Deliverable-3

Team Innovators

a. Requirements
List a set of requirements designated for **phase 1**.

- If the scope for this phase is unchanged, use the same requirements of phase 1 specified in Deliverable 2, but **more detailed descriptions** about the requirements should be provided.
- If not, explain the reasons why the plan is changed and describe the updated plan including the modified set of requirements for all phases including the **detailed descriptions** of the requirements for phase 1.

Updated Requirements for Phase 1

1) **Login/Logout for HR and Employees:**
Employee's data is stored in company's database, which includes their username as well as their encrypted password for data security, the password from database is decrypted and pattern matched while logging in and based on it the user is rendered their dashboard, more over we will be having two user screens one would be for HR and the other would be for Employee, each has different Api's exposed to them, this differentiation is done by checking the Employee's role, logout feature has also been added, once the user clicks on the logout button they will be redirected to the user login page.

2) **Add Job:**
Adding job is a HR functionality, this allows HR to add a job listing to the job board, that an employee can access for referring candidates. HR can fill out details of the job and hit the add button.

3) **Add User:**
We have created an API to add user to the system, currently as our project is under development we only have developer's access to this API, we are debating about giving this access to HR dashboard or not, as the user database can be directly fetched from the existing company database when companies integrate this portal with their already built environment.

4) **Update Job:**
Update Job is also a HR functionality, where a particular job is fetched and it's details can be updated and stored in the database.

5) **Delete Job:**
This is also an HR functionality, this will allow HRs to delete a job listing, this could be for any reason, maybe the requirement has been fulfilled, or if there was any human error.

6) **Get All Referred Candidates:**
HR will be able to view all the candidates that were referred to jobs by employees.

7) **Get All Jobs:**
This feature is available for both HR as well as the Employee, it is used to view all the jobs that have been added by HR.

8) **Refer a Candidate:**

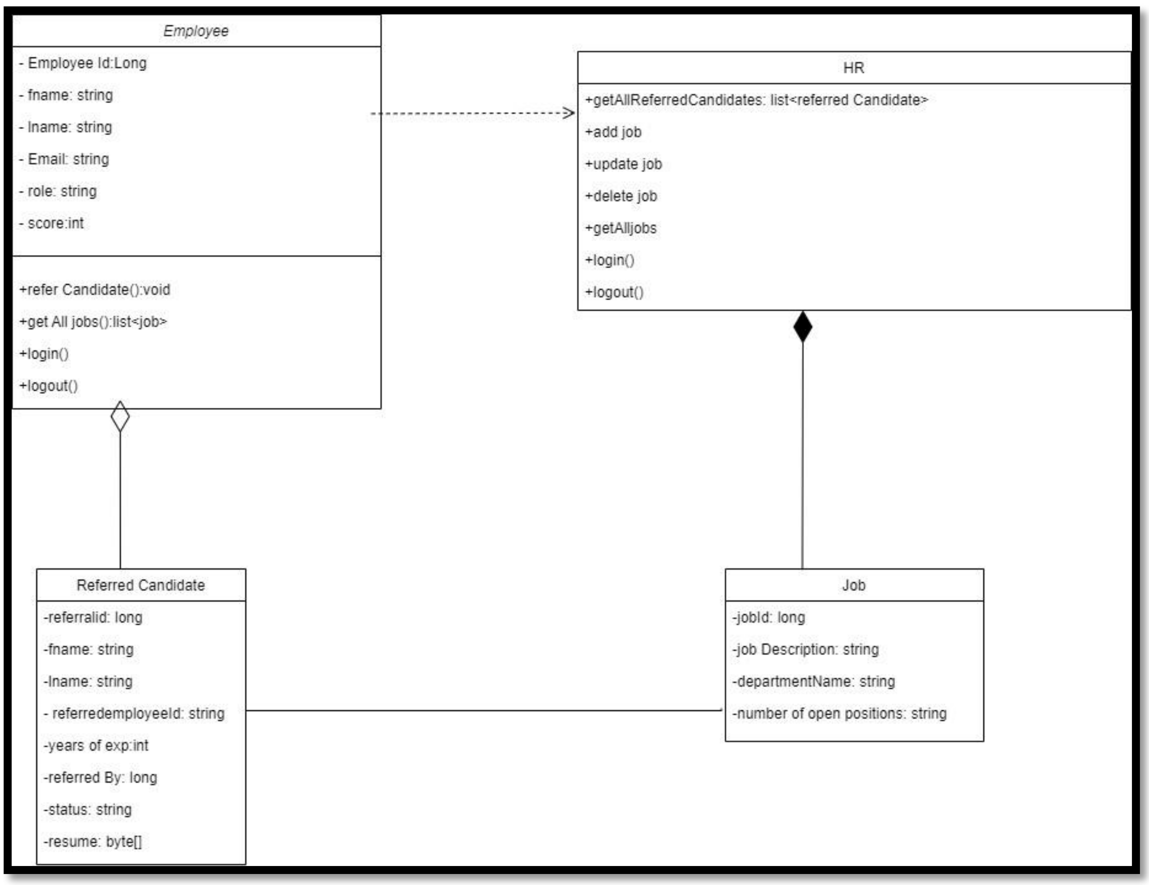
This is a feature which is available for employees which will allow users to refer their friends or known people to an open job listing.

9) **Database Setup:**
The application is connected to an Oracle Autonomous DB, the application has read/write access to the Database, and entities such as Employee, Job, Referred Candidate have been created and tested which are essential for functioning of the Application.

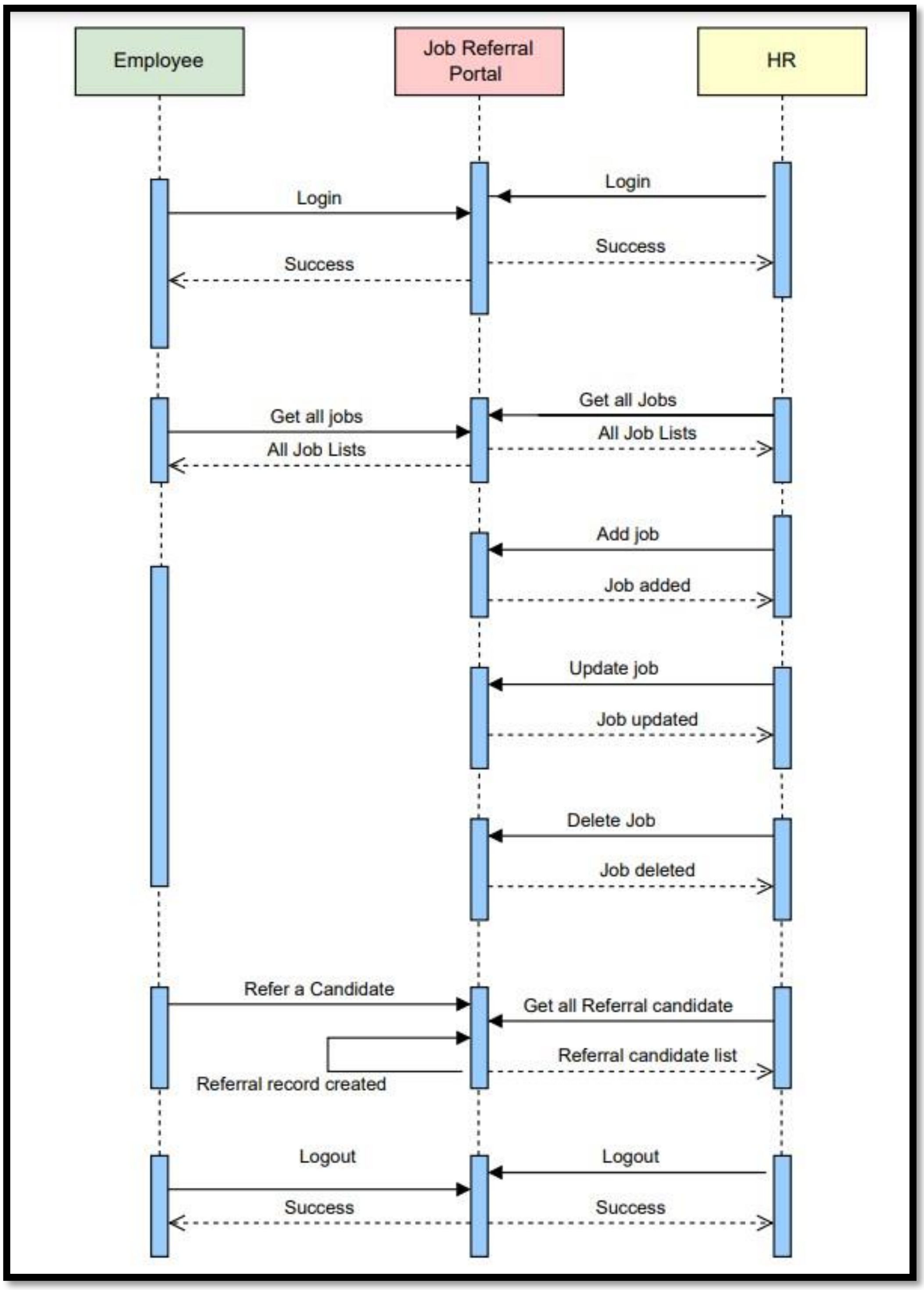
Most of the requirements are in line with the Phase 1 requirements that were submitted as part of the deliverable two, but some features that our team has been discussing and some feedback that was received during the peer review session, we wanted to add them to the phase 2, features such as search function and leaderboard will be covered in the later phases, initially we believed bulk upload of jobs would be a difficult task hence we estimated large effort and put it in phase 2, whereas this wasn't the case and we were able to find a way to do this easily, which resulted in us deciding just focusing on login/logout features and the core functionalities first, hence the main goal of the phase 1 is to get the system up and running we had the essential features for this prototype ready, even though the business logic for search function and leaderboard, resume downloading is ready we don't plan on making it part of this release because we are looking into getting the system running and create a proof of concept also with the feedback we decided to create search based on key words such as location, hence we wanted to be sure and the features to be working properly before we release it, so pushing it to phase2.

b. UML design for **phase 1**. You must include the following diagrams:
- Class diagram
- Sequence diagram
- Use case diagram – at least one normal case and one error case should be included.

Class Diagram:

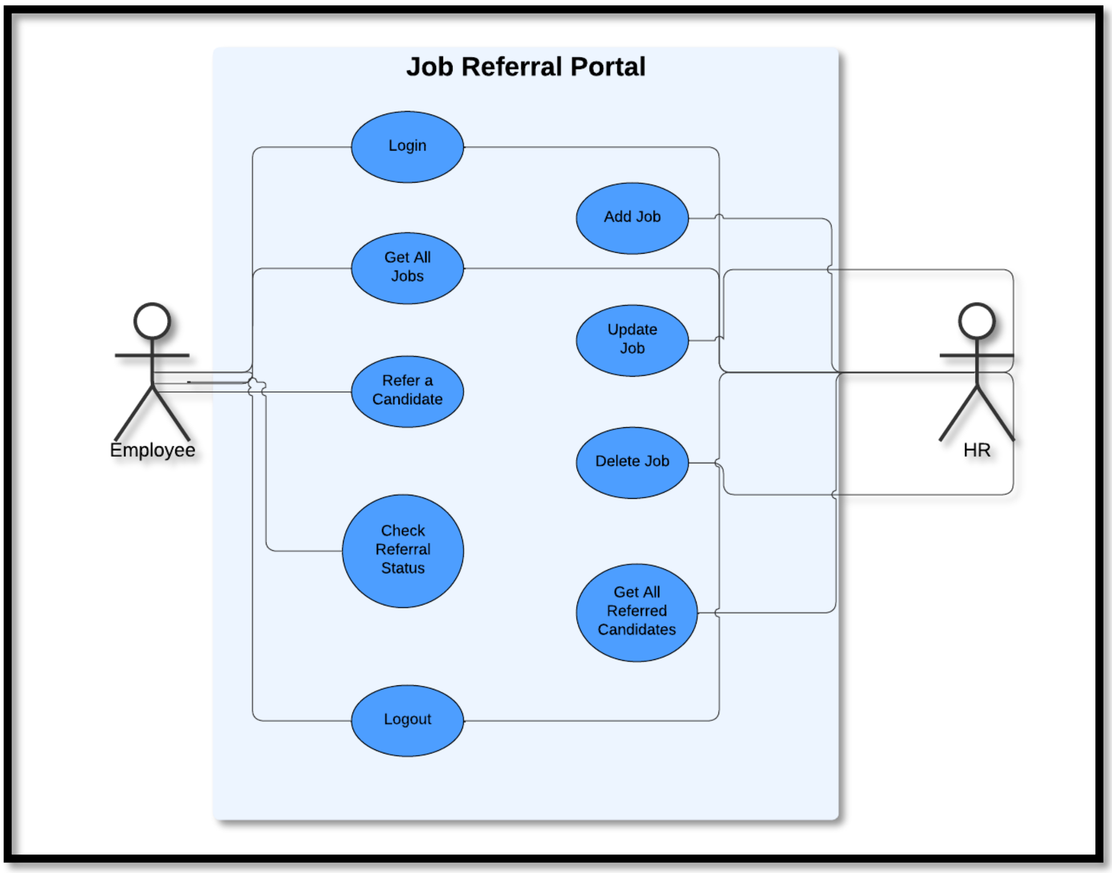


Sequence Diagram:

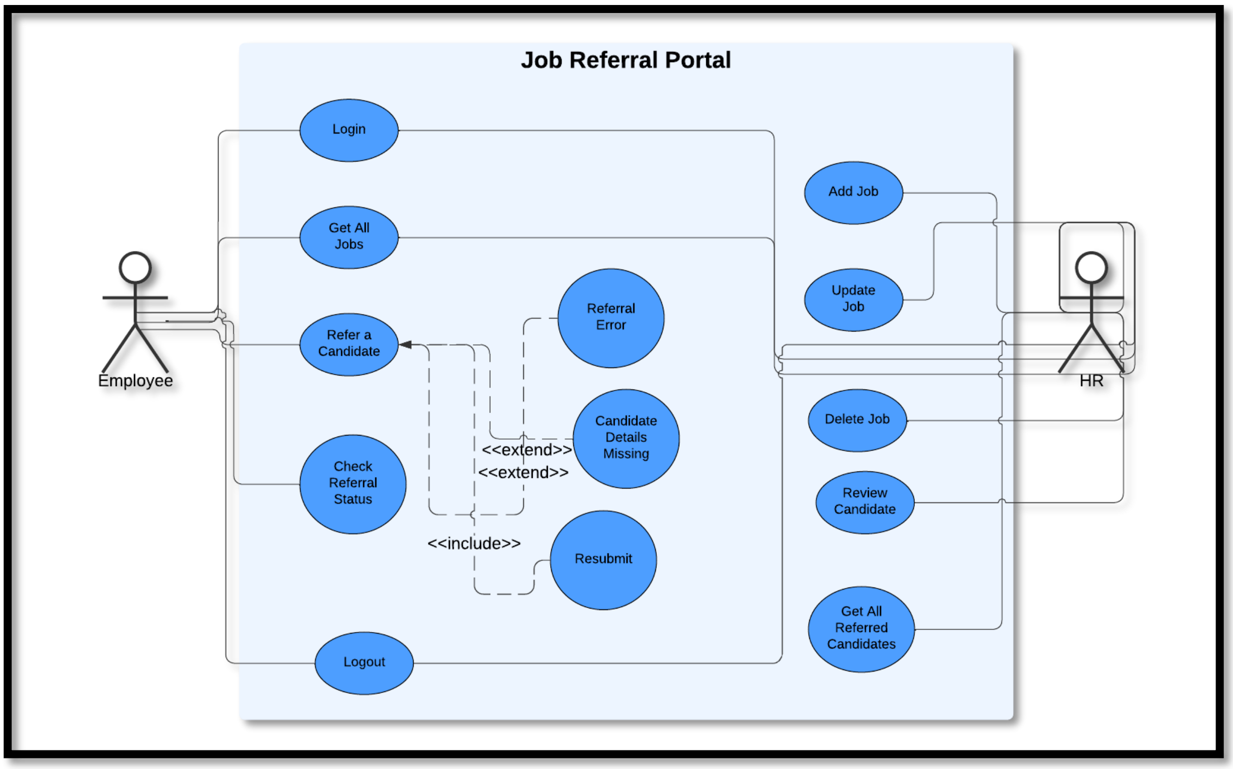


Use Case Diagram:

1. **Normal Case:**



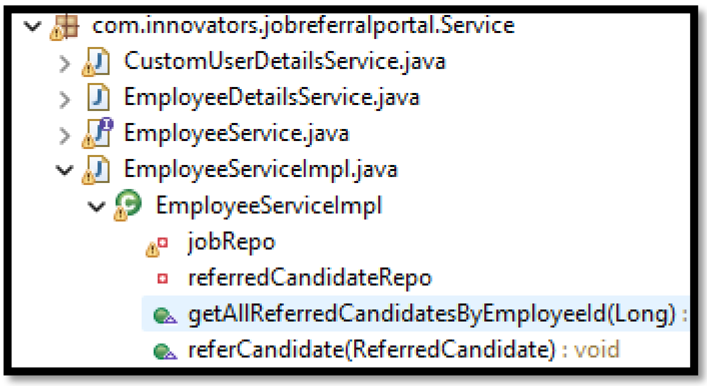
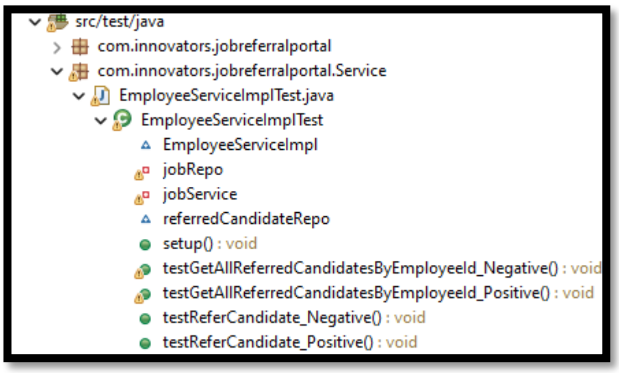
2. **Error Case:**



c. Test Cases (unit tests) for **phase 1**.
List a set of test cases used for testing the working program including descriptions of tests (e.g., what functionality they test, and inputs/outputs for them).

We are performing the method level testing in our project using the JUnit's.

Unit testing for the EmployeeServiceImpl Class Methods:

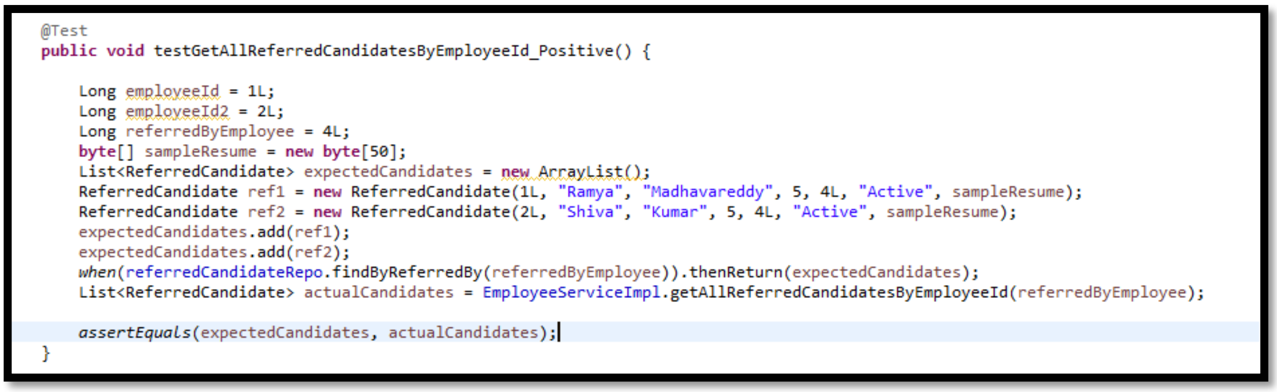
	

```
com.innovators.jobreferralportal.Service
  CustomUserDetailsService.java
  EmployeeDetailsService.java
  EmployeeService.java
  EmployeeServiceImpl.java
    EmployeeServiceImpl
      jobRepo
      referredCandidateRepo
      getAllReferredCandidatesByEmployeeId(Long) :
      referCandidate(ReferredCandidate) : void
```

```
src/test/java
  com.innovators.jobreferralportal
  com.innovators.jobreferralportal.Service
    EmployeeServiceImplTest.java
      EmployeeServiceImplTest
        EmployeeServiceImpl
        jobRepo
        jobService
        referredCandidateRepo
        setup() : void
        testGetAllReferredCandidatesByEmployeeId_Negative() : void
        testGetAllReferredCandidatesByEmployeeId_Positive() : void
        testReferCandidate_Negative() : void
        testReferCandidate_Positive() : void
```

In the EmployeeServiceImpl class we have 2 methods, one for getting all the candidates referred to by the employee and another is to refer to a candidate for the job. For every method we have a positive and negative test case to evaluate a happy path and negative scenario too.

testGetAllReferredCandidatesByEmployeeId_Positive ():

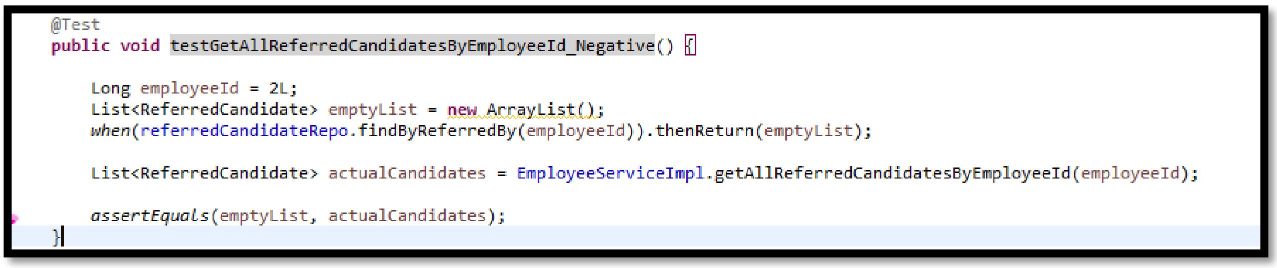


```
@Test
public void testGetAllReferredCandidatesByEmployeeId_Positive() {

    Long employeeId = 1L;
    Long employeeId2 = 2L;
    Long referredByEmployee = 4L;
    byte[] sampleResume = new byte[50];
    List<ReferredCandidate> expectedCandidates = new ArrayList();
    ReferredCandidate ref1 = new ReferredCandidate(1L, "Ramya", "Madhavareddy", 5, 4L, "Active", sampleResume);
    ReferredCandidate ref2 = new ReferredCandidate(2L, "Shiva", "Kumar", 5, 4L, "Active", sampleResume);
    expectedCandidates.add(ref1);
    expectedCandidates.add(ref2);
    when(referredCandidateRepo.findByReferredBy(referredByEmployee)).thenReturn(expectedCandidates);
    List<ReferredCandidate> actualCandidates = EmployeeServiceImpl.getAllReferredCandidatesByEmployeeId(referredByEmployee);

    assertEquals(expectedCandidates, actualCandidates);
}
```

This test case is designed to validate the functionality of the getAllReferredCandidatesByEmployeeId method when the employee has referred candidates. So we are creating an expected candidate list and using the when().then() in junits we are asking it to return the expected list whenever the method is called .Next, we are making a call to the impl class method for fetching the list of employees referred by employeeId 4L.

To Compare if the method is returning the details correctly we are using the assertEquals() methods which compare if both the parameters are equal. If equal, then it indicates that our method is working well.

testGetAllReferredCandidatesByEmployeeId_Negative():

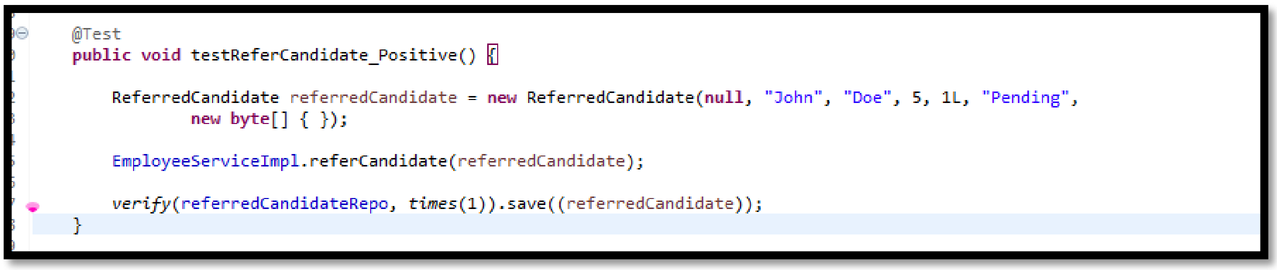


```
@Test
public void testGetAllReferredCandidatesByEmployeeId_Negative() {

    Long employeeId = 2L;
    List<ReferredCandidate> emptyList = new ArrayList();
    when(referredCandidateRepo.findByReferredBy(employeeId)).thenReturn(emptyList);

    List<ReferredCandidate> actualCandidates = EmployeeServiceImpl.getAllReferredCandidatesByEmployeeId(employeeId);

    assertEquals(emptyList, actualCandidates);
}
```

This test case checks the behavior when there are **no referred candidates** for a given employee. In general, we should expect an empty list. So, to verify this we have created a empty list and using the when().then() we are asking it to return the empty list when called.

To verify it's returning the empty list we are using the assert Equals() method .

testReferCandidate_Positive():

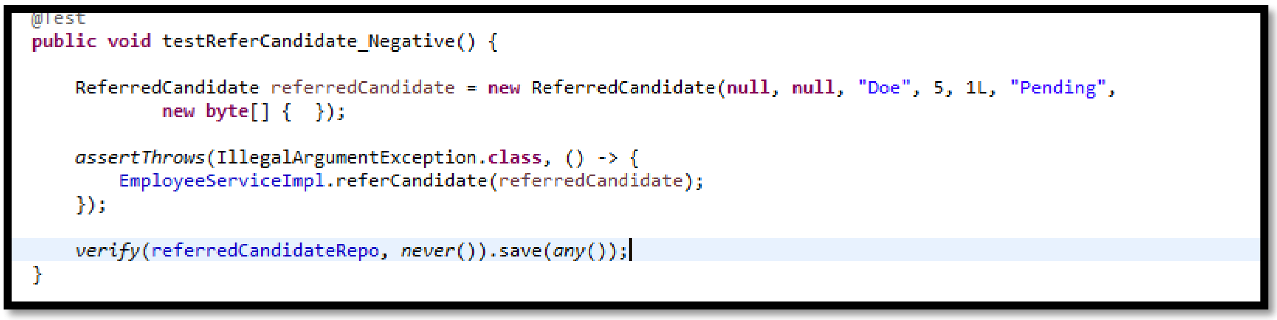


```
@Test
public void testReferCandidate_Positive() {

    ReferredCandidate referredCandidate = new ReferredCandidate(null, "John", "Doe", 5, 1L, "Pending",
            new byte[] { });

    EmployeeServiceImpl.referCandidate(referredCandidate);

    verify(referredCandidateRepo, times(1)).save((referredCandidate));
}
```

In this refercandidate method we will be giving a valid candidate object with the all the parameters and verify if the database save is called at least once when the referCandidate()method is called.

For verifying we are using the verify() method from junit plugin.

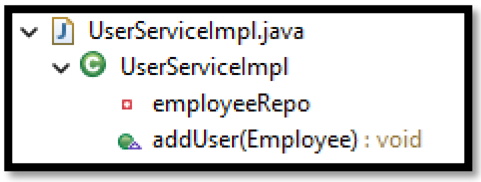
testGetAllReferredCandidatesByEmployeeId_Negative():

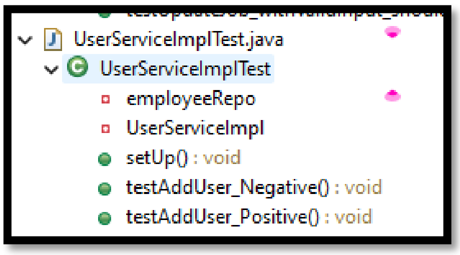


```
@Test
public void testReferCandidate_Negative() {

    ReferredCandidate referredCandidate = new ReferredCandidate(null, null, "Doe", 5, 1L, "Pending",
            new byte[] {  });

    assertThrows(IllegalArgumentException.class, () -> {
        EmployeeServiceImpl.referCandidate(referredCandidate);
    });

    verify(referredCandidateRepo, never()).save(any());
}
```

In the negative scenario testing for this method we are verifying that whenever the object has any null values in it then it should not be saved into the database and an exception should be raised by the method. To verify this we are checking if the method is throwing any exception using the assertThrows() method and also using the verify() method we are making sure not database call is made to save the object.

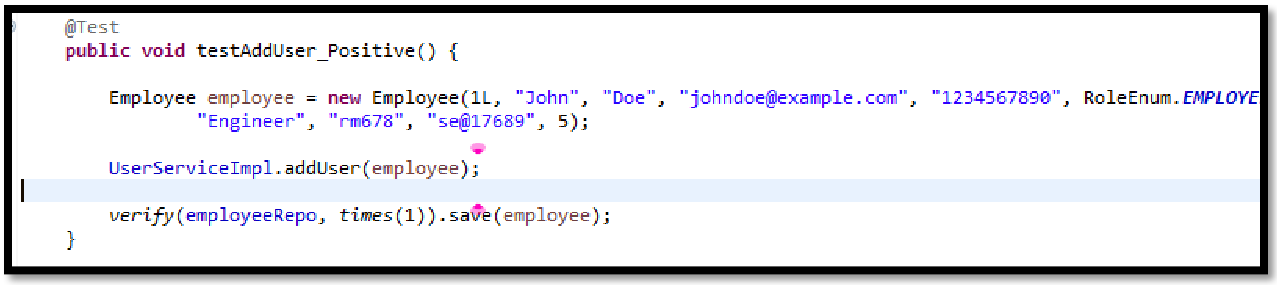
Unit testing for the UserServiceImpl Class Methods:


UserServiceImpl.java
UserServiceImpl
employeeRepo
addUser(Employee) : void


UserServiceImplTest.java
UserServiceImplTest
employeeRepo
UserServiceImpl
setUp() : void
testAddUser_Negative() : void
testAddUser_Positive() : void

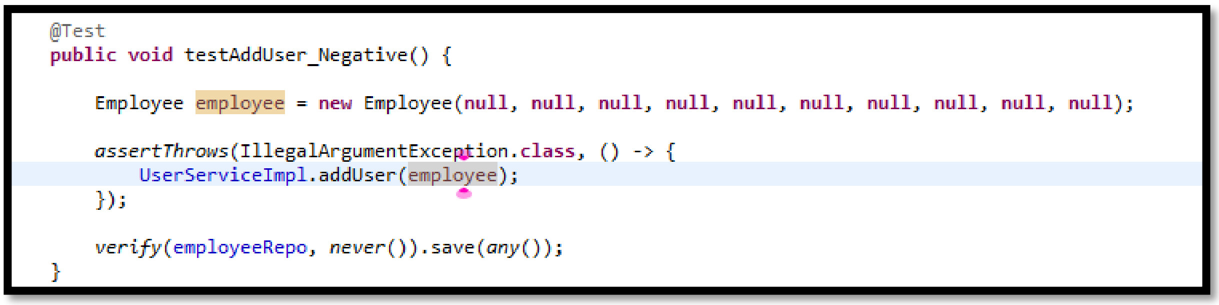
The UserServiceImpl has only 1 method which actually saves the employee object into the database which is technically adding the user.So we have created 2 test cases one for positive and one for negative.

testAddUser_Positive():


```
@Test
public void testAddUser_Positive() {

    Employee employee = new Employee(1L, "John", "Doe", "johndoe@example.com", "1234567890", RoleEnum.EMPLOYE
            "Engineer", "rm678", "se@17689", 5);

    UserServiceImpl.addUser(employee);

    verify(employeeRepo, times(1)).save(employee);
}
```

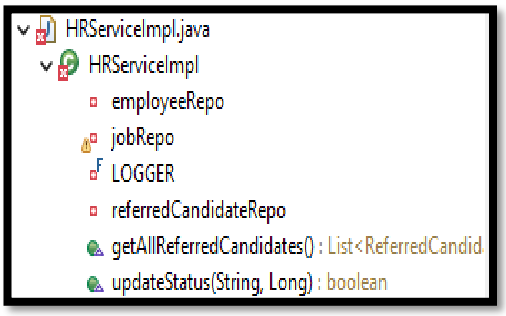
In the positive testcase of this method we are validating if the database save method is called atleast once during the execution.We have created an employee object and gave it to the addUser() method as the input and then we are verifying using the verify method if the save method is called.

testAddUser_Negative():

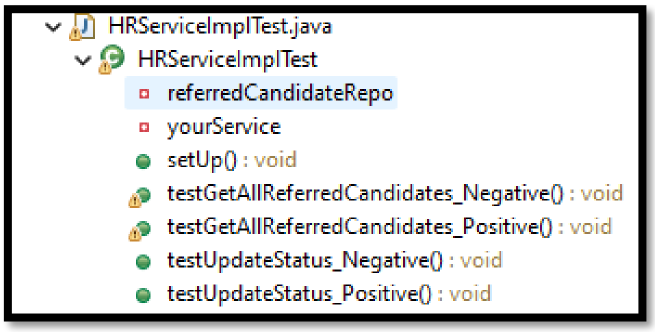

```
@Test
public void testAddUser_Negative() {

    Employee employee = new Employee(null, null, null, null, null, null, null, null, null, null);

    assertThrows(IllegalArgumentException.class, () -> {
        UserServiceImpl.addUser(employee);
    });

    verify(employeeRepo, never()).save(any());
}
```

This test case is created to verify how the method is reacting for an employee object with some null values in it. In general, whenever there are some null values it should be saved into the database instead an exception should be raised. Here we are creating an employee with some null values and giving it as input to the addUser method and then we are checking if an exception is thrown and also verifying that no database save method call is made using the verify method.

JUnit's for the HRServiceImpl Class Methods:



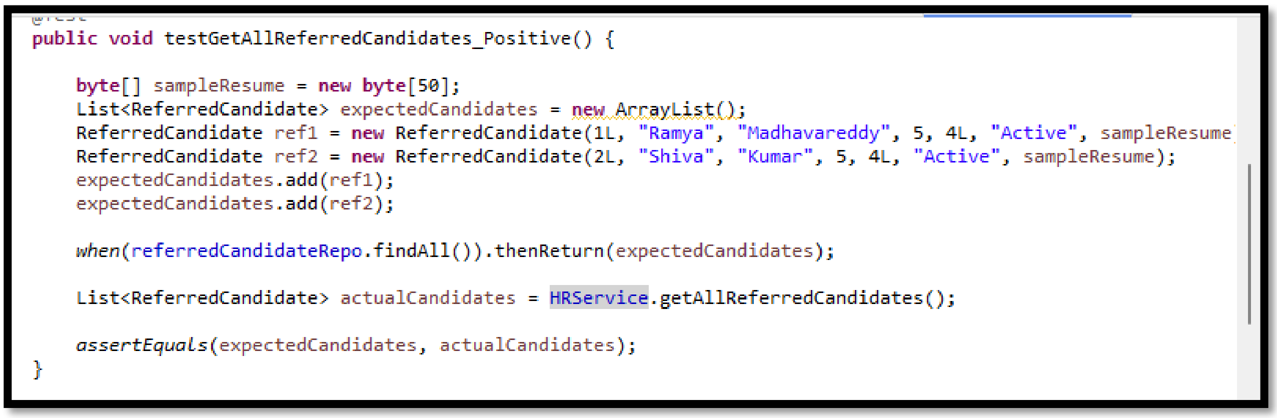
```
HRServiceImpl.java
  HRServiceImpl
    employeeRepo
    jobRepo
    LOGGER
    referredCandidateRepo
    getAllReferredCandidates() : List<ReferredCandid
    updateStatus(String, Long) : boolean
```



```
HRServiceImplTest.java
  HRServiceImplTest
    referredCandidateRepo
    yourService
    setUp() : void
    testGetAllReferredCandidates_Negative() : void
    testGetAllReferredCandidates_Positive() : void
    testUpdateStatus_Negative() : void
    testUpdateStatus_Positive() : void
```

In this HRServiceImpl class we have 2 methods one method is for pulling out all the persons who got referred and a method to update the status of the referred persons.

testGetAllReferredCandidates_Positive():

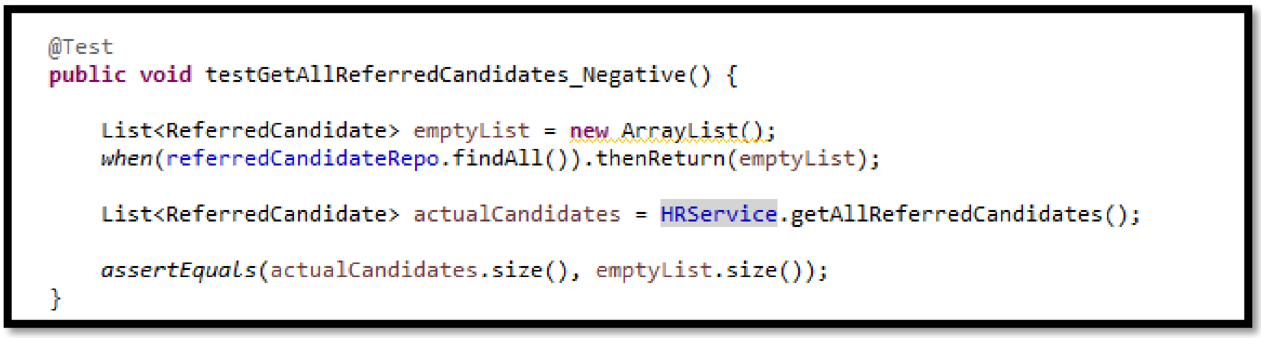


```
public void testGetAllReferredCandidates_Positive() {

    byte[] sampleResume = new byte[50];
    List<ReferredCandidate> expectedCandidates = new ArrayList();
    ReferredCandidate ref1 = new ReferredCandidate(1L, "Ramya", "Madhavareddy", 5, 4L, "Active", sampleResume
    ReferredCandidate ref2 = new ReferredCandidate(2L, "Shiva", "Kumar", 5, 4L, "Active", sampleResume);
    expectedCandidates.add(ref1);
    expectedCandidates.add(ref2);

    when(referredCandidateRepo.findAll()).thenReturn(expectedCandidates);

    List<ReferredCandidate> actualCandidates = HRService.getAllReferredCandidates();

    assertEquals(expectedCandidates, actualCandidates);
}
```

In this test case we are validating if we are successfully able to pull out all the referred persons records from the database base. We have created a list of employees and assigned it to the expected list and using the Mockito’s when() function we are asking that whenever the method is called then return this expected output.

And we are verifying if it is returning the actual values which is nothing but the same expected values. Validation of the actual and expected output is done using the assertEquals() method which verifies if the both the given parameters are same.

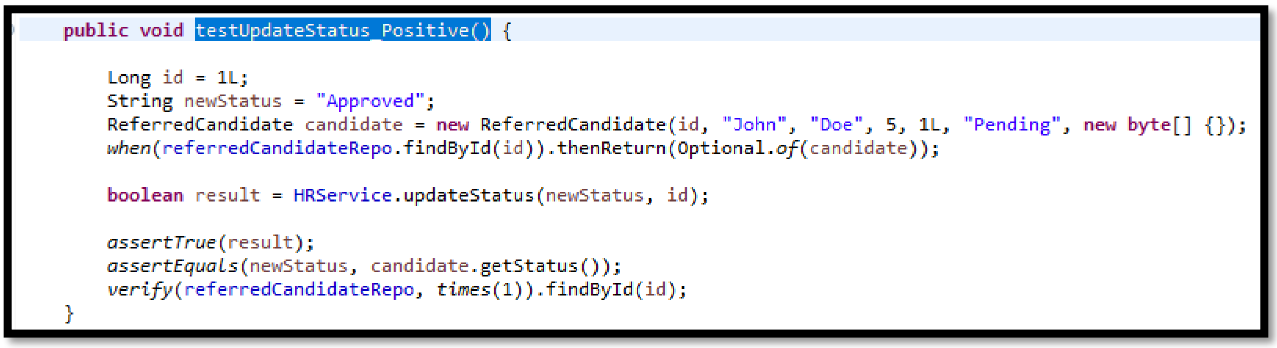
testGetAllReferredCandidates_Negative():



```
@Test
public void testGetAllReferredCandidates_Negative() {

    List<ReferredCandidate> emptyList = new ArrayList();
    when(referredCandidateRepo.findAll()).thenReturn(emptyList);

    List<ReferredCandidate> actualCandidates = HRService.getAllReferredCandidates();

    assertEquals(actualCandidates.size(), emptyList.size());
}
```

In the negative testcase for fetching the records we are expecting an empty list from the method so hence we are validating if we are getting an empty list as the output whenever we call the method and check that everything goes well with no exceptions. We are using the assertEquals() method to make the comparison.

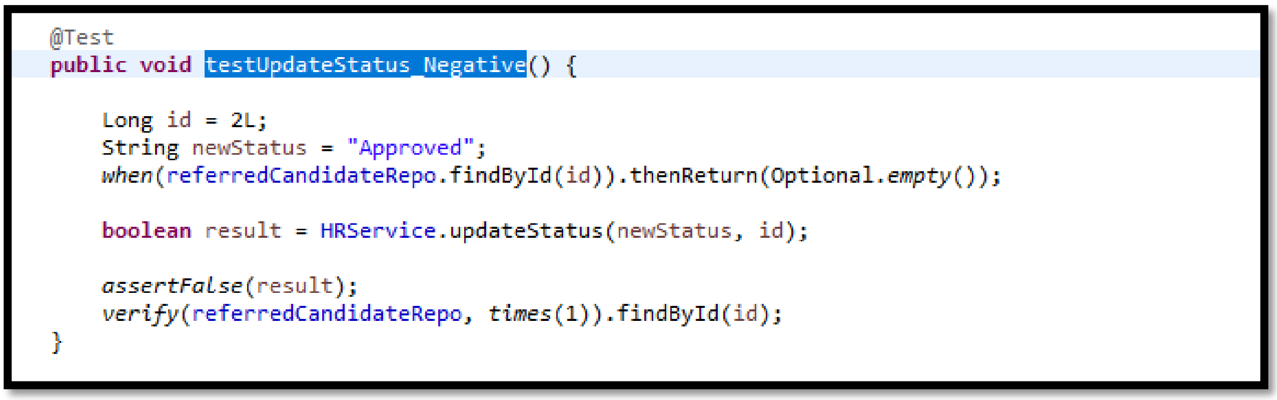
testUpdateStatus_Positive():



```java
    public void testUpdateStatus_Positive() {

        Long id = 1L;
        String newStatus = "Approved";
        ReferredCandidate candidate = new ReferredCandidate(id, "John", "Doe", 5, 1L, "Pending", new byte[] {});
        when(referredCandidateRepo.findById(id)).thenReturn(Optional.of(candidate));

        boolean result = HRService.updateStatus(newStatus, id);

        assertTrue(result);
        assertEquals(newStatus, candidate.getStatus());
        verify(referredCandidateRepo, times(1)).findById(id);
    }
```

In this method we are actually verifying if the status updation is done correctly in this method. If the updation is done, then the status would change and also the database is called atleast once to save the status or fetch the user from the db. We are creating a candidate object and providing the status of that employee and the id of the employee as input to the updateStatus method which is expected to give true as output and so we are verifying if the status is approved using the assertEquals() method.

testUpdateStatus_Negative():

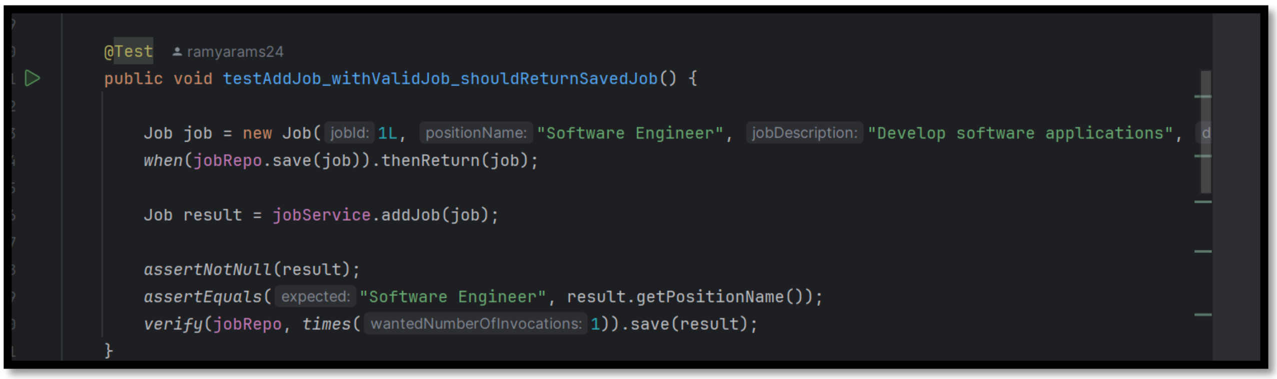


```java
    @Test
    public void testUpdateStatus_Negative() {

        Long id = 2L;
        String newStatus = "Approved";
        when(referredCandidateRepo.findById(id)).thenReturn(Optional.empty());

        boolean result = HRService.updateStatus(newStatus, id);

        assertFalse(result);
        verify(referredCandidateRepo, times(1)).findById(id);
    }
```

In this method we are actually trying to test the negative scenario where there is no user for the specified id to update the status.So we mocked the findById method to return empty as ouput so we get the result as False and we are validating if it is False using the assertFalse() and additionally we are also verifying that the database is called in the negative scenario too to fetch the employeedetails.

Unit testing for the JobServiceImpl Class Methods:

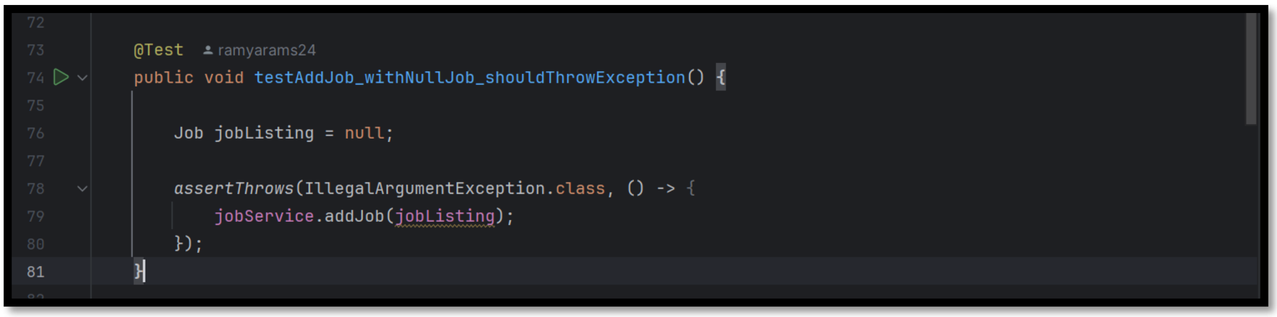
testAddJob_withValidJob_shouldReturnSavedJob():



```java
    @Test  ramyarams24
    public void testAddJob_withValidJob_shouldReturnSavedJob() {

        Job job = new Job( jobId: 1L,  positionName: "Software Engineer",  jobDescription: "Develop software applications",
        when(jobRepo.save(job)).thenReturn(job);

        Job result = jobService.addJob(job);

        assertNotNull(result);
        assertEquals( expected: "Software Engineer", result.getPositionName());
        verify(jobRepo, times( wantedNumberOfInvocations: 1)).save(result);
    }
```

In this method, we give a valid Job object to the system and then we test if our method returns the object that has been saved, first we assert that the returned job object is not null, then next we check if the values have been stored correctly by getting the position name and then comparing it with the string of the original job title that was passed to the method.

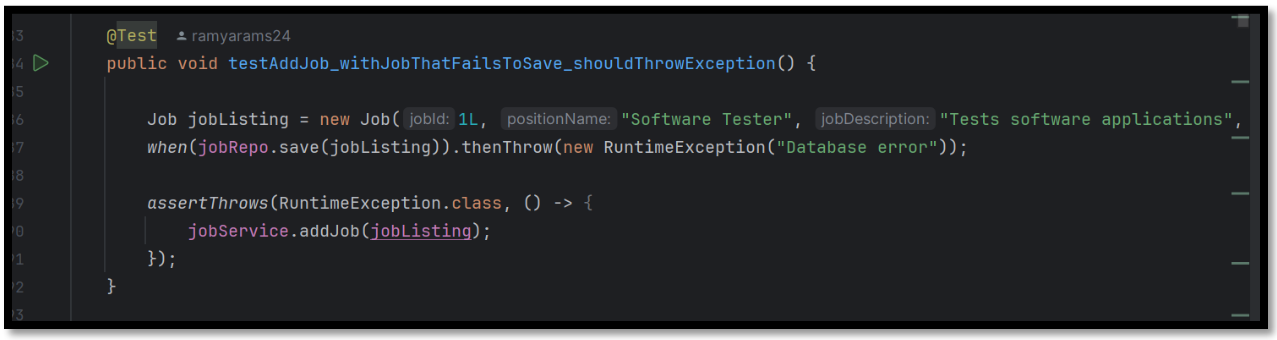
testAddJob_withNullJob_shouldThrowException():



```
@Test  ramyarams24
public void testAddJob_withNullJob_shouldThrowException() {

    Job jobListing = null;

    assertThrows(IllegalArgumentException.class, () -> {
        jobService.addJob(jobListing);
    });
}
```

In this test case, we try to pass a null value as a Job object and check if our method throws an exception.

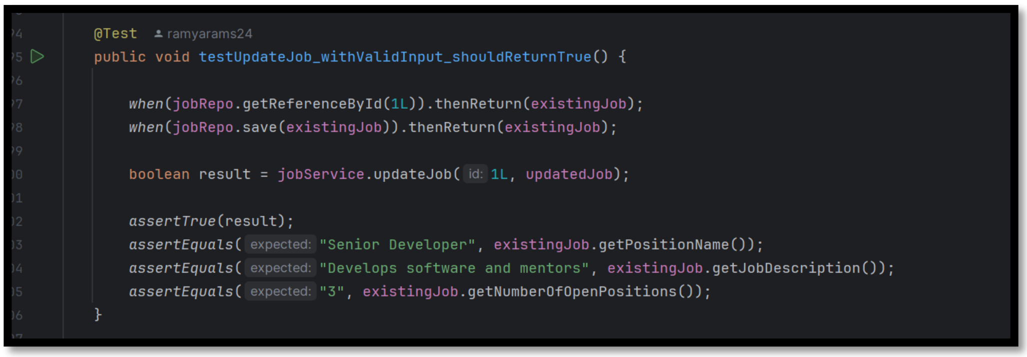
testAddJob_withJobThatFailsToSave_shouldThrowException():



```
@Test  ramyarams24
public void testAddJob_withJobThatFailsToSave_shouldThrowException() {

    Job jobListing = new Job( jobId: 1L,  positionName: "Software Tester",  jobDescription: "Tests software applications",
    when(jobRepo.save(jobListing)).thenThrow(new RuntimeException("Database error"));

    assertThrows(RuntimeException.class, () -> {
        jobService.addJob(jobListing);
    });
}
```

In this test case, we try to simulate a case where failure occurs when we try to save the job, this is to make sure that exceptions are not getting eaten or suppressed by the method.

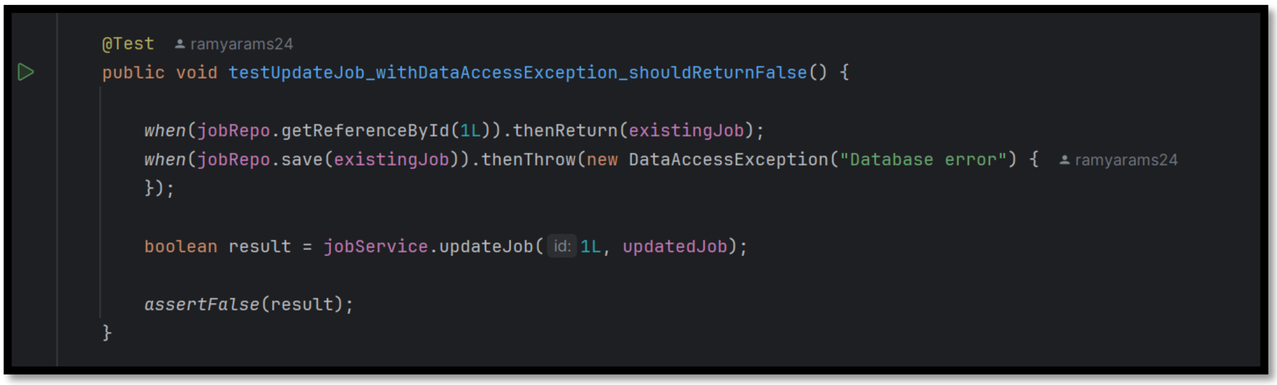
testUpdateJob_withValidInput_shouldReturnTrue():



```
@Test  ramyarams24
public void testUpdateJob_withValidInput_shouldReturnTrue() {

    when(jobRepo.getReferenceById(1L)).thenReturn(existingJob);
    when(jobRepo.save(existingJob)).thenReturn(existingJob);

    boolean result = jobService.updateJob( id: 1L, updatedJob);

    assertTrue(result);
    assertEquals( expected: "Senior Developer", existingJob.getPositionName());
    assertEquals( expected: "Develops software and mentors", existingJob.getJobDescription());
    assertEquals( expected: "3", existingJob.getNumberOfOpenPositions());
}
```

In this test case we update a job and then we check if the job is actually updated, we ensure that the job is updated by asserting that the name, Description and number of open positions of the existing job is matching with the updated job requirements.

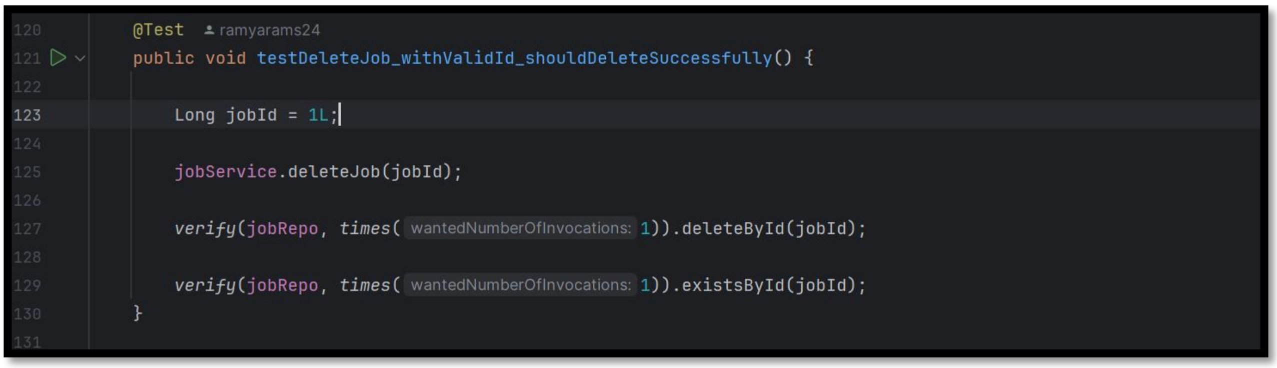
testUpdateJob_withDataAccessException_shouldReturnFalse():



```
@Test  ramyarams24
public void testUpdateJob_withDataAccessException_shouldReturnFalse() {

    when(jobRepo.getReferenceById(1L)).thenReturn(existingJob);
    when(jobRepo.save(existingJob)).thenThrow(new DataAccessException("Database error") {  ramyarams24
    });

    boolean result = jobService.updateJob( id: 1L, updatedJob);

    assertFalse(result);
}
```

In this test case, we check if the original method is handling the exception properly and it's not getting suppressed or eaten, whenever we pass update request, then if in case where DataAccessException might occur, the method shall throw the exception rather than suppressing it.

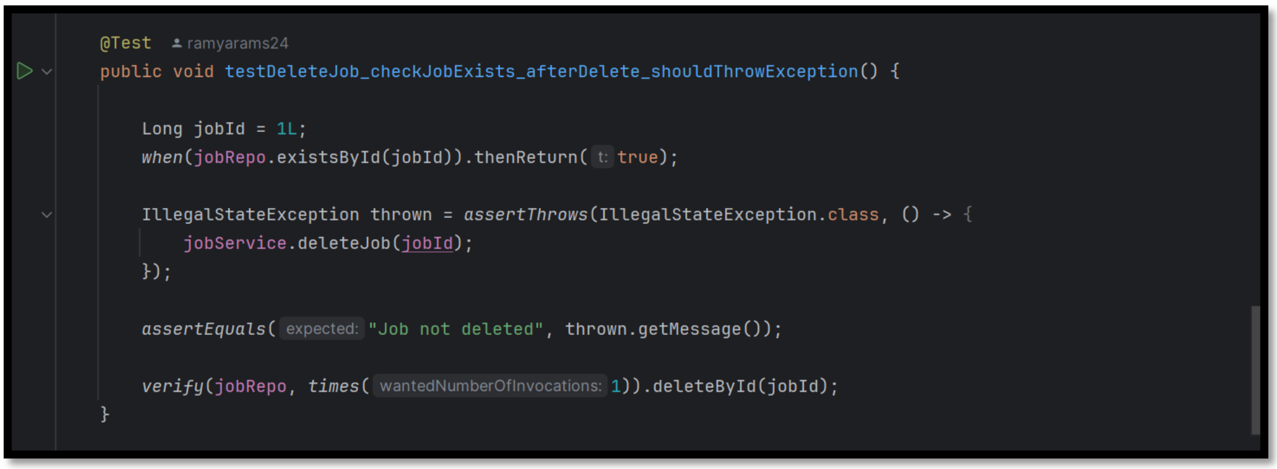
testDeleteJob_withValidId_shouldDeleteSuccessfully():



```
@Test  ramyarams24
public void testDeleteJob_withValidId_shouldDeleteSuccessfully() {

    Long jobId = 1L;

    jobService.deleteJob(jobId);

    verify(jobRepo, times( wantedNumberOfInvocations: 1)).deleteById(jobId);

    verify(jobRepo, times( wantedNumberOfInvocations: 1)).existsById(jobId);
}
```

in this test case we pass the id of the job that needs to be deleted and whenever we hit the delete API, we want to see if the method internally calls the two prebuilt methods, that are existsById and deleteById and these methods must be only invoked once, that is checked.

testDeleteJob_checkJobExists_afterDelete_shouldThrowException():



```java
@Test  ramyarams24
public void testDeleteJob_checkJobExists_afterDelete_shouldThrowException() {

    Long jobId = 1L;
    when(jobRepo.existsById(jobId)).thenReturn( t: true);

    IllegalStateException thrown = assertThrows(IllegalStateException.class, () -> {
        jobService.deleteJob(jobId);
    });

    assertEquals( expected: "Job not deleted", thrown.getMessage());

    verify(jobRepo, times( wantedNumberOfInvocations: 1)).deleteById(jobId);
}
```

In this test case, we try to get the job that has been deleted and we ensure that an exception is thrown and we also check if the method deleteById has been invoked.

d. A user manual that tells us how to install/use your program. This is meant for the end-user of the software. You may include screen shots, where appropriate.

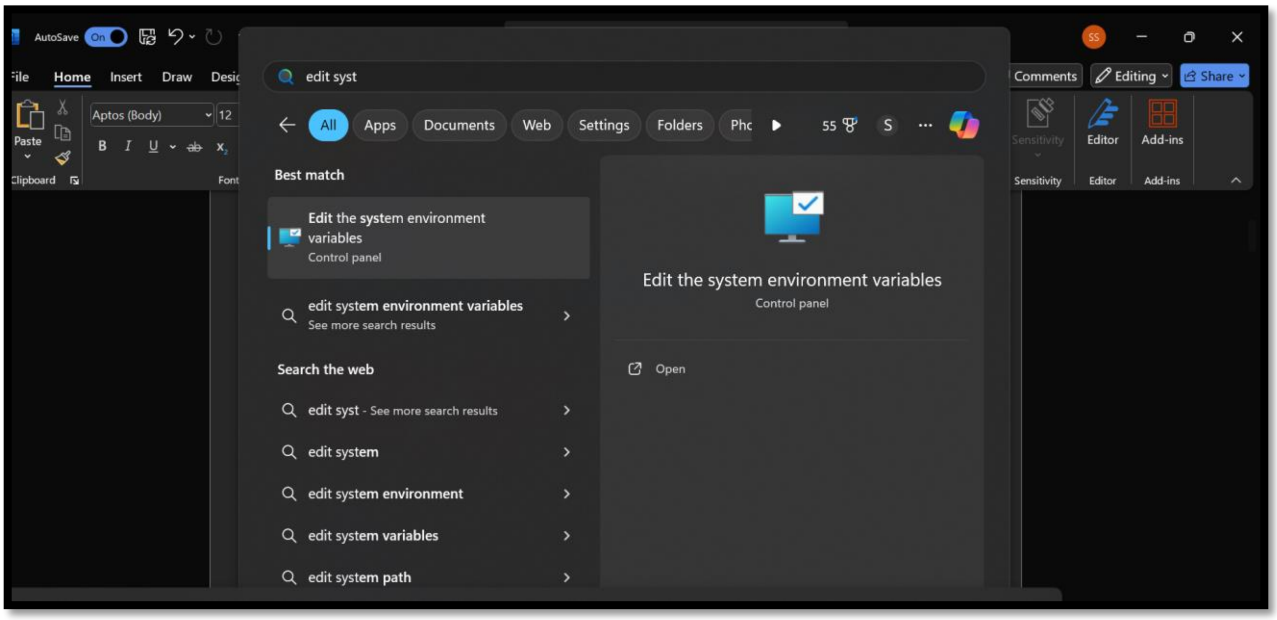
Installing and Running Software

Note: Ensure to have the correct java and angular versions to download correct dependencies. The college Wi-Fi is having an issue with oracle , so use personal hotspot to execute.

For Windows(**Backend**):

- Step 1) Install jdk-17 https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

- Step 2) Right-click This PC or Computer on your desktop or in File Explorer, and go to Properties (or press windows key and search for edit the system environment variables like in the attached screenshot below). Then select Advanced system settings on the left, and then click on Environment Variables button.In the System Variables section, click New, and create a system variable named JAVA_HOME and set the path of downloaded jdk.



Edit the system environment variables
Control panel

edit system environment variables
See more search results

Search the web

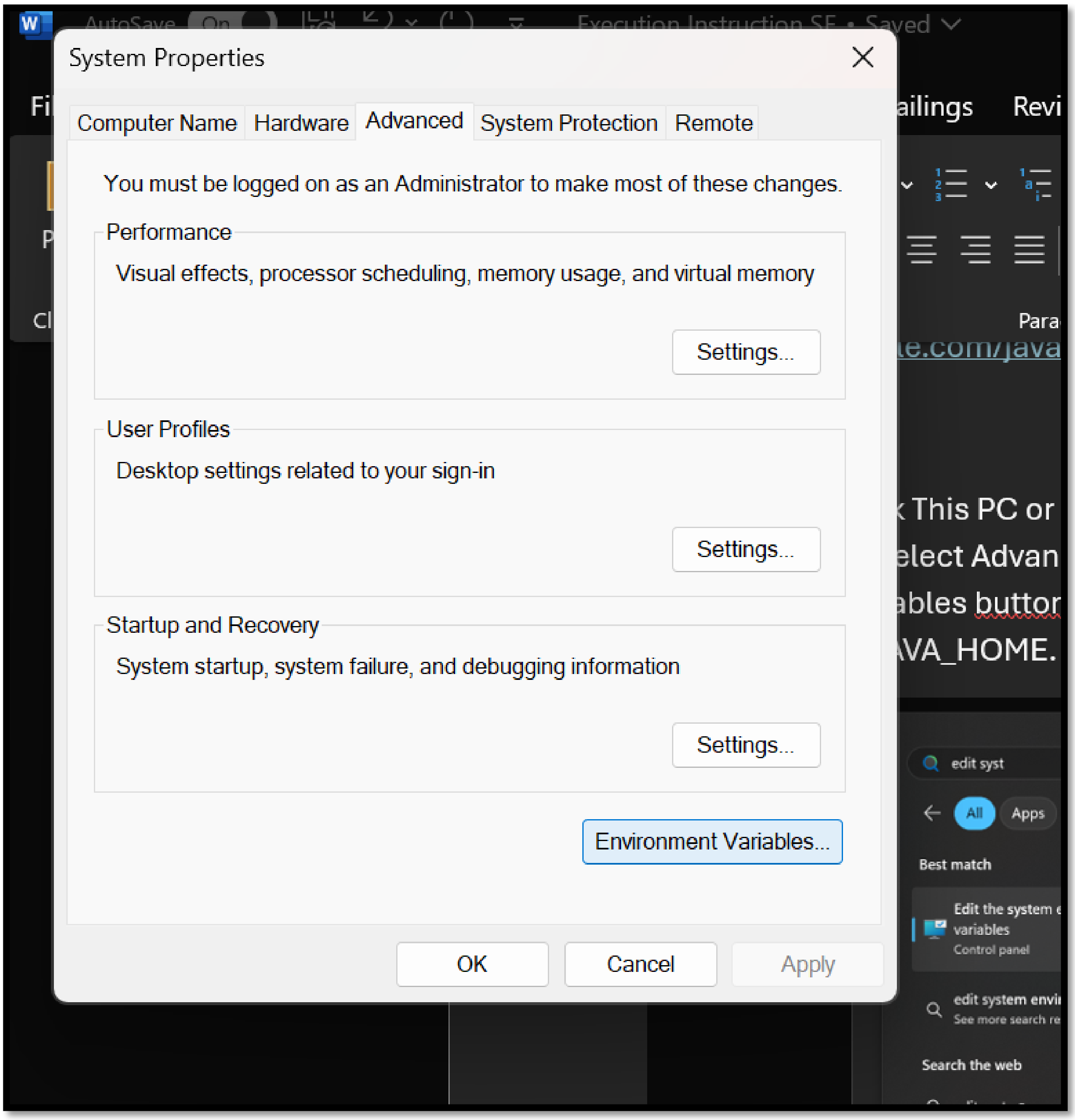
edit syst - See more search results

edit system

edit system environment

edit system variables

edit system path



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

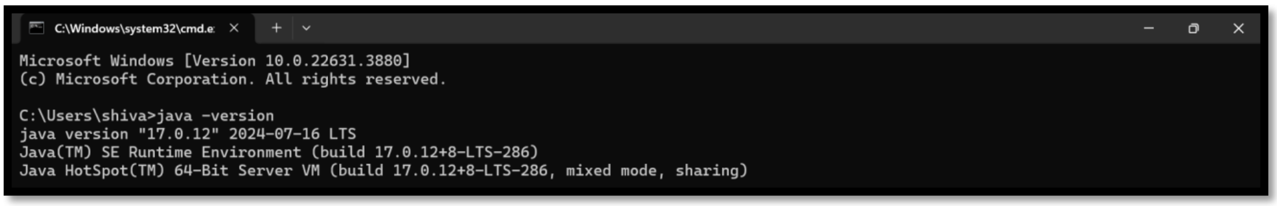
System startup, system failure, and debugging information

Set the variable value to the path where JDK 17 is installed (e.g., C:\Program Files\Java\jdk-17). Variable name as JAVA_HOME and path to the jdk as variable value, and also add %JAVA_HOME%\bin to the Path system variable.

Restart the command prompt.

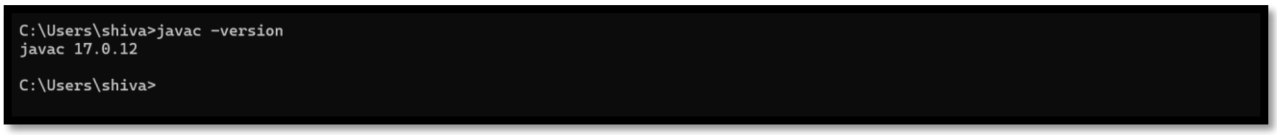
Verification (open command prompt)

Use this command to check java's version: java -version



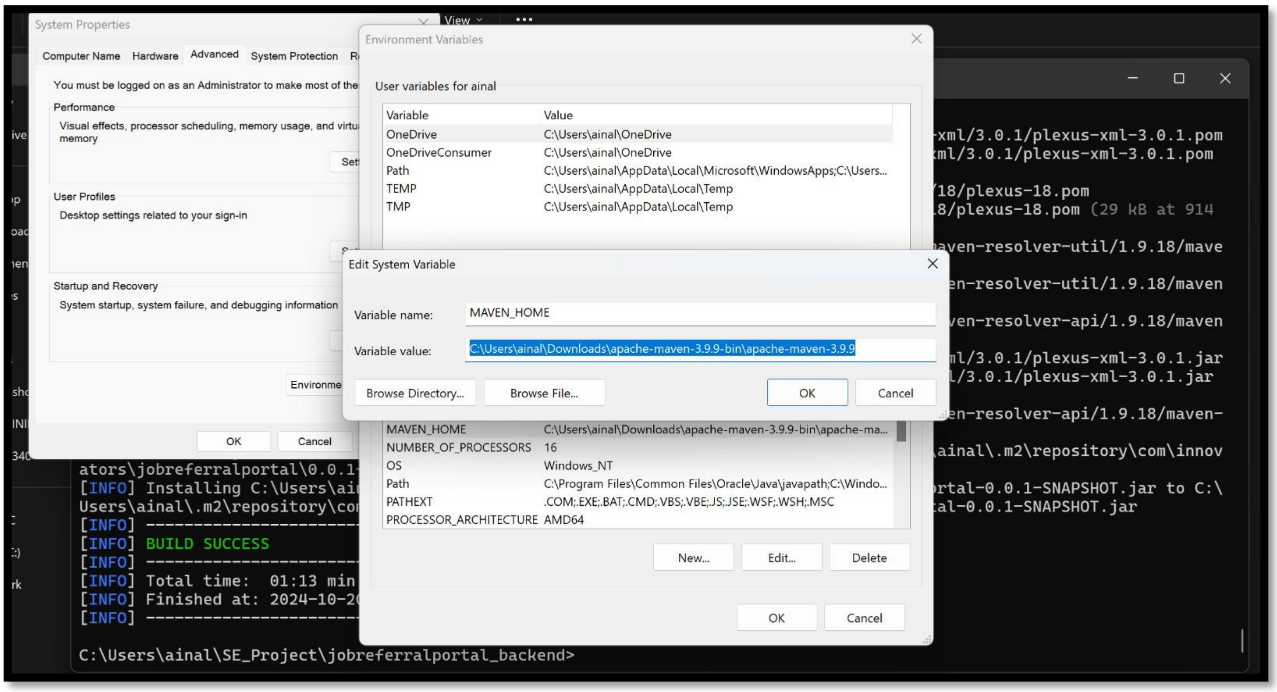
```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shiva>java -version
java version "17.0.12" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

For checking if jdk is available : javac -version



```
C:\Users\shiva>javac -version
javac 17.0.12

C:\Users\shiva>
```

- Step 3) Install maven
  https://maven.apache.org/download.cgi
  imilar to JAVA_HOME, create MAVEN_HOME and Add %MAVEN_HOME%\bin to the **Path** system variable. And restart the command prompt.



Edit System Variable

Variable name: MAVEN_HOME

Variable value: C:\Users\ainal\Downloads\apache-maven-3.9.9-bin\apache-maven-3.9.9

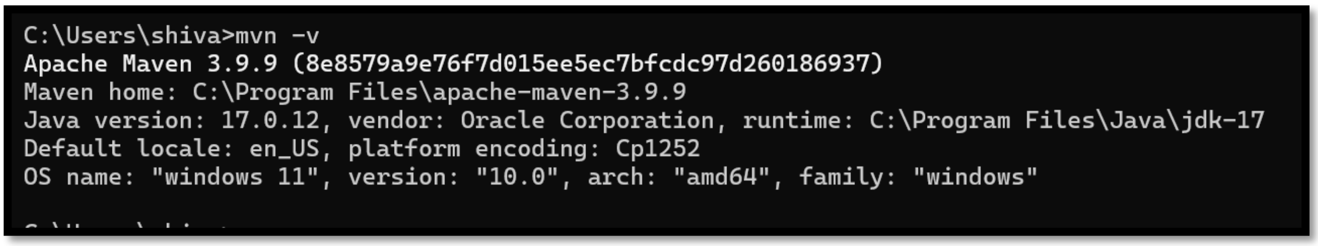
User variables for ainal

Variable	Value
OneDrive	C:\Users\ainal\OneDrive
OneDriveConsumer	C:\Users\ainal\OneDrive
Path	C:\Users\ainal\AppData\Local\Microsoft\WindowsApps;C:\Users...
TEMP	C:\Users\ainal\AppData\Local\Temp
TMP	C:\Users\ainal\AppData\Local\Temp

MAVEN_HOME	C:\Users\ainal\Downloads\apache-maven-3.9.9-bin\apache-ma...
NUMBER_OF_PROCESSORS	16
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Windo...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

```
ators\jobreferralportal\0.0.1-
[INFO] Installing C:\Users\ai
Users\ainal\.m2\repository\co
[INFO] ----------------------
[INFO] BUILD SUCCESS
[INFO] ----------------------
[INFO] Total time:  01:13 min
[INFO] Finished at: 2024-10-2
[INFO] ----------------------

C:\Users\ainal\SE_Project\jobreferralportal_backend>
```

For verification use : mvn -v
you should get



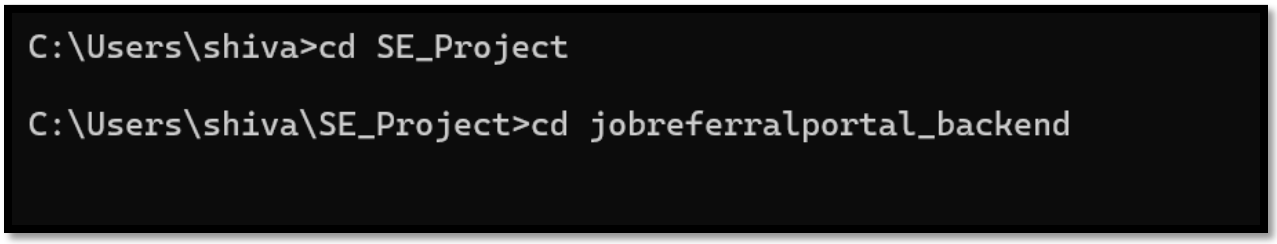
```
C:\Users\shiva>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Program Files\apache-maven-3.9.9
Java version: 17.0.12, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

- Step 4) create fork of the repository and clone it into your local https://github.com/ravali-kudaravalli24/SE_Project/tree/main

- Step 5) edit the application.properties file in the src code pring.datasource.url=jdbc:oracle:thin:@referralportaldb_high?TNS_ADMIN=C:/Users/shiva/SE_Project/jobreferralportal_backend/Wallet_ReferralPortalDB

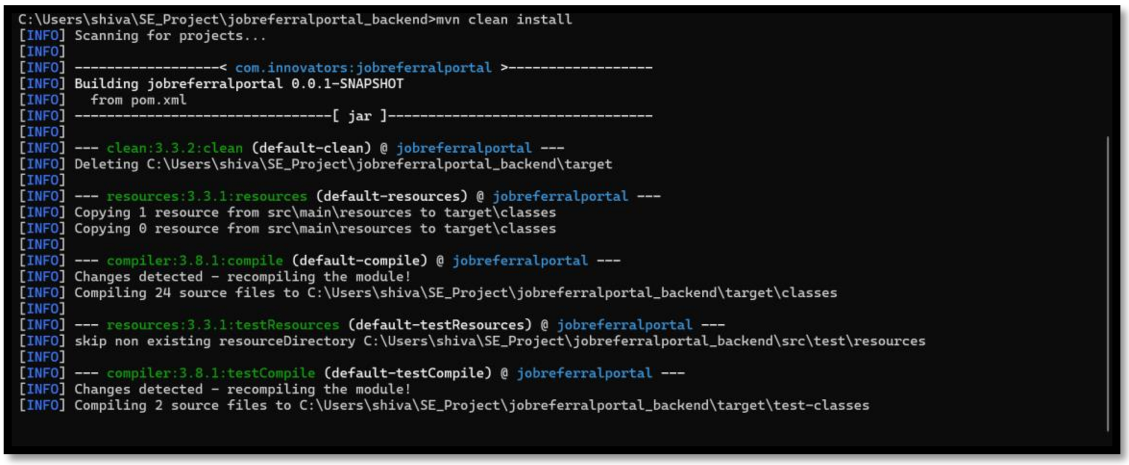
Update the above line, to point to the location of the connection wallet in your system.

- Step 6) cd into the jobreferralportal_backend directory



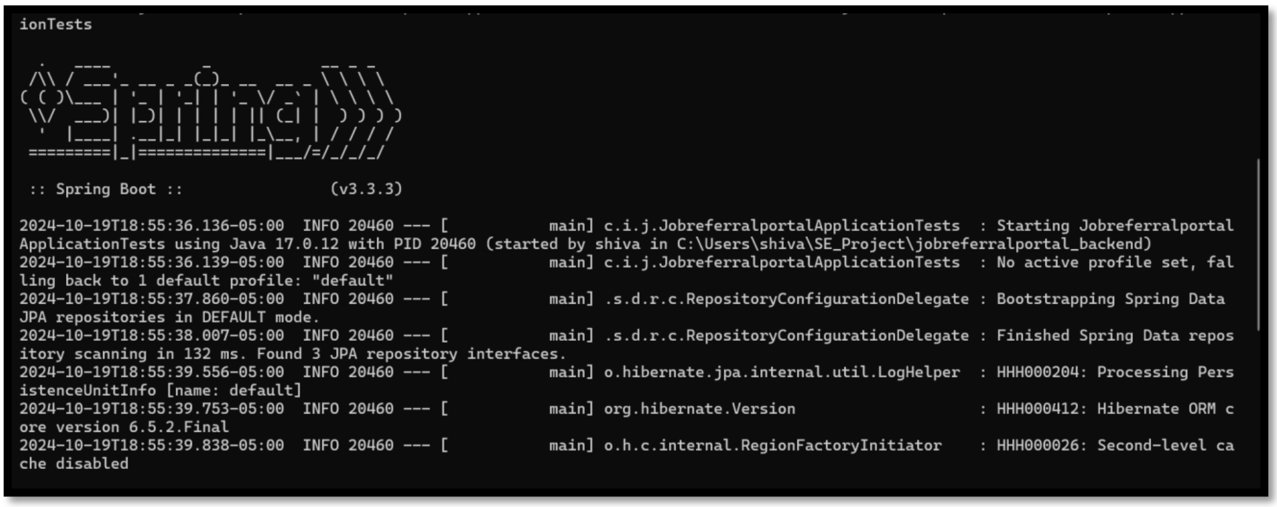
```
C:\Users\shiva>cd SE_Project

C:\Users\shiva\SE_Project>cd jobreferralportal_backend
```

- Step 7) Downloading all the required dependencies : mvn clean install



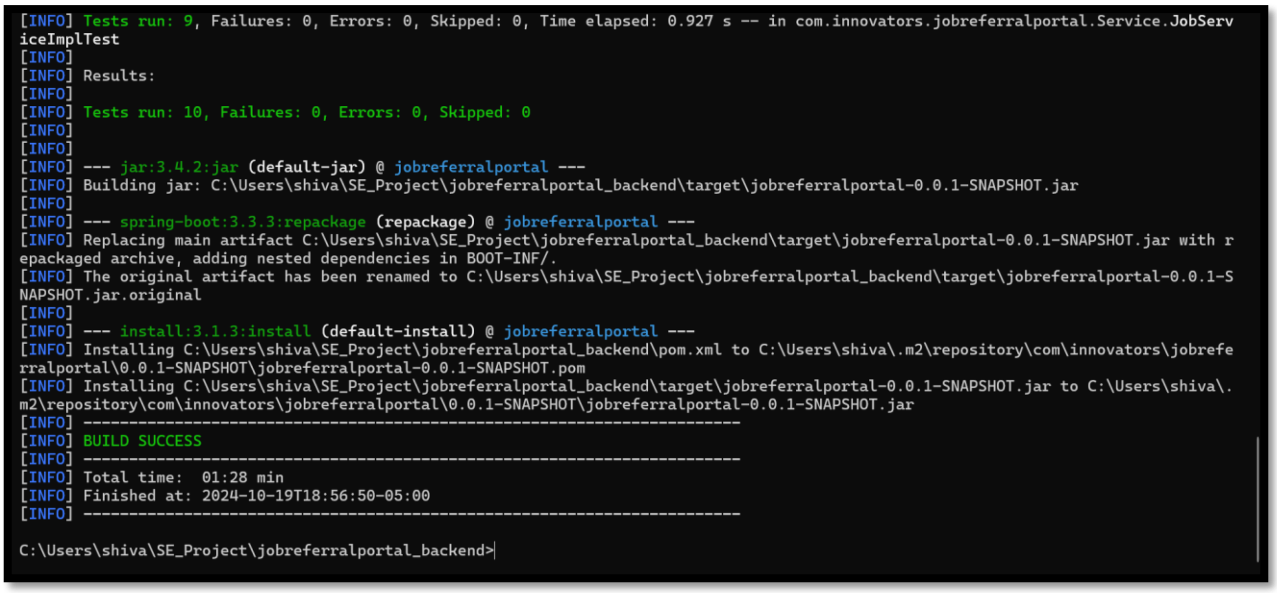
```
C:\Users\shiva\SE_Project\jobreferralportal_backend>mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.innovators:jobreferralportal >------------------
[INFO] Building jobreferralportal 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ jobreferralportal ---
[INFO] Deleting C:\Users\shiva\SE_Project\jobreferralportal_backend\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ jobreferralportal ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ jobreferralportal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 24 source files to C:\Users\shiva\SE_Project\jobreferralportal_backend\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ jobreferralportal ---
[INFO] skip non existing resourceDirectory C:\Users\shiva\SE_Project\jobreferralportal_backend\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ jobreferralportal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to C:\Users\shiva\SE_Project\jobreferralportal_backend\target\test-classes
```

The application should start running



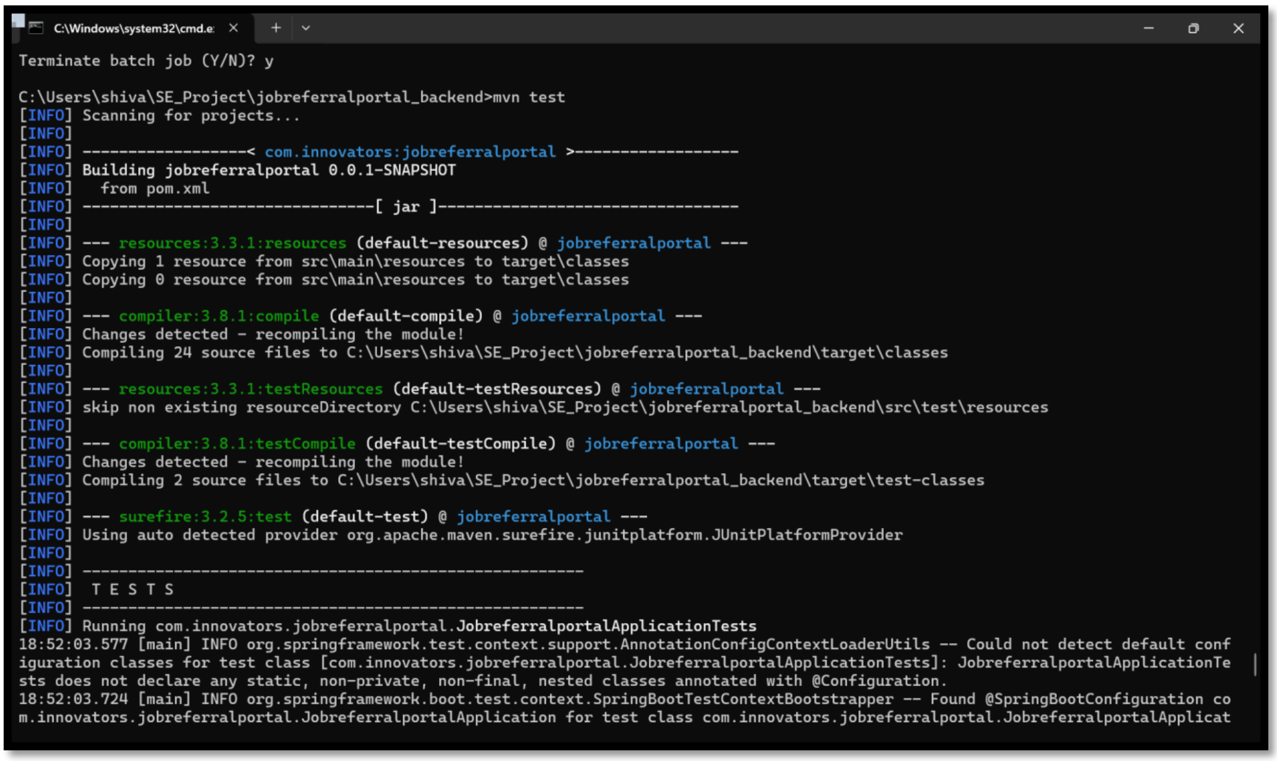
```
ionTests

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.3)

2024-10-19T18:55:36.136-05:00  INFO 20460 --- [           main] c.i.j.JobreferralportalApplicationTests  : Starting Jobreferralportal
ApplicationTests using Java 17.0.12 with PID 20460 (started by shiva in C:\Users\shiva\SE_Project\jobreferralportal_backend)
2024-10-19T18:55:36.139-05:00  INFO 20460 --- [           main] c.i.j.JobreferralportalApplicationTests  : No active profile set, fal
ling back to 1 default profile: "default"
2024-10-19T18:55:37.860-05:00  INFO 20460 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data
JPA repositories in DEFAULT mode.
2024-10-19T18:55:38.007-05:00  INFO 20460 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repos
itory scanning in 132 ms. Found 3 JPA repository interfaces.
2024-10-19T18:55:39.556-05:00  INFO 20460 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing Pers
istenceUnitInfo [name: default]
2024-10-19T18:55:39.753-05:00  INFO 20460 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM c
ore version 6.5.2.Final
2024-10-19T18:55:39.838-05:00  INFO 20460 --- [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level ca
che disabled
```

All the test cases will run, and you will get a build successful message and the job will end.

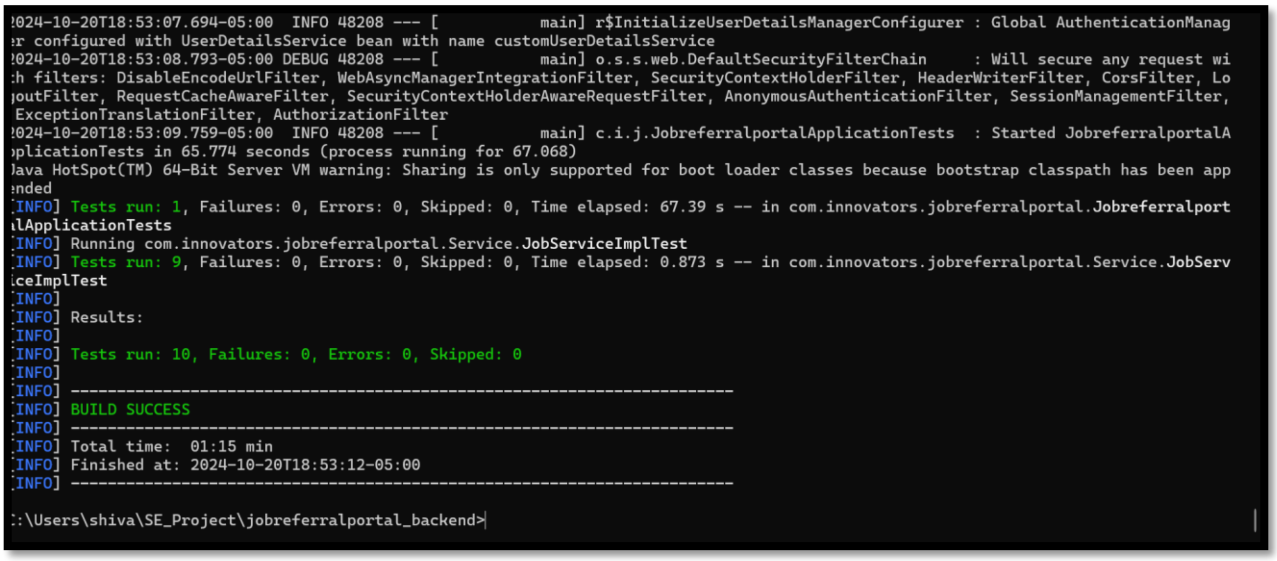


```
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.927 s -- in com.innovators.jobreferralportal.Service.JobServ
iceImplTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ jobreferralportal ---
[INFO] Building jar: C:\Users\shiva\SE_Project\jobreferralportal_backend\target\jobreferralportal-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.3.3:repackage (repackage) @ jobreferralportal ---
[INFO] Replacing main artifact C:\Users\shiva\SE_Project\jobreferralportal_backend\target\jobreferralportal-0.0.1-SNAPSHOT.jar with r
epackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\shiva\SE_Project\jobreferralportal_backend\target\jobreferralportal-0.0.1-S
NAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.3:install (default-install) @ jobreferralportal ---
[INFO] Installing C:\Users\shiva\SE_Project\jobreferralportal_backend\pom.xml to C:\Users\shiva\.m2\repository\com\innovators\jobrefe
rralportal\0.0.1-SNAPSHOT\jobreferralportal-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\shiva\SE_Project\jobreferralportal_backend\target\jobreferralportal-0.0.1-SNAPSHOT.jar to C:\Users\shiva\.
m2\repository\com\innovators\jobreferralportal\0.0.1-SNAPSHOT\jobreferralportal-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:28 min
[INFO] Finished at: 2024-10-19T18:56:50-05:00
[INFO] ------------------------------------------------------------------------

C:\Users\shiva\SE_Project\jobreferralportal_backend>
```

- Step 8) running the test cases:
  cd into jobreferralportal_backend
  command: mvn test



```
C:\Windows\system32\cmd.e
Terminate batch job (Y/N)? y

C:\Users\shiva\SE_Project\jobreferralportal_backend>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< com.innovators:jobreferralportal >-----------------
[INFO] Building jobreferralportal 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ jobreferralportal ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ jobreferralportal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 24 source files to C:\Users\shiva\SE_Project\jobreferralportal_backend\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ jobreferralportal ---
[INFO] skip non existing resourceDirectory C:\Users\shiva\SE_Project\jobreferralportal_backend\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ jobreferralportal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to C:\Users\shiva\SE_Project\jobreferralportal_backend\target\test-classes
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ jobreferralportal ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.innovators.jobreferralportal.JobreferralportalApplicationTests
18:52:03.577 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default conf
iguration classes for test class [com.innovators.jobreferralportal.JobreferralportalApplicationTests]: JobreferralportalApplicationTe
sts does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
18:52:03.724 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguration co
m.innovators.jobreferralportal.JobreferralportalApplication for test class com.innovators.jobreferralportal.JobreferralportalApplicat
```

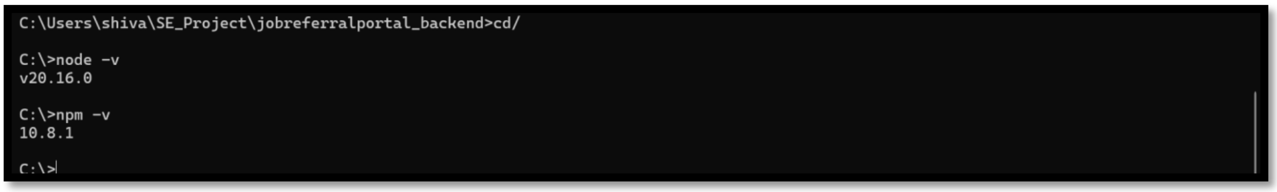


```
024-10-20T18:53:07.694-05:00  INFO 48208 --- [           main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManag
r configured with UserDetailsService bean with name customUserDetailsService
024-10-20T18:53:08.793-05:00 DEBUG 48208 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request wi
h filters: DisableEncodeUrlFilter, WebAsyncManagerIntegrationFilter, SecurityContextHolderFilter, HeaderWriterFilter, CorsFilter, Lo
outFilter, RequestCacheAwareFilter, SecurityContextHolderAwareRequestFilter, AnonymousAuthenticationFilter, SessionManagementFilter,
ExceptionTranslationFilter, AuthorizationFilter
024-10-20T18:53:09.759-05:00  INFO 48208 --- [           main] c.i.j.JobreferralportalApplicationTests  : Started JobreferralportalA
plicationTests in 65.774 seconds (process running for 67.068)
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been app
nded
INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 67.39 s -- in com.innovators.jobreferralportal.Jobreferralport
lApplicationTests
INFO] Running com.innovators.jobreferralportal.Service.JobServiceImplTest
INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.873 s -- in com.innovators.jobreferralportal.Service.JobServ
ceImplTest
INFO]
INFO] Results:
INFO]
INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
INFO]
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  01:15 min
INFO] Finished at: 2024-10-20T18:53:12-05:00
INFO] ------------------------------------------------------------------------

:\Users\shiva\SE_Project\jobreferralportal_backend>
```

- Step 9) Run the application
  run the command
  mvn spring-boot:run
  Application will be running at http://localhost:8090 : but you won't be able to access it as you haven't been authorized yet.
  Running Frontend
  Make sure that backend is up and running before the frontend.

step 1) for running angular project, first we need Node.js and npm, for that download https://nodejs.org/

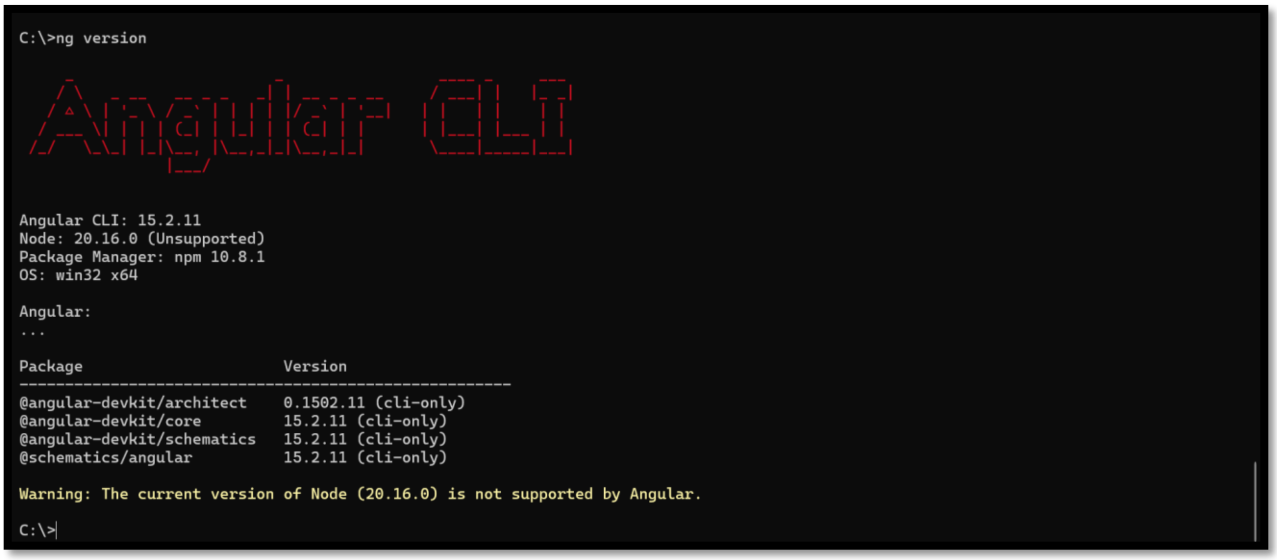
verification: run commands node-v and npm -v



```
C:\Users\shiva\SE_Project\jobreferralportal_backend>cd/

C:\>node -v
v20.16.0

C:\>npm -v
10.8.1

C:\>
```

step 2) install angular cli globally
ensure that you download angular 15
npm install -g @angular/cli@15
verify this by running

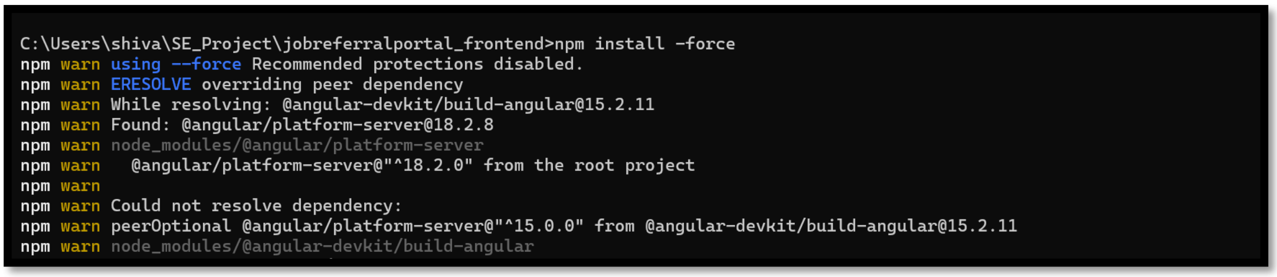
ng version



```
C:\>ng version

Angular CLI: 15.2.11
Node: 20.16.0 (Unsupported)
Package Manager: npm 10.8.1
OS: win32 x64

Angular:
...

Package                      Version
------------------------------------------------------
@angular-devkit/architect    0.1502.11 (cli-only)
@angular-devkit/core         15.2.11 (cli-only)
@angular-devkit/schematics   15.2.11 (cli-only)
@schematics/angular          15.2.11 (cli-only)

Warning: The current version of Node (20.16.0) is not supported by Angular.

C:\>
```

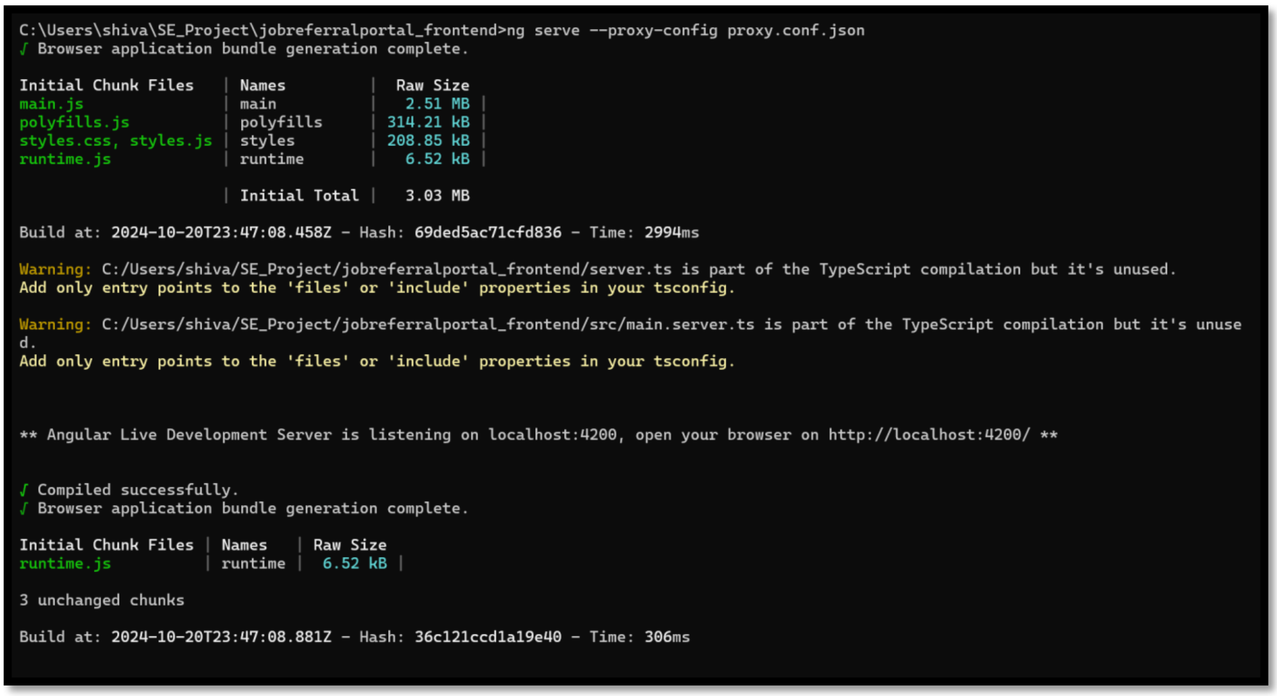
step 3) cd into jobreferralportal_frontend dir and install required dependencies.

run: npm install --force (please make sure you use --force to resolve pdeps issue)



```
C:\Users\shiva\SE_Project\jobreferralportal_frontend>npm install -force
npm warn using --force Recommended protections disabled.
npm warn ERESOLVE overriding peer dependency
npm warn While resolving: @angular-devkit/build-angular@15.2.11
npm warn Found: @angular/platform-server@18.2.8
npm warn node_modules/@angular/platform-server
npm warn   @angular/platform-server@"^18.2.0" from the root project
npm warn
npm warn Could not resolve dependency:
npm warn peerOptional @angular/platform-server@"^15.0.0" from @angular-devkit/build-angular@15.2.11
npm warn node_modules/@angular-devkit/build-angular
```

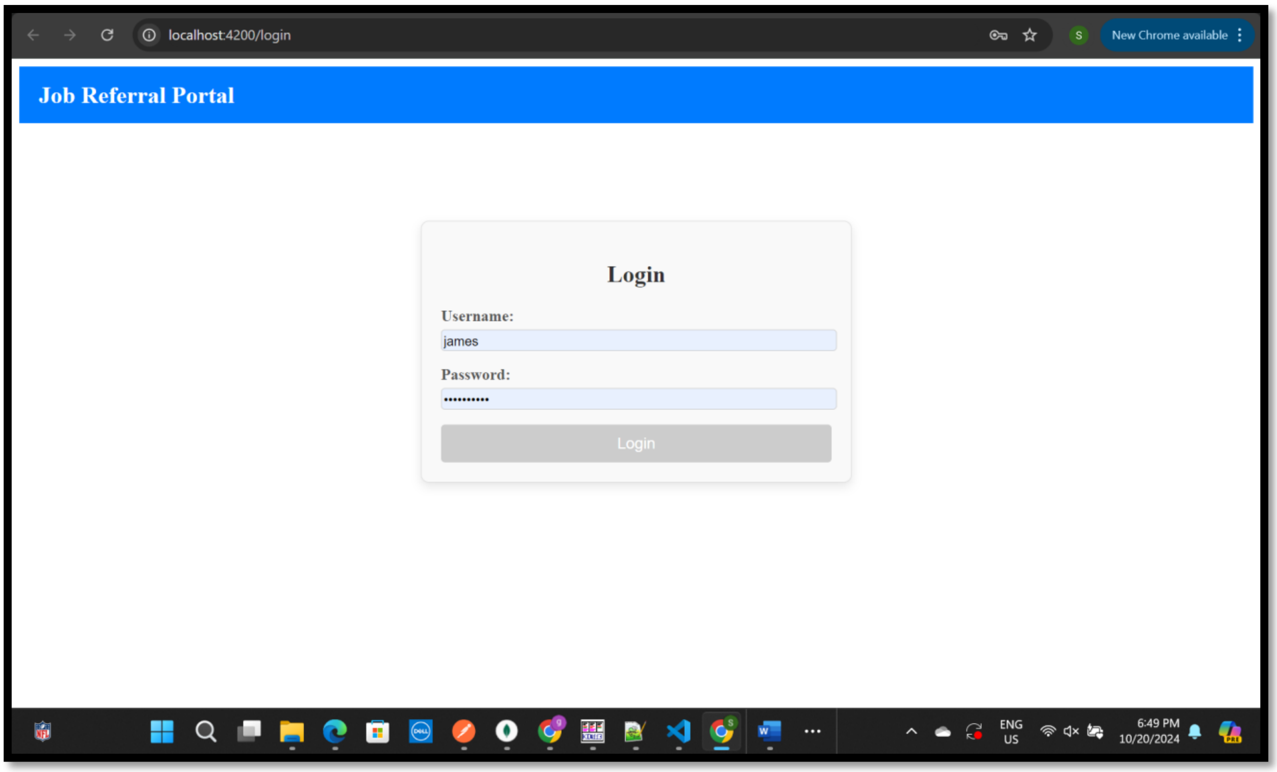
now run
ng serve --proxy-config proxy.conf.json



```
C:\Users\shiva\SE_Project\jobreferralportal_frontend>ng serve --proxy-config proxy.conf.json
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
main.js               | main          |   2.51 MB |
polyfills.js          | polyfills     | 314.21 kB |
styles.css, styles.js | styles        | 208.85 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   3.03 MB

Build at: 2024-10-20T23:47:08.458Z - Hash: 69ded5ac71cfd836 - Time: 2994ms

Warning: C:/Users/shiva/SE_Project/jobreferralportal_frontend/server.ts is part of the TypeScript compilation but it's unused.
Add only entry points to the 'files' or 'include' properties in your tsconfig.

Warning: C:/Users/shiva/SE_Project/jobreferralportal_frontend/src/main.server.ts is part of the TypeScript compilation but it's unuse
d.
Add only entry points to the 'files' or 'include' properties in your tsconfig.


** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


√ Compiled successfully.
√ Browser application bundle generation complete.

Initial Chunk Files | Names   |  Raw Size
runtime.js          | runtime |   6.52 kB |

3 unchanged chunks

Build at: 2024-10-20T23:47:08.881Z - Hash: 36c121ccd1a19e40 - Time: 306ms
```

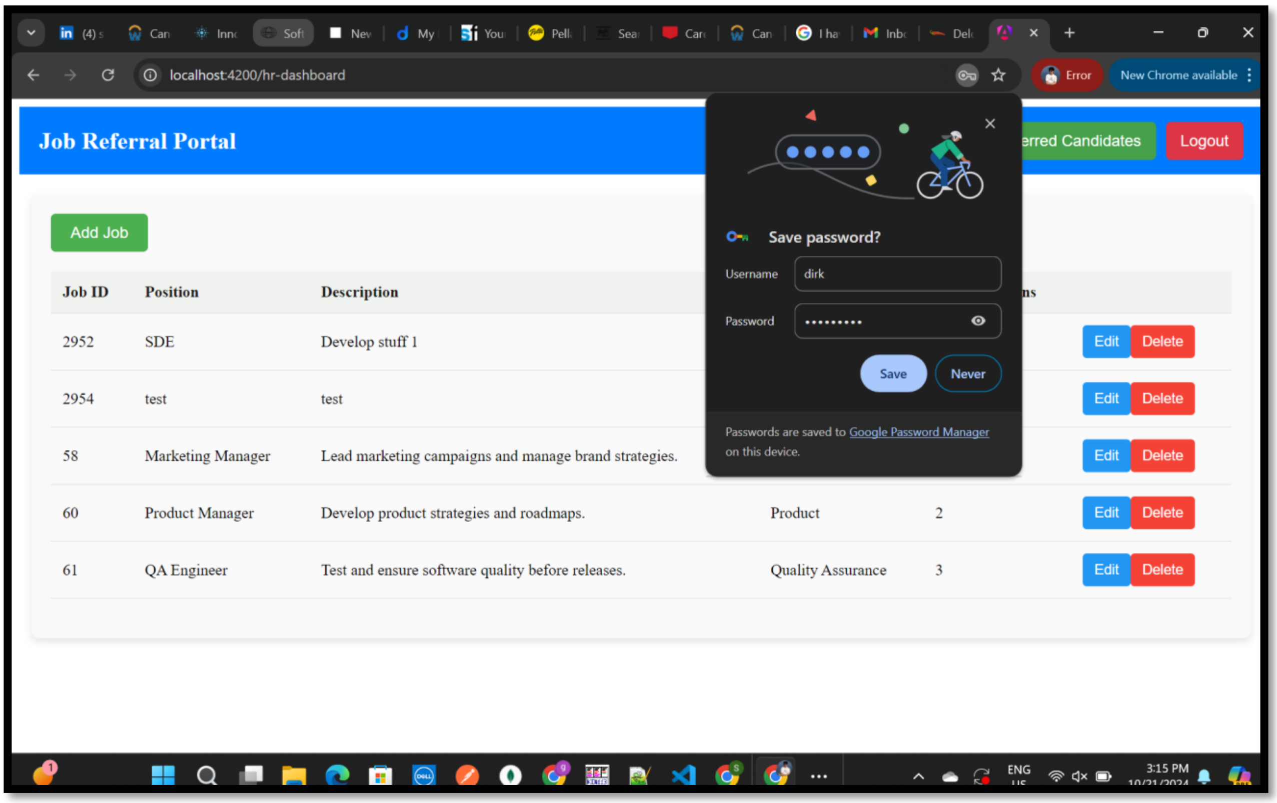
now open a web browser and go to localhost:4200/

http://localhost:4200/



for Employee view : use credentials username : james, password: James@2000

for HR view: use credentials username: dirk, password: Dirk@2000



localhost:4200/hr-dashboard

Job Referral Portal

Add Job

Job ID	Position	Description			
2952	SDE	Develop stuff 1			Edit Delete
2954	test	test			Edit Delete
58	Marketing Manager	Lead marketing campaigns and manage brand strategies.			Edit Delete
60	Product Manager	Develop product strategies and roadmaps.	Product	2	Edit Delete
61	QA Engineer	Test and ensure software quality before releases.	Quality Assurance	3	Edit Delete

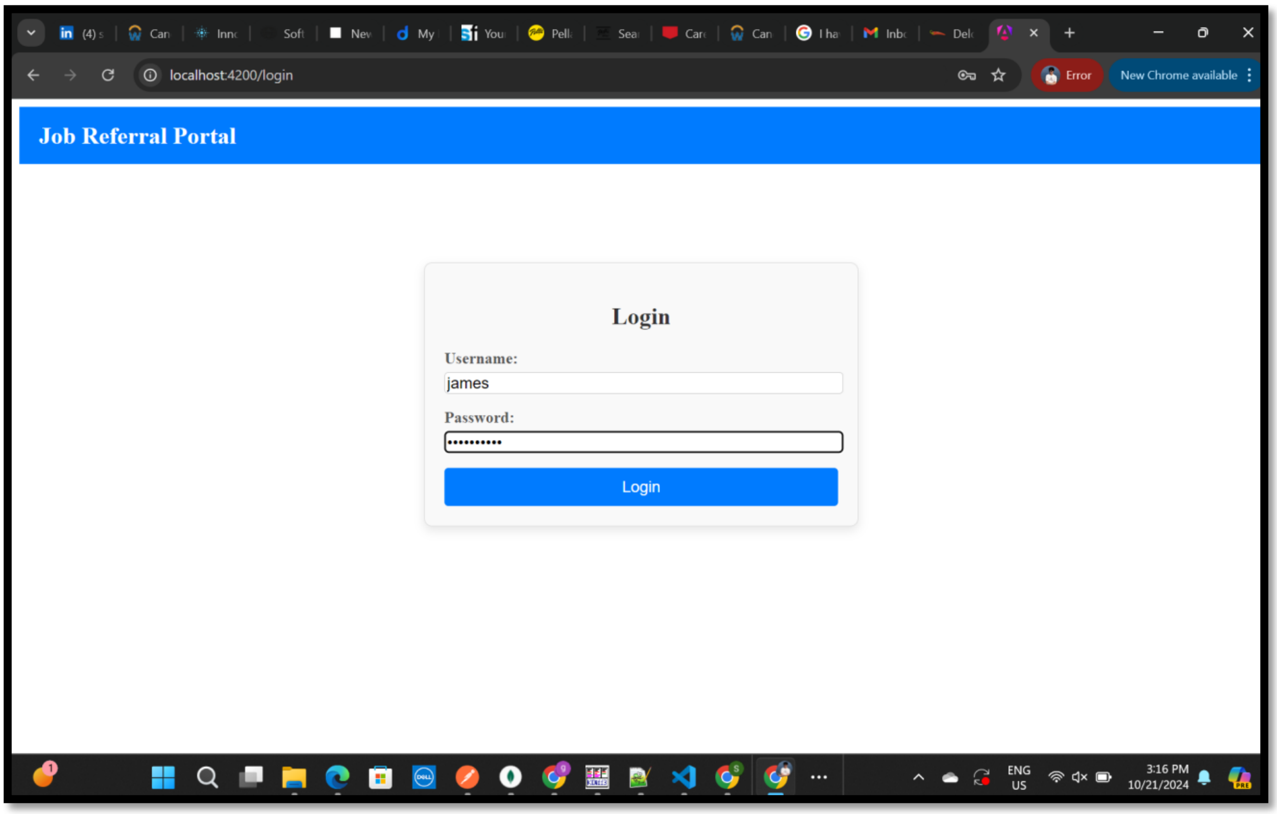
Save password?

Username: dirk

Password: •••••••••

Save Never

Passwords are saved to Google Password Manager on this device.



localhost:4200/login

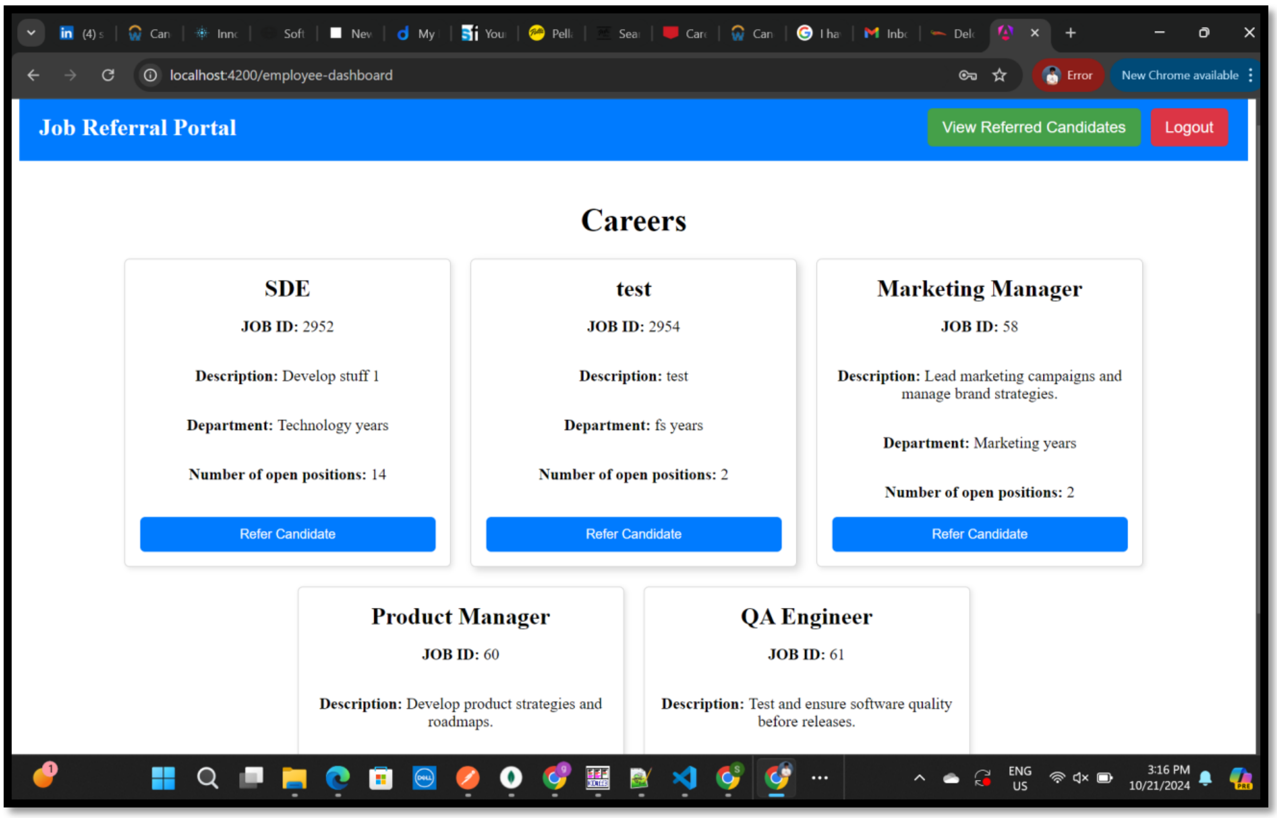
Job Referral Portal

Login

Username: james

Password: ••••••••••

Login


localhost:4200/employee-dashboard

Job Referral Portal

View Referred Candidates | Logout

Careers

SDE
JOB ID: 2952
Description: Develop stuff 1
Department: Technology years
Number of open positions: 14
Refer Candidate

test
JOB ID: 2954
Description: test
Department: fs years
Number of open positions: 2
Refer Candidate

Marketing Manager
JOB ID: 58
Description: Lead marketing campaigns and manage brand strategies.
Department: Marketing years
Number of open positions: 2
Refer Candidate

Product Manager
JOB ID: 60
Description: Develop product strategies and roadmaps.

QA Engineer
JOB ID: 61
Description: Test and ensure software quality before releases.

f. A section that briefly describes feedback received during **the peer review** session and actions taken based on the feedback.

Feedback given by the Team -8 during the peer review session:

Cross-checking candidate qualifications:

Team 8 have suggested us to improve efficiency by cross-checking candidate's qualifications against job requirements, by using AI for recommendations, but the aim of the project is to make the referral process easy. We thought it is a valuable addition to ensure accuracy in referrals. But there are many steps after referring a candidate, the HR team will manually go through the applications and move them to the next rounds, so we don't want to use AI for this step and want to continue it manually. The suggestion was partially accepted, but in our project, it would not be the right choice as manual steps help in maintaining controlled evaluation process, so that HR team can make decisions based on their performance.

Referral history tracking:

They recommended to provide both HR and employees visibility into the progress for each referral at every stage. This totally aligns with our view and makes total sense as it is very essential and useful. Hence this feature was accepted by the team as it allows users to track the status of referrals, ensuring transparency and helping both parties stay informed throughout the referral process. Tracking the referral progress was agreed by the team upon as a useful feature.

Advanced search criteria:

Team 8 suggested to add an advanced filters based on factors like experience, or salary expectations, location. We have implemented a search operation where an employee can access the portal and can search for a particular job opening. Taking an example, a search for a job opening for the position of Java developer.

So, when the employee searches for Java, then the jobs containing "Java" will be displayed. This functionality has been implemented. Based on the location, the employee can refer a candidate. We thought this suggestion as a valuable enhancement to make job searches more efficient.

Automated-notifications:

They have asked us to initiate notifications. As suggested that both the referrer and referred candidate receive automatic updates on status changes, as being moved to the next round or rejected.

Automated notifications may not entirely implement due to manual steps and some personalized updates in certain situations. The employer can see the status of the referred candidate in the dashboard.

Personalized job recommendations:

The opposite team suggested to add a feature to provide personalized job recommendations on the basis of user's abilities and experiences. Our team felt this recommendation is not a useful thing, as referrals are made by employees who already know the candidates, so this was considered unnecessary.

Filter jobs based on location:

They have also recommended us to include location-based filtering to aid employees in referring candidates. As being said earlier, we have implemented search operations only on position names. It was partially accepted to implement this as part of the enhanced search feature. So before going ahead we need to discuss more about this feature and finalize it with team member and include this in the up-coming steps.

Feedback and contribution during the peer review session:

In the team, we assigned works for each of us and collaboratively worked for the better outcome. Every member in the team gave their possible suggestions and some were accepted and rejected by having discussions. We would like to continue our contribution for the project and want to learn different skills each week.

Rishika being the project management lead, assigned roles and took lead in meetings. Involved in video presentation and gave detailed description of the project. She planned the team meetings and worked on functional and non-functional requirements and gave her suggestions. She has also written an API about the search operation. Rishika's effort towards the project and for the team is appreciative.

Aravind's work for the project has been outstanding. He made changes in PowerPoint presentation and took initiative to collect the requirements. Joined the video presentation and described about the project's frontend roles. Helped in reviewing the documentation and submitted files. Involved in assigning the tasks for the team members. He contributed in documentation for functional and non- functional requirements and interface.

Shiva's work towards the project is exceptional. Worked on back-end, started the spring-boot project and created the entities and pushed them to the repository. Actively participates in the team meetings and reviewed the documentation. Additionally, improved the operation of the project. He worked on three development phase and also on user-authentication. Shiva has finished an API called refer a candidate.

Ravali's developed the documentation and gave all the detailed explanation. Also drafted the Readme File. Her work on developing code and gave her hand in the front-end department. Created the GitHub project's repository and ensured that everyone committed. Ravali being the design lead, she involved in designing the development phase plan for the project. She also finished writing APIs for the job listing.

Ramya's amazing skills in programming for the front-end code made backbone of our project. She developed the basic project structure and added that to GitHub. Furthermore, she prepared the PowerPoint presentation slides and provided detailed information. She involved in designing the overall structure of the system and also written the documentation for system diagrams and sub- systems. On the other hand, she created and added the meeting minutes and committed to the GitHub.

Rajini identified the risks involved in the project and ensured that the system is running efficiently without errors. Being the Demo and Presentation lead I created a video that gives detailed information of project risks and outcomes. Attended each meeting that related to the project and gave suggestions. Contributed in writing the documentation for the sub-systems and also non-functional requirements. I made a helping hand in designing the overall structure of the system.

Shresta's effort for the project is immense. Helped in developing the documentation and created the summary report. Took initiative and designed the timeline chart which was well-structured and organized. Sreshta has always been supportive for the project development. She gave her hand in written documentation of three development phase. On the other side, she reviewed the deliverable 2 and submitted that file in the GitHub.

Vigna's work has been tremendous as she is working on database connections like API. She has set up the application configuration and ensured that system is running smoothly in all environments. She started to work on testing side for the development of the project. She has done her part by involving in the documentation of three development phases. She also involved in the team discussions and gave her suggestions for the improvement of the project.

Trisha has been supportive for the project development. She worked on the database connection. She attended the team meetings and gave her perspective which increased the ideology of the project. She involved in documentation for the system diagram and sub-systems. She made a great effort in designing the overall structure of the system.

g. brief reflection on what has been accomplished, what went well and could be improved.

<u>Accomplishments</u>

- **Core functionalities:** Our Team have successfully designed and implemented core functionalities for both kinds of users i.e. HR and Employees. Employees can now refer candidates, view job listing, and also look at the referrals made by them, HR can add jobs, view all the jobs, and also update them.

- **System Architecture and Database Integration**: As we have implemented the system with three-tier architecture, with user interface, business logic and database, we created a well-structured platform. This system is capable of handling multiple users simultaneously. The data is securely stored in an Autonomous Oracle database, and the passwords are encrypted and decrypted with the help of strong encryption algorithms.

- **Security and Performance**: Non-functional requirements such as security and performance were given importance. The project included encryption for sensitive data like passwords, role-based access i.e HR should not be able to access employee dashboard and employee must not access HR dashboard with their credentials, we performed different type of actions, like trying to access HR dashboard with employee token, we were successful in stopping from this unauthorized actions.

- **Collaboration and Planning**:
  The work was divided properly among teammates, everyone has put equal effort in their own areas, including designing the system architecture, writing APIs, testing them and creating documentation, the meetings were smooth this time, with individuals meeting their deadlines and delivering their works.
  <u>Room for Improvement</u>

- **User Interface Customization:** Basic functionality of UI has been built and is working effectively, but feedback was take from group members and we all believed that there could be more styling present in the UI, that would make the user interface better.

- **Api Documentation:** While the Api's functionality has been documented properly, comments on code are limited, though the readability of code is pretty decent, comments on the code will help understand the code better, we have decided to start commenting to explain what is happening, this would be helpful for other developers who might be working on the API written by another developer.

- **Exception Handling:** Even though after taking several measures to handle all exceptions some edge cases were overlooked, but due to unit testing we were able to notice all the edge cases and fixed the APIs to work seamlessly.

- **Communication:** The teams communication and coordination has seen a great improvement, yet we still need to find common time for all the teammates to work together, everyone is busy with their other assignments and mid-terms, made it difficult to find time to discuss and coordinate properly.

h. Member contribution table (should describe who wrote what components or classes of the system and what parts of the report).

Member name	Contribution description	Overall Contribution (%)	Note (if applicable)
Rishika Reddy	Involved in the main design of phase-1, reviewing all the documents and works for deliverable 3 and detailed descriptions of Phase-1	13	
Aravind Chitiprolu	Involved in designing UML diagrams of Class, Sequence, Use case diagrams, Reviewing the documents for Deliverable-3 and Initiated team meetings	11.5	
Shiva Sai Kondapuram	Involved in phase 1 design, unit test cases and user manual explain that how to use the program	13	
Ravali Kudaravalli	Involved in designing UML diagrams of Class,	12	

	Sequence, Use case diagrams, Reviewing the documents for Deliverable-3 and phase 1		
Ramya Madhavareddy	Involved in testing the unit test cases for the working program including description	11.5	
Sreshta Chityala	Involved in designing UML diagrams of Class, Sequence, Use case diagrams, and collecting all the required information about the project status and minutes of meeting.	11.5	
Vigna Penmetsa	Involved in feedback of peer review sessions	6	
Trisha Reddy Nidjintha	Involved in instructing to run/compile the program and user manual installation	10.5	
Rajini Vijetha Rudrarapu	Involved in designing sequence diagram and brief review of the project status and also peer review and accomplishments of the project.	11	