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.

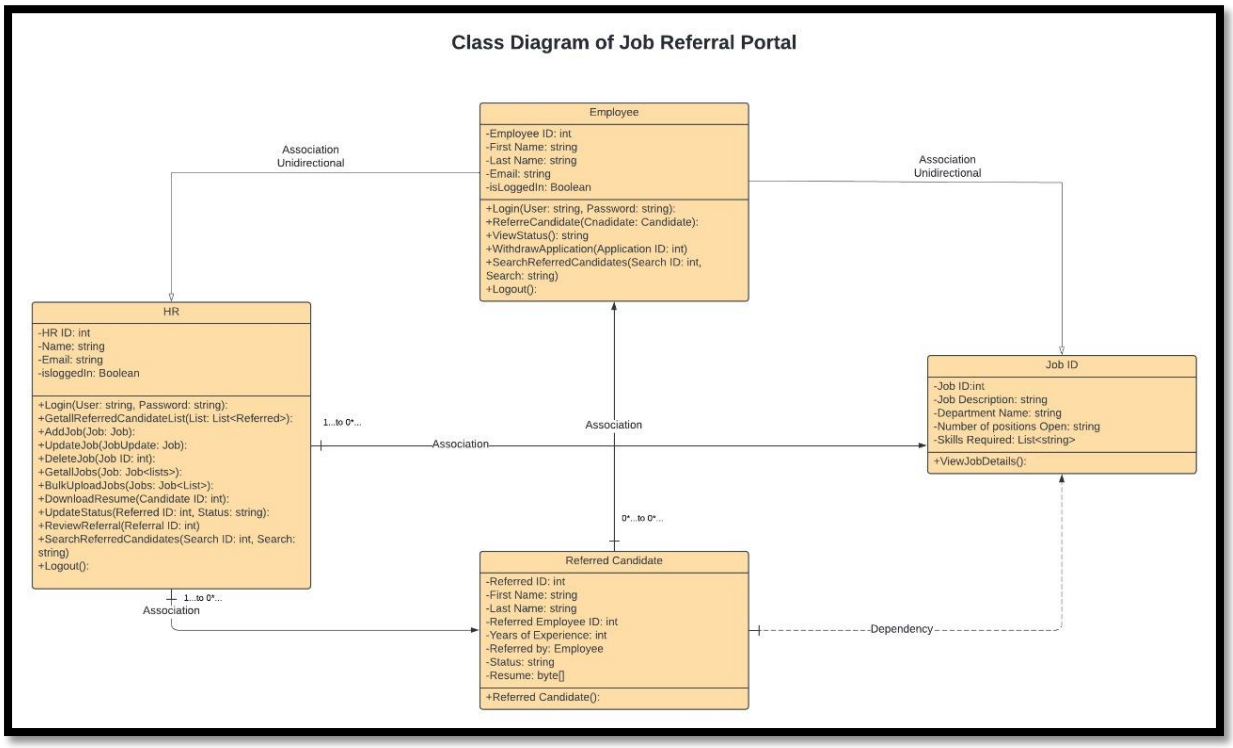
4. Search in Referred Candidates: We also have implemented a search operation in referred candidates for employees and HRs as well, employees can now search the candidates they referred by name, and HRs can search all the referred candidates by their name.

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.

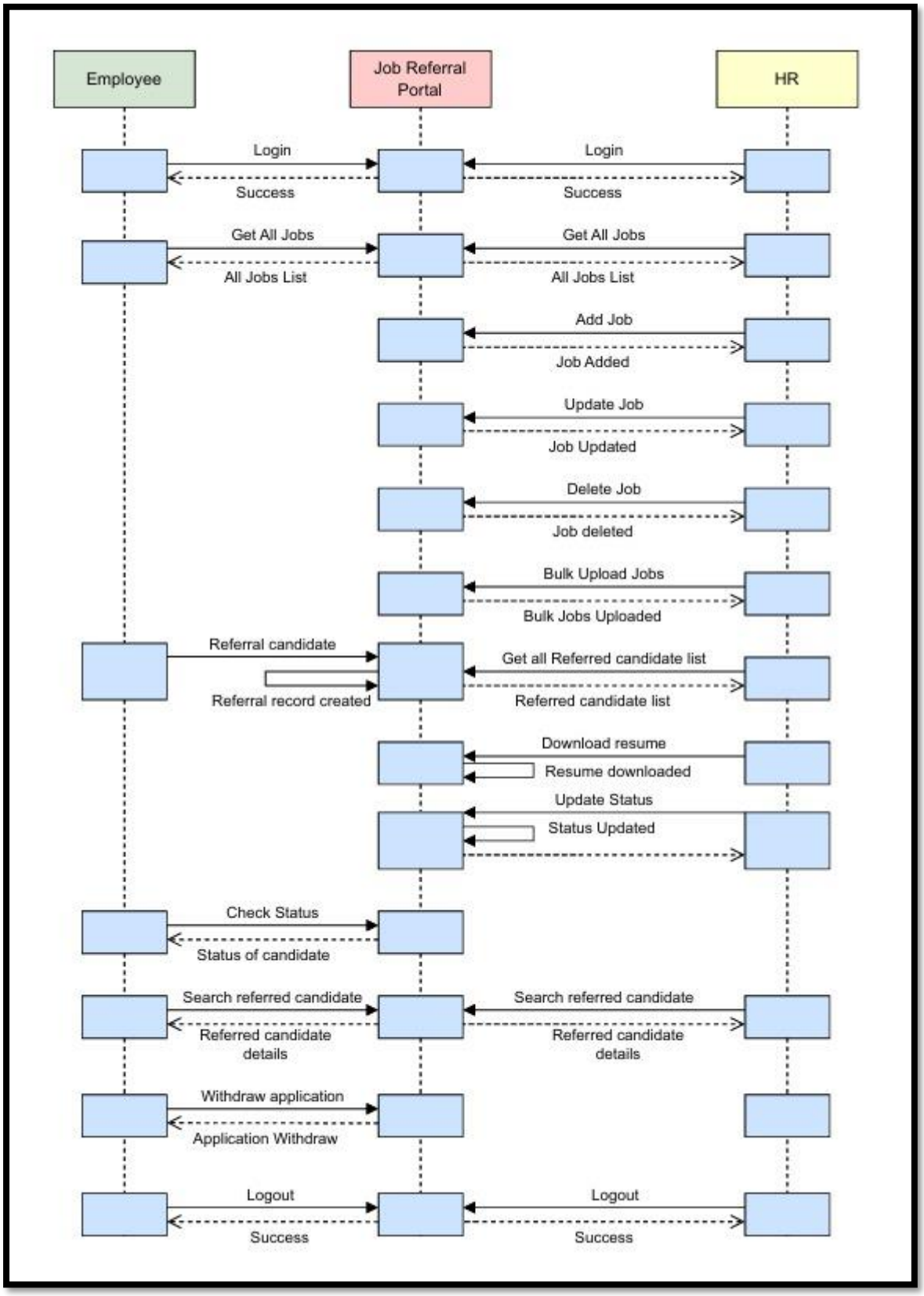
6. Status Update: Initially we had an option for HR to put Custom status for referrals, but we decided to restrict this to few options, like UNDER CONSIDERATION, ONLINE ASSESSMENT etc but finally we also have ACCEPTED and REJECTED, this also core logic for our score incrementor sub method, as whenever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

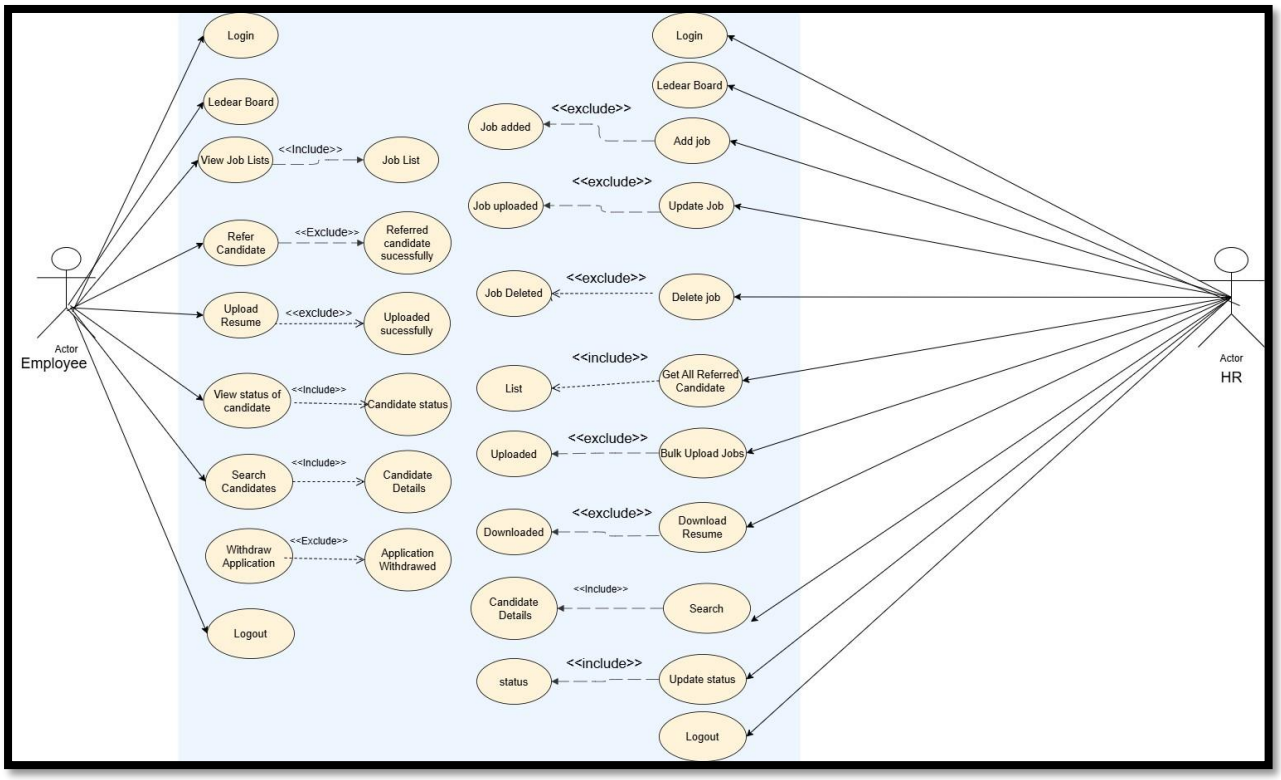


Sequence Diagram:

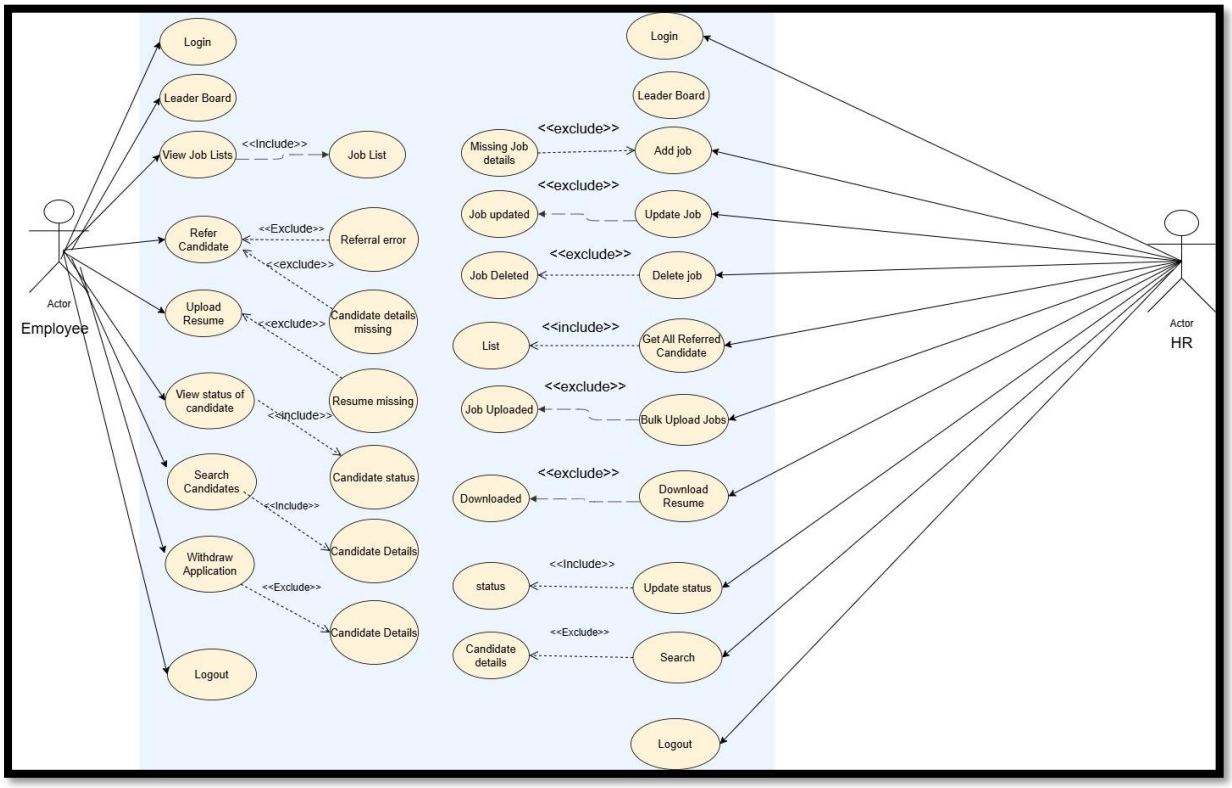


Use Case Diagram:

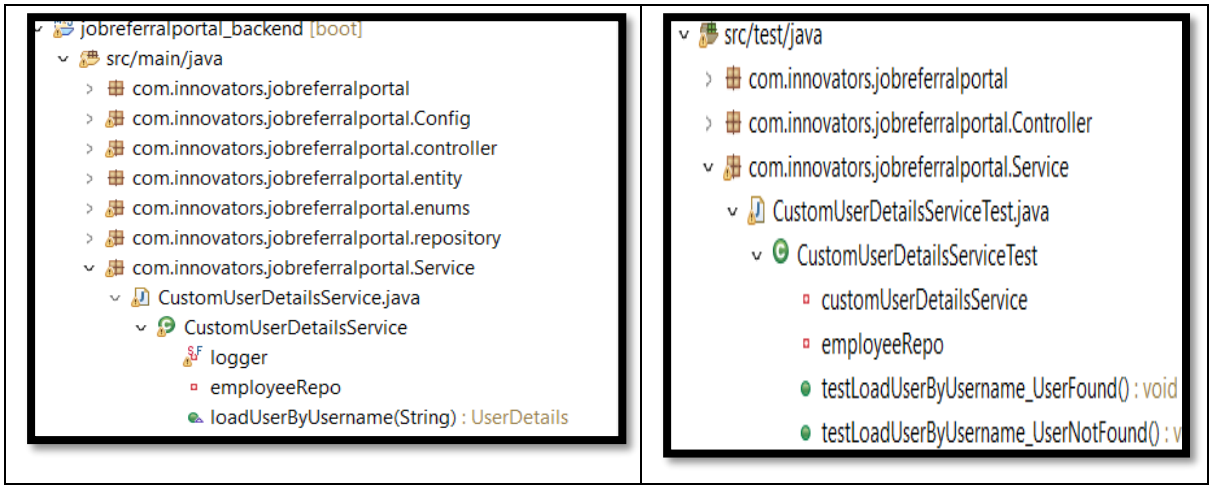
1. Normal Case:



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).



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 ():

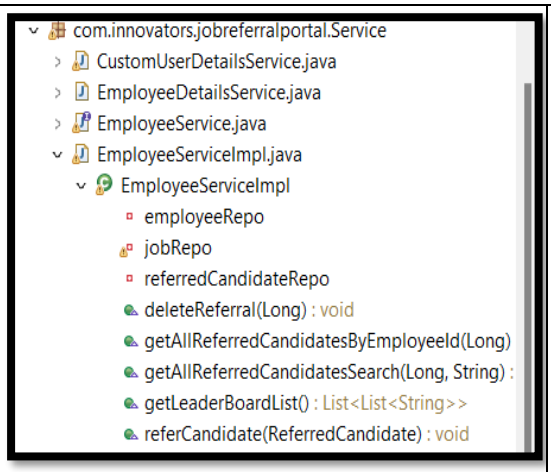
This is actually the positive case test case where we are mocking the method findByUsername() which is called when we call the loadUserByUsername() method using the current injected mock object of the current class .And we are verifying if we have received the same object ,using the assertEquals() method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

```java
@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.

```java
@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.

testGetAllReferredCandidatesSearch_FoundCandidates ():

```java
@Test
public void testGetAllReferredCandidatesSearch_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.

testGetAllReferredCandidatesSearch_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.

```java
@Test
public void testGetAllReferredCandidatesSearch_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.

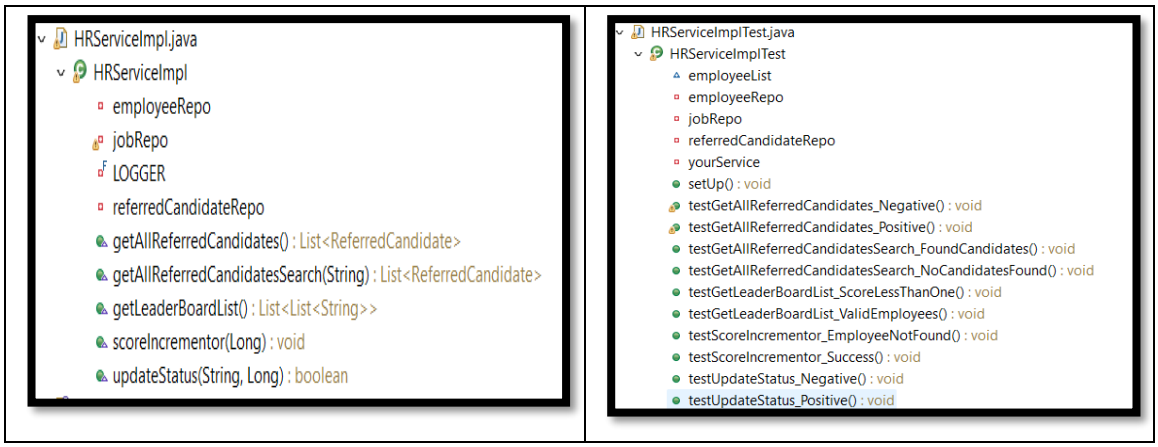
```
public void testGetLeaderBoardList_ValidEmployees() {

    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 to increase any score any 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.

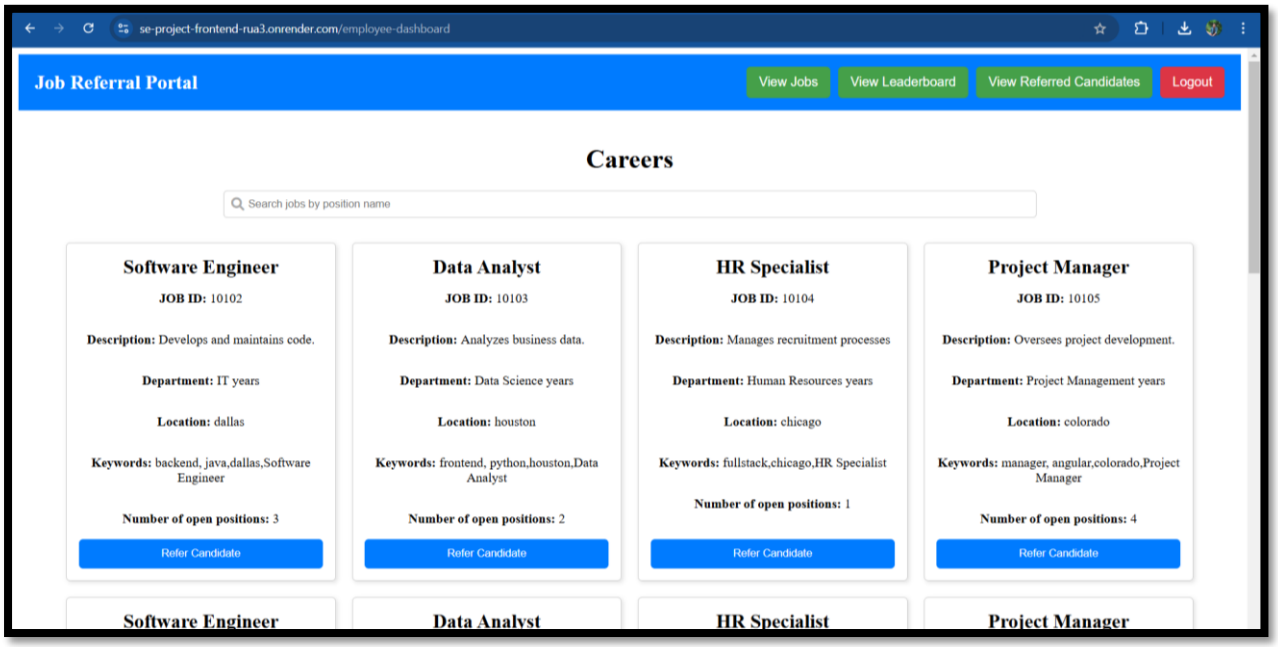
Functional Test 1:

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.

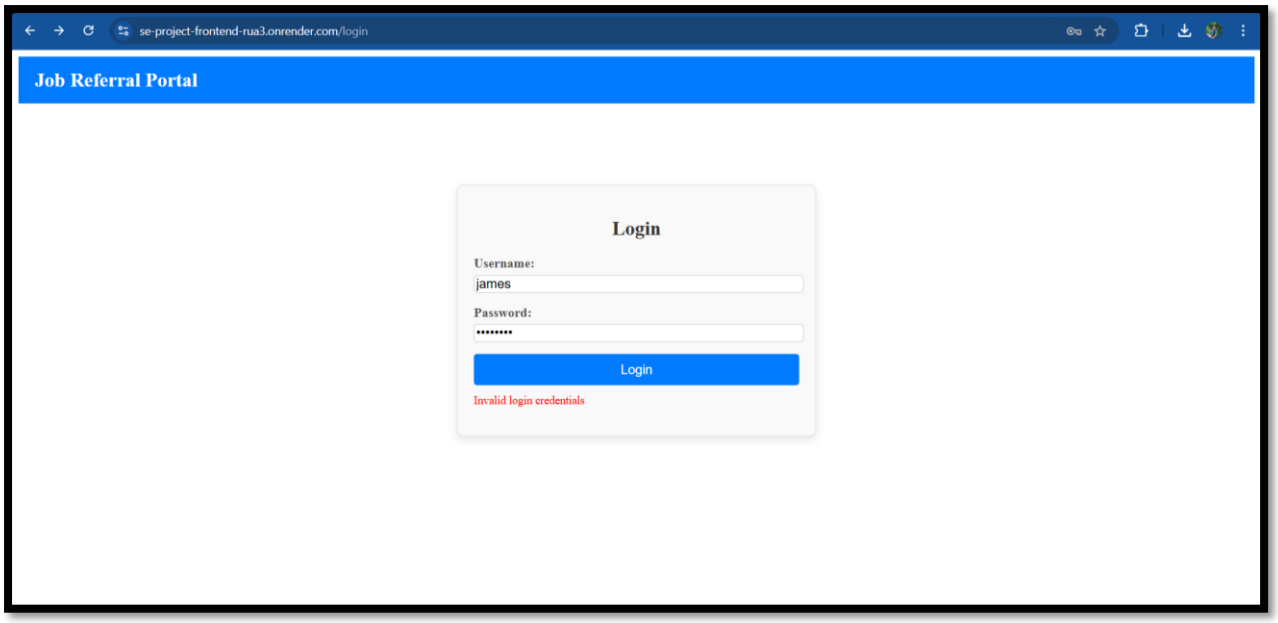
I am giving a valid username james and password as James@2000 as it is a valid username it should login.

Result:



So, the valid username case passed.

Let us now test the invalid username.

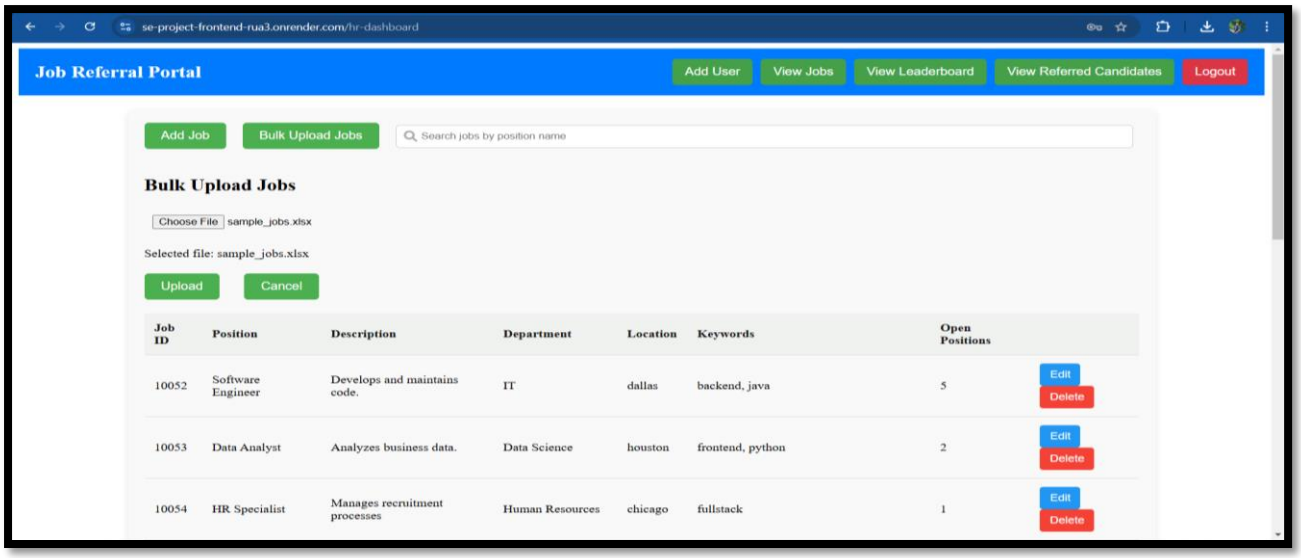


So, the login functionality is working as expected here.

Functionality Test 2:

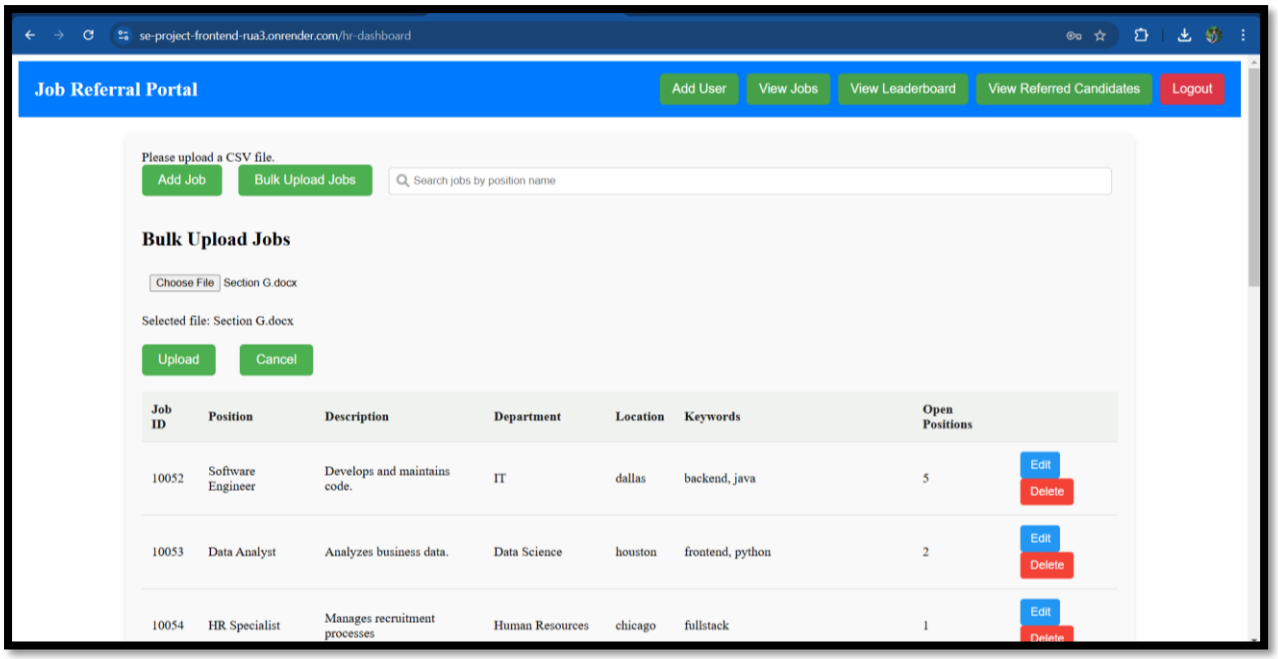
Now let us make a test to add a file which is used to bulk upload numerous jobs in it.

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



We can upload a file successfully.

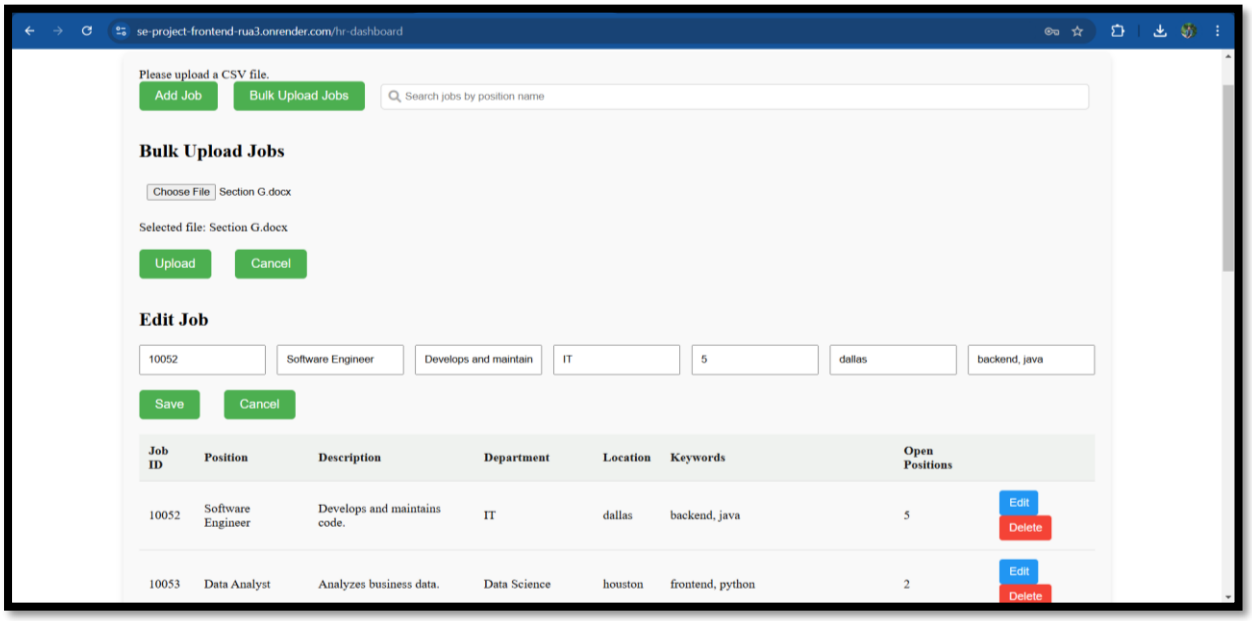
Now let us try to upload a doc file and see if it is allowing doc files.



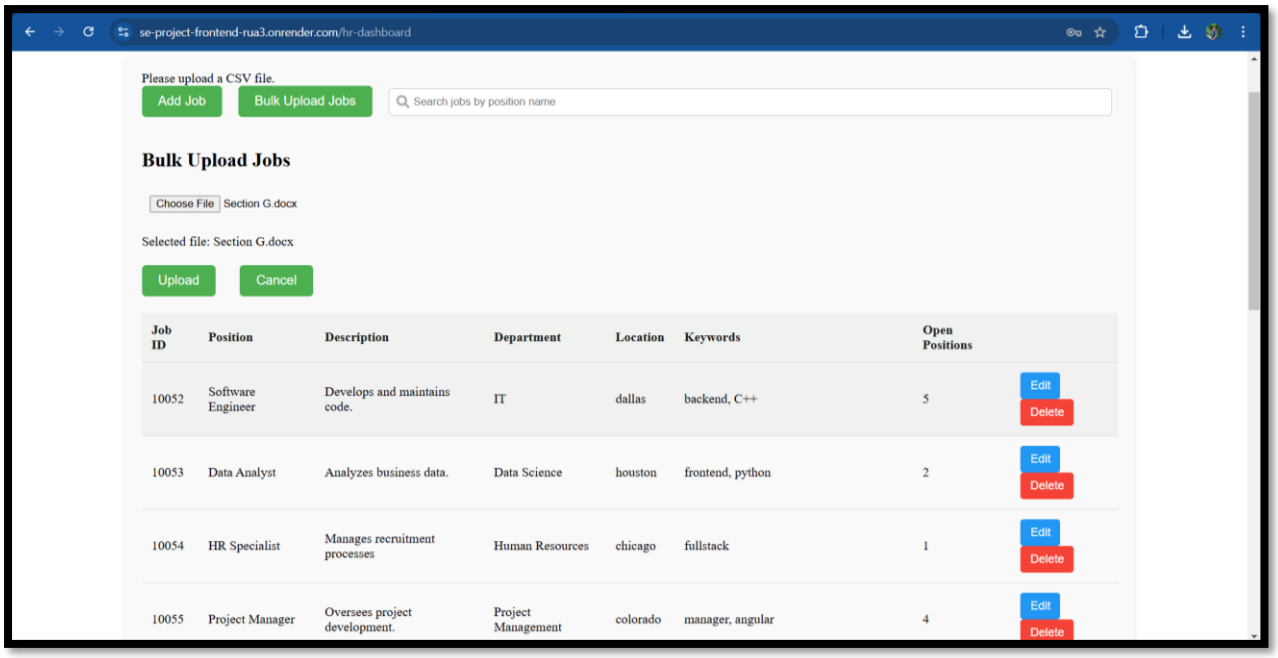
So, our upload function is working properly as expected.

Functionality Test 3:

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



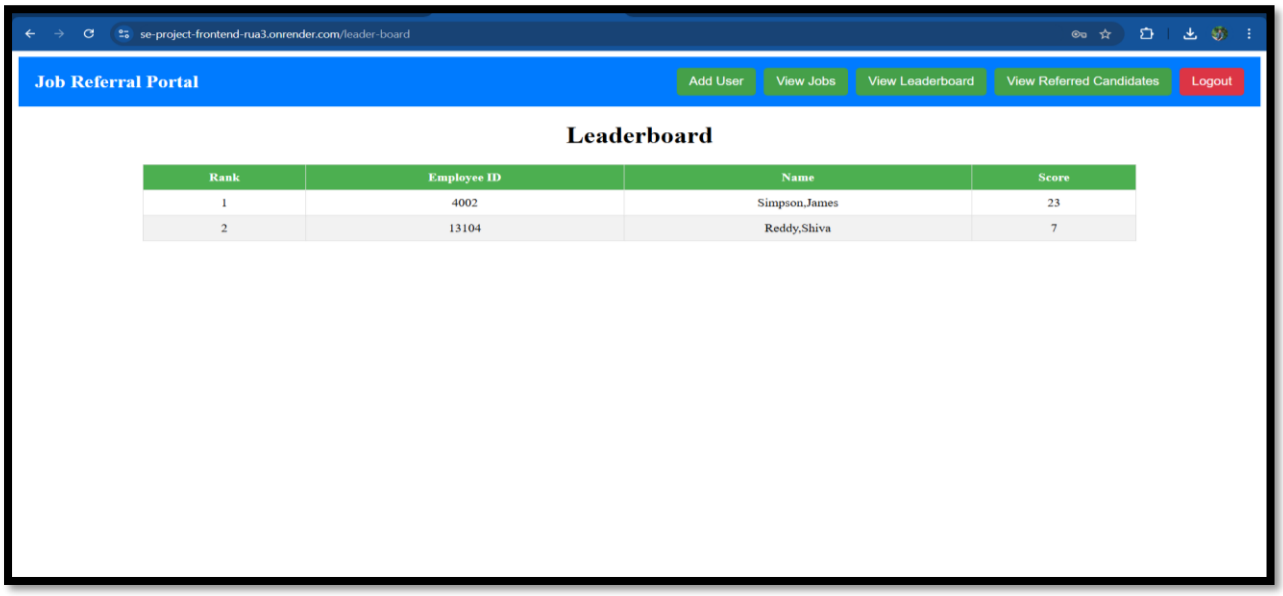
We have tried to change the technologies in the above job as backend, c++ and let us see if it changed successfully.



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

Functional Test 4:

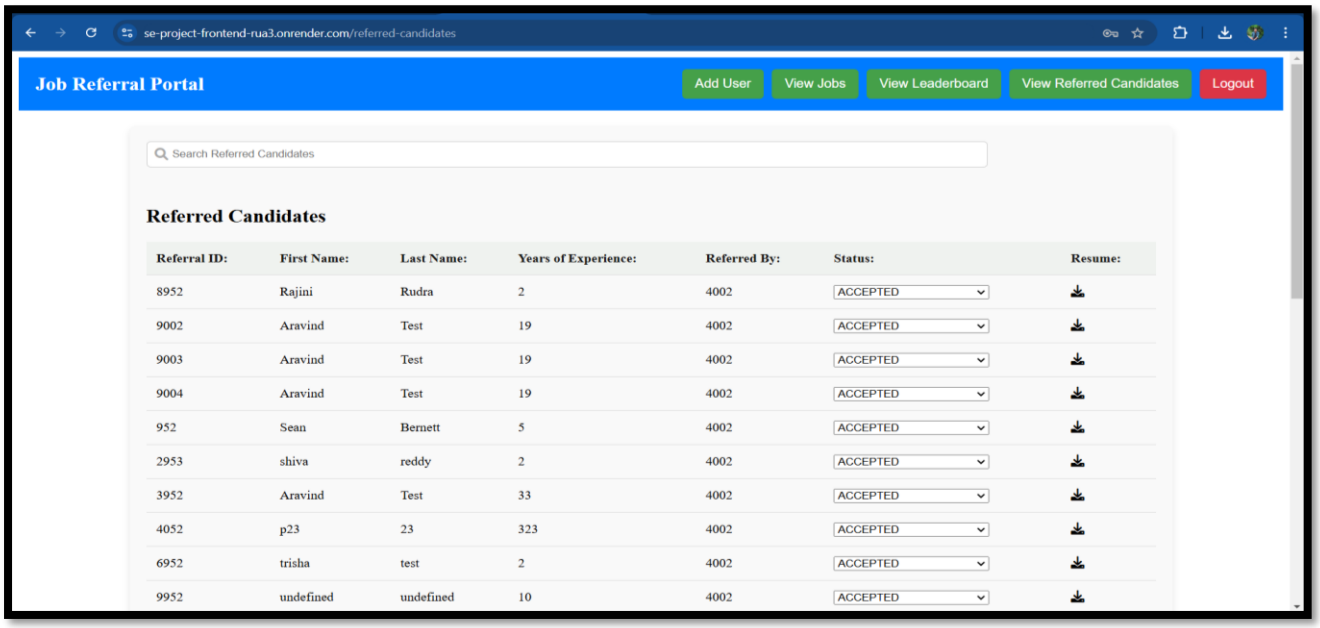
Now let us check if we can go to the leader board and see employees in it with the score calculated to them. We have clicked on the view leader board option in the screen and are able to see the dashboard below.



Hence the leaderboard is working for the HR page.

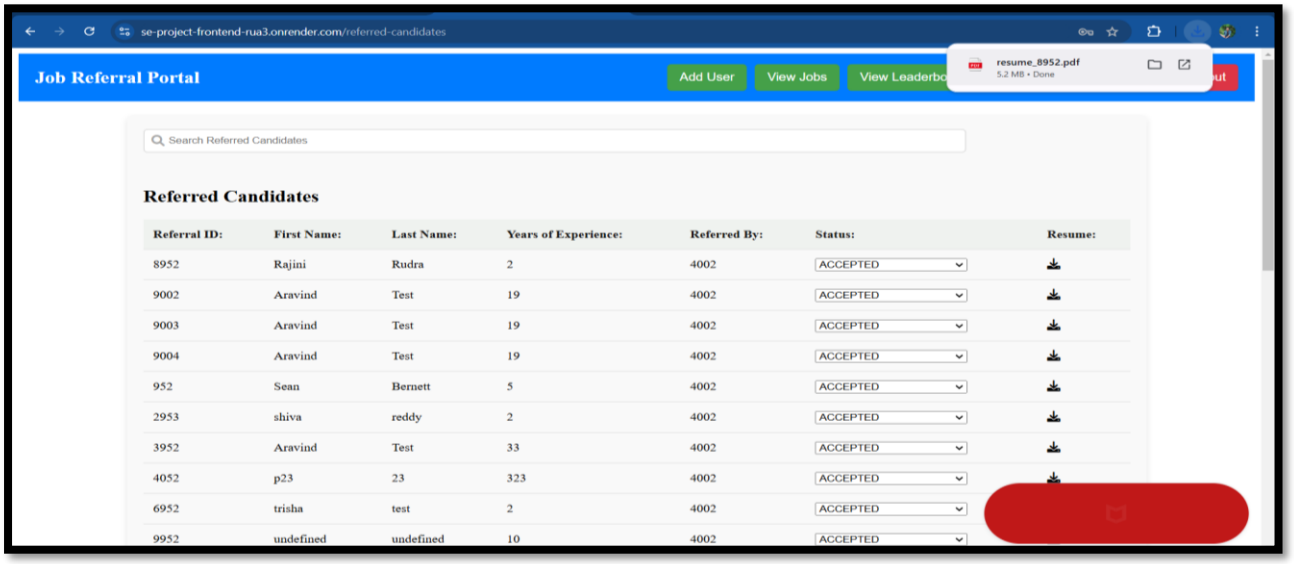
Functionality Test 5:

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.



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.



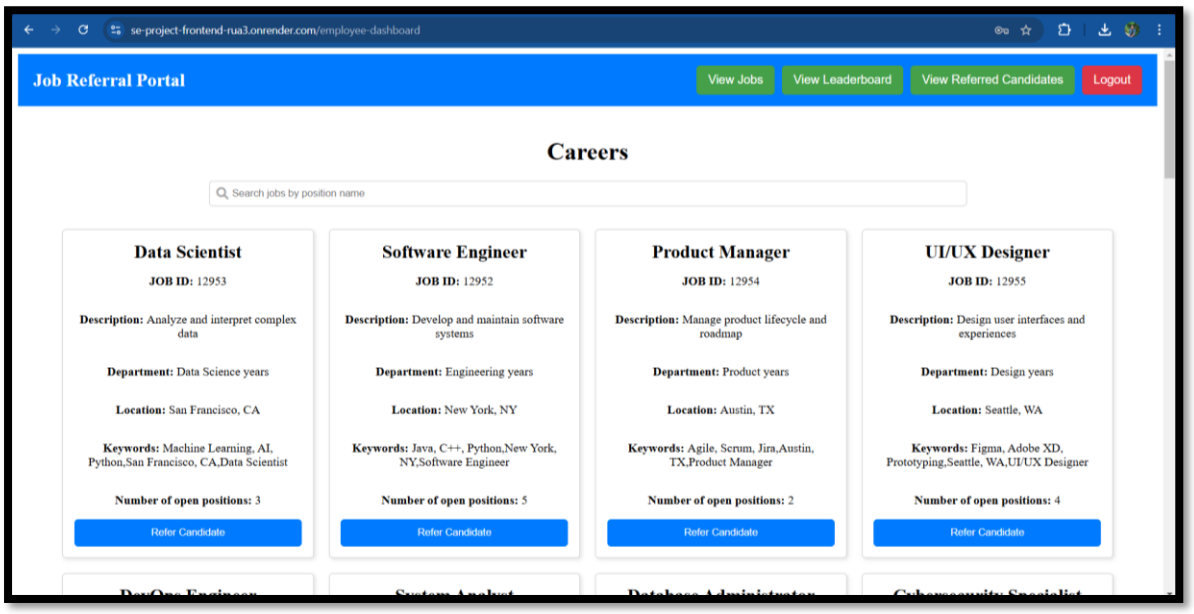
The above screen has the resume downloading in our system. Hence the functionality is implemented successfully.

Functionality Test 7:

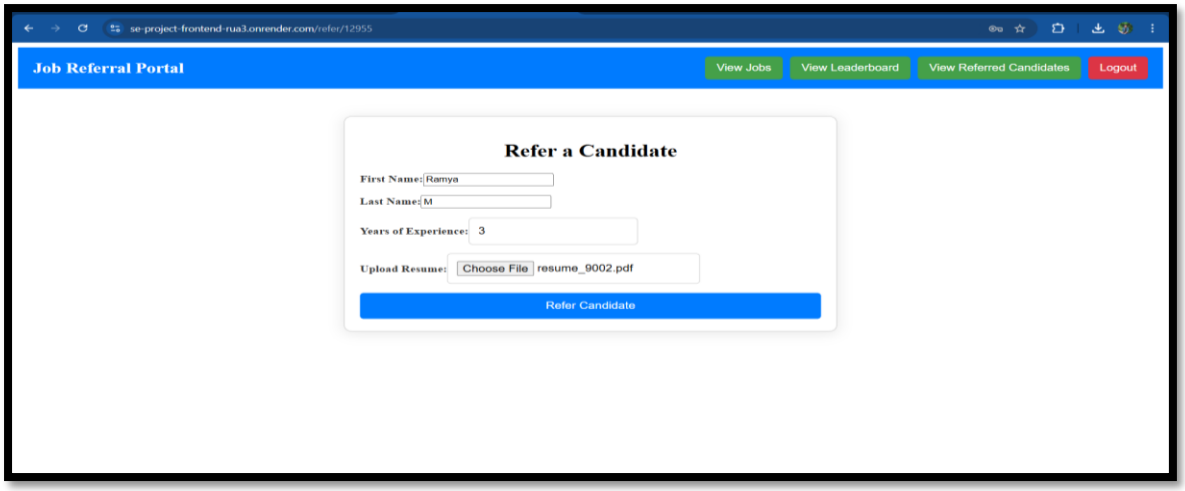
Now let's test the employee page and the employee functionalities.

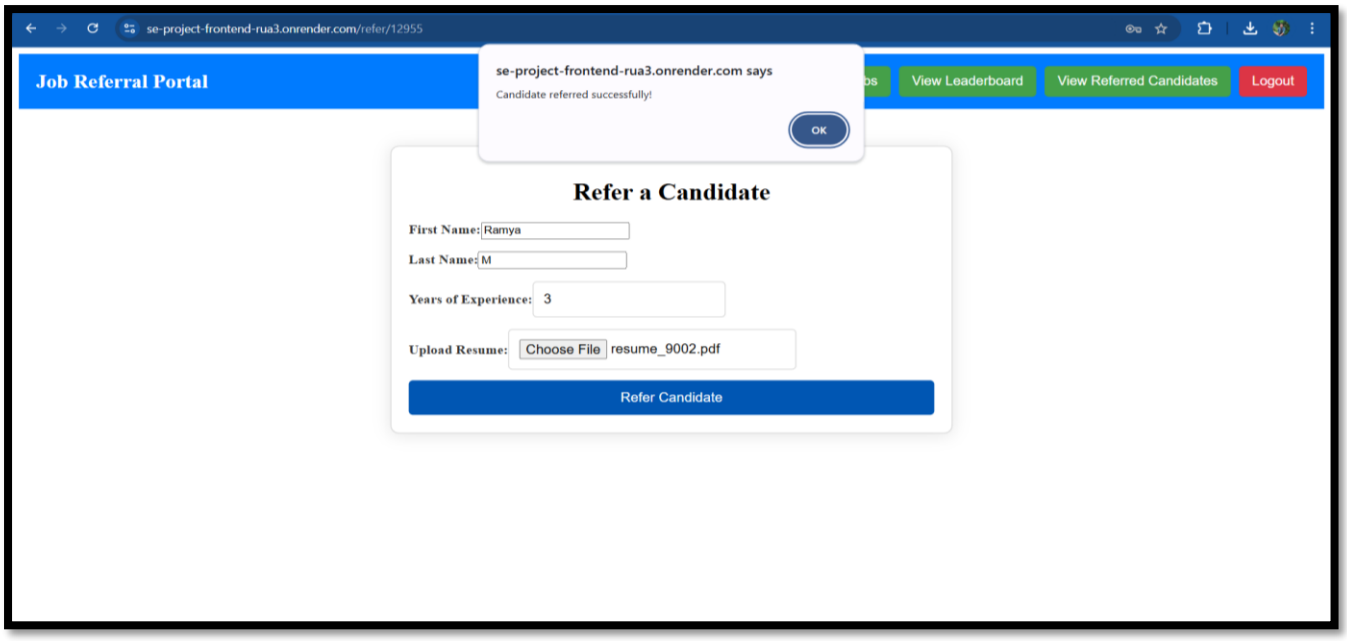
In the employee page we can refer a candidate and withdraw the application.

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



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



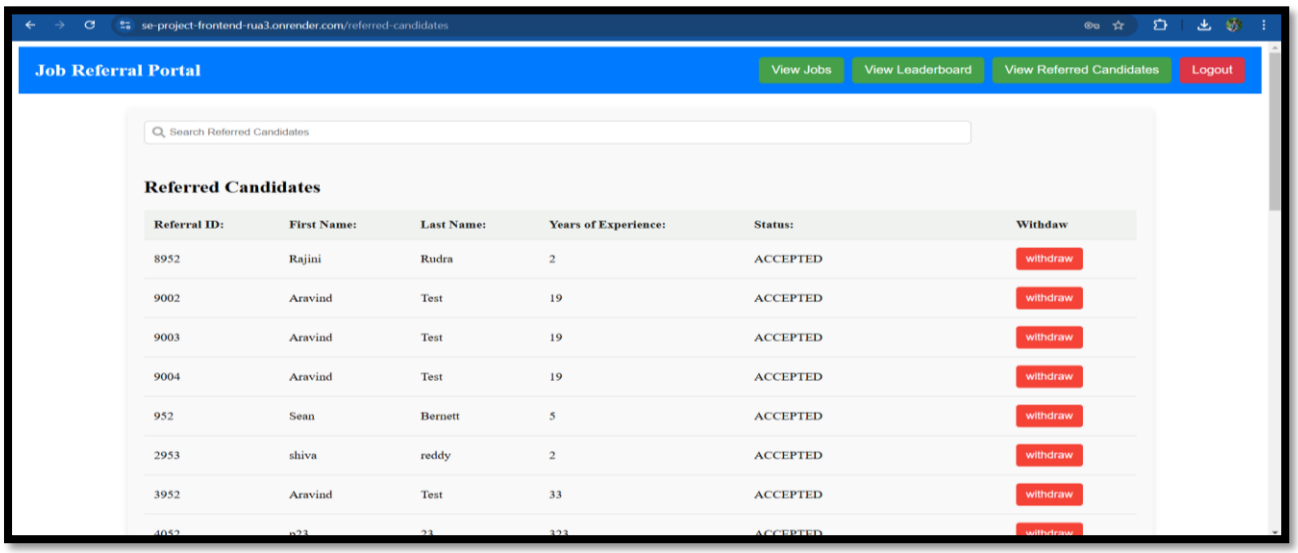


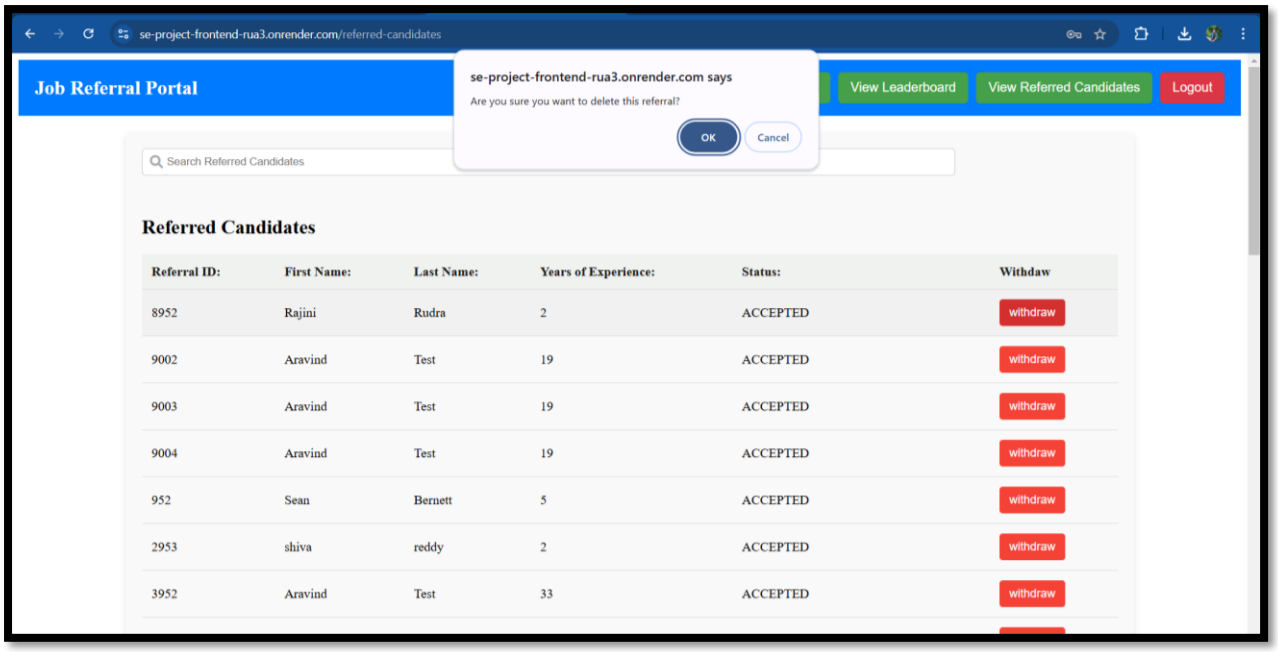
The person is referred successfully now. Hence the functionality is working as expected.

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.





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).

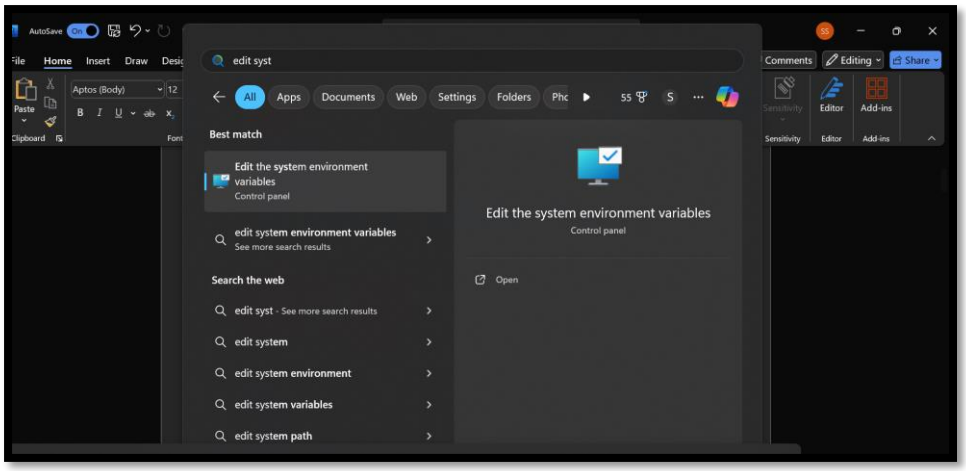
Installing and Running Software:

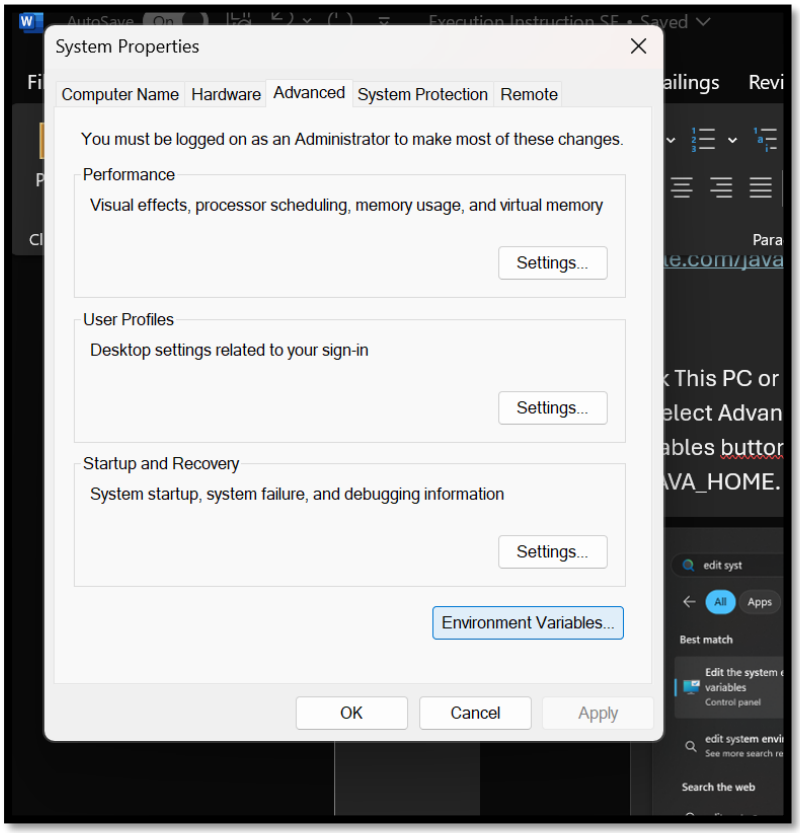
Note: Ensure to have the correct java and angular versions to download correct dependencies.
The college Wi-Fi is having an issue with oracle , so use personal hotspot to execute.

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.

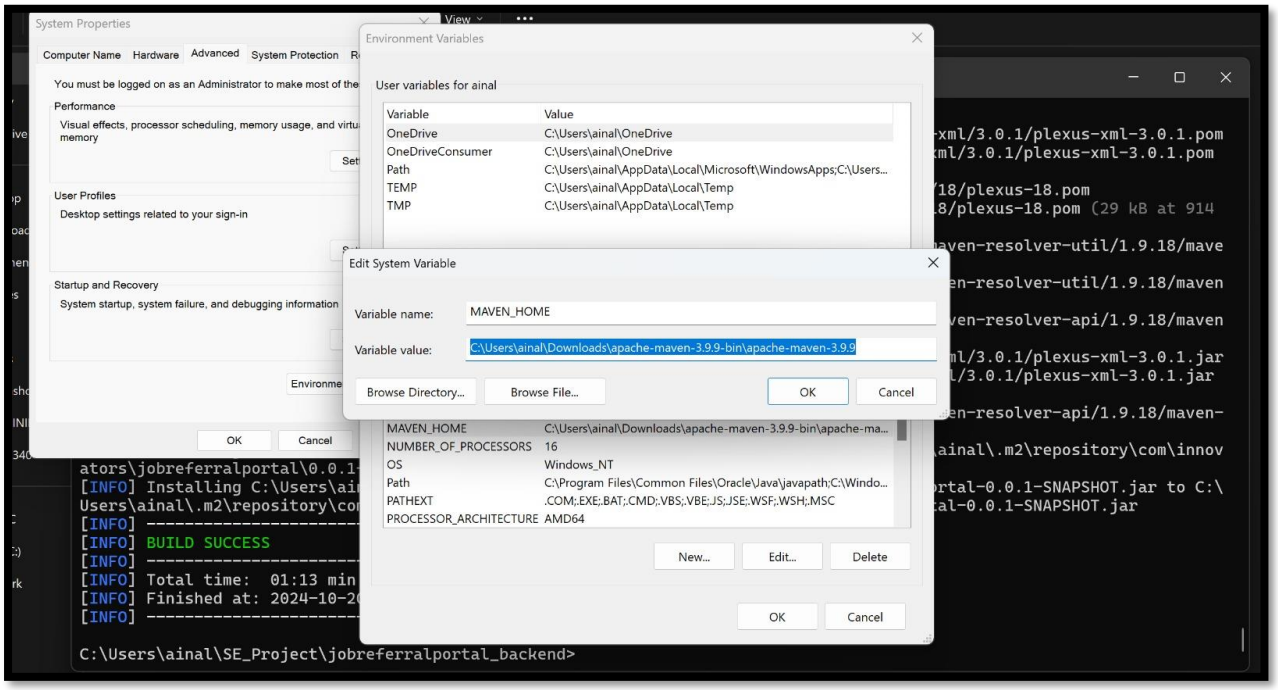
Restart the command prompt.

Verification (open command prompt)

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

For checking if jdk is available : javac -version

- 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.



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

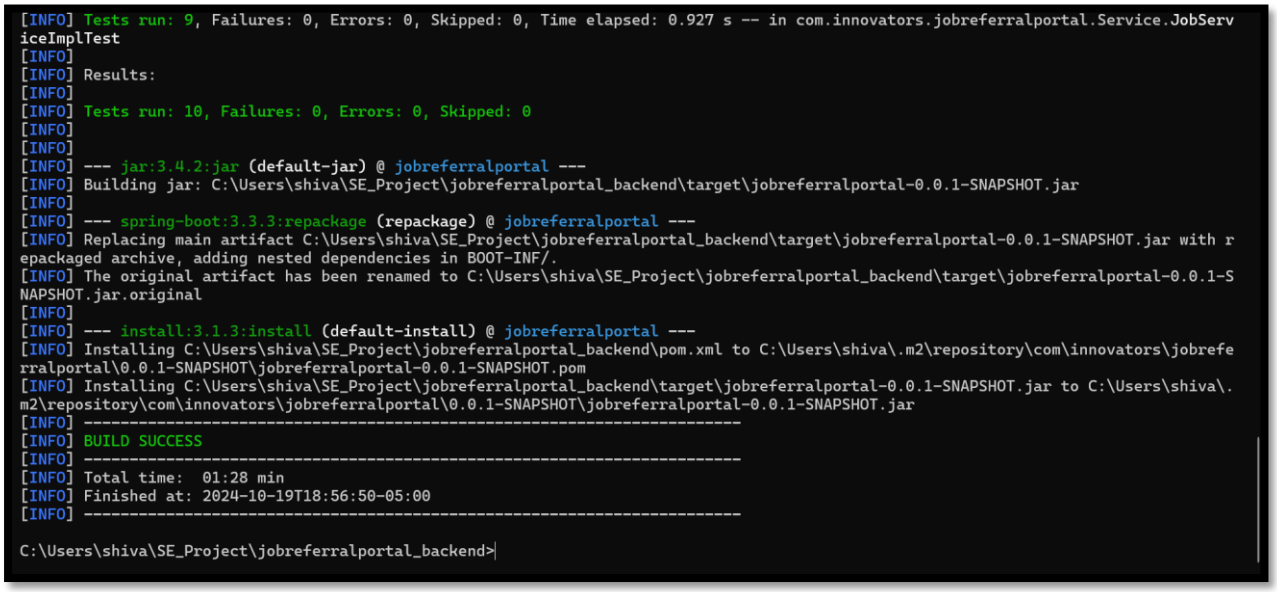
```
C:\Users\shiva>cd SE_Project

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

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

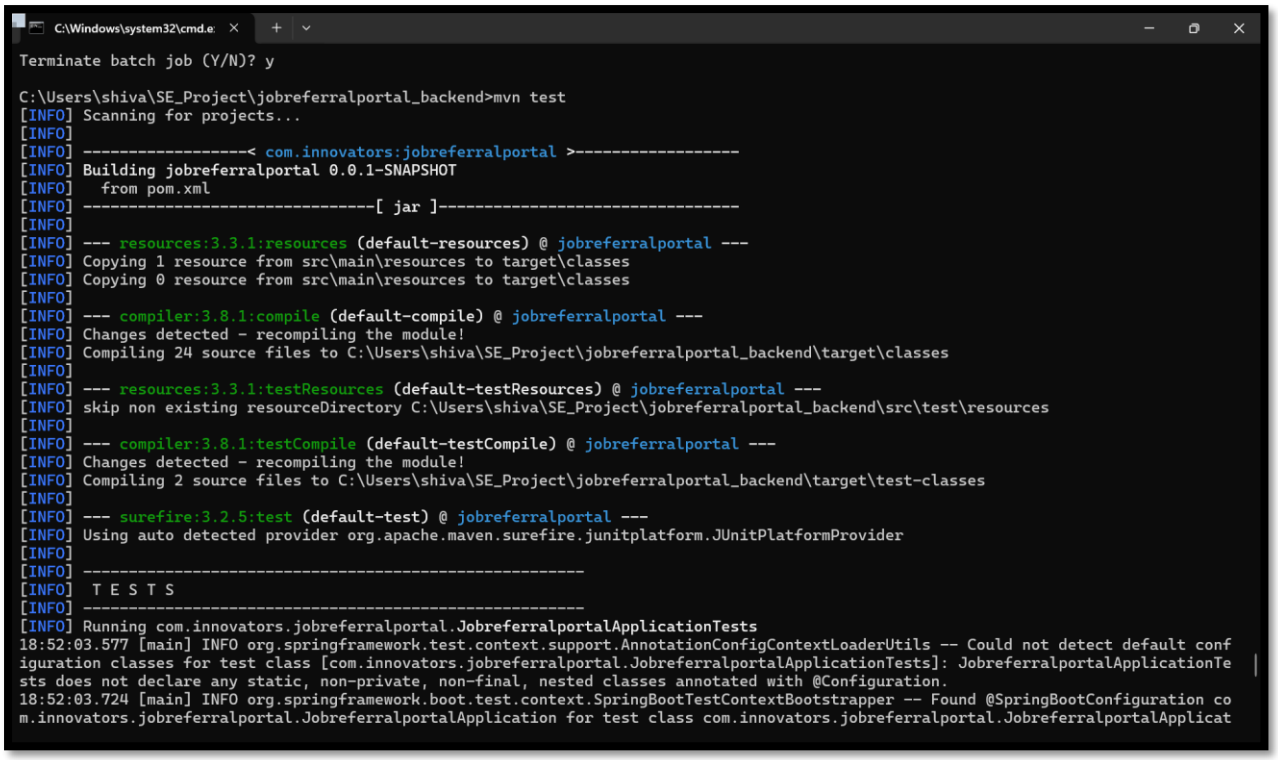
```
C:\Users\shiva\SE_Project\jobreferralportal_backend>mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.innovators:jobreferralportal >------------------
[INFO] Building jobreferralportal 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ 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
```

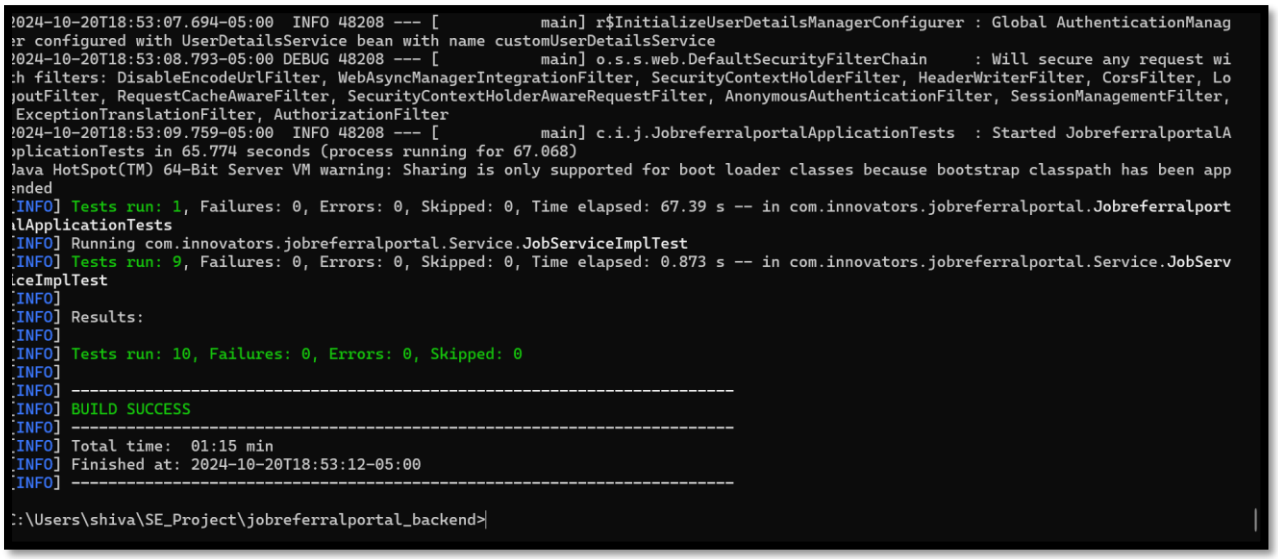
The application should start running

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



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





- 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)

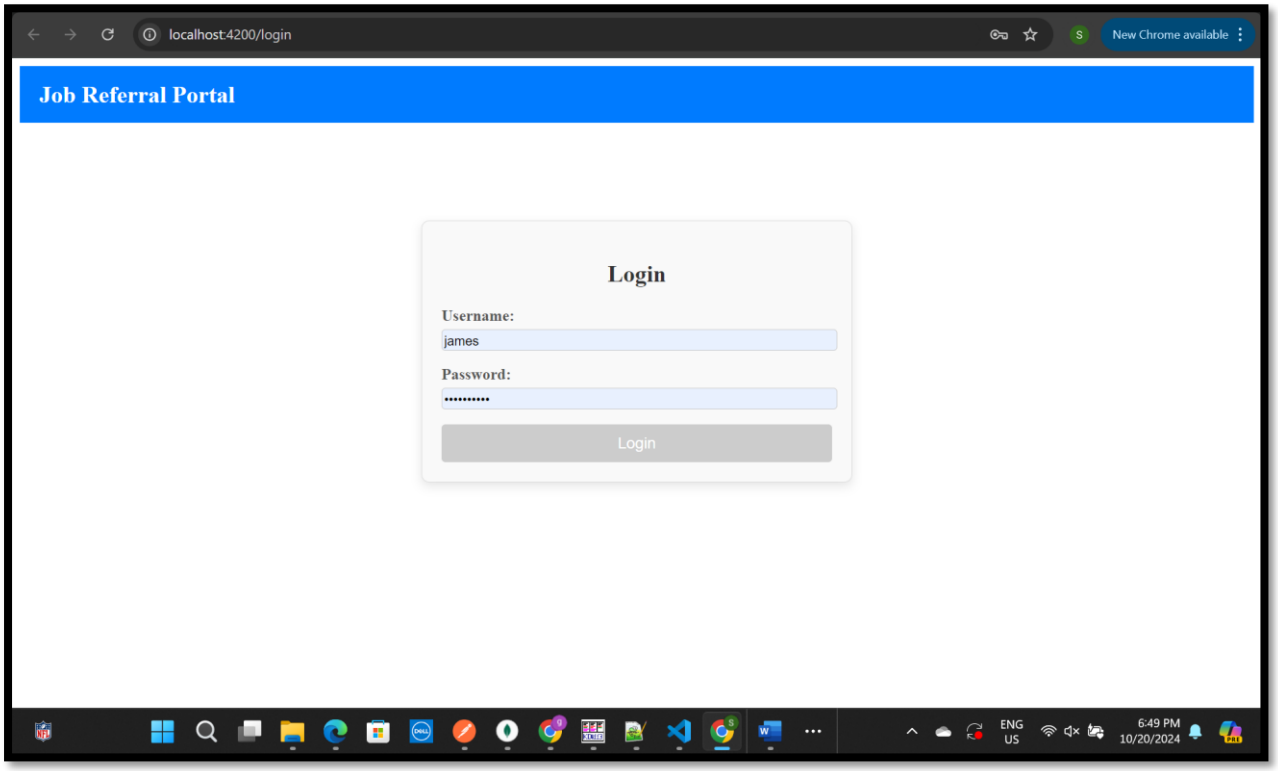
```
C:\Users\shiva\SE_Project\jobreferralportal_frontend>npm install -force
npm warn using --force Recommended protections disabled.
npm warn ERESOLVE overriding peer dependency
npm warn While resolving: @angular-devkit/build-angular@15.2.11
npm warn Found: @angular/platform-server@18.2.8
npm warn node_modules/@angular/platform-server
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
```

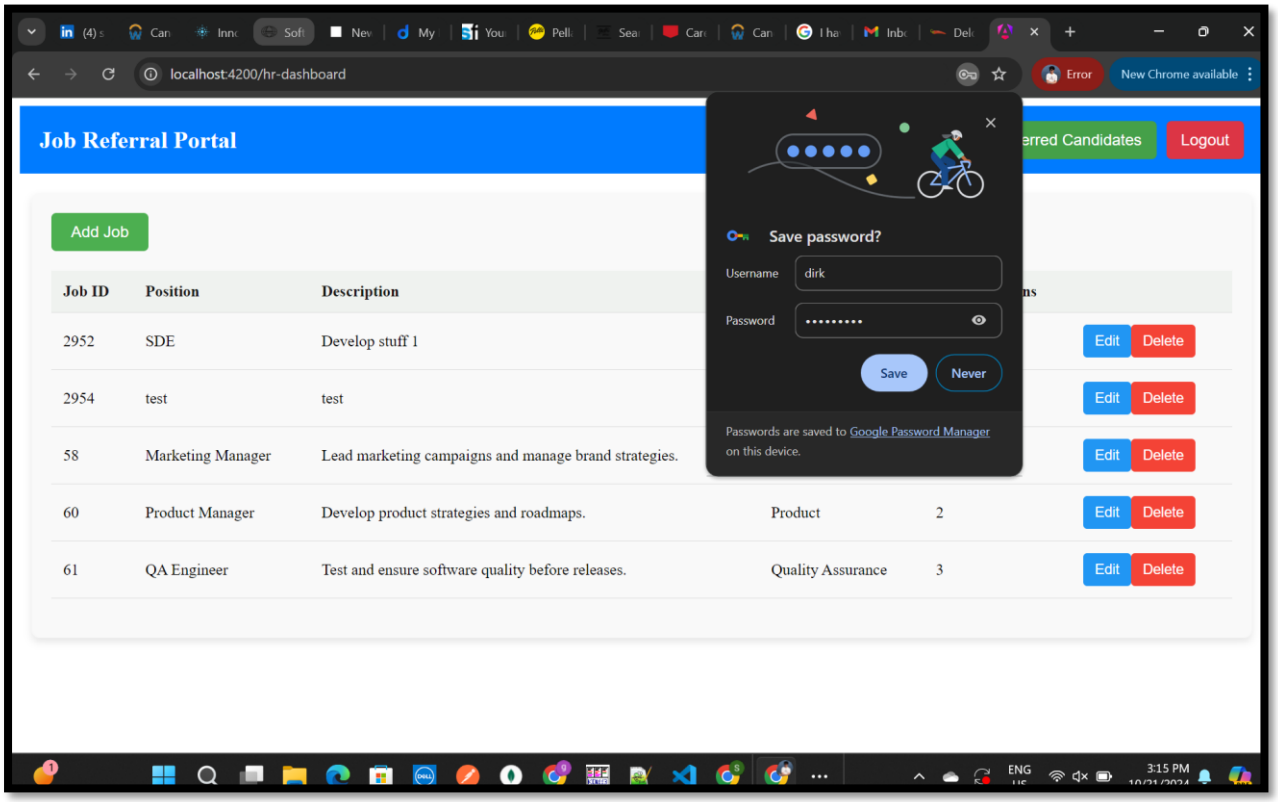
now open a web browser and go to localhost:4200/

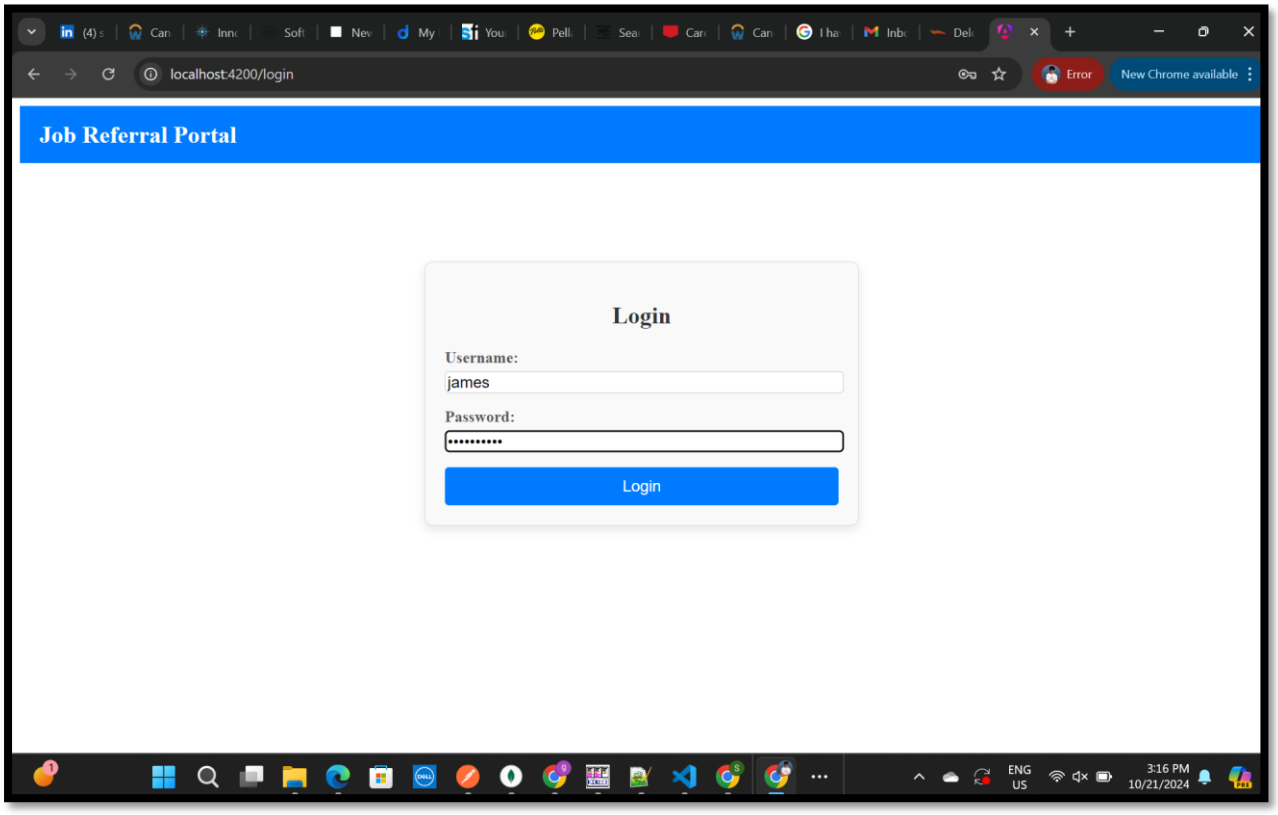
http://localhost:4200/

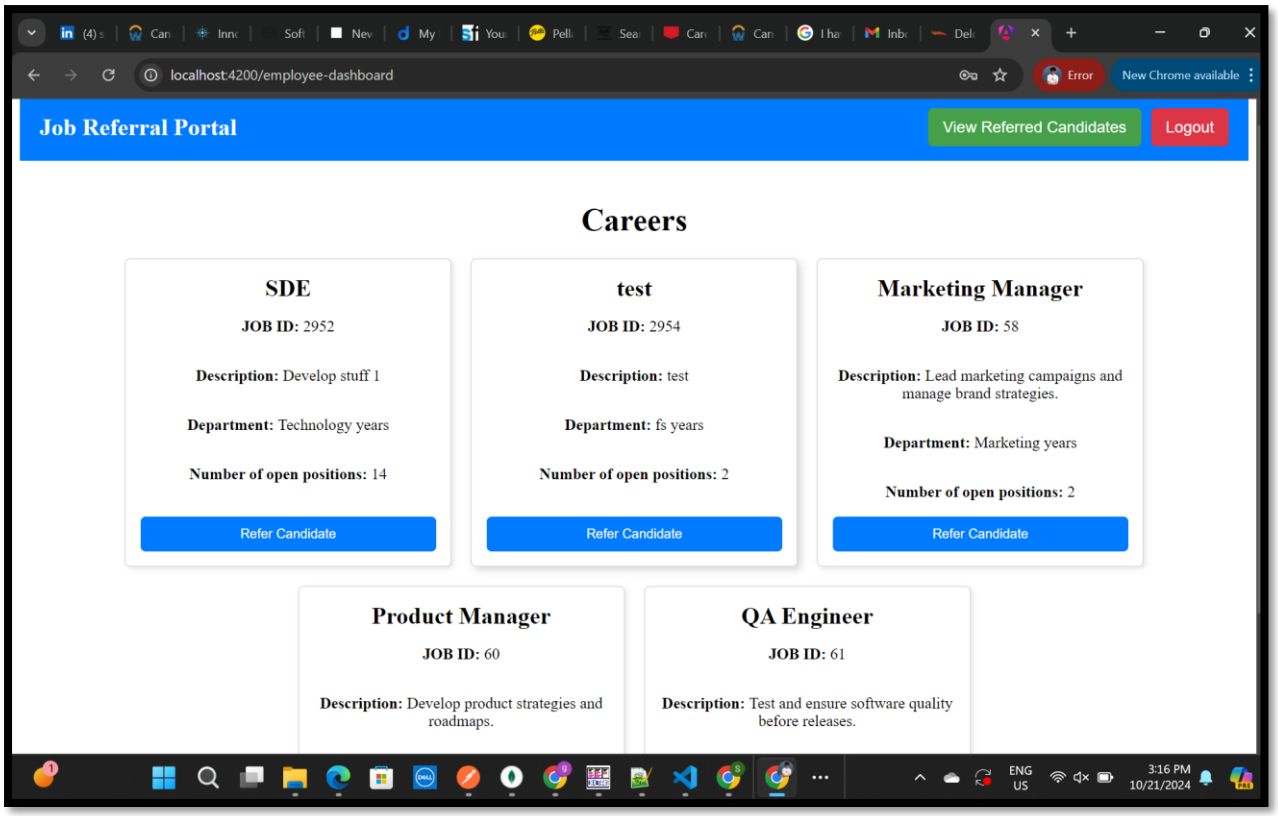


for Employee view : use credentials username : james, password: James@2000

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


localhost:4200/hr-dashboard
Error
New Chrome available
Job Referral Portal
erred Candidates
Logout
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.
ENG
US
3:15 PM
10/21/2024


localhost:4200/login
Error
New Chrome available
Job Referral Portal
Login
Username:
james
Password:
Login
ENG
US
3:16 PM
10/21/2024



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	