-May-2023, 12:06 am

■ Jenkins Console Output

Dashboard > jenkins clone task > #15 > Console Output

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#15'

Git Build Data

Previous Build

Console Output

Started by user admin

Running as SYSTEM

Building in workspace C:\Users\dell\.jenkins\workspace\jenkins clone task

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\Users\dell\.jenkins\workspace\jenkins clone task\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url <https://github.com/gourav-dot/jenkins-clone-task.git> # timeout=10

Fetching upstream changes from <https://github.com/gourav-dot/jenkins-clone-task.git>

> git.exe --version # timeout=10

> git --version # 'git version 2.39.0.windows.2'

> git.exe fetch --tags --force --progress -- <https://github.com/gourav-dot/jenkins-clone-task.git> +refs/heads/*:refs/remotes/origin/* # timeout=10

> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10

Checking out Revision 6987b12a3205072742852f2f36eafd7cc535bd97 (refs/remotes/origin/master)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 6987b12a3205072742852f2f36eafd7cc535bd97 # timeout=10

Commit message: "first"

> git.exe rev-list --no-walk 6987b12a3205072742852f2f36eafd7cc535bd97 # timeout=10

[jenkins clone task] \$ cmd.exe /C "C:\apache-maven-3.9.1\bin\mvn.cmd -f pom.xml clean test && exit %ERRORLEVEL%"

[INFO] Scanning for projects...

[INFO]

[INFO] -----< groupId:helloci >-----

[INFO] Building helloci 1.0-SNAPSHOT

[INFO] from pom.xml

[INFO] -----[jar]-----

[INFO]

[INFO] --- clean:3.2.0:clean (default-clean) @ helloci ---

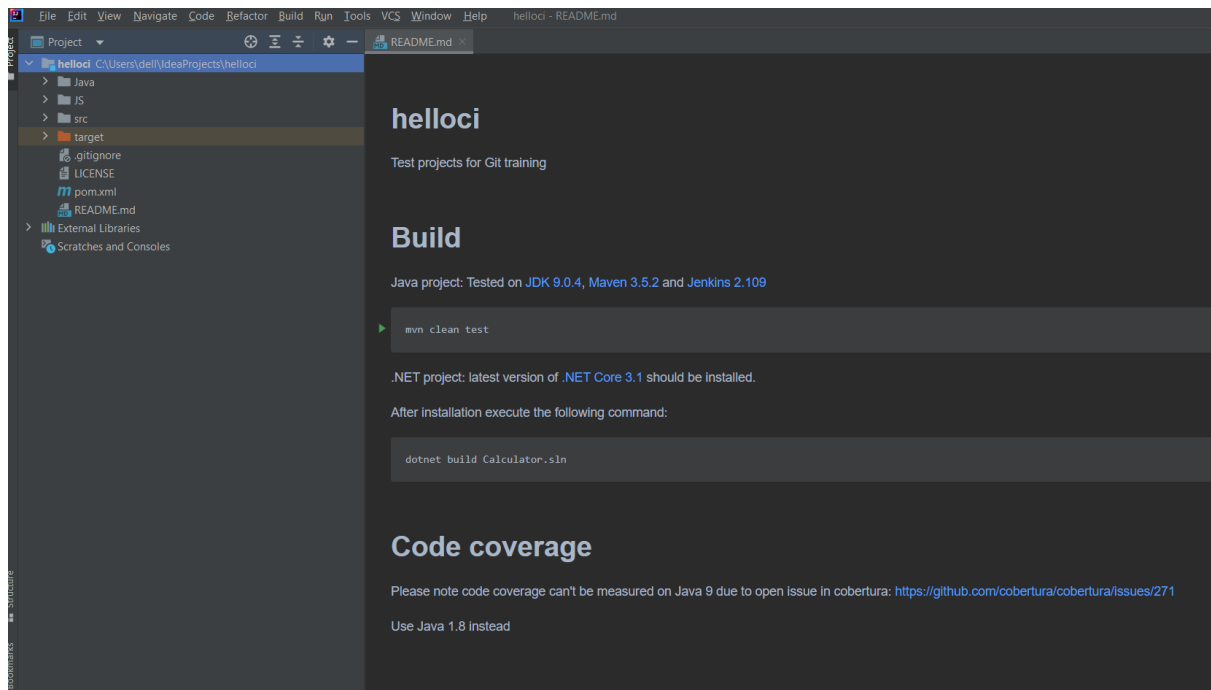
[INFO]

■ Jenkins Console Output

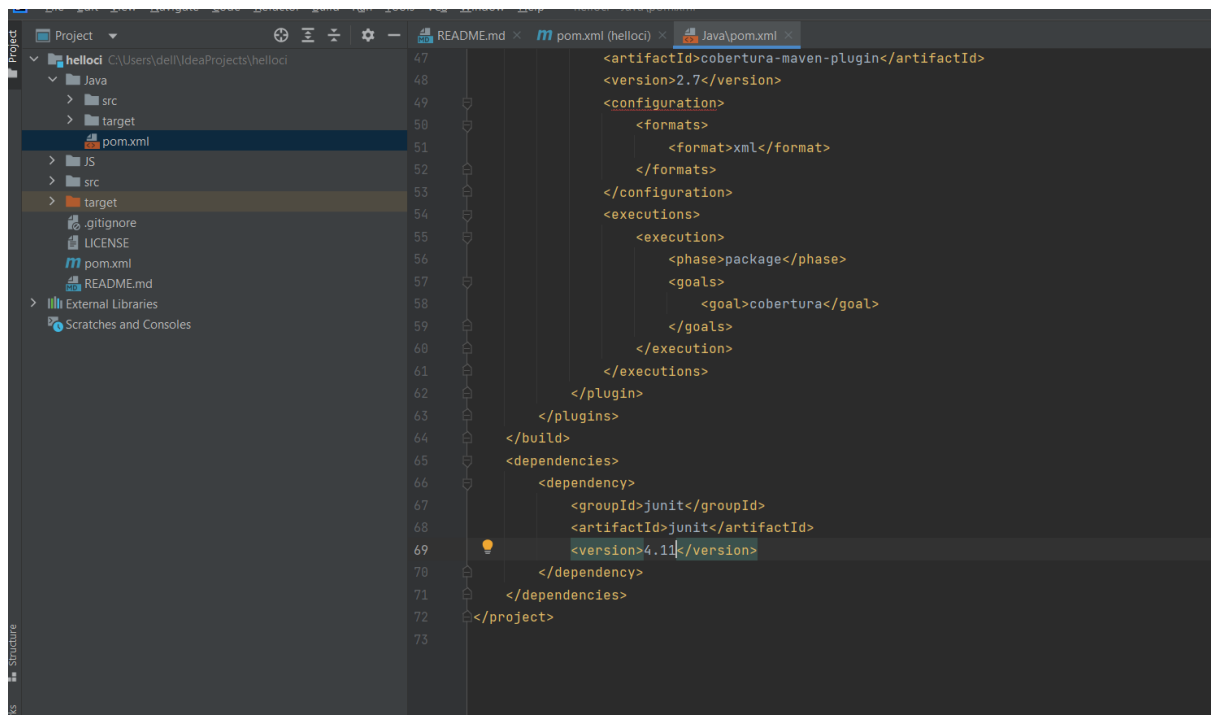
```
[INFO]
[INFO] --- resources:3.3.0:resources (default-resources) @ helloci ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent
[INFO] skip non existing resourceDirectory C:\Users\dell\.jenkins\workspace\jenkins clone task\src\main\resources
[INFO]
[INFO] --- compiler:3.10.1:compile (default-compile) @ helloci ---
[INFO] No sources to compile
[INFO]
[INFO] --- resources:3.3.0:testResources (default-testResources) @ helloci ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent
[INFO] skip non existing resourceDirectory C:\Users\dell\.jenkins\workspace\jenkins clone task\src\test\resources
[INFO]
[INFO] --- compiler:3.10.1:testCompile (default-testCompile) @ helloci ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.0.0:test (default-test) @ helloci ---
[INFO] No tests to run.
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.759 s
[INFO] Finished at: 2023-05-02T00:07:30+05:30
[INFO]
Finished: SUCCESS
```

Maven junit version change Task

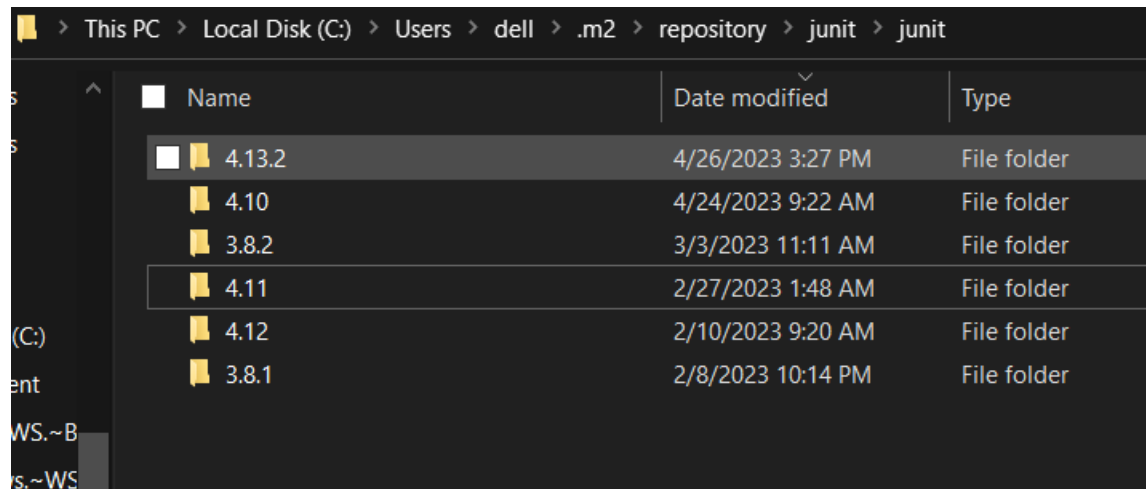
Project downloaded from git repo.



Version change to 4.11



Check junit version is created



Name	Date modified	Type
4.13.2	4/26/2023 3:27 PM	File folder
4.10	4/24/2023 9:22 AM	File folder
3.8.2	3/3/2023 11:11 AM	File folder
4.11	2/27/2023 1:48 AM	File folder
4.12	2/10/2023 9:20 AM	File folder
3.8.1	2/8/2023 10:14 PM	File folder

4.11 and 4.12 and 4.13 already present

CHAPTER-3

TECHNOLOGIES LEARNT

Software Development Methodologies:

- **High Level Overview:** An overview of software development methodologies, their benefits, and their various types.
- **Waterfall:** An introduction to the traditional Waterfall methodology, which is a linear and sequential approach to software development.
- **Agile:** An overview of the Agile methodology, which emphasizes flexibility, customer collaboration, and incremental and iterative development.
- **Scrum:** An in-depth look at the Scrum framework, which is a popular Agile methodology that uses sprints, backlogs, and daily stand-ups to manage projects.
- **Kanban:** An introduction to the Kanban methodology, which emphasizes visual management, continuous flow, and limiting work in progress.
- **Extreme Programming:** An overview of Extreme Programming (XP), which is an Agile methodology that emphasizes customer involvement, continuous testing, and frequent releases.
- **Test-Driven Development:** An introduction to Test-Driven Development (TDD), which is a software development approach that focuses on creating automated tests before writing code.
- **Behavior-Driven Development:** An overview of Behavior-Driven Development (BDD), which is a software development approach that focuses on describing the behavior of a system in natural language.
- **Summary:** A summary of the main points of each methodology, their strengths and weaknesses, and how to choose the right methodology for your project

Version Control with GIT:

- **Version control concept:** Understanding the basics of version control, its benefits, and its various types.
- **Download, install and configure GIT:** Installing GIT on your local machine and configuring it with your user details and preferences.
- **GitHub:** Introduction to GitHub, a web-based hosting service for version control, and its features such as repositories, issues, and pull requests.
- **Git graphical tools:** Overview of graphical user interfaces (GUI) and Integrated Development Environments (IDEs) that can be used to work with Git.
- **Git internals:** Understanding the inner workings of Git, including how Git stores and manages versions of files, branches, commits, and merges.
- **Undoing changes:** How to undo changes made to files or the repository using Git commands such as revert, reset, and checkout.
- **Branching and merge:** Creating and managing branches in Git, and merging changes from one branch to another.
- **Tags:** Creating and managing tags in Git to mark specific points in the repository's history, such as release versions.
- **Stash:** How to use Git stash to temporarily save changes that are not yet ready to be committed.
- **Remotes:** Working with remote repositories in Git, such as cloning, pushing, and pulling changes from remote repositories.
- **Branching strategies:** Overview of different branching strategies and workflows such as Gitflow and Github Flow.

Software Testing Introduction:

- **Introduction to Software Functional Testing:** An overview of software functional testing, its importance, and the different types of functional testing.
- **Test Planning:** An overview of test planning, which involves identifying test objectives, test strategies, and test schedules.
- **Requirements Testing:** An introduction to requirements testing, which involves verifying that the software meets the specified requirements.
- **Test Cases and Test Scenarios:** An in-depth look at test cases and test scenarios, which are used to define the conditions under which software will be tested and the expected results.
- **Defect Reporting:** An overview of defect reporting, which involves identifying and documenting defects in the software.
- **Test Results Reporting:** An introduction to test results reporting, which involves analyzing and documenting the results of the software testing.
- **Test Automation Basics:** An overview of test automation, which involves using software tools to automate the testing process and reduce manual effort.

Java Basics:

- **Introduction to the Java Basics Course:** An overview of the course, its objectives, and the topics covered.
- **Data Types:** An introduction to data types in Java, including primitive and reference types, and how to declare and use them.

- **Conditions and Loops:** An introduction to conditional statements, such as if-else statements and switch statements, and loops such as for loops and while loops.
- **Arrays:** An introduction to arrays in Java, including how to declare and initialize arrays, and how to access and modify their elements.
- **Classes:** An introduction to classes in Java, including how to declare and instantiate classes, and how to use them to create objects.
- **Introduction to OOP:** An introduction to object-oriented programming (OOP) concepts, such as encapsulation, inheritance, and polymorphism.
- **Abstract Classes and Interfaces:** An introduction to abstract classes and interfaces, which are used to define abstract types that can be implemented by other classes.
- **Nested Classes:** An introduction to nested classes in Java, including static nested classes, inner classes, and anonymous classes.
- **Strings:** An introduction to strings in Java, including how to create and manipulate string objects.
- **Collections and Maps:** An introduction to collections and maps in Java, including how to use them to store and manipulate groups of objects.
- **Exceptions:** An introduction to exceptions in Java, including how to use try-catch blocks to handle exceptions and how to create custom exceptions.
- **Annotations:** An introduction to annotations in Java, which are used to provide metadata about code elements.
- **Generics:** An introduction to generics in Java, which allow you to define classes and methods that can work with different types of objects.
- **Enum:** An introduction to enumerations in Java, which are used to define a fixed set of values.

- **Wrapper Classes and Optional Classes:** An introduction to wrapper classes, which are used to represent primitive data types as objects, and optional classes, which are used to represent values that may be null.
- **Code Documentation:** An introduction to code documentation in Java, including how to use Javadoc to document your code.

Data & Analytics - Introduction to SQL:

- **Database Basics:** An introduction to databases, their types, components, and architecture.
- **SQL Foundation:** An introduction to Structured Query Language (SQL), including how to create and manipulate tables, perform queries, and use aggregate functions.
- **SQL for Analysis:** An introduction to using SQL for data analysis, including how to filter, sort, and group data, and how to join tables

Clean Code - Introduction to Clean Code:

- **Introduction to Clean Code:** An overview of clean code and why it is important in software development.
- **Writing Clean Functions:** Best practices for writing clean functions, including how to make them small, focused, and testable.
- **Naming:** Best practices for naming code elements, including classes, methods, and variables.
- **Comments:** Best practices for writing comments, including when to use them and what to include.

- **Error Handling:** Best practices for error handling, including how to handle exceptions and write code that is resilient to errors..

Cloud Computing - Introduction to Cloud Computing:

- **Introduction to Cloud Computing:** An overview of cloud computing and its benefits.
- **Cloud Deployment Models:** An introduction to different cloud deployment models, including public, private, and hybrid clouds.
- **Cloud Service Models:** An introduction to different cloud service models, including infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).
- **Cloud Providers:** An introduction to different cloud providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).
- **Cloud Security:** An overview of cloud security, including best practices for securing cloud-based applications and data.

Automated Testing Basics + Java:

OO Design Principles & Patterns

- **Patterns in General:** An introduction to different types of design patterns, including structural, behavioral, and creational patterns.
- **Factory Pattern:** A creational pattern that provides a way to create objects without specifying their concrete classes.

- **Strategy Pattern:** A behavioral pattern that enables the selection of an algorithm at runtime.
- **Builder Pattern:** A creational pattern that separates the construction of a complex object from its representation.
- **Singleton Pattern:** A creational pattern that ensures a class has only one instance and provides a global point of access to it.

Introduction to Test Automation and xUnit Test Framework:

- **Introduction to Test Automation:** An overview of test automation and its benefits.
- **UI/API/Performance/Security/Mobile Testing:** An introduction to different types of test automation, including user interface (UI), application programming interface (API), performance, security, and mobile testing.
- **Build Tools (Maven):** An introduction to Maven, a build automation tool used primarily for Java projects.
- **TestNG:** An xUnit test framework for Java that supports parameterized, data-driven, and parallel testing.

API Automation:

- **Client-Server Architecture:** An overview of client-server architecture, which is the basis of most web applications.
- **HTTP and HTTP Request/Response:** An introduction to Hypertext Transfer Protocol (HTTP) and HTTP request/response messages.
- **JSON and XML:** An introduction to JavaScript Object Notation (JSON) and Extensible Markup Language (XML), two commonly used data formats for web applications.

- **Postman:** A popular tool for testing and debugging API requests.

Selenium WebDriver (Basic+Advanced):

- **Introduction:** An overview of Selenium WebDriver, a popular web testing tool.
- **HTML and CSS:** An introduction to HTML and CSS, the building blocks of web pages.
- **XPath:** A language used for selecting nodes in an XML or HTML document.
- **Selenium WebDriver:** An introduction to the WebDriver API and its methods for interacting with web pages.
- **WebDriver Waiters:** An introduction to the wait methods in WebDriver, which can help synchronize tests with the page under test.
- **JS Executor:** An advanced feature of WebDriver that allows JavaScript code to be executed within the context of a web page.

Automation Framework:

- **Automation Framework:** Maven + xUnit + WebDriver: This involves the use of Maven as a build tool, xUnit as the testing framework, and WebDriver for automating the web application. This helps in creating a structured and scalable test automation framework.
- **Page Object:** This is a design pattern that separates the page elements of a web application from the test scripts, making the tests more maintainable and reducing code duplication.
- **Page Factory:** This is an extension of the Page Object pattern that uses annotations to initialize web elements, making the code more readable and maintainable.

- **Singleton:** This is a design pattern that ensures only one instance of a class is created, which can be useful in maintaining the state of the application during the test.
- **Production AT Framework:** This includes various approaches like TDD (Test Driven Development), KDT (Keyword Driven Testing), DDT (Data Driven Testing), DDD (Domain Driven Design), BDD (Behavior Driven Development), and BDD + Cucumber. These approaches help in creating a robust and maintainable test automation framework that aligns with the development process.
- **ATF Architecture:** This is an architecture that helps in structuring the test automation framework and integrating it with the development process.
- **Continuous Integration with Jenkins:** This involves integrating the test automation framework with Jenkins, a popular continuous integration tool, to run the tests automatically and generate reports. This helps in achieving continuous testing and faster feedback.

CHAPTER 3

CONCLUSION

The completion of the Maven practical task, as well as the subsequent tasks, showcases a strong grasp of the Maven build tool and the ability to use it efficiently. The task involved downloading a test project and collecting it with Maven, changing the junit version, and ensuring that the new library version was added to the repository. These tasks demonstrate proficiency in managing dependencies and building projects with Maven.

The Webdriver module's practical tasks demonstrate a solid understanding of Selenium WebDriver, framework unit tests, and Page Object concepts. The "I can win" task involved automating the creation of a new paste on a service like Pastebin with specific attributes, while the "Hurt Me Plenty" task automated the use of the Google Cloud Platform Pricing Calculator with specific parameters. Both tasks demonstrate a solid understanding of the Page Object model and the ability to create efficient, maintainable code.

The successful completion of the Framework practical task involves the development of a robust automation framework for the "Hurt Me Plenty" task. The framework includes a WebDriver manager for managing browser connections, page abstractions using Page Object/Page Factory, models for business objects of the required elements, and property files with test data for at least two different environments. The framework also includes XML suites for smoke tests and other tests, screenshot capture when a test fails, and the ability to run with Jenkins, parameterize browsers, test suites, and environments.

The Spotify UI test involved testing various functionalities of the Spotify web application, including login, playlist creation, adding songs, playing and pausing music, and deleting songs and playlists. The test demonstrates proficiency in using Selenium WebDriver, TestNG, and Page Object pattern to automate various tasks, ensuring that the application functions as expected. The successful completion of the test shows that the test cases were comprehensive, and the application meets the necessary requirements.

Overall, the completion of these tasks and the successful execution of the Spotify UI test demonstrate a strong understanding of various technologies such as Maven, Selenium WebDriver, TestNG, Jenkins, and Page Object pattern. These skills are essential for developing and maintaining robust automation frameworks, ensuring that web applications meet the necessary requirements and function as expected.