

Deliverable-5

Team Innovators

a. Requirements

List a set of requirements designated for this development phase.

- 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 3.

Updated Requirements for Phase 3

1. Leaderboard API: This API enables Employees as well as HRs to look at the scores, here by scores we mean, whenever a referral gets accepted the score of the employee who referred that candidate gets incremented by 1, this API returns a sorted list on the basis of the score, it also returns name of the employee and also their score, their ID and also gives their rank, Organizations can give incentives to the employees based on the score they have.

2. Refined Search Function (for jobs) : After getting feedback from other team and also from our internal discussions, we enhanced our search function, prior to this job search function was done on basis of the position title but now we have also added location, as well as keywords, HR can add keywords when uploading a job, it acts like a filter , when user searches for a job now the search API will go through the position title, keywords and also location, making this operation powerful, and also gives freedom to users to be less confined in searching.

3. Updated BulkUpload/AddJob: We have updated the API to accommodate the changes in the Entity Job, now that we have two variables added that are location and keywords, some adjustments were to be made to these APIs, they were made and tested.

4. Search in Referred Candidates: We also have implemented a search operation in referred candidates for employees and HRs as well, employees can now search the candidates they referred by name, and HRs can search all the referred candidates by their name.

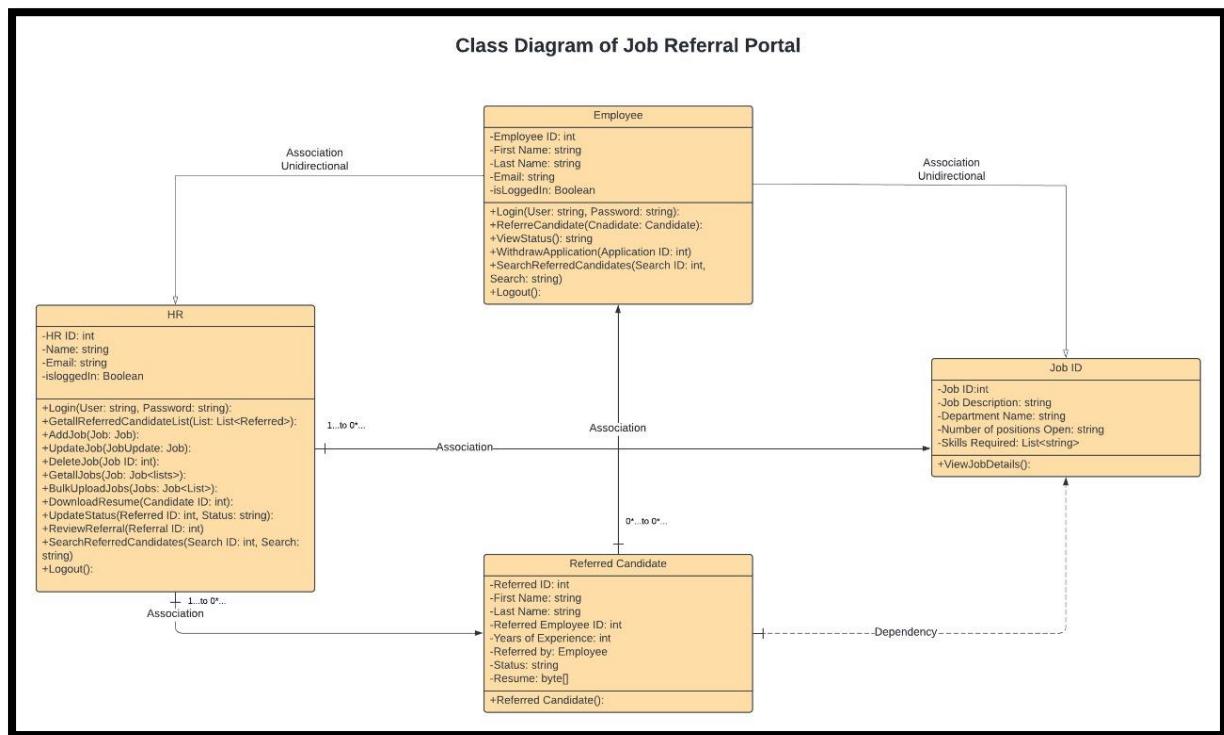
5. Withdraw Application: Now employees have an option to remove a referral that they have made, this option is available when a candidate has been referred and the employee no longer wants this application to be active, no matter what stage of candidature the referral is, the employee can withdraw their application.

6. Status Update: Initially we had an option for HR to put Custom status for referrals, but we decided to restrict this to few options, like UNDER CONSIDERATION, ONLINE ASSESSMENT etc but finally we also have ACCEPTED and REJECTED, this also core logic for our score incrementor sub method, as whenever ACCEPTED status comes into db our system will detect it and increment the score.

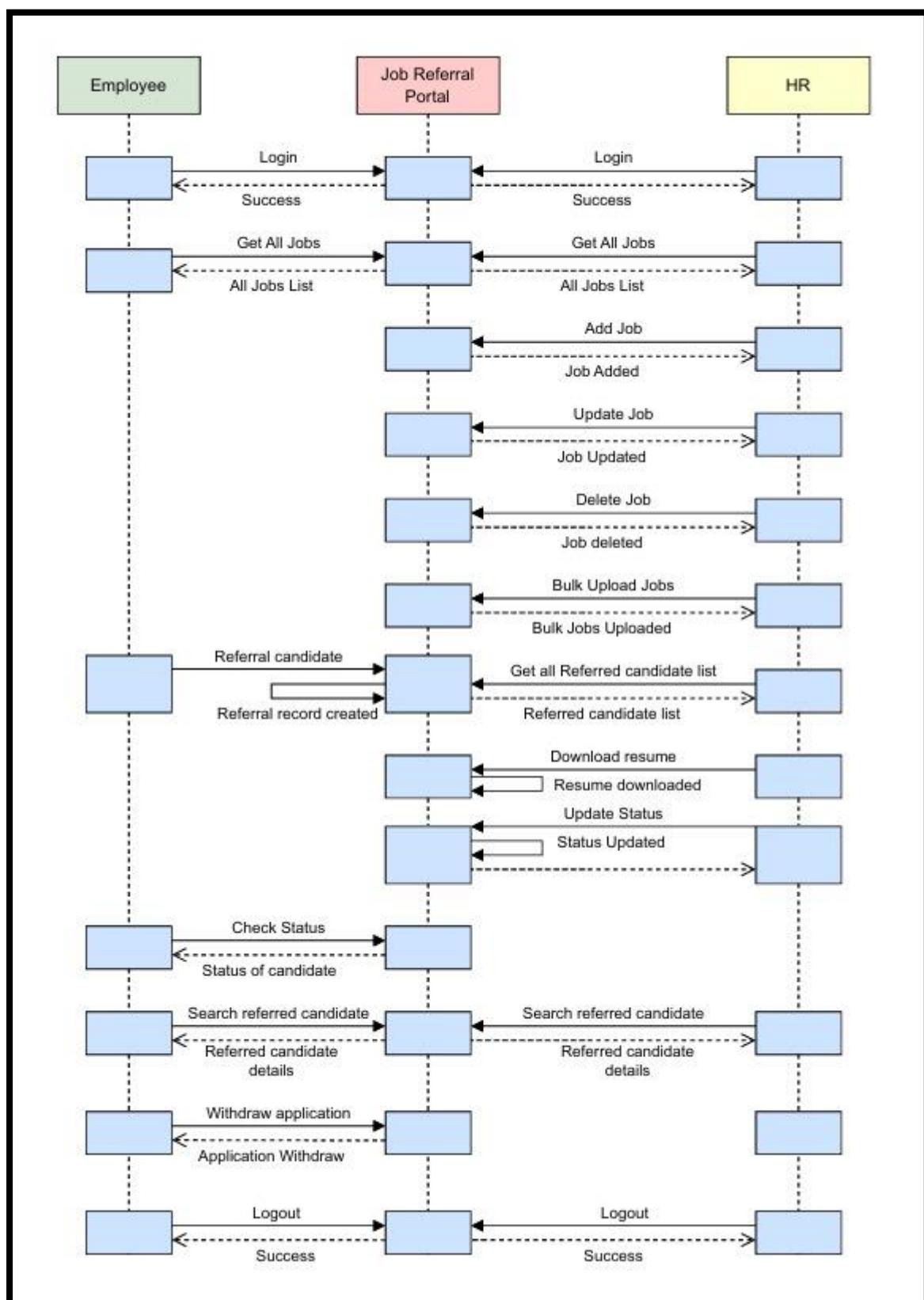
b. UML design. You must include the following diagrams:

- Class diagram
- Sequence diagram
- Use case diagram – one normal case and one error case should be included.

Class Diagram:

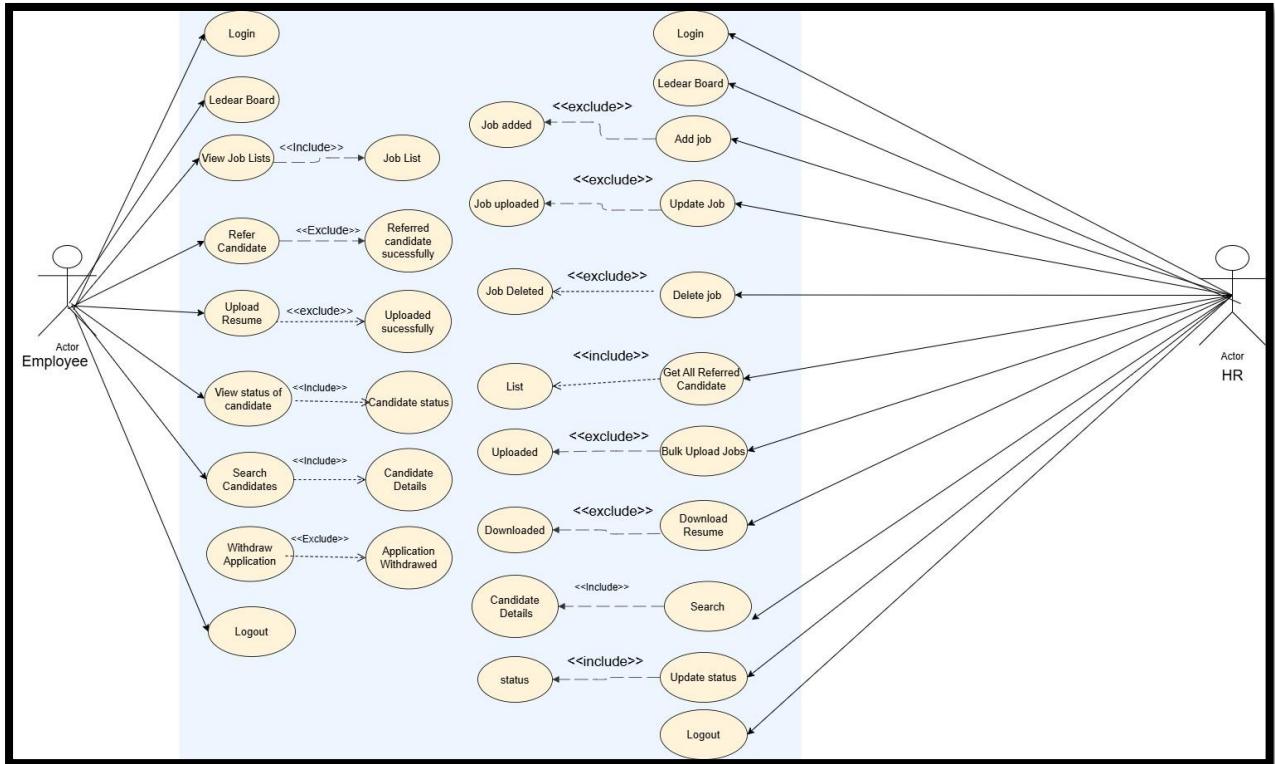


Sequence Diagram:

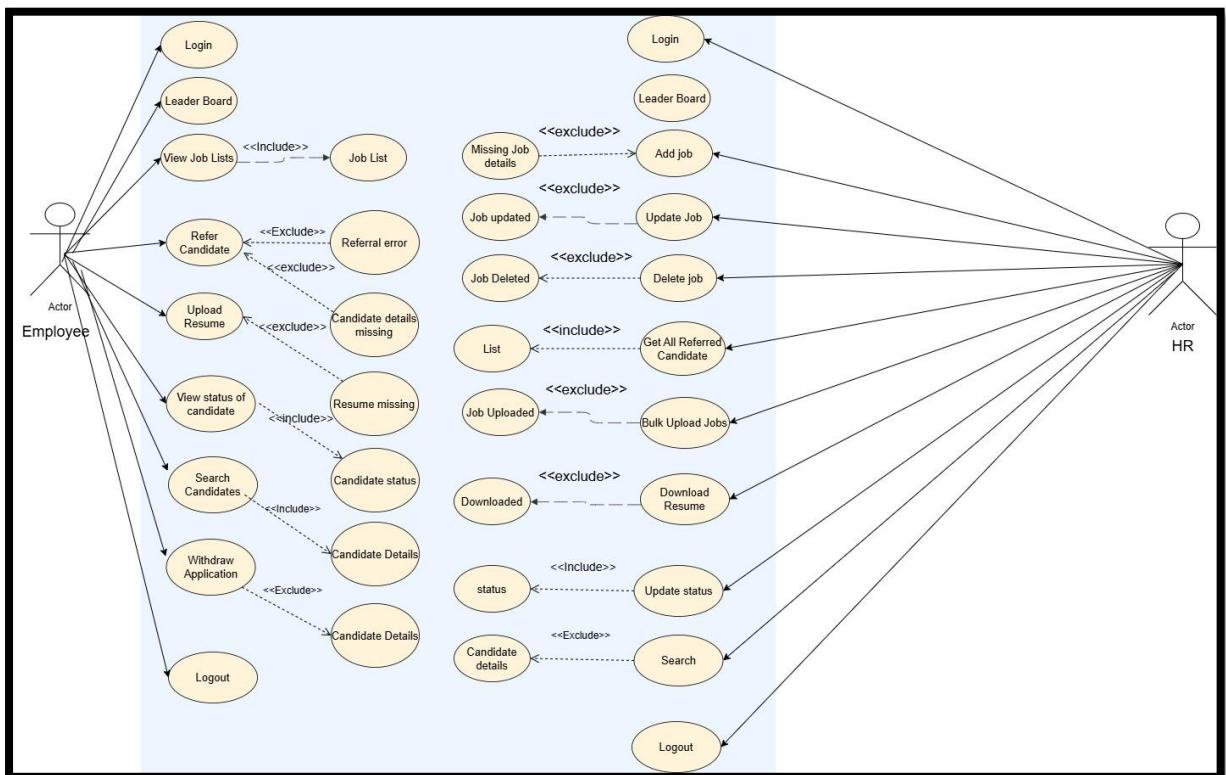


Use Case Diagram:

1. Normal Case:



2. Error Case:



c. Test Cases (**unit tests + system (functional) tests**)

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). While unit testing focuses on the components that have been implemented in phase 3, system testing should consider **the entire project** (components that have been implemented through all three phases).

The screenshot shows two panels of a file explorer. The left panel displays the main source code structure under 'jobreferralportal_backend [boot]'. It includes packages like com.innovators.jobreferralportal, com.innovators.jobreferralportal.Config, com.innovators.jobreferralportal.controller, com.innovators.jobreferralportal.entity, com.innovators.jobreferralportal.enums, com.innovators.jobreferralportal.repository, and com.innovators.jobreferralportal.Service. The 'Service' package contains a 'CustomUserDetailsService.java' file with methods 'CustomUserDetailsService' (containing 'logger' and 'employeeRepo') and 'loadUserByUsername(String)' which returns 'UserDetails'. The right panel shows the test directory 'src/test/java'. It includes packages com.innovators.jobreferralportal and com.innovators.jobreferralportal.Controller. Under 'com.innovators.jobreferralportal.Service', there is a 'CustomUserDetailsServiceTest.java' file containing test methods: 'testLoadUserByUsername_UserFound()' and 'testLoadUserByUsername_UserNotFound()'. The 'CustomUserDetailsServiceTest' class has fields 'customUserDetailsService', 'employeeRepo', and methods 'loadUserByUsername(String)'.

Test cases for the APIs in this phase:

The CustomUserDetailService actually holds the code for the method `loadUserByname()` which is already a existing method in the spring security framework. This method loads all the user data based on the username given as parameter. It returns the user's username, password and the authorities allocated to the user.

For this method we have used a positive and negative test case to define how it works in successful case and how it behaves in the negative scenario.

testLoadUserByUsername_UserFound ():

This is actually the positive case test case where we are mocking the method `findByUsername()` which is called when we call the `loadUserByUsername()` method using the current injected mock object of the current class .And we are verifying if we have received the same object ,using the `assertEquals()` method .

```

@Test
public void testLoadUserByUsername_UserFound() {
    String username = "dirk";
    Employee mockEmployee = new Employee();
    mockEmployee.setUsername(username);
    mockEmployee.setPassword("password123");
    mockEmployee.setRole(RoleEnum.EMPLOYEE);
    when(employeeRepo.findByUsername(username)).thenReturn(Optional.of(mockEmployee));

    UserDetails userDetails = customUserDetailsService.loadUserByUsername(username);
    assertNotNull(userDetails);
    assertEquals(username, userDetails.getUsername());
    assertEquals("password123", userDetails.getPassword());
}

```

testLoadUserByUsername_UserNotFound ():

In the negative scenario according to the actual implementation code it should raise the UserNameNotFoundException when the given username is not available in the database. So to mock the input we have taking a random user “something” as the input and and using the when statement we are indicating it to return an empty object as output when the repo is called up for the username search. Using the assertThrows we are verifying if we are getting the exception as expected.

```

@Test
public void testLoadUserByUsername_UserNotFound() {
    String username = "something";

    when(employeeRepo.findByUsername(username)).thenReturn(Optional.empty());

    assertThrows(UserNameNotFoundException.class, () -> {
        customUserDetailsService.loadUserByUsername(username);
    });
}

```



For this deliverable we have added 3 new methods in the class.

a) *deleteReferral (Long):*

This method is created to implement the functionality withdraw application.

testDeleteReferral_Success():

```
@Test  
public void testDeleteReferral_Success() {  
    Long referralId = 1L;  
    EmployeeServiceImpl.deleteReferral(referralId);  
  
    verify(referredCandidateRepo, times(1)).deleteById(referralId);  
}
```

According to the code implemented whenever we find a row in the database with the referral given it should delete it from the database so the database will be called at least once. So, to test the positive case we have called the method with the injected class and we are verifying if the database is called at least once using the verify () method.

testDeleteReferral_Fail_NotFound():

```
@Test  
public void testDeleteReferral_Fail_NotFound() {  
    Long referralId = 999L;  
    doThrow(new EmptyResultDataAccessException(1)).when(referredCandidateRepo).deleteById(referralId);  
    try {  
        EmployeeServiceImpl.deleteReferral(referralId);  
    } catch (EmptyResultDataAccessException ex) {  
        assertNotNull(ex);  
    }  
  
    verify(referredCandidateRepo, times(1)).deleteById(referralId);  
}
```

In the negative scenario when there is no id found for the given Id it is supposed to raise the exception and to validate the exception, we have incorporated the try catch block in the negative scenario.

b) *getAllReferredCandidatesSearch():*

This method is coded to search the jobs based on the name given in the search bar and employeeId. Whenever it finds the name in the db it outputs the list of employees which matches the id and the name. The name is matching in case Sensitive manner.

test GetAllReferredCandidatesSearch_FoundCandidates ():

```
@Test
public void test GetAllReferredCandidatesSearch_FoundCandidates() {
    Long employeeId = 1L;
    String name = "ramya";

    ReferredCandidate candidate1 = new ReferredCandidate();
    candidate1.setFirstName("Ramya");

    ReferredCandidate candidate2 = new ReferredCandidate();
    candidate2.setFirstName("Sravan");

    List<ReferredCandidate> expectedCandidates = Arrays.asList(candidate1, candidate2);

    when(referredCandidateRepo.findByReferredByAndFirstNameContainingIgnoreCase(employeeId, name.toLowerCase()))
        .thenReturn(expectedCandidates);

    List<ReferredCandidate> actualCandidates = EmployeeServiceImpl.getAllReferredCandidatesSearch(employeeId, name);

    assertEquals(2, actualCandidates.size());
    assertTrue(actualCandidates.stream().anyMatch(c -> c.getFirstName().equals("Ramya")));
    assertTrue(actualCandidates.stream().anyMatch(c -> c.getFirstName().equals("Sravan")));

    verify(referredCandidateRepo, times(1)).findByReferredByAndFirstNameContainingIgnoreCase(employeeId,
        name.toLowerCase());
}
```

This is the positive test case where we are creating a dummy employeeid and name and asking the when statement to return the expected candidates as the output as its just the test case we are returning 2 different users which do not match with the name Ramya .Next we are actually calling the method being tested which returns the then block in when() statement and we are verifying if we got the same using the assertEquals() and the assertTrue() methods.

test GetAllReferredCandidatesSearch_NoCandidatesFound():

This is the negative test case written for the method mentioned and, in this usage, the when statement we are returning an empty list as output indicating we do not have any users for the match. When we do not have any match it is okay and we can quit with no issues so we are just verifying if the database is at least called once during the execution of the program using the verify () method.

```
@Test
public void test GetAllReferredCandidatesSearch_NoCandidatesFound() {
    Long employeeId = 1L;
    String name = "Ramya";
    when(referredCandidateRepo.findByReferredByAndFirstNameContainingIgnoreCase(employeeId, name.toLowerCase()))
        .thenReturn(Collections.emptyList());

    List<ReferredCandidate> actualCandidates = EmployeeServiceImpl.getAllReferredCandidatesSearch(employeeId, name);

    assertTrue(actualCandidates.isEmpty());

    verify(referredCandidateRepo, times(1)).findByReferredByAndFirstNameContainingIgnoreCase(employeeId,
        name.toLowerCase());
}
```

c) *getLeaderBoardList()*:

This method is actually used for the dashboard implementation at the UI end. In this method we are actually getting the scores associated with the employee and sorting them to create the dashboard. We have both positive and negative test cases for this method.

testGetLeaderBoardList_ValidEmployees ():

In the positive scenario we are actually creating few dummy employees and then we are using the then method to return these employees when the findAll() method is called. Next we are calling the method being implemented using the inject mock object which in turn calls the findAll() and returns the employee list. And finally we check if the returned users and the expected users are the same using the assertEquals() methods.

```
public void testGetLeaderBoardList_ValidEmployees() {  
    Employee emp1 = new Employee();  
    emp1.setEmployeeID(1L);  
    emp1.setFirstName("Ramya");  
    emp1.setLastName("Madhavareddy");  
    emp1.setScore(10);  
  
    Employee emp2 = new Employee();  
    emp2.setEmployeeID(2L);  
    emp2.setFirstName("Rishika");  
    emp2.setLastName("Dudipala");  
    emp2.setScore(5);  
  
    Employee emp3 = new Employee();  
    emp3.setEmployeeID(3L);  
    emp3.setFirstName("Sravan");  
    emp3.setLastName("Nadipalli");  
    emp3.setScore(8);  
  
    employeeList.add(emp1);  
    employeeList.add(emp2);  
    employeeList.add(emp3);  
  
    when(employeeRepo.findAll()).thenReturn(employeeList);  
    List<List<String>> leaderBoard = EmployeeServiceImpl.getLeaderBoardList();  
  
    assertEquals(3, leaderBoard.size());  
    assertEquals("1", leaderBoard.get(0).get(0));  
    assertEquals("Madhavareddy,Ramya", leaderBoard.get(0).get(1));  
    assertEquals("10", leaderBoard.get(0).get(2));  
  
    assertEquals("3", leaderBoard.get(1).get(0));  
    assertEquals("Nadipalli,Sravan", leaderBoard.get(1).get(1));  
    assertEquals("8", leaderBoard.get(1).get(2));  
  
    assertEquals("2", leaderBoard.get(2).get(0));  
    assertEquals("Dudipala,Rishika", leaderBoard.get(2).get(1));
```

testGetLeaderBoardList_ScoreLessThanOne():

This is the negative test case for the method in which we are verifying if we got an employee with no score associated with it then our leaderboard would be empty as they do not have score.

So, we have created an employee with the name and the score where the score is 0. And we then called the method being tested which should have not added any user to the leaderboard. So, to verify that we have used the assertTrue() which validates if the leaderboard is empty.

```
@Test
public void testGetLeaderBoardList_ScoreLessThanOne() {

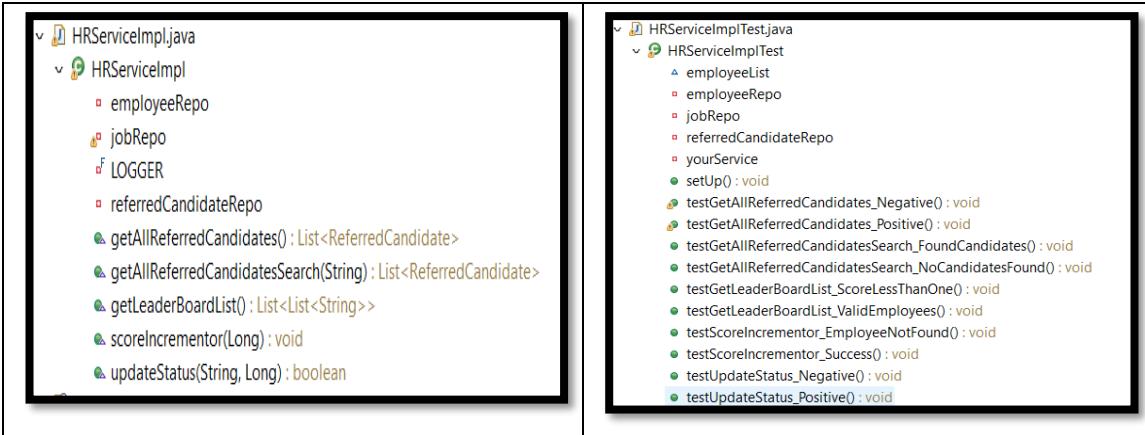
    Employee emp1 = new Employee();
    emp1.setEmployeeID(1L);
    emp1.setFirstName("John");
    emp1.setLastName("Doe");
    emp1.setScore(0);

    employeeList.add(emp1);

    when(employeeRepo.findAll()).thenReturn(employeeList);

    List<List<String>> leaderBoard = EmployeeServiceImpl.getLeaderBoardList();

    assertTrue(leaderBoard.isEmpty(), "Leaderboard should be empty when score is less than 1");
}
```



For this deliverable we have 3 new methods in the class to implement the dashboard for HR.

- a) getAllReferredCandidatesSearch (String)
- b) getLeaderBoardList ()
- c) ScoreIncrementer (Long)

The A&B methods work the same as the employeeServiceImpl and the same kind of test case is implemented. For the ScoreIncrementer () method the testcase are as below.

The scoreincrementer is for increasing the score of the employee who referred others to the jobs. Based on the number of referrals they add in the system their score gets increased. This is an internal logic for the dashboard implementation.

testScoreIncrementor_Success ():

In the success scenario we will get all the referrals of the employee, and his score get increased. We have created 2 ids, and we have asked the repo to return the dummy employee created when we ask for the person who referred. Then we call the method being tested and then we are checking if the returned candidate is same, and we also verify if the database is at least called once when we are performing this.

```
@Test
public void testScoreIncrementor_Success() {
    Long referralId = 1L;
    Long referredById = 2L;

    ReferredCandidate referredCandidate = new ReferredCandidate();
    referredCandidate.setReferredBy(referredById);

    Employee employee = new Employee();
    employee.setEmployeeID(referredById);
    employee.setScore(10);
    when(referredCandidateRepo.getReferenceById(referralId)).thenReturn(referredCandidate);
    when(employeeRepo.getReferenceById(referredById)).thenReturn(employee);

    yourService.scoreIncrementor(referralId);
    assertEquals(11, employee.getScore());

    verify(referredCandidateRepo, times(1)).getReferenceById(referralId);
    verify(employeeRepo, times(1)).getReferenceById(referredById);
}
```

testScoreIncrementor_EmployeeNotFound():

When there are no employees found for the id, we are expected to throw the exception so that no score gets incremented. As there are no employees found for the referred person, we do not increase any score for any employee. We just make sure we are calling the db. in the negative scenario too in this test case.

```
@Test
public void testScoreIncrementor_EmployeeNotFound() {

    Long referralId = 1L;

    ReferredCandidate referredCandidate = new ReferredCandidate();
    referredCandidate.setReferredBy(2L);

    when(referredCandidateRepo.getReferenceById(referralId)).thenReturn(referredCandidate);
    when(employeeRepo.getReferenceById(2L)).thenThrow(new EntityNotFoundException("Employee not found"));

    assertThrows(EntityNotFoundException.class, () -> {
        yourService.scoreIncrementor(referralId);
    });

    verify(referredCandidateRepo, times(1)).getReferenceById(referralId);
    verify(employeeRepo, times(1)).getReferenceById(2L);
}
```

Functional testing:

In the functionality testing we test the entire system if it's performing the expected operations with no deviation in it and, we test if the total system is working properly, and flow is maintained with no discrepancy in it.

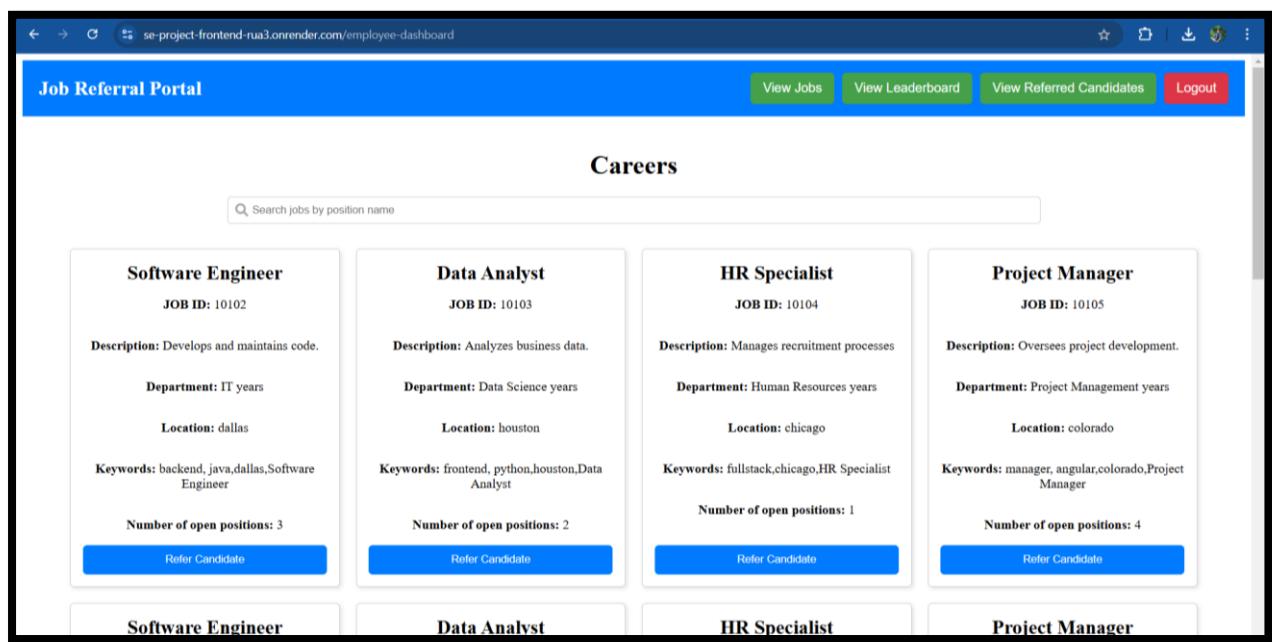
Functional Test 1:

In this we are verifying if the user login is performed correctly with no issues in it.

So first let us test the positive scenario where when we enter a valid username it should allow the login.

I am giving a valid username james and password as James@2000 as it is a valid username it should login.

Result:



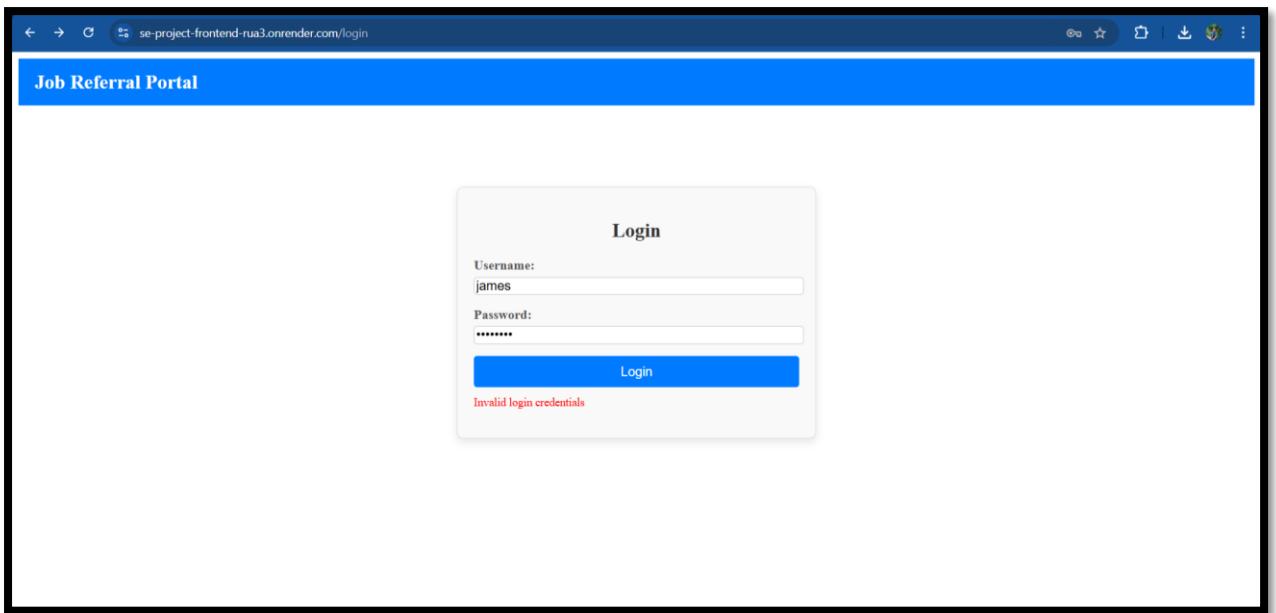
The screenshot shows a web browser window for the 'Job Referral Portal' at the URL 'se-project-frontend-nua3.onrender.com/employee-dashboard'. The page has a blue header bar with the portal name and navigation links for 'View Jobs', 'View Leaderboard', 'View Referred Candidates', and 'Logout'. Below the header is a search bar labeled 'Search jobs by position name'. The main content area is titled 'Careers' and displays four job listings in cards:

- Software Engineer** (JOB ID: 10102)
 - Description: Develops and maintains code.
 - Department: IT years
 - Location: dallas
 - Keywords: backend, java,dallas,Software Engineer
 - Number of open positions: 3
- Data Analyst** (JOB ID: 10103)
 - Description: Analyzes business data.
 - Department: Data Science years
 - Location: houston
 - Keywords: frontend, python,houston,Data Analyst
 - Number of open positions: 2
- HR Specialist** (JOB ID: 10104)
 - Description: Manages recruitment processes
 - Department: Human Resources years
 - Location: chicago
 - Keywords: fullstack,chicago,HR Specialist
 - Number of open positions: 1
- Project Manager** (JOB ID: 10105)
 - Description: Oversees project development.
 - Department: Project Management years
 - Location: colorado
 - Keywords: manager, angular,colorado,Project Manager
 - Number of open positions: 4

At the bottom of each card is a blue 'Refer Candidate' button. Below the cards is a horizontal row with buttons for 'Software Engineer', 'Data Analyst', 'HR Specialist', and 'Project Manager'.

So, the valid username case passed.

Let us now test the invalid username.



So, the login functionality is working as expected here.

Functionality Test 2:

Now let us make a test to add a file which is used to bulk upload numerous jobs in it.

In the positive scenario we are considering an excel file as input with one job in it and uploading it.

Job ID	Position	Description	Department	Location	Keywords	Open Positions	
10052	Software Engineer	Develops and maintains code.	IT	dallas	backend, java	5	Edit Delete
10053	Data Analyst	Analyzes business data.	Data Science	houston	frontend, python	2	Edit Delete
10054	HR Specialist	Manages recruitment processes	Human Resources	chicago	fullstack	1	Edit Delete

We can upload a file successfully.

Now let us try to upload a doc file and see if it is allowing doc files.

Please upload a CSV file.

Add Job Bulk Upload Jobs Search jobs by position name

Bulk Upload Jobs

Choose File Section G.docx

Selected file: Section G.docx

Upload Cancel

Job ID	Position	Description	Department	Location	Keywords	Open Positions	
10052	Software Engineer	Develops and maintains code.	IT	dallas	backend, java	5	<button>Edit</button> <button>Delete</button>
10053	Data Analyst	Analyzes business data.	Data Science	houston	frontend, python	2	<button>Edit</button> <button>Delete</button>
10054	HR Specialist	Manages recruitment processes	Human Resources	chicago	fullstack	1	<button>Edit</button> <button>Delete</button>

So, our upload function is working properly as expected.

Functionality Test 3:

Now we are trying to edit the job and see if the job is edited successfully.

Please upload a CSV file.

Add Job Bulk Upload Jobs Search jobs by position name

Bulk Upload Jobs

Choose File Section G.docx

Selected file: Section G.docx

Upload Cancel

Edit Job

10052	Software Engineer	Develops and maintain	IT	5	dallas	backend, java
-------	-------------------	-----------------------	----	---	--------	---------------

Save Cancel

Job ID	Position	Description	Department	Location	Keywords	Open Positions	
10052	Software Engineer	Develops and maintains code.	IT	dallas	backend, java	5	<button>Edit</button> <button>Delete</button>
10053	Data Analyst	Analyzes business data.	Data Science	houston	frontend, python	2	<button>Edit</button> <button>Delete</button>

We have tried to change the technologies in the above job as backend, c++ and let us see if it changed successfully.

The screenshot shows a web application interface for bulk job upload. At the top, there's a message: "Please upload a CSV file." Below it are two green buttons: "Add Job" and "Bulk Upload Jobs". To the right is a search bar with the placeholder "Search jobs by position name".

The main area is titled "Bulk Upload Jobs". It displays a table with the following data:

Job ID	Position	Description	Department	Location	Keywords	Open Positions	
10052	Software Engineer	Develops and maintains code.	IT	dallas	backend, C++	5	<button>Edit</button> <button>Delete</button>
10053	Data Analyst	Analyzes business data.	Data Science	houston	frontend, python	2	<button>Edit</button> <button>Delete</button>
10054	HR Specialist	Manages recruitment processes	Human Resources	chicago	fullstack	1	<button>Edit</button> <button>Delete</button>
10055	Project Manager	Oversees project development.	Project Management	colorado	manager, angular	4	<button>Edit</button> <button>Delete</button>

From the above screenshot it is visible that the job got edited successfully.

Functional Test 4:

Now let us check if we can go to the leader board and see employees in it with the score calculated to them. We have clicked on the view leader board option in the screen and are able to see the dashboard below.

The screenshot shows a web application interface for the leader board. At the top, there's a blue header bar with the title "Job Referral Portal" and several navigation buttons: "Add User", "View Jobs", "View Leaderboard" (which is highlighted in blue), "View Referred Candidates", and "Logout".

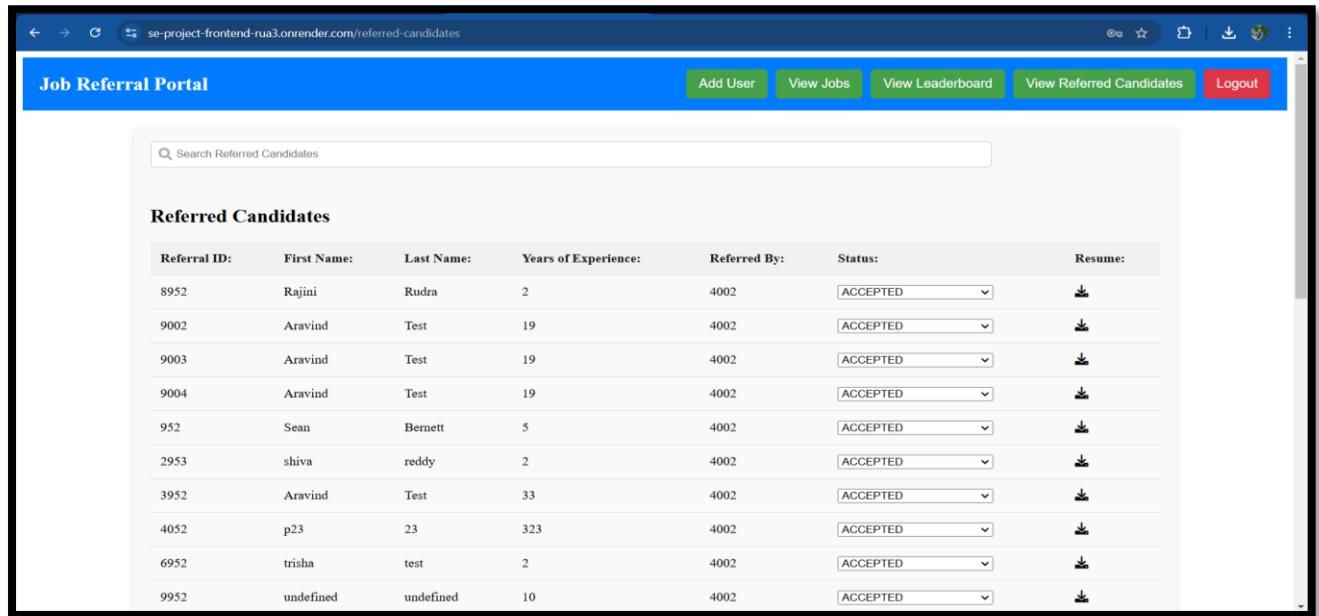
The main area is titled "Leaderboard" and displays a table with the following data:

Rank	Employee ID	Name	Score
1	4002	Simpson,James	23
2	13104	Reddy,Shiva	7

Hence the leaderboard is working for the HR page.

Functionality Test 5:

In this we are checking if we can view the existing jobs as HR. We have clicked on the view jobs option as we could see from the screen below, which makes it evident that this functionality is working properly.

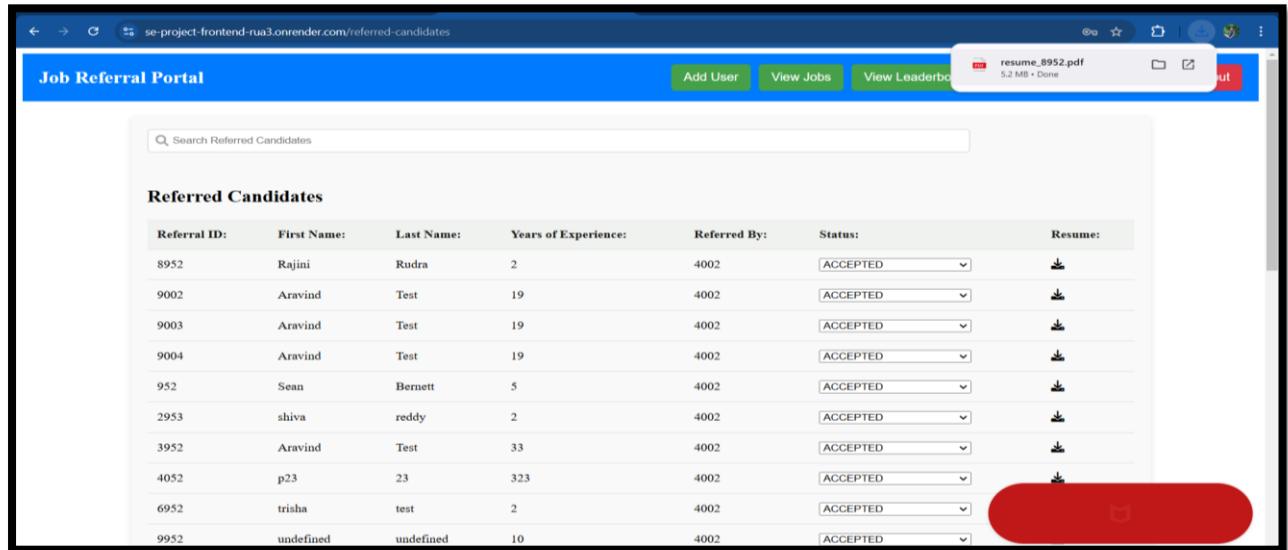


The screenshot shows a web browser window for the "Job Referral Portal". The URL is "se-project-frontend-rua3.onrender.com/referred-candidates". The page has a blue header bar with the portal name and several navigation buttons: "Add User", "View Jobs", "View Leaderboard", "View Referred Candidates", and "Logout". Below the header is a search bar with the placeholder "Search Referred Candidates". The main content area is titled "Referred Candidates" and contains a table with the following data:

Referral ID:	First Name:	Last Name:	Years of Experience:	Referred By:	Status:	Resume:
8952	Rajini	Rudra	2	4002	ACCEPTED	
9002	Aravind	Test	19	4002	ACCEPTED	
9003	Aravind	Test	19	4002	ACCEPTED	
9004	Aravind	Test	19	4002	ACCEPTED	
952	Sean	Bernett	5	4002	ACCEPTED	
2953	shiva	reddy	2	4002	ACCEPTED	
3952	Aravind	Test	33	4002	ACCEPTED	
4052	p23	23	323	4002	ACCEPTED	
6952	trisha	test	2	4002	ACCEPTED	
9952	undefined	undefined	10	4002	ACCEPTED	

Functionality Test 6:

Now we are testing the download resume feature so whenever I click on the download option it should start the resume download automatically.



The above screen has the resume downloading in our system. Hence the functionality is implemented successfully.

Functionality Test 7:

Now let's test the employee page and the employee functionalities.

In the employee page we can refer a candidate and withdraw the application.

Let us see if we can refer a person using the refer candidate button.

The screenshot shows the 'Employee Dashboard' of the 'Job Referral Portal'. At the top, there are navigation links: 'View Jobs', 'View Leaderboard', 'View Referred Candidates', and 'Logout'. Below the header, the title 'Careers' is displayed. A search bar with the placeholder 'Search jobs by position name' is present. Four job listings are shown in cards:

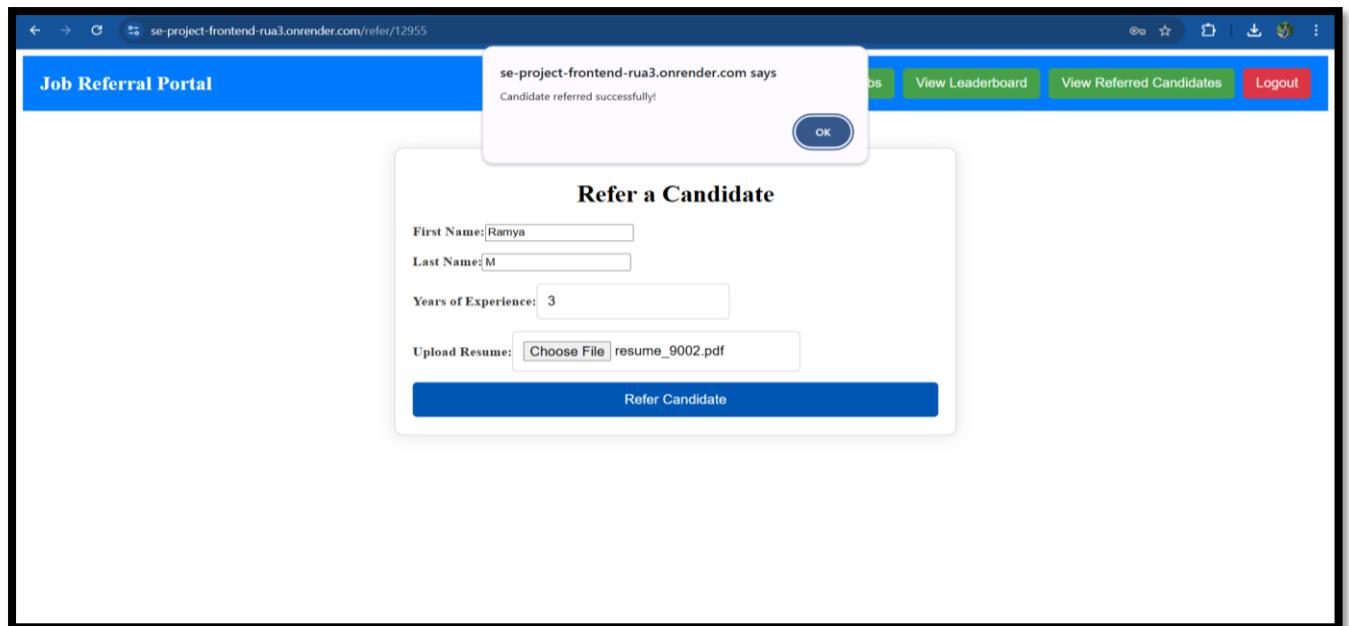
- Data Scientist** (JOB ID: 12953)
Description: Analyze and interpret complex data
Department: Data Science years
Location: San Francisco, CA
Keywords: Machine Learning, AI, Python, San Francisco, CA. Data Scientist
Number of open positions: 3
Refer Candidate button
- Software Engineer** (JOB ID: 12952)
Description: Develop and maintain software systems
Department: Engineering years
Location: New York, NY
Keywords: Java, C++, Python, New York, NY. Software Engineer
Number of open positions: 5
Refer Candidate button
- Product Manager** (JOB ID: 12954)
Description: Manage product lifecycle and roadmap
Department: Product years
Location: Austin, TX
Keywords: Agile, Scrum, Jira, Austin, TX. Product Manager
Number of open positions: 2
Refer Candidate button
- UI/UX Designer** (JOB ID: 12955)
Description: Design user interfaces and experiences
Department: Design years
Location: Seattle, WA
Keywords: Figma, Adobe XD, Prototyping, Seattle, WA. UI/UX Designer
Number of open positions: 4
Refer Candidate button

Below the cards, there are tabs for 'Data Entry', 'Custom Analyst', 'Database Administration', and 'Customer Specialist'.

We are referring to a person to job id 12955. While referring we should submit the below details.

The screenshot shows the 'Refer a Candidate' form for job ID 12955. The form fields are:

- First Name: Ranjana
- Last Name: M
- Years of Experience: 3
- Upload Resume: Choose File resume_9002.pdf
- Refer Candidate button

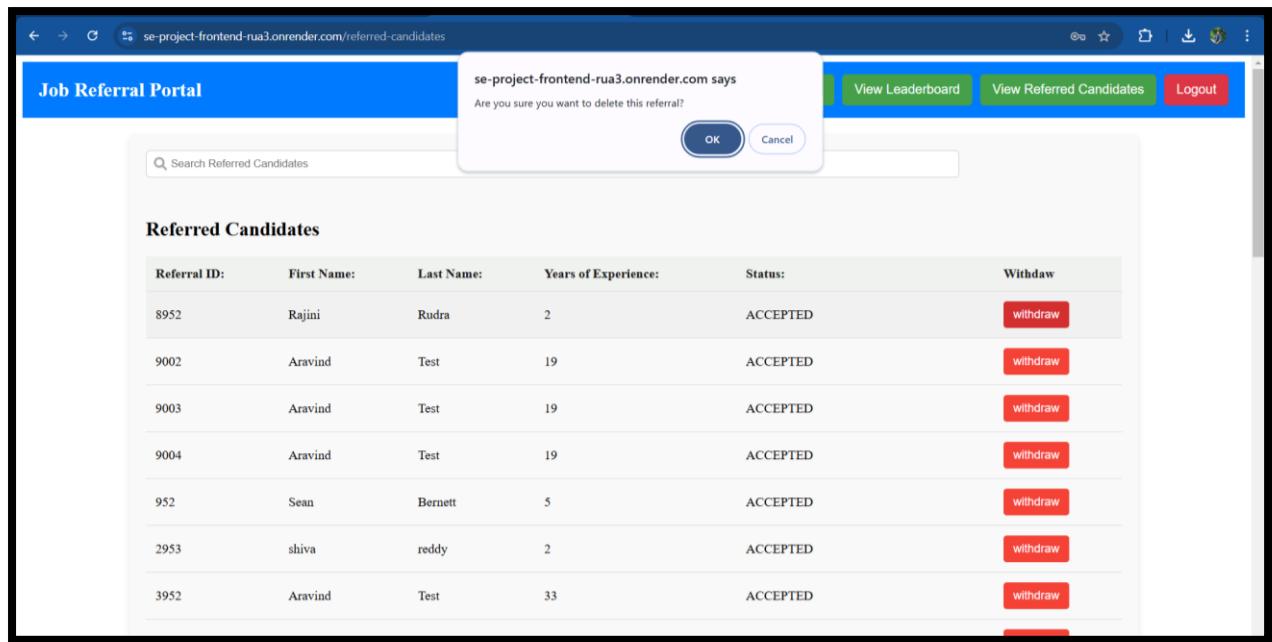


The person is referred successfully now. Hence the functionality is working as expected.

Functional Test 8:

We are checking here to see if we can withdraw the application for the Referral using the withdraw option available in the below screen.

Referred Candidates					
Referral ID:	First Name:	Last Name:	Years of Experience:	Status:	Withdraw
8952	Rajini	Rudra	2	ACCEPTED	<button>withdraw</button>
9002	Aravind	Test	19	ACCEPTED	<button>withdraw</button>
9003	Aravind	Test	19	ACCEPTED	<button>withdraw</button>
9004	Aravind	Test	19	ACCEPTED	<button>withdraw</button>
952	Sean	Bennett	5	ACCEPTED	<button>withdraw</button>
2953	shiva	reddy	2	ACCEPTED	<button>withdraw</button>
3952	Aravind	Test	33	ACCEPTED	<button>withdraw</button>
4052	p23	23	123	ACCEPTED	<button>withdraw</button>



And hence we have successfully working withdraw feature in our application.

- d. A user manual that tells us how to use your program. This is meant for the end-user of the software. You may include screen shots, where appropriate.
- e. Clear instructions on how to compile/run both your program and your test cases (the program must compile/run).

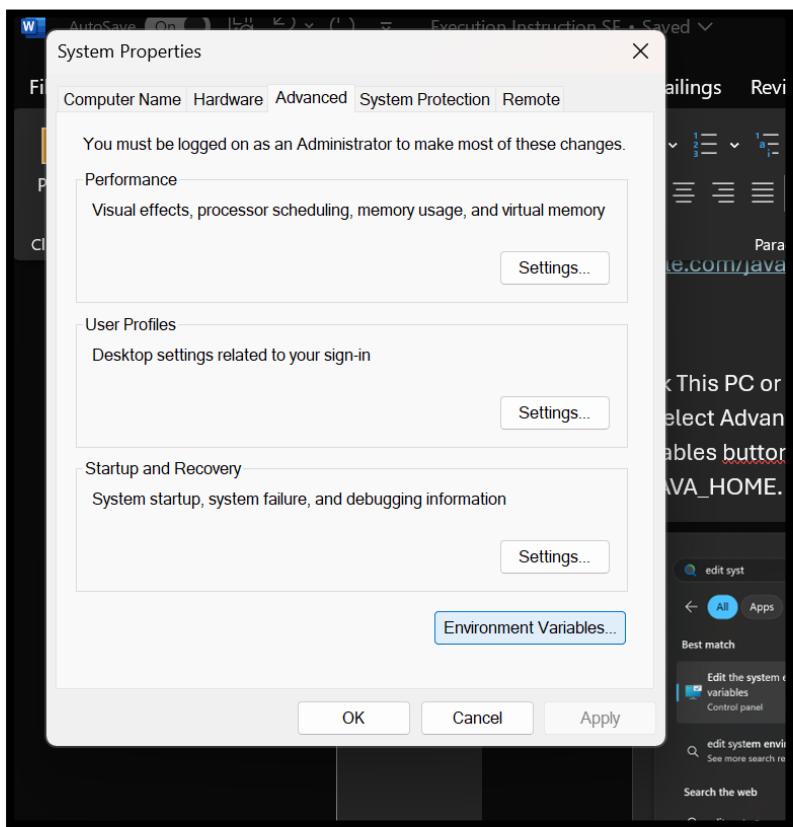
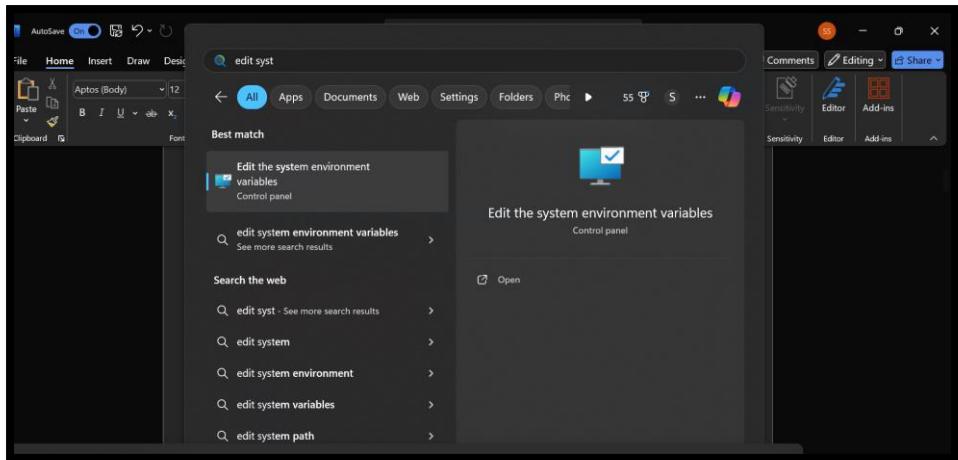
Installing and Running Software:

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.

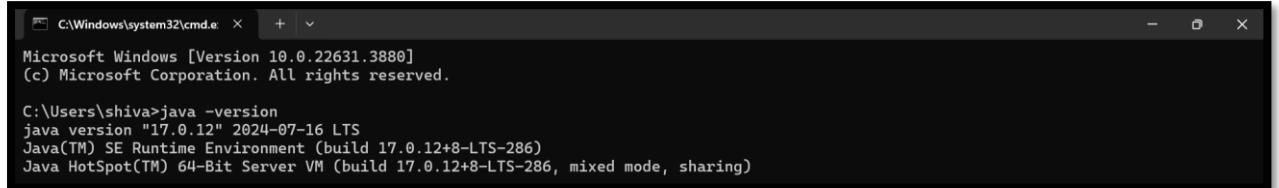


Set the variable value to the path where JDK 17 is installed (e.g., C:\Program Files\Java\jdk-17). Variable name as JAVA_HOME and path to the jdk as variable value, and also add %JAVA_HOME%\bin to the Path system variable.

Restart the command prompt.

Verification (open command prompt)

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



```
C:\Windows\system32\cmd.exe + 
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`

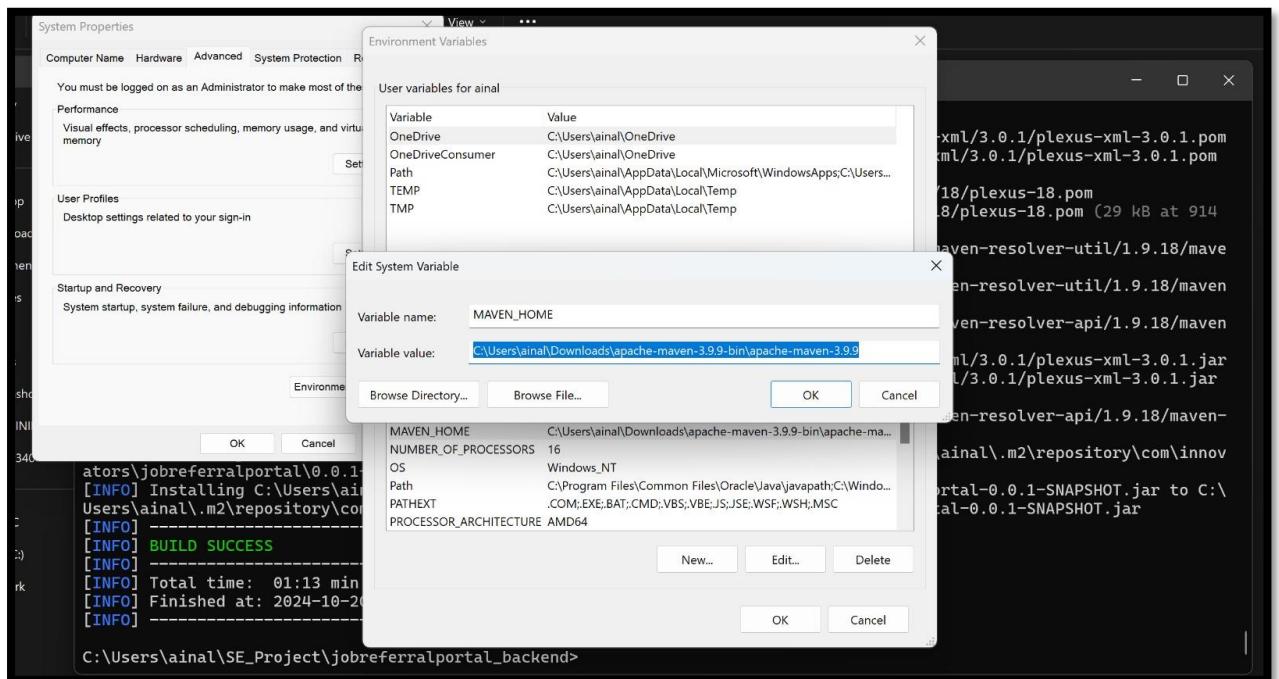


```
C:\Users\shiva>javac -version
javac 17.0.12
C:\Users\shiva>
```

- Step 3) Install maven

<https://maven.apache.org/download.cgi>

similar to JAVA_HOME, create MAVEN_HOME and Add %MAVEN_HOME%\bin to the Path system variable. And restart the command prompt.



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, falling 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 repository 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 PersistenceUnitInfo [name: default]
2024-10-19T18:55:39.753-05:00  INFO 20460 --- [           main] org.hibernate.Version                 : HHH000412: Hibernate ORM core version 6.5.2.Final
2024-10-19T18:55:39.838-05:00  INFO 20460 --- [           main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache 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.JobServiceImplTest
[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 repackaged 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-SNAPSHOT.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\jobreferralportal\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] -----
```

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

```

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 configuration classes for test class [com.innovators.jobreferralportal.JobreferralportalApplicationTests]: JobreferralportalApplicationTests 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.SpringBootTestBootstrapper -- Found @SpringBootConfiguration com.innovators.jobreferralportal.JobreferralportalApplication for test class com.innovators.jobreferralportal.JobreferralportalApplicat

```

```

1024-10-20T18:53:07.694-05:00 INFO 48208 --- [           main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with UserDetailsService bean with name customUserDetailsService
1024-10-20T18:53:08.793-05:00 DEBUG 48208 --- [           main] o.s.s.web.DefaultSecurityFilterChain      : Will secure any request with filters: DisableEncodeUrlFilter, WebAsyncManagerIntegrationFilter, SecurityContextHolderFilter, HeaderWriterFilter, CorsFilter, LogoutFilter, RequestCacheAwareFilter, SecurityContextHolderAwareRequestFilter, AnonymousAuthenticationFilter, SessionManagementFilter, ExceptionTranslationFilter, AuthorizationFilter
1024-10-20T18:53:09.759-05:00 INFO 48208 --- [           main] c.i.j.JobreferralportalApplicationTests  : Started JobreferralportalApplicationTests 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 appended
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 67.39 s -- in com.innovators.jobreferralportal.JobreferralportalApplicationTests
[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.JobServiceImplTest
[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.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: 69ded5ac71cf836 - 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 unused.
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

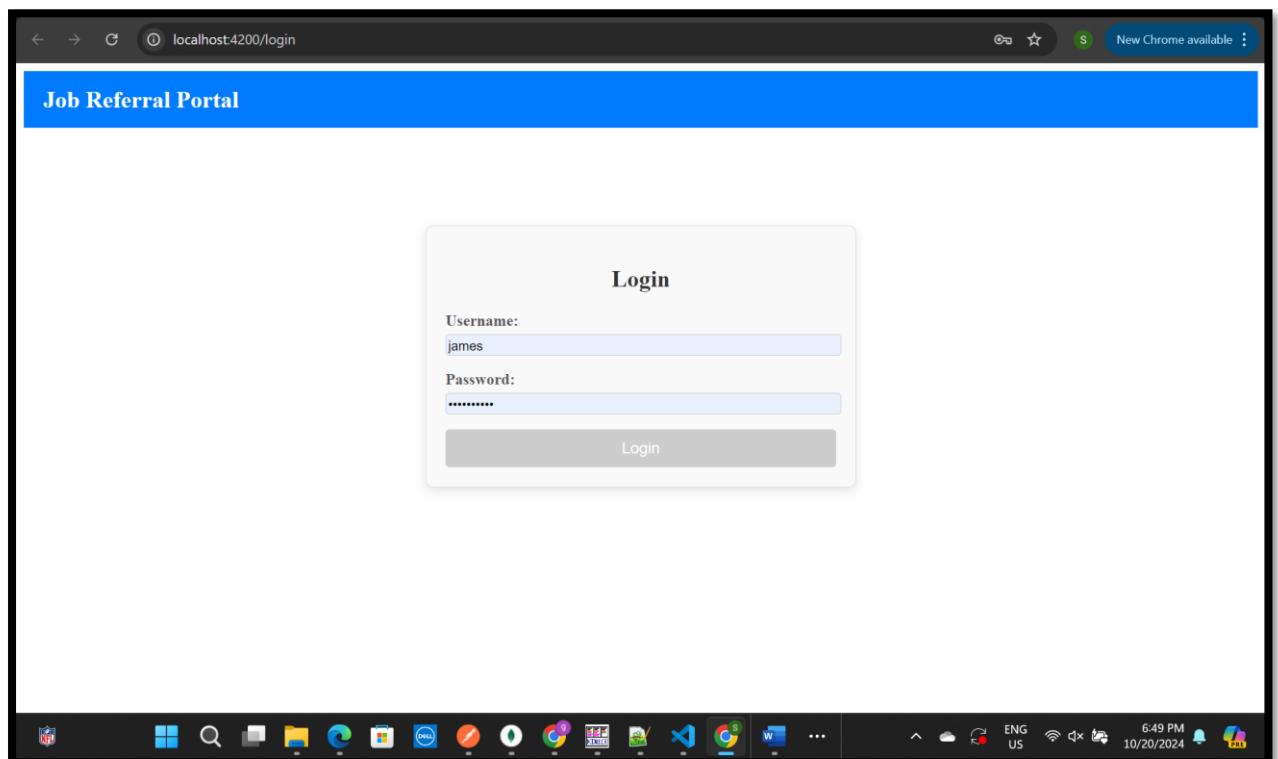
3 unchanged chunks

Build at: 2024-10-20T23:47:08.881Z - Hash: 36c121cc1a19e40 - Time: 306ms

```

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

<http://localhost:4200/>



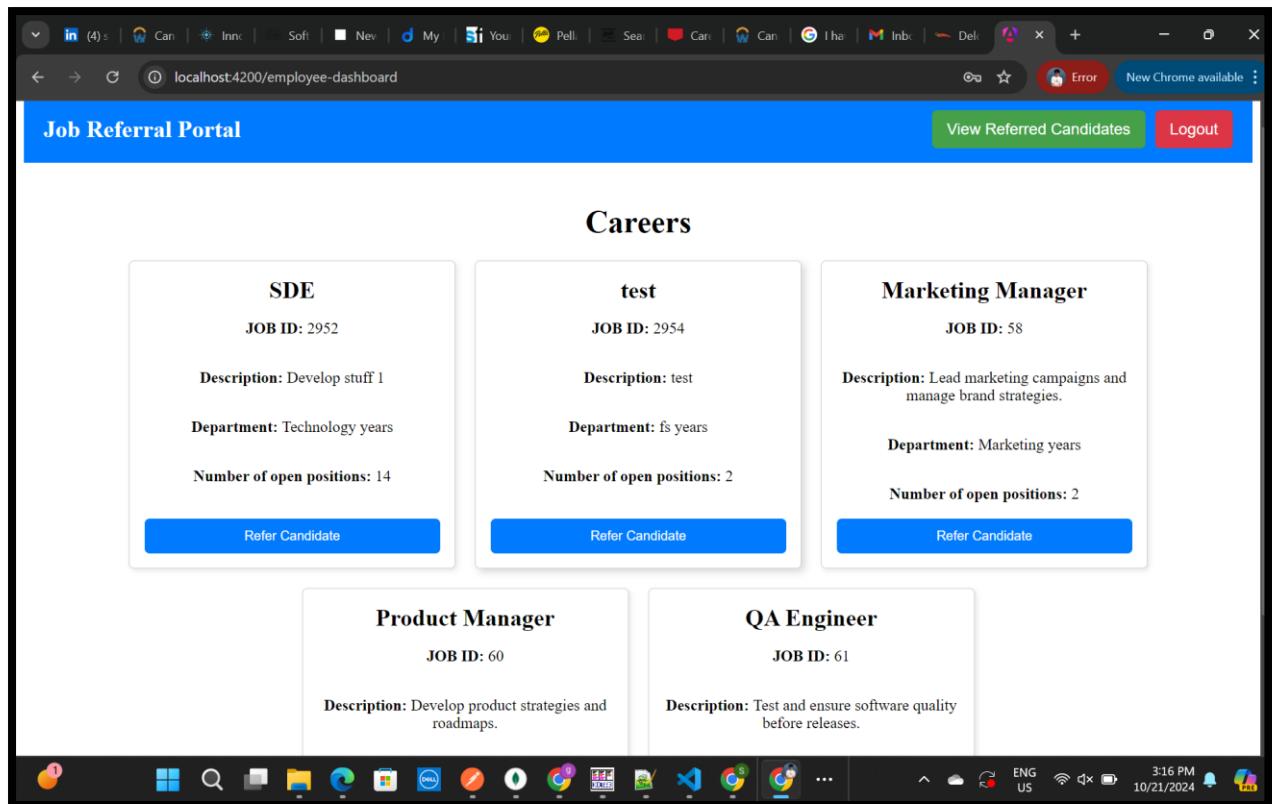
for Employee view : use credentials username : james, password: James@2000

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

The screenshot shows a web browser window for the 'Job Referral Portal' at localhost:4200/hr-dashboard. The main content area displays a table of job listings with columns for Job ID, Position, Description, Product, and Quality Assurance. A modal dialog titled 'Save password?' is overlaid on the page, prompting the user to save their credentials for the current session. The dialog includes fields for 'Username' (dirk) and 'Password' (redacted), with 'Save' and 'Never' buttons. A note at the bottom states: 'Passwords are saved to Google Password Manager on this device.'

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

The screenshot shows a web browser window for the 'Job Referral Portal' at localhost:4200/login. The main content area displays a 'Login' form. The form has two input fields: 'Username' (james) and 'Password' (redacted). Below the fields is a blue 'Login' button.



- f. A section that describes which features you have successfully implemented and any limitations (i.e., any known problems or issues with your program) such as features that you were unable to complete on time including feedback received during peer review sessions. It should include discussions of your plans for the next phase of your project. These are the next features and/or improvements you plan to make to your program (assume that you will continue working on this project next semester).

Features That Have Been Successfully Incorporated

- Leaderboard feature has been implemented in the portal, the working of leaderboard is when an employee refers a candidate, and that candidate gets selected then we increment the score of the employee by 1.
- The HR can update the status(Accepted, Rejected, Next Round, In-Progress) of the referral job application. This feature helps the HR to update the status and the employee can view their candidate referral status
- An employee can search by name from their candidate referral list. This feature is useful to look at the status of their referral candidate when there are many referrals in the list
- For add job feature, where the HR posts a new job, we have added the location and keywords input. These location details and keywords can be used to search for the jobs from the provided open jobs list.

- Similar to the previous point, location and keywords can be uploaded from the bulk upload feature by the HR. A list of jobs can be added using an excel file at once.
- Application withdrawal – An employee can withdraw their referral application at any point. This feature is useful when an employee refers a wrong candidate or position

Next Phase of the project:

- The Leaderboard feature should be reset after every hiring process in a fiscal year, so that the employee with highest score can get reward or bonus every year. Team has thought of implementing it but due to time constraint we couldn't implement. This would be a good scope to work on in the future to enhance the application
- User interface can be improved with more interactive screens, engagement and effectiveness. Clear Menu options, search options can be more improvised.
- Pagination can be implemented to retrieve bulk data to improve the performance of the application.

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

Accomplishments:

- We have successfully gathered requirements, designed, developed, tested, and deployed the “Job Referral Portal” application in production. The application provides features for the HR and employees which makes the hiring and referral process easy.
- HR functionalities like add, update, and delete jobs are implemented. HR can view leaderboard, referral applications, resumes and lot more functionalities have been implemented.
- Employee functionalities like viewing jobs, refer a candidate, search referred candidates, view referral candidate status and lot more functionalities have been implemented.
- Choosing right technologies and infrastructure has made the deployment process smoothly.
- Collaborative teamwork with proper communication helped to reduce the duplicate task and in addition to it prioritizing tasks to avoid dependencies and blockers for other team members helped to complete the deliverables in the given timelines.

Improvements:

- As discussed earlier, the Leaderboard feature can be reset after each hiring phase.
- User interface can be improved with more interactive screens, engagement and effectiveness. Clear Menu options, search options can be more improvised.

- Pagination can be implemented to retrieve bulk data to improve the performance of the application.

- 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-3, reviewing all the documents and works for deliverable 5 and source code of program	13	
Aravind Chitiprolu	Involved in designing UML diagrams of Class, Sequence, Use case diagrams of normal and error cases, Reviewing the documents for Deliverable-5 and Initiated team meetings and involved in PPT, video and user manual.	12	
Shiva Sai Kondapuram	Involved in phase 3 design, unit test cases and requirements designated for development phase 3. Did the deployment of final project and helped the teammates in every phase. Reviewed all the documents for deliverable 5	14.5	

Ravali Kudaravalli	Involved in designing UML diagrams of Use case diagrams, Reviewing the documents for Deliverable-5 and phase 3 and involved in requirements designated for development phase and PPT, Video and User manual.	12	
Ramya Madhavareddy	Involved in testing the unit test cases and functional test cases for the working program including description and PPT, Video and User manual.	12.5	
Sreshta Chityala	Involved in designing UML 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 and also detailed description of requirements for Phase 3	7	

Rajini Vijetha Rudrarapu	Involved in designing sequence diagram and brief review of the project status and also peer review and accomplishments of the project and involved in making PPT.	11.5		
-----------------------------	---	------	--	--