1.write code for simple user registration form for an event.
Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>REGISTRATION EVENT</title>
</head>
<body bgcolor='wheat'>

          <h2>REGISTRATION FOR AN EVENT </h2>


<form action="./register" method="get" >
<table>
<tr><td>NAME</td><td><input type="text" name="name"
id="name"></td></tr>
<tr><td>MOBILE</td><td><input type="text" name="phone"
id="phone"></td></tr>
<tr><td>EMAIL ID</td><td><input type="text" name="email"
id="email"></td></tr>
<tr>
<td>ATTENDEES</td><td>
<select name='attendeesno' id='attendeesno'>
<option value='1'>ONE PERSON</option>
<option value='2'>TWO PERSON</option>
<option value='3'>THREE PERSON</option>
<option value='4'>FOUR PERSON</option>
</select>
</td></tr>
<tr><td><input type="submit" value="register"></td></tr>
</table>
</form>
</body>
</html>
```

Registration.java

```
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

/**
 * Servlet implementation class Registration
 */
public class Registration extends HttpServlet {
     private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Registration() {
        super();
        // TODO Auto-generated constructor stub
    }

     /**
      * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
      */
     protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
          PrintWriter out=response.getWriter();
          String name=request.getParameter("name");
          String phone=request.getParameter("phone");
          String email=request.getParameter("email");
          String
attendeesno=request.getParameter("attendeesno");
          Connection con=null;
          Statement s=null;
          try
          {
```

```
            Class.forName("com.mysql.cj.jdbc.Driver");

        con=DriverManager.getConnection("jdbc:mysql://localho
st:3303/realdb","root","root");
            s=con.createStatement();
            int result=s.executeUpdate("insert into event
values('"+name+"','"+phone+"','"+email+"','"+attendeesno+"
')");

            out.println("<html><body
bgcolor='wheat'><center><h1>");
            if(result!=0)
            out.println(name+"you are sucessfully
registered");
            else
                out.println(name+"you are unable to
register,please fill form correctly");

            out.println("</h1></center></body></html>");
            s.close();
            con.close();
            }
            catch(Exception e)
            {
            e.printStackTrace();
            }


        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
        }

}
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="https://jakarta.ee/xml/ns/jakartaee"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
version="6.0">
  <servlet>
    <description></description>
    <display-name>Registration</display-name>
    <servlet-name>Registration</servlet-name>
    <servlet-class>Registration</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Registration</servlet-name>
    <url-pattern>/register</url-pattern>
  </servlet-mapping>
  <display-name>Registration</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

2.Explore git and github commands.

git commands.

Git init : To create git repository

Git add file: adding file to staging

Git commit –m “msg”: moving file from staging to .git repository. We can keep track of changes by commiting.

Git branch branchname: creating ing new branch as copy of master.

Git branch : shows all branches and shows where is pointer now.

Git checkout branchname: moving pointer from present branch to specified branch.
Git log: displays all commits with info.

Git log --oneline: displays all commits with each in single line.

Git merge branchname: copy from branch to pointer which is in head state now.

GITHUB COMMANDS

GITHUB FORK:

Fork is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

Fork is not a command in Git, but something offered in GitHub and other repository hosts.

logging in to GitHub, and fork the repository, Now you have your own copy of repository.

GITHUB PULL REQUEST:

On GitHub, and we see that the repository has a new commit. And we can send a Pull Request to the original repository

GITHUB BRANCH:

On GitHub, access your repository and click the "master" branch button.

There you can create a new Branch. Type in a descriptive name, and click Create branch:

Executing process
Create a folder with your roll no as ex12 in any drive

```
Narendarkumboji@DESKTOP-5HUH47P MINGW64 ~
$ cd F:/ex12

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
$ git init
EXPECTED OUTPUT:
Initialized empty Git repository in F:/ex12/.git/

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
$ git add file1.txt

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
$ git commit -m "master 1 version"
EXPECTED OUTPUT:
[master (root-commit) 85310a0] master 1 version
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
$ git branch br1

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
```

```
$ git branch
EXPECTED OUTPUT:
  br1
* master


Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
$ git checkout br1
EXPECTED OUTPUT:
Switched to branch 'br1'

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git add file1.txt

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git commit -m "br1 1 version"
EXPECTED OUTPUT:
[br1 65cf90c] br1 1 version
 1 file changed, 1 insertion(+), 1 deletion(-)

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git log -oneline
EXPECTED OUTPUT:
65cf90c (HEAD -> br1) br1 1 version
85310a0 (master) master 1 version

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git add file1.txt

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
EXPECTED OUTPUT:
$ git commit -m "br1 2 version"
[br1 4cb7a65] br1 2 version
 1 file changed, 1 insertion(+), 1 deletion(-)

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git log -oneline
EXPECTED OUTPUT:
4cb7a65 (HEAD -> br1) br1 2 version
65cf90c br1 1 version
85310a0 (master) master 1 version

Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12 (br1)
$ git checkout master
EXPECTED OUTPUT:
Switched to branch 'master'
```

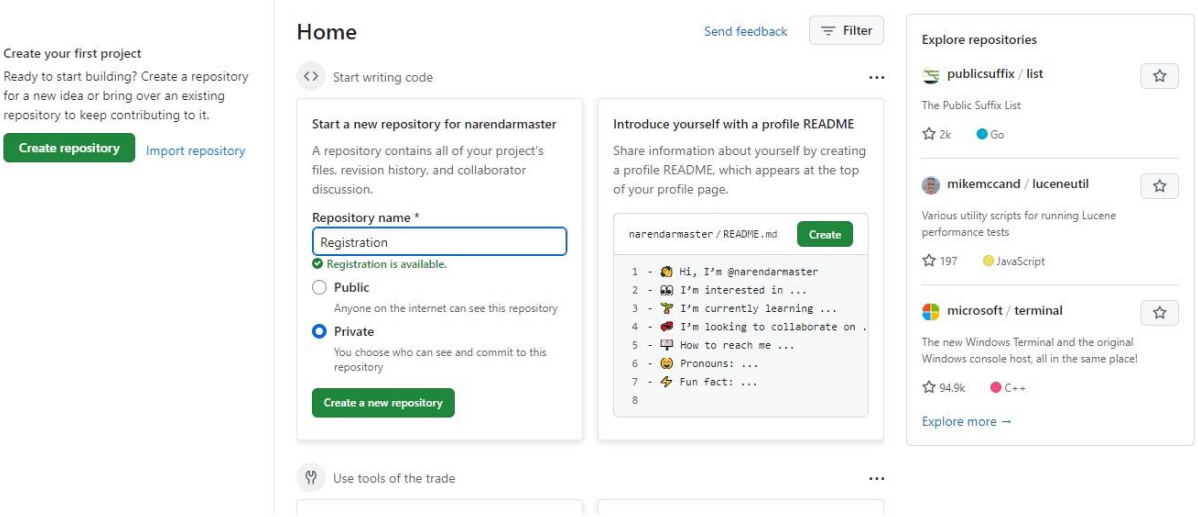
```
Narendarkumboji@DESKTOP-5HUH47P MINGW64 /f/ex12
(master)
$ git merge br1
EXPECTED OUTPUT:
Updating 85310a0..4cb7a65
Fast-forward
 file1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

3. Practice sorce code management on github.

Create a account on github (https://github.com/) with your emailid and create own password which you can remember.

Verify your emailid with otp sent by entering it.

Create a repo on git hub.



After that create git repository on eclipse id

Right click on Registration folder

Select team and share project and

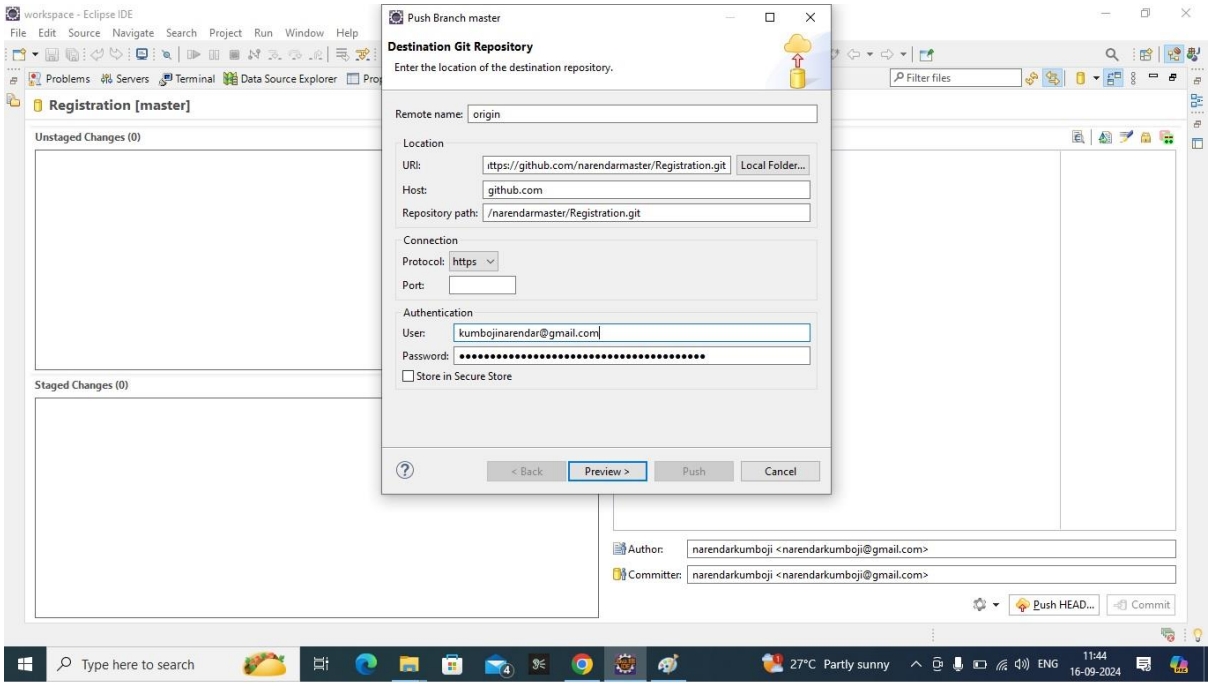
Create .git eclipse itself(recommended) and finish.

Pushing to git hub:

Right click on Registration folder

Select team and commit and

Paste path and access token from github and finish.

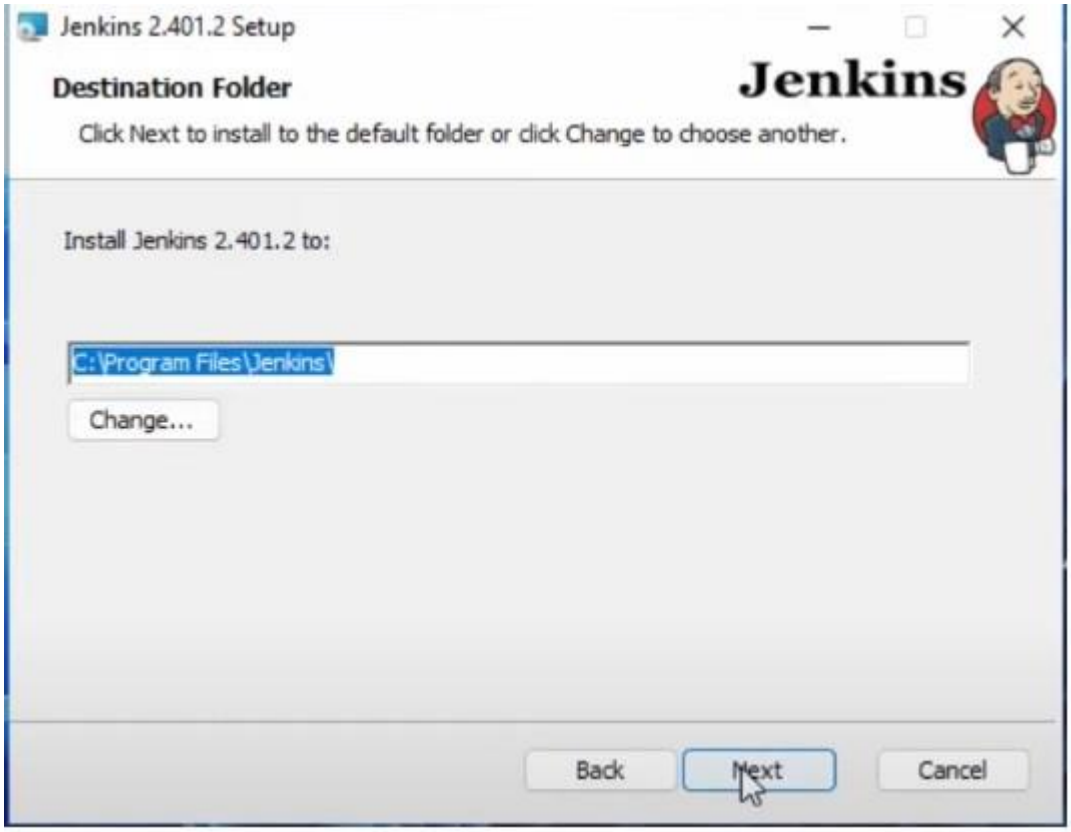


Now you can see project on github.

4. JENKINS INSTALLATION AND SETUP,EXPLORE THE ENVIRONMENT.

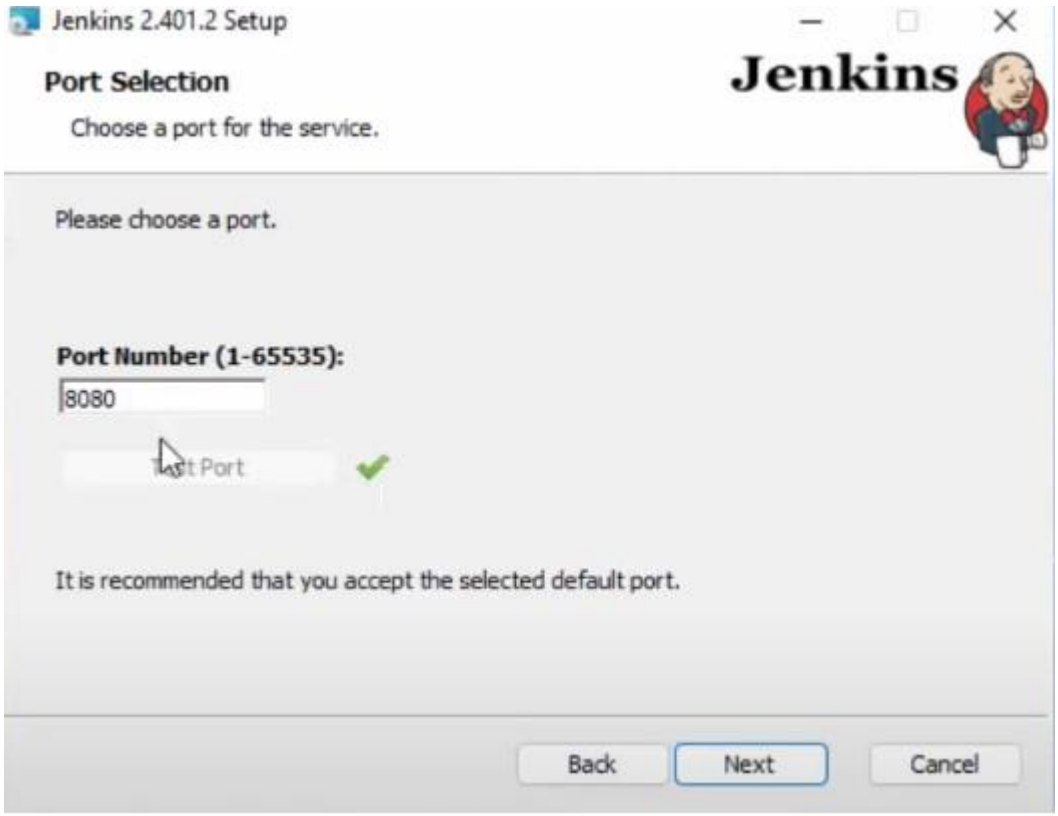
RUN JENKINS.EXE FILE

PRESS NEXT AND SELECT JENKINS DEFAULT PATH TO INSTALL AS SHOWN



PRESS NEXT AND SELECT RUN SERVICE AS LOCAL SYSTEM.

PRESS NEXT AND TEST PORT 8080 IS AVAILABLE SELECT NEXT.IF NOT VAILABLE CHANGE PORT NO AND PRESS NEXT.



PRESS NEXT AND SELECT JAVA PATH IN THE SYSTEM

PRESS NEXT AND INSTALL.

AFTER INSTALLATION LOG IN WITH INTIAL ADMIN PASSWORD FROM C:\ProgramData\Jenkins\.jenkins\secrets

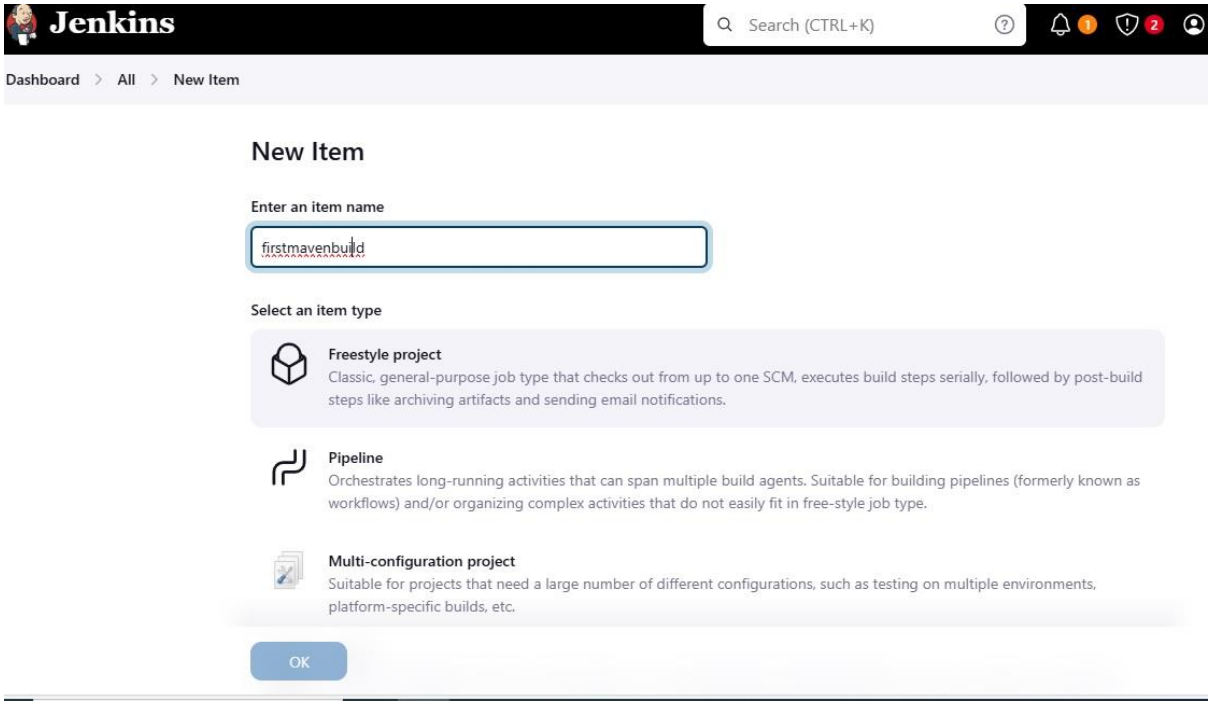
SELECT RECOMMNENED PLUGINS TO INSTALL AFTER

CHANGE USERNAME AND PASSWORD TO ADMIN.

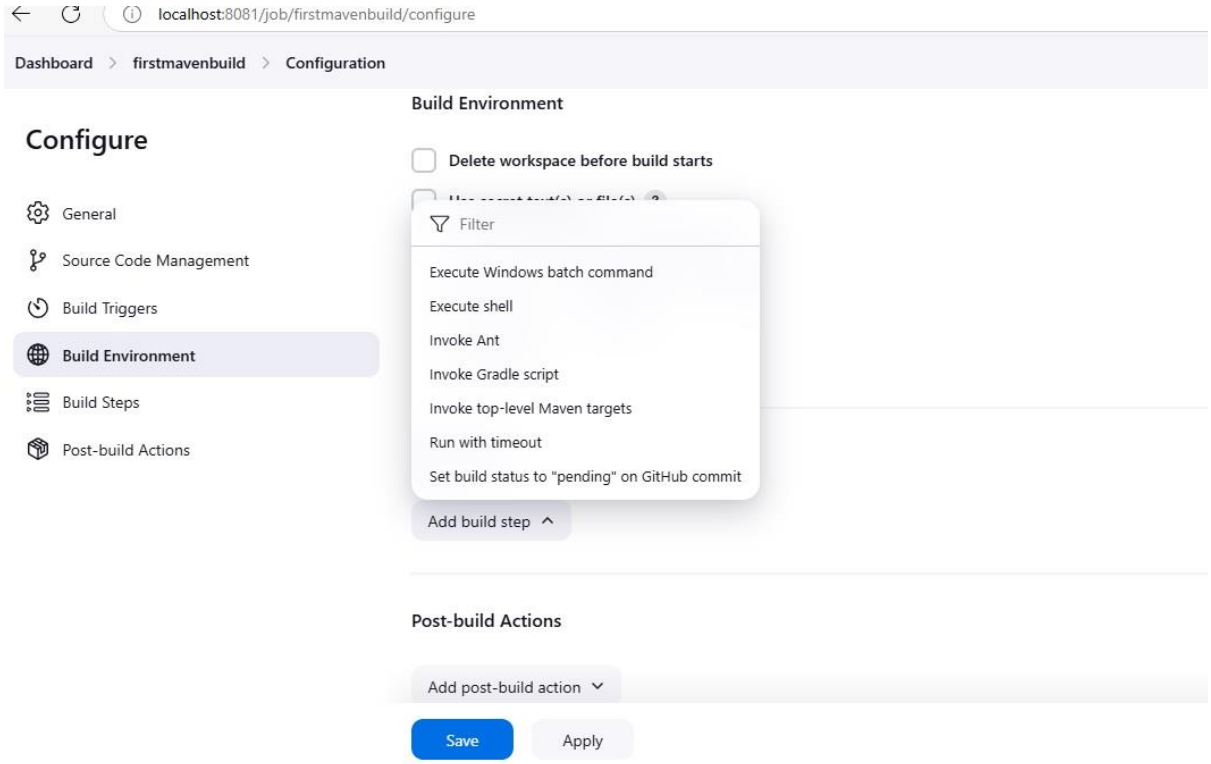
5. DEMONSTRATE CONTINOUS INTEGRATION AND DEVELOPMENT USING JENKINS.

OPEN http://localhost:8080/ on browser login with admin as username and password

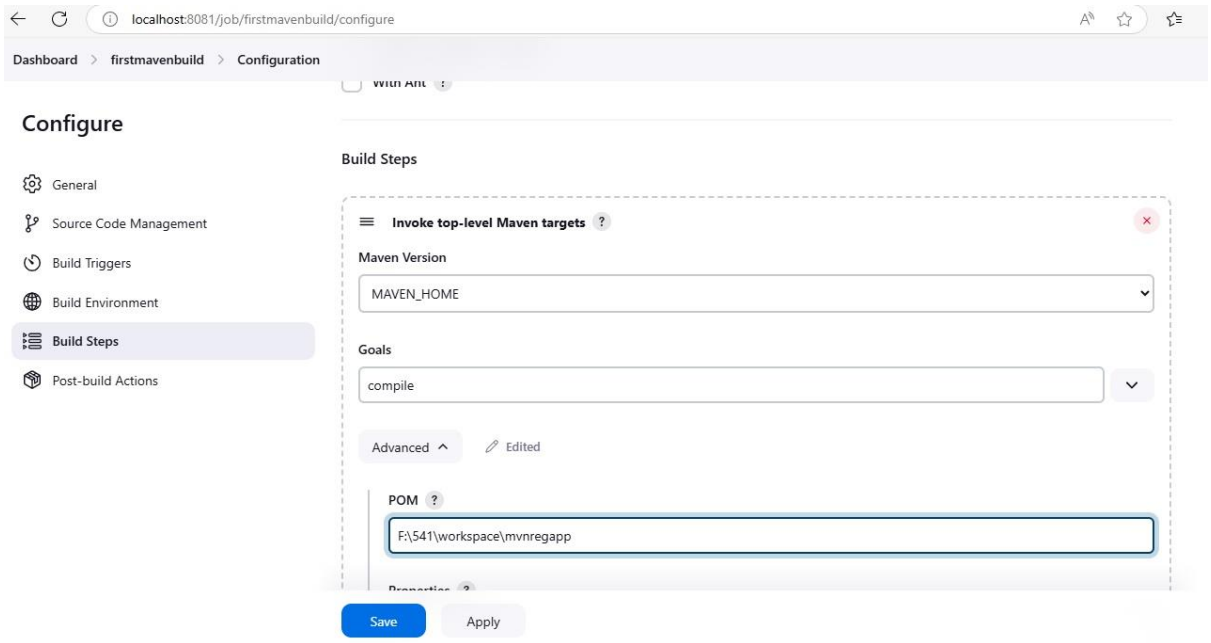
Select new item enter item as shown and select as free style project.



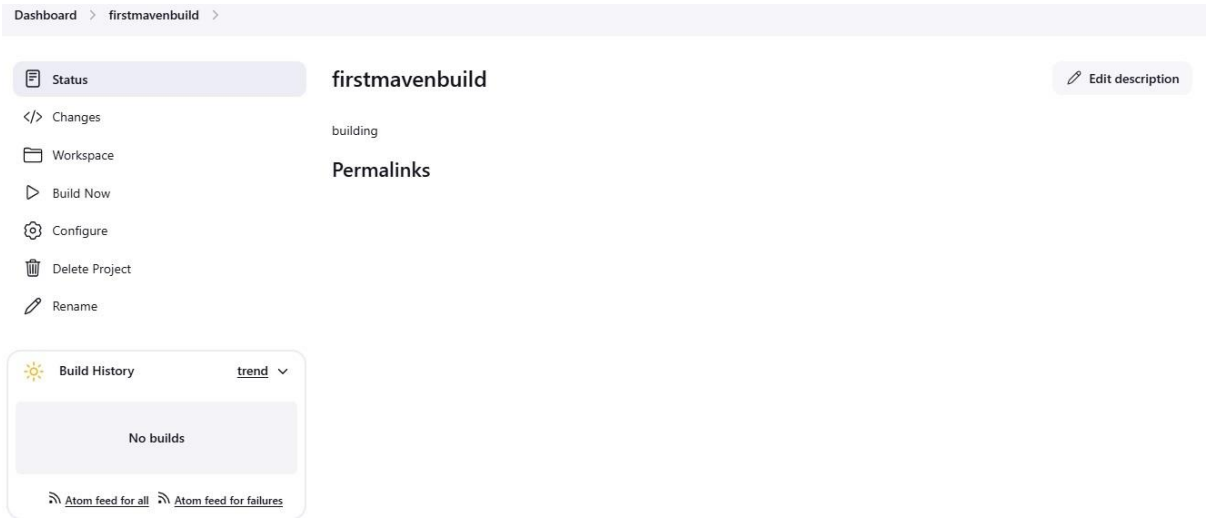
Press ok and enter description as building select build step invoke top level maven targets.



Select maven environment and goal as compile and give pom path as F:\541\workspace\mvnregapp



Click apply and save.



Press build now