**Software Engineering II**

**Lab Manual**

***By:***

***Srushti Shah***

***191071902***

***Faculty: Dr. B. B. Meshram***

*Veermata Jijabai Technological Institute***Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Experiment No.** | **AIM** | **Page No.** |
| 1 | Problem Statement | **2** |
| 2 | Write SRS | **3** |
| 3 | OOAD USING UML | **33** |
| 4 | OO System Testing | **79** |

**Problem Statement**

***Job Recruitment System***

The proposed system bridges the gap between the recruiters and job seekers, by reducing the time and cost of running for job interviews from one place to another. By uploading resumes and setting job search criteria by the job seekers, the recruiters can browse the web application. Recruiters will have the freedom to post the particular jobs with their requirements, can filter their search query, check out the resumes of job seekers, upload their logos, can create their profiles. Job seekers can able to search job on the basis of various category such as organization wise, using their skills, location wise, job type etc.

***Aim:*** To develop an object-oriented system for Job Recruitment Process using Java to bridge the gap between recruiters and job seekers

***Objective***

1. Understand the object-oriented approach for designing a software application
2. Identifying the classes for the application, the main being the recruiter and job seeker
3. Give the system requirements along with reports using questionnaire and interviews
4. Make a User-Friendly Interface
5. Get the Maximum Jobs Listing
6. Minimum Cost
7. Minimum Time to Find Desired Job for Jobseekers
8. Minimum Time to Find Desired Jobseekers for Employer
9. Information Sharing Within Jobseeker and Employer
10. Proper Security and Privacy To All Users
11. Release the project after testing and modification for enhance user experience throughout.

**Experiment No*.: 1***

***AIM:*** Write SRS

***THEORY:***

**Software Requirements Specification**

**For**

**Job Recruitment System**

**Prepared by:**

Srushti Shah

191071902

**Instructor: Dr. B. B. Meshram**

**Course: Computer Engineering**

**Lab section: SE II Lab**

**College: VJTI**

**Table of contents**

1. **Introduction**
   1. Purpose
   2. Document conventions
   3. Intended audience and reading suggestions
   4. Product Scope
   5. References
2. **Overall description**
   1. Product perspective
   2. Product features
   3. User classes and characteristics
   4. Operating environment
   5. Design and implementation constraints
   6. Assumptions and dependencies
3. **System Features**
   1. Registration
   2. Login
   3. Manage Profile
   4. Upload CV
   5. Apply/Search for Job
   6. Job Application Status
   7. Post a Job
   8. View Job Seeker Profile
   9. Hire Jobseeker
   10. Handle Users
   11. Feedback
4. **Function Requirements**
   1. Job Seeker Module
   2. Recruiter Module
   3. Admin Module
5. **External interface requirements**
   1. User interfaces
   2. Hardware interfaces
   3. Software interfaces
   4. Communication interfaces
6. **Non-functional requirements**
   1. Performance requirements
   2. Security requirements
   3. Safety requirements
   4. Business Rules
   5. Data Integrity requirement
   6. Regulatory requirement
   7. Environmental requirement
   8. Software Quality Attributes
      1. Usability requirement
      2. Availability requirement
      3. Capacity requirement
      4. Serviceability requirement
      5. Manageability requirement
      6. Recoverability requirement
      7. Interoperability requirement
      8. Reliability requirement
      9. Maintainability requirement
      10. Scalability requirement
7. **Introduction**
   1. **Purpose**

The proposed system bridges the gap between the recruiters and job seekers, by reducing the time and cost of running for job interviews from one place to another. By uploading resumes and setting job search criteria by the job seekers, the recruiters can browse the web application. Recruiters will have the freedom to post the particular jobs with their requirements, can filter their search query, check out the resumes of job seekers, can create their profiles. Job seekers can able to search job on the basis of various category such as organization wise, using their skills, location wise, job type.

* 1. **Document Conventions**

For Main Heading:

Font size: 18, Font: Times New Roman, Font Type: Bold

For Sub Heading:

Font size: 14; Font: Times New Roman, Font Type: Bold

For Content:

Font size: 12, Font: Times New Roman, Font Type: Regular

* 1. **Intended Audience**

The intended audience of this document includes faculty members in the Department, the students looking for placements, company HR and job seeker looking for jobs.

* 1. **Product Scope**

The project targets the HR Departments of hiring companies, where the recruiters can upload jobs and look out for candidates with the required skills for the job. On the other hand, those seeking for jobs choose from various jobs available. The main modules of the system are Admin, Recruiter and Job Seeker

**Aim**: To develop an object-oriented system for Job Recruitment Process using Java to bridge the gap between recruiters and job seekers

**Objective**

* + 1. To allow the user to enter the system through registering through forms
    2. To allow only authentic users having valid ID and Password with the specified type to enter into the system
    3. To check for any invalid entry into the system
    4. To allow the user to modify his details using the profile tab
    5. To allow user to upload his Resume through his profile
    6. To allow the recruiter to check the Resume of a job seeker
    7. To allow the job seeker to search for jobs by pplying different filters available
    8. To get the Find Desired Job for Jobseekers in Minimum Time
    9. To get the maximum Jobs Listing for job seekers to apply for
    10. To apply for the job if the job seeker is eligible and has valid skills
    11. To Find Desired Jobseekers for Employer in Minimum Time
    12. To check for interviews and status of a job applied to
    13. To post jobs as recruiter
    14. To allow admin to manage the overall system
    15. To make a User-Friendly Interface
    16. To release the project after testing and modification for enhance user experience throughout.

**Scope (Modules)**

1. Registration
2. Login
3. Manage Profile
4. Upload Resume
5. Apply/Search for Job
6. Job Application Status
7. Post a Job
8. View Job Seeker Profile
9. Hire Jobseeker
10. Handle Users
11. Feedback
    1. **References**
12. Vivek Kumar Sehgal; Akshay Jagtiani; Meha Shah; Anupriya Sharma; Arpit Jaiswal; Dhananjay Mehta, Job Portal - A Web Application for Geographically Distributed Multiple Clients, IEEE, 2012
13. : Keethana Kopuri1, Gulam Mujtaba Hussain Aqueel2, Azbar Sadiqa Jabeen3, Dr.T.K.Shaik Shavali, A Online Job portal management system
14. Dr. Bandu Meshram ,Software Engineering PPT
15. Urmi Chakravarty, Report on Online Job Portal
16. Grady Booch, J. Rambaugh, Ivar Jacobson,”The UML Users guide”, Pearson
17. Stephen Cooper (2008). Software Requirements Specification. Retrieved June 6, 2008, from www.sju.edu/~scooper/spring01se/SoftwareRequirementsSpecification.htm

1. **Overall Description**
   1. **Product Perspective**

The Job Recruitment System provides searching of jobs by the job seekers listed. The Recruiter/HR Departments of the companies posting jobs available. The Admin approving the recruiters and job seekers to keep authenticity. It manages the database of jobs, companies, HR Departments, job seekers, skills, resumes.

The external view of the project

* Admin: He is the central manager of the system who manages the users on the system mainly being job seeker and recruiter. He/she decides if a user is valid or is it a bogus account, if he sees any malicious activity or illegal jobs posted he caters to its management.
* Job Seeker: The main user of the system, who is looking for job. He/she fills in details about skills and uploads the resume, after which he can search for the jobs that have been posted on the system according to the filters to be applied. Later, if suitable he can apply for the job and thereafter wait for the response from the recruiter for interview.
* Recruiter: The stakeholder of the system, the one that uploads the jobs on the system for employment. He/she can search for a particular job seeker based on his skills. He/she can schedule the interviews for the job.

|  |
| --- |
|  |
| ***Components of the System*** |

* 1. **Product Features**

Admin

* 1. Login/Logout: This will help users to login into the system using id and password. A user who has the valid id and password can only login to their respective accounts.
  2. Manage recruiters/job Seekers: It verifies and authorizes the accounts of a user, that can be either a recruiter or a job seeker, to ensure minimum fake jobs and scams to occur.

Recruiter

* + 1. Register: In this segment, we will register the new user of the system by entering the required details, this will then be approved by the admins.
    2. Login/Logout: This will help users to login into the system using id and password. A user who has the valid id and password can only login to their respective accounts.
    3. Create Job: In this interface, the recruiter who has registered themselves in the system can post the jobs. They can give the requisition specifying the post and skill needed for that post. It also implies the criteria of recruitment the Recruiter has planned for the post.
    4. Schedule Interview: Once, a recruiter is satisfied with the skills and resume of a job seeker he can schedule an interview for the job of that job seeker, for knowing him further, if he is a feasible candidate or not
    5. Edit Profile: The recruiter can make modifications in his profile containing personal details, job details and other information
    6. View Job Seekers: The recruiter can search a particular job seeker according to his skills or past positions
    7. Feedback: The recruiter can give a feedback about the system or report any bugs if found

Job Seeker

* 1. Register: In this segment, we will register the new user of the system, by entering the required details.
  2. Login/Logout: This will help users to login into the system using id and password. A user who has the valid id and password can only login to their respective accounts.
  3. Search for Jobs
  4. Apply for Jobs: This module is for the job seeker where they can search all the requisition present in the system. One can filter the search based on their skill and experience of the job. Ones selected a requisition applicant can apply for the post.
  5. Edit Profile: This feature allows the job seeker to update his/her skills and profile
  6. Check status for the job application applied to
  7. Upload Resume: The job seeker has an option to upload a file as his resume to let the recruiter know more about his work positions and educational information, projects and more
  8. Feedback: The recruiter can give a feedback about the system or report any bugs if found
  9. **User Classes**

The user classes are: -

* + - 1. Admin: Who administrated the overall system by allowing job seekers and recruiters into the system and ensuring authentic entry into the system
      2. Job Seeker: The main task is to search from jobs available and update the skills and resume.
      3. Recruiter: Uploading of valid vacant jobs for the company and selecting candidate for interview
      4. Interview: A recruiter schedules an interview process for the job seeker
      5. Job: It is posted by a recruiter for a company and job seeker applies for one.
  1. **Operating Environment**

No. of machine: 1.

Processor: Pentium 4 & upward.

Hard disk: 80 GB

RAM: 256 MB & upward

Java: JDK8.1, JRE8.0

Server: Apache tomcat 7.0

Database: Oracle10g

Operating system: Windows-xp & upward

* 1. **Design and Implementation Constraints**

The following semantic integrity and constraint explicitly are imposed by the s/w developed. These constraints are applied by keeping in mind customer requirements as well as consistent working of the system: -

|  |  |
| --- | --- |
| Module Name | Applied constraint to maintain data integrity |
| JOB SEARCH | * + Search query need to be entered else entry denied.   + Only registered job seekers can search for job. |
| POST JOB | * + Only registered recruiters can post jobs.   + Job details need to be entered otherwise denied. |
| JOB STATUS | * + Registered users except admin can see job status.   + Only status of applied job will be shown. |

Security & Other Constraints

* Each profile must be created or deleted by owner of profile or administrator.
* (username, password, type) pair constraint to be unique for each profile.
* Module name can be one of the possible modules existing in the system (viz. Job Search, Job Status, Reports, Post Job etc)
* The user should have sufficient knowledge of computers.
* The users must know the English language, as the user interface will be provided in English.
* The browser using should support JSP
* The user needs internet connectivity

Implementation Constraints: -

### Software Language Used: The languages that shall be used for coding Online Job Portal System are Java Servlets, Java Server Pages (JSP), and HTML. For working on the coding phase of the Online Library System, the Internet Information Services (IIS) Server needs to be installed.

### Development Tools: We will make use of the available Java Development Tool kits for working with Java Beans and Java Server Pages. Also we will make use of the online references available for developing programs in HTML.

### Class Libraries: We will make use of the existing Java libraries available for JSP and Servlets. Also, we need to develop some new libraries for the web-based application. Also, we will develop new programs using scripting languages.

* 1. **Assumptions and Dependencies**
* Jobseeker should be from any fields.
* Only admin have all the privileges.
* Candidate can only fill the registration form.
* Candidate cannot apply for job after deadlines.
* The project is done on the basis of selection of candidates for the company
* Every user should be comfortable working with computer and net browsing
* Since the application is a web based application there is a need for the internet browser. It will be assumed that the users will possess decent internet connectivity.

1. **System Features**

## Use Case: Registration

|  |  |
| --- | --- |
| Name | Register |
| Summary | A new user (a user which doesn’t have an account) should first register into the system to use it. This feature will allow the user to enroll into the system if the user is a new user and doesn’t already have an existing account. |
| Rationale | The user can access the system |
| Actor | Job Seeker and Recruiter |
| Pre-Condition | None |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the registration. 3. User enters the valid details required to join the system 4. Submits the form |
| Post- Condition | User is registered |
| Alternate Flow | * + - 1. User goes back to the Login Page |

## Use Case: Login

|  |  |
| --- | --- |
| Name | Login |
| Summary | Job seeker can change information into their profiles which include profile edit, delete, and update. |
| Rationale | After registration of the user, the valid user can login into the system |
| Actor | All Users |
| Pre-Condition | The login name and password should match with the login name and password provided while registering.  If the username and or password do not match, the user cannot login successfully into the system. |
| Basic course of Event | * 1. User opens the desired module of the system.   2. The system displays GUI the login form   3. The user writes its username, password and type   4. If valid, user can login into the system. If the username and or password do not match, the user cannot login successfully into the system. |
| Post- Condition | User goes to his/her dashboard |
| Alternate Flow | * + 1. The username and password is blank     2. The type of user is not specified     3. The username and password do not match |

## Use Case: Manage Profile

|  |  |
| --- | --- |
| Name | Manage Profile |
| Summary | User can change information into their profiles which include profile edit, delete, and update. |
| Rationale | If there is unnecessary date and fake profiles that are running on the site then Admin will have rights to remove or delete the profile. |
| Actor | All Users |
| Pre-Condition | Profile and record must exist |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI of different pages. 3. User will input the necessary data into the fields. 4. System will update the record as per requirements. |
| Post- Condition | User receives the notification that profile has been approved or profile has been updated. |
| Alternate Condition | * + - 1. The details enter to not validate, e.g. age is <18, date of bitrth is out of range. |

## Use Case: Upload CV

|  |  |
| --- | --- |
| Name | Upload CV |
| Summary | Job seeker can post his/her CV on job portal |
| Actor | Job seeker |
| Pre-Condition | New Jobs and other features should be displayed. |
| Basic course of Event | 1. Job seeker search for a specific job category. 2. The system displays GUI for the new jobs and features. 3. Job seeker clicks on the new jobs option and upload his/her CV there. 4. System will show all the CV on the admin and job seeker end. |
| Post- Condition | CV successfully uploaded. |
| Alternate Flow | * + - 1. The type of file uploaded does not match the required file type       2. The size of file is more than the maximum available size |

## Use Case: Apply/Search for Job

|  |  |
| --- | --- |
| Name | Search/Apply Jobs |
| Summary | Job seeker can search for jobs related to his interest using filters, if interested he can apply for the job |
| Rationale | Job Seeker is searching for a job |
| Actor | Job Seeker |
| Pre-Condition | Job Seeker must have an account |
| Basic course of Event | 1. Jobseeker access the required page to search the job. 2. System will show the appropriate GUI for searching and filtering the jobs 3. Job Seeker will fill all the required fields and describe the category and type of a job. 4. If, he/she is interested he will Apply for the job |
| Post- Condition | Apply for the job and wait for Interview notification. |
| Alternate Flow | * + - 1. The search criteria entered has no jobs listed |

## Use Case: Job Application Status

|  |  |
| --- | --- |
| Name | Job application status |
| Summary | Job seeker can check its status either their application is accepted. |
| Rationale | If their application is accepted and rejected, then they must know about the status, so that they can further proceed. |
| Actor | Job seeker |
| Pre-Condition | Job should have been applied for |
| Basic course of Event | 1. The job seeker login to system and check status of his application 2. The system displays GUI for the job application page. 3. Job seeker clicks on the page and check its status. 4. System will show all the results regarding their post on the jobs. |
| Post- Condition | Result successfully shown to the job seeker |

## Use Case: Post jobs

|  |  |
| --- | --- |
| Name | Post Jobs |
| Summary | Recruiter can post the job |
| Rationale | Recruiter wants to find the right employees for specific job. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must have a task and reasonable price for that task |
| Basic course of Event | 1. Recruiter access the required page to post the job. 2. System will show the appropriate GUI for posting the project. 3. Recruiter will fill all the required fields and describe the category and type of a project. 4. System will post the job. |
| Post- Condition | Anyone can see the posted job and apply for it after registration. |
| Alternate Flow | * + - 1. The job details are not sufficient enough to be posted |

## User Case: View Job seeker profile

|  |  |
| --- | --- |
| Name | View Job seeker profile |
| Summary | Recruiter can search the job seeker by name and can view the whole portfolio of job seeker. |
| Rationale | If a Recruiter has a job then he can search the job seeker and see entire profile of all employees or jobseekers. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must be existing in a list and Job Seeker has applied for the Job |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results. 3. System will show the entire job seeker in a relevant field in which Recruiter wants. |
| Post- Condition | Recruiter can see whole information about the job seeker. |

## Use Case: Hire Jobseeker

|  |  |
| --- | --- |
| Name | Hire Job seeker |
| Summary | Recruiter can hire the employee for a specific job required |
| Rationale | Recruiter has a job and wants to search the employee for the project if found, Recruiter will hire schedule the interview. |
| Actor | Recruiter |
| Pre-Condition | Recruiter should have posted a job. |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results 3. System will show the entire list of job seeker. 4. System will send the interview notification to the job seeker. |
| Post- Condition | Job seeker will receive the notification that he or she has been asked for interview. |

## User Case: Handle Users

|  |  |
| --- | --- |
| Name | Handle Users |
| Summary | Admin can view all the user’s proceedings. |
| Rationale | If admin wants to job seeker and recruiter to be added to the system or denied. |
| Actor | Admin |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. Admin verifies the details and credentials of the user 3. If valid, he admits them into the system, else removes them |
| Post- Condition | Job Seeker and Recruiter’s account will be created |

## User Case: Feedback

|  |  |
| --- | --- |
| Name | Feedback |
| Summary | To get a feedback from the user of the system |
| Rationale | To improve the system and understand the bugs |
| Actor | Recruiter and Job Seeker |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. GO to the feedback tab from dashboard 3. Enter and submit the feedback |
| Post- Condition | The developer receive the feedback |

1. **Functional Requirements**
   * + 1. Login/Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR001 | Login approved | The system will provide functionality to all the users once they log onto the system with their user name and password. On the basis of their type, they will be directed to different pages or section after matching username and password | Essential |
| FR002 | Login declined | If user enters wrong username and password it will show the error message. | Essential |
| FR003 | Forgot/Change password | Due to privacy concerns any registered user in the system can change their login password | Essential |
| FR004 | Logout | The user can logout from the system to avoid unauthentic access | Essential |

* + - 1. Manage recruiters/job Seekers

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR005 | Admin approval  (Job Seeker) | The admin can approve the request of job seeker to the system | Essential |
| FR006 | Admin approval  (Recruiter) | The admin can approve the request of recruiter to the system | Essential |
| FR007 | Admin Rejection  (Job Seeker) | The admin if suspicious can deny the request of job seeker to join the system | Essential |
| FR008 | Admin Rejection  (Recruiter) | The admin if suspicious can deny the request of recruiter to join the system | Essential |
| FR009 | View/Suspend  (Job Seeker) | Admin can suspend a job seeker from system as required and view reports on it. | Essential |
| FR010 | View/Suspend  (Recruiter) | Admin can suspend a recruiter from system as required and view its reports | Essential |

* + - 1. Register (Recruiter)

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR011 | Registration form | A recruiter can register online to put up jobs | Essential |
| FR012 | Email confirmation of registration | Recruiter will get an email after registration is completed. | Essential |
| FR013 | Submit registration form | After filling registration form recruiter can submit the application | Essential |
| FR014 | Cancel Registration form | If recruiter does not want to submit the form, they can cancel it | Essential |
| FR015 | Select Type of User | Choose the type of user while filling the registration form | Essential |
| FR016 | Password and Confirm Password Field Should Match | The characters of both fields should be identical to avoid error in typing of password | Essential |

* + - 1. Create Job

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR021 | Job Posting | Recruiters can post jobs, and specific details | Essential |
| FR022 | Update Job Details | Recruiter can make changes to the job listed already | Essential |
| FR023 | View Posted Job | A recruiter can view the summary of the job | Optional |

* + - 1. Schedule Interview

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR024 | Create Interview | The recruiter can schedule an interview | Essential |
| FR025 | Edit Interview | Recruiter can change the date, time or place of interview | Essential |

* + - 1. Edit Profile (Recrutier)

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR026 | Change address | Recruiter can change their address | Essential |
| FR027 | Change company | Recruiter can edit his details on the profile | Essential |
| FR028 | Change designation | Recruiter can edit his details on the profile | Essential |
| FR029 | Change email | Recruiter can edit his email on the profile | Essential |

* + - 1. View Job Seekers

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR030 | Search Job Seeker | A recruiter can search a job seeker by his name or skills required | Essential |
| FR031 | Send message | If the recruiter wants to get in contact with the job seeker he can leave a message | Essential |

* + - 1. Feedback

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR032 | Feedback form | Recruiter can send feedback to admin by feedback us page by filling in username, email and message | Optional |

* + - 1. Register (Job Seeker)

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR033 | Registration form | A job seeker can register online to apply for the job | Essential |
| FR034 | Email confirmation of registration | Job seeker will get an email after registration is completed. | Essential |
| FR035 | Submit registration form | After filling registration form job seeker can submit the application | Essential |
| FR036 | Cancel Registration form | If job seeker does not want to submit the form, they can cancel it | Essential |
| FR037 | Select Type of User | Choose the type of user while filling the registration form | Essential |
| FR038 | Password and Confirm Password Field Should Match | The characters of both fields should be identical to avoid error in typing of password | Essential |

* + - 1. Search for Jobs

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR043 | Filter for jobs | Job seeker can search for a job according to various filters like type, salary, position, experience needed | Essential |
| FR044 | View Job Details | By clicking on the job from the search results you can check the details of the same. | Essential |

* + - 1. Apply for Jobs

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR045 | Apply for job | Job seeker can apply for job after registration. | Essential |
| FR046 | Reference no for application | System will generate a reference no for job seeker application. | Essential |

* + - 1. Check Interview

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR047 | Job Alerts | Job seeker can receive job alerts by message about specific job category | Essential |
| FR048 | Interview Message | The jobseeker receives a message when he is selected for an interview with its details | Essential |

* + - 1. Edit Profile (Job Seeker)

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR049 | Change address | Job seeker can change their address as required. | Essential |
| FR050 | Change skills, education and other details | The job seeker can edit his details on the profile | Essential |
| FR051 | Change personal details | Job seeker can change their address as required. | Essential |
| FR052 | Change phone | Job seeker can edit his details on the profile | Essential |
| FR053 | Change email | Job seeker can edit his email on the profile | Essential |

* + - 1. Upload Resume

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement ID** | **Requirement Name** | **Details of requirement** | **Priority** |
| FR054 | Upload Resume | Job seeker can upload their Resume including cover letter to apply for job | Essential |
| FR055 | Download Resume | Recruiter can download Resume of the applicant from the system | Essential |

1. **External Interface Requirements**
   1. **User Interfaces**

The key rules ares

* Placed the user in control with program the interface do what the user desire.
* Reduced the user’s memory load with less command based and intuitive shortcuts.
* Make the interface consistent

Fig 1. Login Form

Fig 2. Registration Form (Job Seeker/Recrutier)

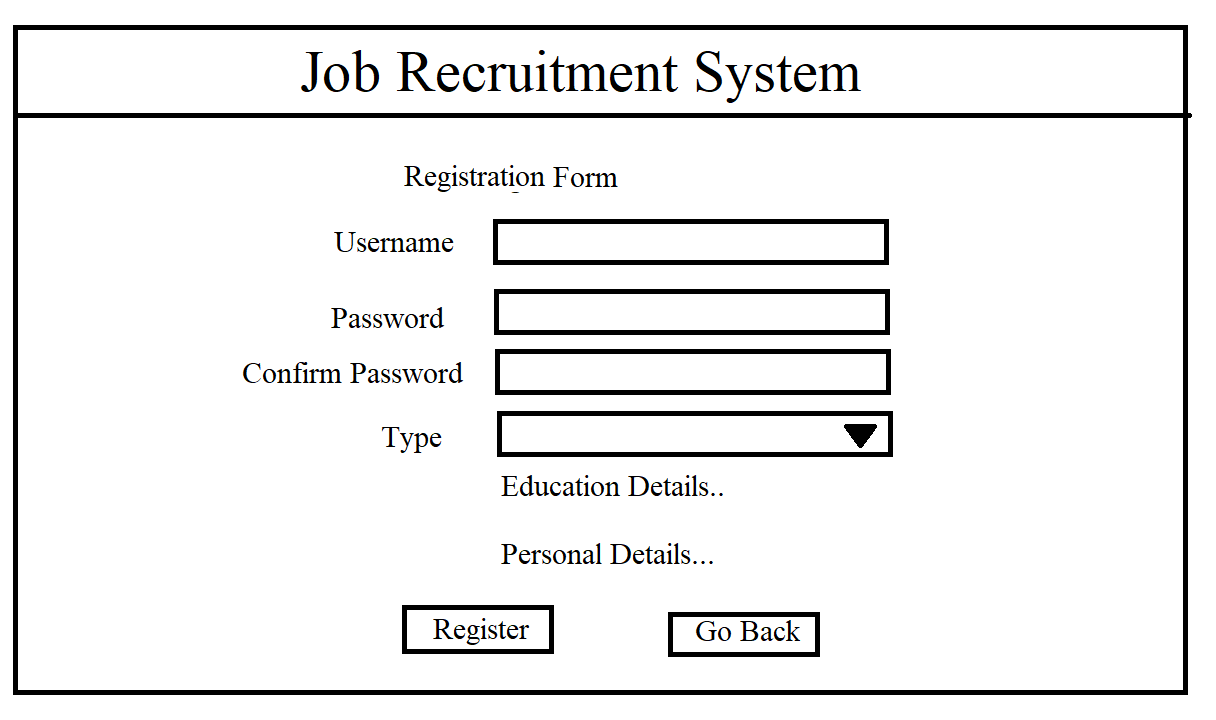


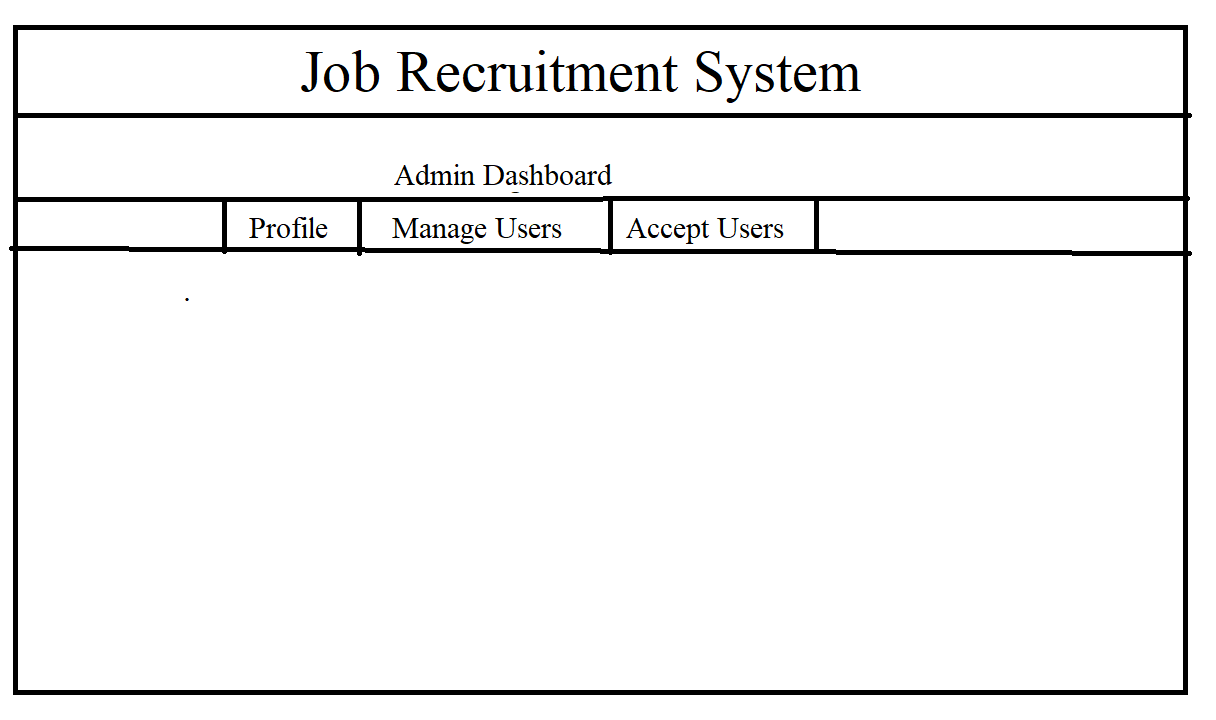
Fig 3. Admin Dashboard

Fig 4. Job Seeker Dashbard

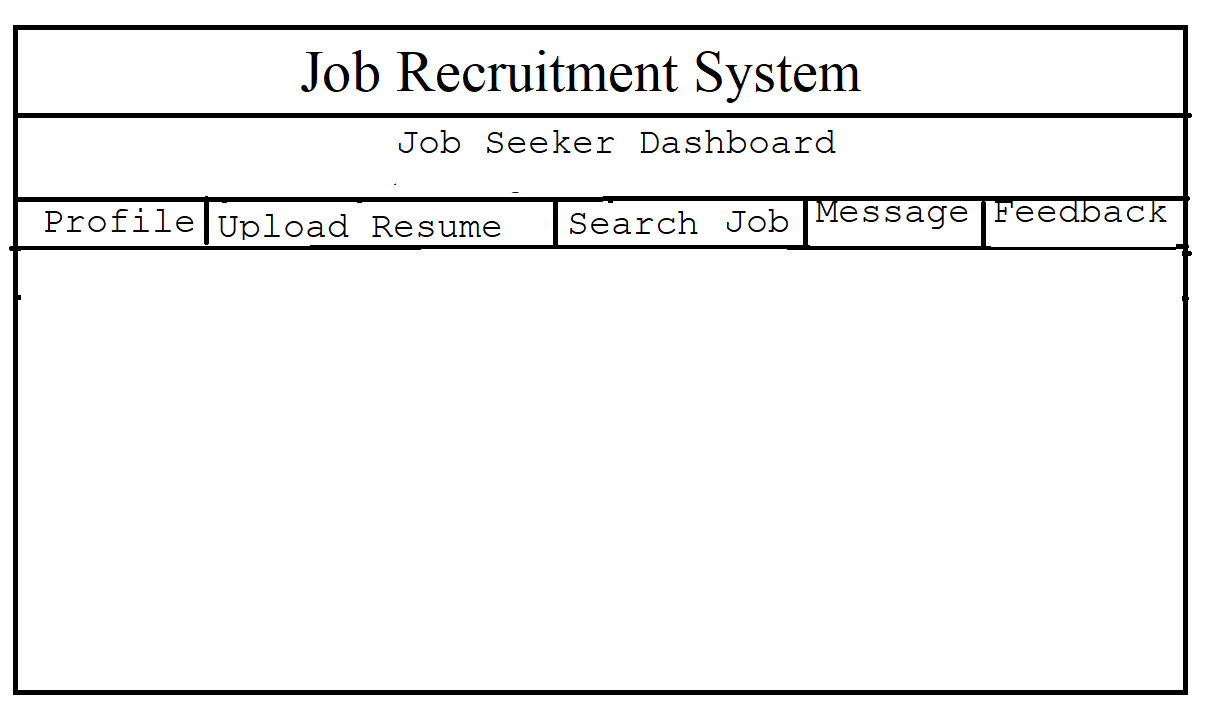


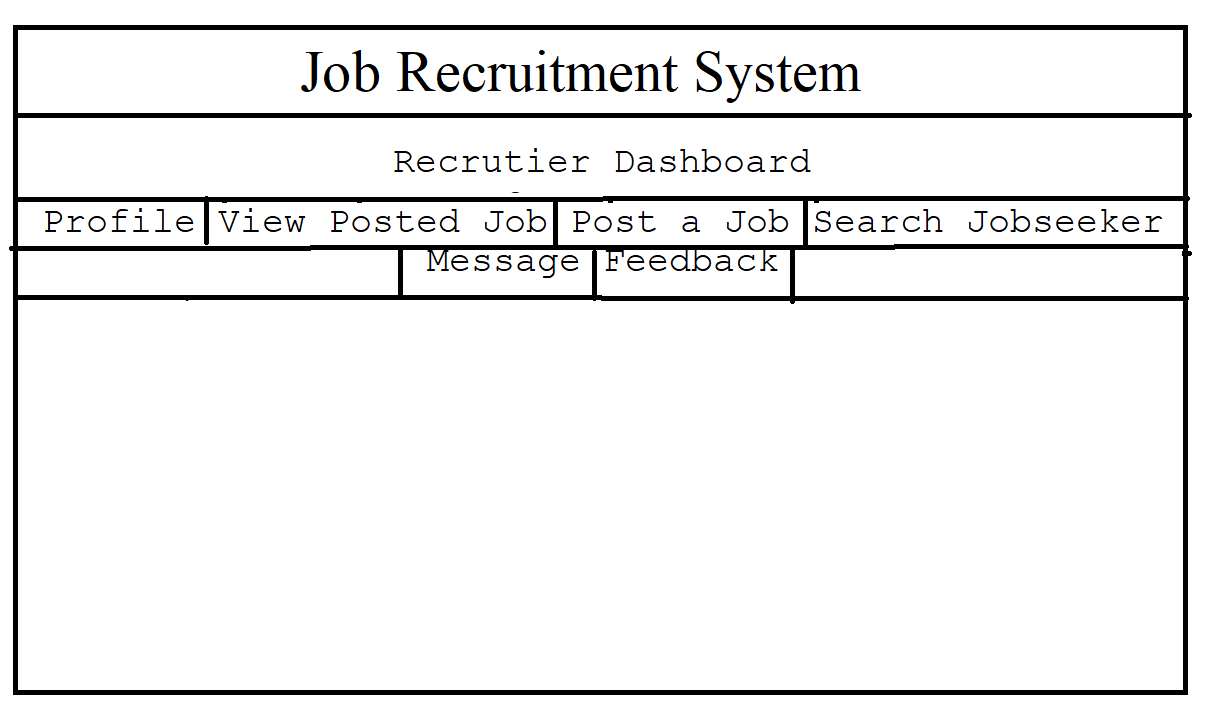
Fig 5. Recruiter Dashboard

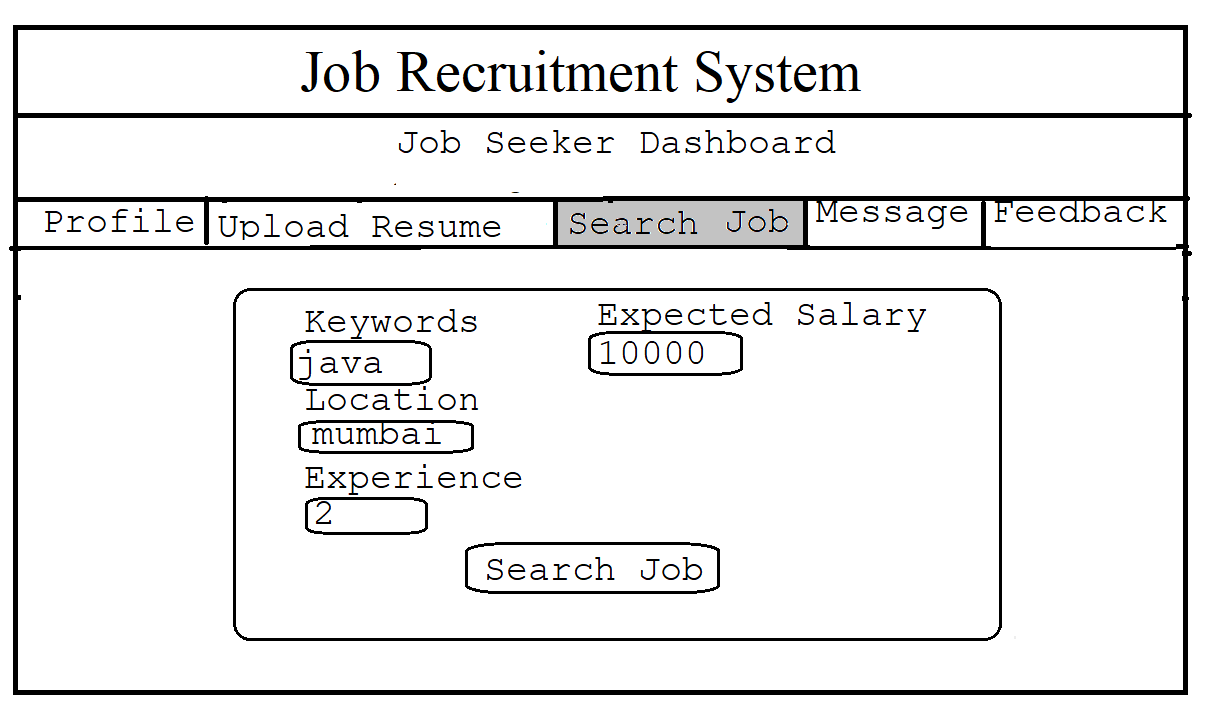
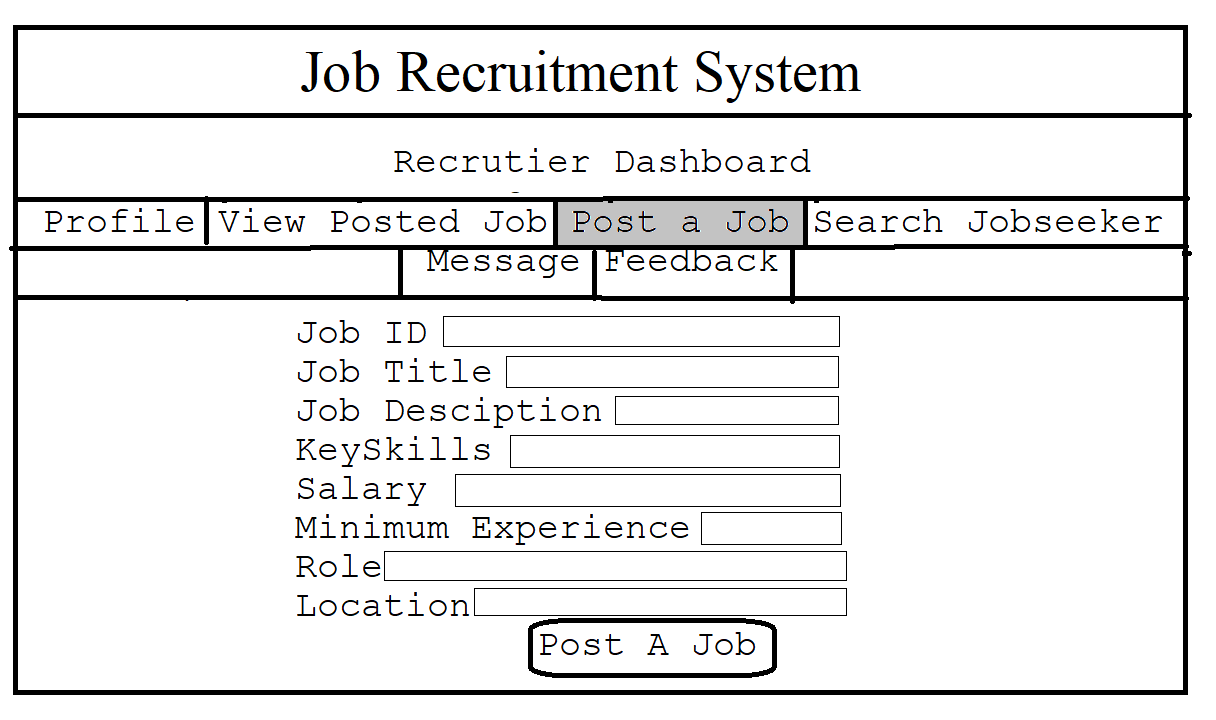
Fig 6. Search Jobs

Fig 7. Post a Job

* 1. **Hardware Interfaces**

No. of machine: More than 1.

Processor: Pentium 4 & upward.

Hard disk: 80 GB

RAM: 256 MB & upward

LAN card (Each system)

Keyboard

Mouse

Color Monitor.

Networking environment

* 1. **Software Interfaces**

**For server:**

Java: JRE5.0

Server: Apache tomcat 5.5

Database: Oracle9i

Operating system: Windows-xp & upward

**For client system:**

New version of java enabled web browser.

Operating system: Windows-xp & upward

* 1. **Communications Interfaces**

The Recruitment System will be connected to the World Wide Web.

The HTTP protocol will be used for communication over web

1. **Non-functional Requirements**
   1. **Performance Requirements**
      * 1. Response Time:

The splash page or information page should be able to be downloaded within a minute using a 56k modem. The information is refreshed every two minutes. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs.

* + - 1. Admin Response:

The system shall take as less time as possible to provide service to the admin.

* + - 1. Throughput:

The number of transactions is directly dependent on the number of users; the users may be the Admin and also the job seekers who use the system for checking-out jobs, agencies checking-out registered job seekers.

* + - 1. Resource Utilization:

The resources are modified according the user requirements and also according to the jobs requested by the users.

* 1. **Security Requirements**
     + 1. Login verification:
* Jobseeker will be asked for username, password and type will provide the detail about the type of user, to access jobseeker home only jobseeker type is accessible.
* For the admin home, it will only demand their username & password and type as admin.
* For the recruiter home, it will request username, password and type as recruiter.
  + - 1. Hiding the Home Page name from external people: Through this, we will provide double security from external users for the Job Seekers, Recruiters and as well as Administrator Home Page: By Hiding home page name. If unfortunately, they know then username verification works as a security for us.
      2. All users should be properly authenticated before allowed entry into the system Authentication will be based on a E-mail address, and a password. All activities on the system must be logged.
      3. Highly recommend Kaspersky 2020 internet security to been Installed in users Pc to prevent the harm that may occurs by Unwanted malicious software’s, phishing URLs and all the types of Virus attacks during using this application.
      4. The system must be protected against the following security threats:-
         * Man-in-the-middle (MitM) attacks, also known as eavesdropping attacks, occur when attackers insert themselves into a two-party transaction. Once the attackers interrupt the traffic, they can filter and steal data. Two common points of entry for MitM attacks:
         1. On unsecure public Wi-Fi, attackers can insert themselves between a visitor’s device and the network. Without knowing, the visitor passes all information through the attacker.
         2. Once malware has breached a device, an attacker can install software to process all of the victim’s information.
    - A denial-of-service attack floods systems, servers, or networks with traffic to exhaust resources and bandwidth. As a result, the system is unable to fulfill legitimate requests. Attackers can also use multiple compromised devices to launch this attack. This is known as a [distributed-denial-of-service (DDoS) attack](https://www.cisco.com/c/en/us/products/security/what-is-a-ddos-attack.html).
    - SQL Injection: A Structured Query Language (SQL) injection occurs when an attacker inserts malicious code into a server that uses SQL and forces the server to reveal information it normally would not. An attacker could carry out a SQL injection simply by submitting malicious code into a vulnerable website search box.

6. The user should be forced to keep a difficult password including upper and lower

case, numbers and symbols, to avoid:-

* + - Brute-force password guessing means using a random approach by trying different passwords and hoping that one work Some logic can be applied by trying passwords related to the person’s name, job title, hobbies or similar items.
    - In a dictionary attack, a dictionary of common passwords is used to attempt to gain access to a user’s computer and network. One approach is to copy an encrypted file that contains the passwords, apply the same encryption to a dictionary of commonly used passwords, and compare the results.
  1. **Safety Requirement**
     + - The system should be protected from excess heating by using cooling mechanisms and fan to release the heat produced from continuous working.
       - The server should always be confirmed to run properly and the data are saved to the database at consecutive intervals.
       - Power is a significant feature and the power supply should be always taken care of.
       - Power supply must be uninterrupted. Unsaved data might be lost due to sudden power failure.
       - The database may get crashed at any point of time due an operating system failure or any other reason. Therefore, a reliable backup system has been implemented.
       - Safeguard the system and server side and database from People, earthquake, fire, temperature, humidity etc.
  2. **Business Rules**
* Frequency of use can be varied depend on where it is used.
* Maintenance: The maintenance of the system shall be done as per the maintenance contract.
* Standards: The coding standards and naming conventions will be as per the American standards.
* The system shall support the UHCL information security requirements and use the same standard as the information security requirements.
* Internet Protocols: The system shall be comply with the TCP/IP protocol standards and shall be designed accordingly.
  1. **Data Integrity requirement**
     + 1. User data is confidential and should be stored in an encrypted way, to prevent unauthorized access and misuse. Information regarding airlines must be treated with utmost confidentiality. Users must be told to upgrade their passwords periodically, to improve security and confidentiality of their data. Encryption keys must not be made publicly available at any cost.
       2. The integrity constraints on the database should be as follows:-
          - Job
          - Primary Key:- JOBID
          - Domain: The job should be listed under a given available domains only
          - Referential Integrity: Each job is created by a recruiter with recrutier username
          - Job Seeker
          - Primary Key: Job Seeker Username
          - Recruiter
          - Primary Key: Recruiter Username
          - Interview:
          - Primary Key: Interview ID
          - Referential: Each interview is for a particular job seeker for a particular job created by a particular recruiter
  2. **Regulatory requirement**
* The system shall be complying with the TCP/IP protocol standards and shall designed accordingly.
* The coding standards and naming conventions will be as per the American standards.
* Legal, Copyright, and Other Notices: Job Recruitment System is a trademark and cannot be used without consent.
* Applicable Standards: The ISO/IEC 6592 guidelines for the documentation of computer-based application systems will be followed.
  1. **Environmental requirement**
* The system should be developed in a well-lit and cool room to maintain the server.
* There should be less interference in the wireless signals in case of wireless connectivity to the internet.
* The temperature within the range of 18 ºC to 23 ºC and the humidity within the range of 50 % to 70 % independent of weather conditions.
  1. **Software quality attributes**
     1. **Usability requirement**
* The system shall allow the users to access the system form the internet using HTML or its derivative technologies. The system uses a web browser as an interface.
* Since all users are familiar with the general usage of browser, no specific training is required.
* The system is user friendly and self-explanatory.
  + 1. **Availability requirement**

1. In case of failure of one server the other must be made available.
2. The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs.
3. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also, in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator.
4. The service should not be denied at any time to any authorized user. Routine maintenance should be performed at hours of low user activity, to reduce the number of users inconvenienced. If the platform is not available for use by any user, at any point, efforts should be taken to make it available with as little delay as possible.
5. The system is available 100% for the user and is used 24 hours a day and 365 days a year.
   * 1. **Capacity requirement**
        + 1. The system is capable of handling 250 users a time.
          2. A large amount of data will be stored on such a platform; hence the data storage should be as efficient as possible. Data access and data management should be efficient. Since many users do not have large amounts of internet data that can be used, the platform should efficiently use network resources, and not consume unnecessary data bytes for communication. It should also not store much cache data on the user side, to keep the application size small.
     2. **Serviceability requirement**
        1. Mean Time between Failures (MTBF): The system will be developed in such a way that it may fail once in a year.
        2. Online help is provided for each of the feature available with the System. All the applications provide an on-line help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, etc) with aspects of technical writing (organization, presentation). Online help is provided for each and every feature provided by the system.
        3. The User Manual describes the use of the system to Admin. It describes the use of the system on mobile systems. The user manual should be available as a hard copy and also as online help.
        4. An installation document will be provided that includes the installation instructions and configuration guidelines, which is important to a full solution offering. Also, a Read Me file is typically included as a standard component. The Read Me includes a “What’s New with This Release” section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the Read Me file.
        5. Since the installation of Online Job Portal System is a complex process, our experts will do it. So, an installation Guide will not be provided to the user.
     3. **Manageability requirement**

The software must be looked upon evey hour for any instability or any unlawful activity. The number of users per hour should be tracked, the number of users returning to the system and failure statistics should be noted.

* + 1. **Recoverability requirement**
       1. Recoverability is the ability to restore function and data in the event of a failure. The failure of system must be found within 30minutes and repair should take less than 2 hours.
       2. The system should be robust and recovery schemes must be provided to restore the system state in case of a crash. User data should be backed up at periodic intervals. The software size should be kept as small as possible, since smaller systems have lower chances of crashing, and are therefore more robust. The code should also be capable of handling edge cases, and the implementation should be abstracted to prevent modification by any outsider.
    2. **Interoperability requirement**
       1. The system can be adapted to run on computers other than the one for which it was designed.
       2. Degree of hardware independence: There is no interdependence between the hardware components
       3. Implementation language: the JSP pages can be run on all web browsers
       4. The agencies balance amount that will be calculated and sent to the billing system shall be compatible with the data types and design constraints of the billing system.
       5. The application is HTML and scripting language based. So The end-user part is fully portable and any system using any web browser should be able to use the features of the system, including any hardware platform that is available or will be available in the future.
       6. An end-user is using this system on any OS; either it is Windows or Linux.
       7. The system shall run on PC, Laptops, and PDA etc.
    3. **Reliability requirement**
       1. Mean Time Between Failures (MTBF): The system will be developed in such a way that it may fail once in a year.
       2. Mean Time to Repair (MTTR): Even if the system fails, the system will be recovered back up within an hour or less.
       3. Accuracy: The accuracy of the system is limited by the accuracy of the speed at which the employees of the agency and job seekers use the system.
       4. Access Reliability: The system shall provide 100% access reliability.
       5. The system provides storage of all databases on redundant computers with automatic switchover.
       6. The reliability of the overall program depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes.
       7. Thus, the overall stability of the system depends on the stability of container and its underlying operating system.
       8. The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do.
    4. **Maintainability requirement**

1. The maintenance of the system shall be done as per the maintenance contract.
2. A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the program will be done.
3. Also, the software design is being done with modularity in mind so that maintainability can be done efficiently.
   * 1. **Scalability requirement**
4. There are several ways of measuring efficiency which is a key component of scalability:
   * 1. Concurrent requests per second (250 users)
     2. Average response time per request (1 second)
     3. Number of records processed per second/minute (500)
5. The scalability can be affected by: -
   * 1. More users hitting the server (i.e., more web traffic)
     2. More data in the database (i.e., queries take longer, or processing takes longer)
     3. Hard drive failure in a RAID (storage performance/reliability is affected)
     4. Network saturation
6. With the increase in users over 500 that is double the capacity of the system the, performance will be affected and would slower down. To integrate greater number of users the bandwidth of the server and ability to respond to queries should be continuously monitored and improved.
7. To “Scale up” is when you upgrade a machine to a more powerful machine (e.g. faster CPU, faster GPU, engine with more HP, etc…) to get more processing power.
8. To increase processing power, you decide to buy a new computer with an 8-core 3.4GHz CPU to replace your old computer.
9. Scale out is when you increase the number of processing machines (computers, processors, servers, etc) to increase processing power. In case of excess traffic the server should be increased from one to three. One must scale out processes to achieve redundancy when you want to configure a highly available environment.
10. To scale database partitioning of large tables, based on ranges of values in a key field. In this manner, the database can be scaled out across a cluster of separate database servers. (MPP)

***CONCLUSION:*** Thus, from this experiment I drafted the software requirement specification document for the case study of Job Recruitment System.

*191071902*

*Srushti Shah*

*T. Y. B. Tech. C. S.*

***Experiment No. 2***

***AIM*:** OOAD USING UML

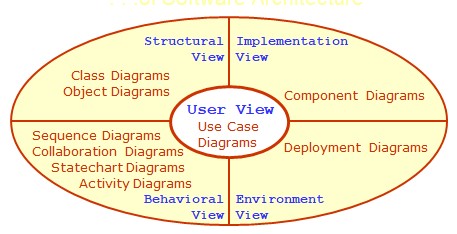
***THEORY:***

(i) Give structural view and Behavioral view.

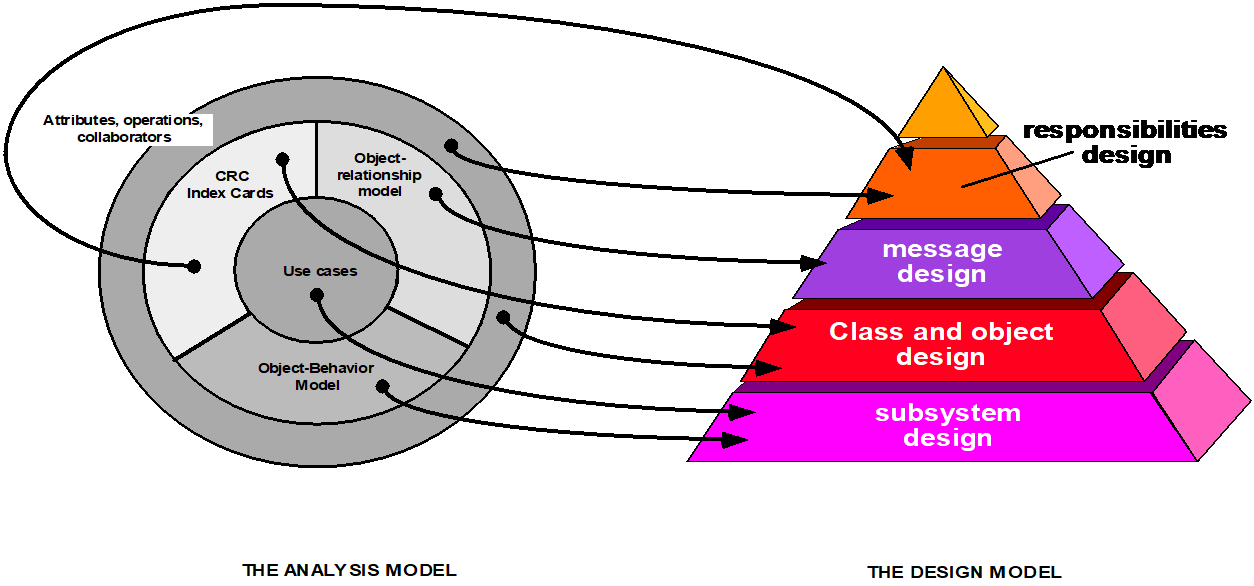
(iii) Implement the project and illustrate implementation view.

(iii) Illustrate the environmental view with specifications.

Use the following dimensions for answering your experiment.



**Map the analysis Model with design Model. Use the following dimensions for the design of object oriented software.**



Design the software architecture, algorithm and data structure design and authoring system for the given project.

* + 1. ***User View***

***USE CASE DIAGRAM***

|  |
| --- |
|  |
| ***Use Case Diagram: Job Recruitment System*** |
|  |
| ***Use Case Diagram for Actor Administration Interaction*** |
|  |
| ***Use Case Diagram for Actor Recruiter*** |
|  |
| ***Use Case Diagram for Actor Job Seeker*** |

***USE CASE TEMPLATE***

## Use Case: Registration

|  |  |
| --- | --- |
| Name | Register |
| Summary | A new user (a user which doesn’t have an account) should first register into the system to use it. This feature will allow the user to enroll into the system if the user is a new user and doesn’t already have an existing account. |
| Rationale | The user can access the system |
| Actor | Job Seeker and Recruiter |
| Pre-Condition | None |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the registration. 3. User enters the valid details required to join the system 4. Submits the form |
| Post- Condition | User is registered |
| Alternate Flow | User goes back to the Login Page |
| Exception | Username already taken |

## Use Case: Login

|  |  |
| --- | --- |
| Name | Login |
| Summary | Job seeker can change information into their profiles which include profile edit, delete, and update. |
| Rationale | After registration of the user, the valid user can login into the system |
| Actor | All Users |
| Pre-Condition | The login name and password should match with the login name and password provided while registering.  If the username and or password do not match, the user cannot login successfully into the system. |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the login form 3. The user writes its username, password and type 4. If valid, user can login into the system. If the username and or password do not match, the user cannot login successfully into the system. |
| Post- Condition | User goes to his/her dashboard |
| Alternate Flow | * + 1. The username and password is blank     2. The type of user is not specified     3. The username and password do not match |

## Use Case: Manage Profile

|  |  |
| --- | --- |
| Name | Manage Profile |
| Summary | User can change information into their profiles which include profile edit, delete, and update. |
| Rationale | If there is unnecessary date and fake profiles that are running on the site then Admin will have rights to remove or delete the profile. |
| Actor | All Users |
| Pre-Condition | Profile and record must exist |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI of different pages. 3. User will input the necessary data into the fields. 4. System will update the record as per requirements. |
| Post- Condition | User receives the notification that profile has been approved or profile has been updated. |
| Alternate Condition | * + - 1. The details enter to not validate, e.g. age is <18, date of bitrth is out of range. |

## Use Case: Upload CV

|  |  |
| --- | --- |
| Name | Upload CV |
| Summary | Job seeker can post his/her CV on job portal |
| Actor | Job seeker |
| Pre-Condition | New Jobs and other features should be displayed. |
| Basic course of Event | 1. Job seeker search for a specific job category. 2. The system displays GUI for the new jobs and features. 3. Job seeker clicks on the new jobs option and upload his/her CV there. 4. System will show all the CV on the admin and job seeker end. |
| Post- Condition | CV successfully uploaded. |
| Alternate Flow | * + - 1. The type of file uploaded does not match the required file type       2. The size of file is more than the maximum available size |

## Use Case: Apply/Search for Job

|  |  |
| --- | --- |
| Name | Search/Apply Jobs |
| Summary | Job seeker can search for jobs related to his interest using filters, if interested he can apply for the job |
| Rationale | Job Seeker is searching for a job |
| Actor | Job Seeker |
| Pre-Condition | Job Seeker must have an account |
| Basic course of Event | 1. Jobseeker access the required page to search the job. 2. System will show the appropriate GUI for searching and filtering the jobs 3. Job Seeker will fill all the required fields and describe the category and type of a job. 4. If, he/she is interested he will Apply for the job |
| Post- Condition | Apply for the job and wait for Interview notification. |
| Alternate Flow | The search criteria entered has no jobs listed |

## Use Case: Job Application Status

|  |  |
| --- | --- |
| Name | Job application status |
| Summary | Job seeker can check its status either their application is accepted. |
| Rationale | If their application is accepted and rejected, then they must know about the status, so that they can further proceed. |
| Actor | Job seeker |
| Pre-Condition | Job should have been applied for |
| Basic course of Event | 1. The job seeker login to system and check status of his application 2. The system displays GUI for the job application page. 3. Job seeker clicks on the page and check its status. 4. System will show all the results regarding their post on the jobs. |
| Post- Condition | Result successfully shown to the job seeker |

## Use Case: Post jobs

|  |  |
| --- | --- |
| Name | Post Jobs |
| Summary | Recruiter can post the job |
| Rationale | Recruiter wants to find the right employees for specific job. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must have a task and reasonable price for that task |
| Basic course of Event | 1. Recruiter access the required page to post the job. 2. System will show the appropriate GUI for posting the project. 3. Recruiter will fill all the required fields and describe the category and type of a project. 4. System will post the job. |
| Post- Condition | Anyone can see the posted job and apply for it after registration. |
| Alternate Flow | * + - 1. The job details are not sufficient enough to be posted |

## User Case: View Job seeker profile

|  |  |
| --- | --- |
| Name | View Job seeker profile |
| Summary | Recruiter can search the job seeker by name and can view the whole portfolio of job seeker. |
| Rationale | If a Recruiter has a job then he can search the job seeker and see entire profile of all employees or jobseekers. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must be existing in a list and Job Seeker has applied for the Job |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results. 3. System will show the entire job seeker in a relevant field in which Recruiter wants. |
| Post- Condition | Recruiter can see whole information about the job seeker. |

## Use Case: Hire Jobseeker

|  |  |
| --- | --- |
| Name | Hire Job seeker |
| Summary | Recruiter can hire the employee for a specific job required |
| Rationale | Recruiter has a job and wants to search the employee for the project if found, Recruiter will hire schedule the interview. |
| Actor | Recruiter |
| Pre-Condition | Recruiter should have posted a job. |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results 3. System will show the entire list of job seeker. 4. System will send the interview notification to the job seeker. |
| Post- Condition | Job seeker will receive the notification that he or she has been asked for interview. |

## User Case: Handle Users

|  |  |
| --- | --- |
| Name | Handle Users |
| Summary | Admin can view all the user’s proceedings. |
| Rationale | If admin wants to job seeker and recruiter to be added to the system or denied. |
| Actor | Admin |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. Admin verifies the details and credentials of the user 3. If valid, he admits them into the system, else removes them |
| Post- Condition | Job Seeker and Recruiter’s account will be created |

## User Case: Feedback

|  |  |
| --- | --- |
| Name | Feedback |
| Summary | To get a feedback from the user of the system |
| Rationale | To improve the system and understand the bugs |
| Actor | Recruiter and Job Seeker |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. GO to the feedback tab from dashboard 3. Enter and submit the feedback |
| Post- Condition | The developer receives the feedback |

1. ***Structural View***

***CLASS DIAGRAM***

1. Identify the Classes
2. User
   * 1. jobseeker
     2. Recruiter
     3. Admin
3. Feedback
4. Job
5. Interview
6. Result
7. Resume
8. Identify Attributes
   1. User

+ Full\_Name: String

+ username: String

# password: String

* + 1. Jobseeker

+JobSeekerID:int

-designation: String

-location:String

-experience:String

-email:String

-mobile:int

-keyskills: Array

-projects: Array

-ITSkills: Array

-Education:String

-Certification:String

-DOB: Date

/age:int

-gender: Gender

-pincode: Int

-LanguagesKnown: String

* + 1. Recruiter

+RecruiterID: int

-Company: String

-Designation: String

-Location:String

-email:String

-mobile:Int

-gender: Gender

-DOB : Date

/age:int

* + 1. Admin
  1. Feedback

- message: String

- name: String

- rating: String

* 1. Job

+JobID:int

-Title:String

-Location:String

-Experience:Int

-Salary:Float

-Description: String

-Role: String

-KeySkills: String

-Vacancy: Boolean

-Recruiter: String

-DateofPosting: Date

* 1. Interview

+InterviewID:int

-InterviewDate: Date

-Interviewer: String

-Interviewee: Job

-IntervewLcoation

* 1. Result

+JobID

+interviewID

* 1. Resume

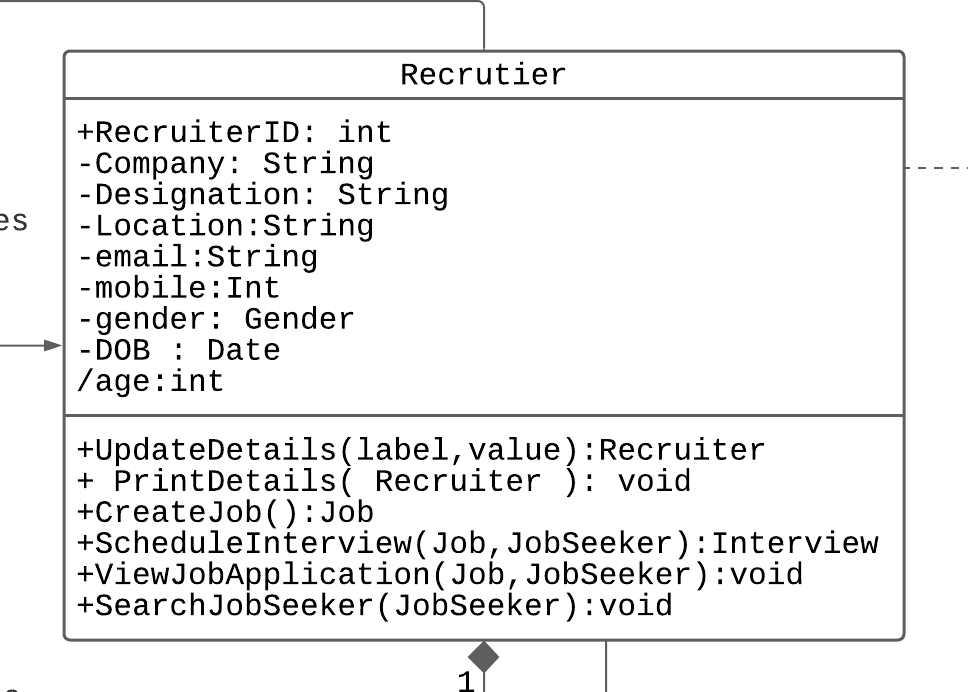
+JobSeekerID: int

-ResumeFIle: file

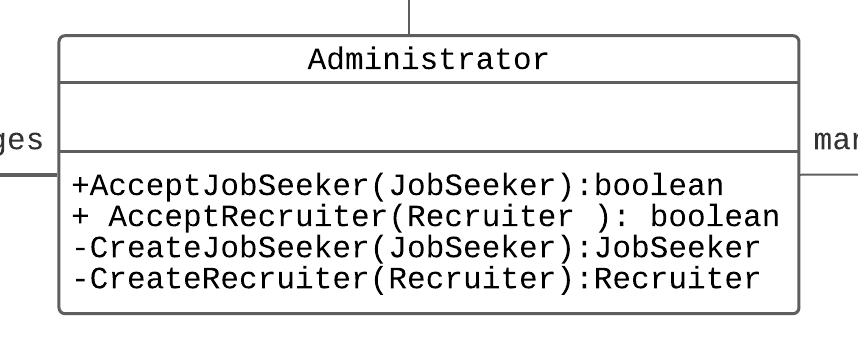
1. Identify functions
2. User



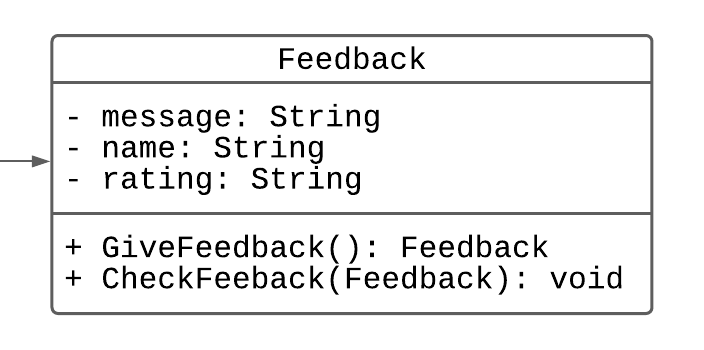
* + 1. jobseeker 
    2. Recruiter



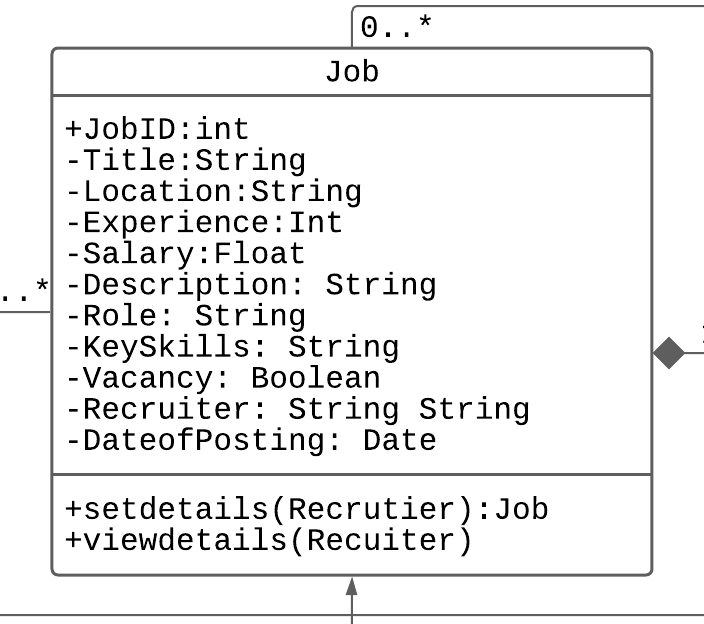
* + 1. Admin



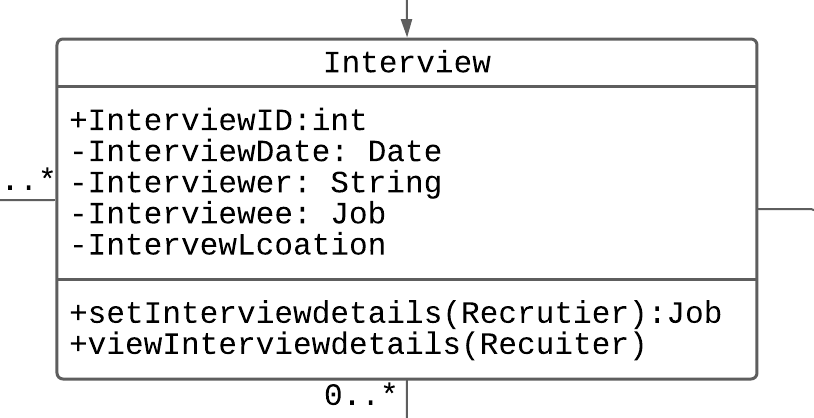
1. Feedback



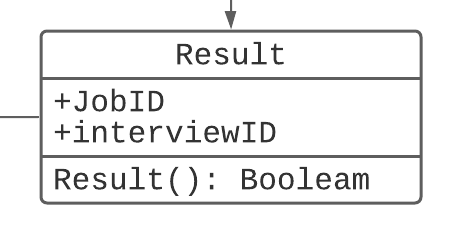
1. Job



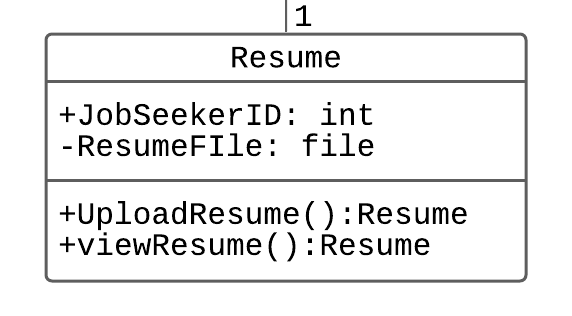
1. Interview



1. Result



1. Resume



1. Identify relationships

1:1

User gives feedback

Resume belongs to a jobseeker

One interview has a result

1:n

A recruiter posts many jobs

A recruiter scehdules many interview

Admin manages many job seekers

Admin manages many recruiters

A job has many interviews

N:n

Jobseeker applies/searches for jobs

1. Constrain

Apply Job: precondition: vacancy>0

After interview if result==true, then c vacancy--;

|  |
| --- |
|  |
| ***Class Diagram: Job Recrutiement System*** |

1. ***Behavioural View***

***3.1 Sequence Diagram***

|  |
| --- |
|  |
| ***3.1.1 Sequence Diagram: Register and Login*** |
| ***Description:***   1. The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. 2. It helps in envisioning several dynamic scenarios. 3. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. 4. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. 5. It incorporates the iterations as well as branching. 6. Here, the login and register usecases are shown 7. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system 8. While, loggining in only if his username and password match with the user type he has selected, he is allowed to login 9. On Login he is directed to the Menu 10. Before logging in a user has to be registered with system |

|  |
| --- |
|  |
| * + 1. ***Sequence Diagram: Jobseeker Activity*** |
| ***Description:***   1. The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. 2. It incorporates the iterations as well as branching. 3. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system, whose segment is added as reference 4. It shows the sequence of use case:   Search for the job by applying filters  Apply for the job  Management for profile using the database and updation of details   1. Checking of job status of applied jobs, which can be done only when you have applied for a job |

|  |
| --- |
|  |
| * + 1. ***Sequence Diagram: Recruiter Activity*** |
| ***Description:***   1. The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. 2. It incorporates the iterations as well as branching. 3. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system, whose segment is added as reference 4. The Recruiter can post a job, where in he has to enter the details for the same, after posting the job he can also view it 5. Check the applications that have arrived for a particular job 6. The Recruiter can check for the jobseeker’s profile, if he likes it, he can send them a message saying they are selected for the interview |

* 1. ***Collaboration Diagram***

|  |
| --- |
|  |
| ***3.2.1 Collaboration Diagram: Login*** |
| ***Description:***   1. The collaboration diagram is used to show the relationship between the objects in a system. 2. Both the sequence and the collaboration diagrams represent the same information but differently. 3. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. 4. Multiple objects present in the system are connected to each other. 5. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system 6. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system 7. While, loggining in only if his username and password match with the user type he has selected, he is allowed to login 8. On Login he is directed to the Menu |

|  |
| --- |
|  |
| ***3.2.2 Collaboration Diagram : JobSeeker and Recruiter Activity*** |
| ***Description:***   1. The collaboration diagram is used to show the relationship between the objects in a system. 2. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system 3. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system 4. The Job Seeker who wants to apply for a job, searches for it uses filters 5. The system return with the list of jobs after filtering the data 6. If a Job Seeker is interested in applying fro job, he does so by sending an applciation via message to the Recruiter 7. The recrutier on recepeit of this, checks the profile of jobseeker 8. The recruteir schedules an interview via sending a message to the Jobseeker about being selected |

***3.3 State Transition Diagram***

|  |
| --- |
|  |
| ***3.3.1 State Transition: Login*** |

|  |
| --- |
| ***Description of State Transition Diagram for Login***   1. ***State: Idle***    1. Entry: initialise the application    2. Do: wait for action 2. ***State: Authentication***    1. Entry: Click Login after entering username, password and usertype    2. Do: verify details    3. exit: authentication check 3. ***State: Home Page***    1. Entry: valid credentials    2. Do: activity (depending on the user)    3. Exit: logout 4. ***State: Logout***     1. Entry: clickLogout    2. Do: delete cookies from the browser    3. Exit: end |

|  |
| --- |
|  |
| ***3.3.2 State Transition: Recruiter Activity*** |

|  |
| --- |
| ***Description for: State Transition: Recruiter Activity***   1. ***Idle***     1. Entry: after login wait for activity    2. Do: wait for action 2. ***Post job***    1. Entry: details of the job are entered like the JOBID, position, location, experience required, salary, etc    2. Do: create job and add it to the database    3. Exit: created 3. ***View posted job***    1. Do: view details of the posted job 4. ***Check Candidate***    1. Entry: check message from job seeker    2. Do: check profile for the job seeker that has applied    3. Exit: go to JobProfile 5. ***Select Job Seeker***    1. Entry: sendmessage to the candidate if he is selected    2. Do: If the candidate is appropriate, schedule interview |

|  |
| --- |
|  |
| ***3.3.3 State Transition:JobSeeker Activity*** |

|  |
| --- |
| ***Description for: State Transition: Recruiter Activity***   1. ***Idle***    1. Entry: initialise    2. Do: wait for action 2. ***Search for Job***    1. Entry: filter details like location, salary, experience should be entered    2. Do: filter jobs based on the details from the job database    3. Exit: choose a particular job of your choice 3. ***Display Details***    1. Entry: Click on the job of your choice to know more about it    2. Do: Check the details    3. Exit: apply for the job, if the job matches your requiremnet 4. ***Apply Jobs*** 5. Entry: click apply 6. Do: send the recruiter a message |

***3.4 Actvity Diagram***

|  |
| --- |
|  |
| ***Activity Diagrams: Job Recuritement System*** |

|  |
| --- |
| **Description: Activity Diagram**   1. In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. 2. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. 3. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system 4. While, loggining in only if his username and password match with the user type he has selected, he is allowed to login 5. On Login he is directed to the Menu   **Admin:** Manage recruiters/job Seekers: He can view/remove the users from system  Recruiter   * + - 1. Create Job: In this interface, the recruiter who has registered themselves in the system can post the jobs.       2. Schedule Interview: Once, a recruiter is satisfied with the skills and resume of a job seeker he can schedule an interview for the job of that job seeker, for knowing him further, if he is a feasible candidate or not       3. Edit Profile: The recruiter can make modifications in his profile containing personal details, job details and other information       4. View Job Seekers: The recruiter can search a particular job seeker according to his skills or past positions       5. Feedback: The recruiter can give a feedback about the system or report any bugs if found   Job Seeker   * + - 1. Search for Jobs and Apply for Jobs       2. Edit Profile: This feature allows the job seeker to update his/her skills and profile       3. Check status for the job application applied to       4. Upload Resume: The job seeker has an option to upload a file as his resume t       5. Feedback: The recruiter can give a feedback about the system or report any bugs if found |

1. ***Implementation View***
   1. ***Component Diagram***

|  |
| --- |
| ***Component Diagram: Job Recrutierment System*** |
|  |

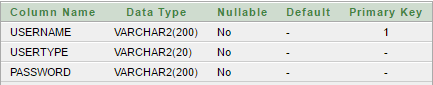
|  |
| --- |
| **Description: Component Diagram**   1. A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. 2. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. 3. It visualizes the relationships as well as the organization between the components present in the system. 4. It helps in forming an executable system. 5. A component is a single unit of the system, which is replaceable and executable. 6. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces. 7. Here, the User (Job Seeker, Admin or Recruiter) is logins into the system 8. While, loggining in only if his username and password match with the user type he has selected, he is allowed to login 9. On Login he is directed to the respective Menu 10. Each Menu has its own subroutines as shown in the package of Admin, Jobseeker and Recruiter 11. Each user is connected to the database tables for communication |

* 1. ***Deployment Diagram***

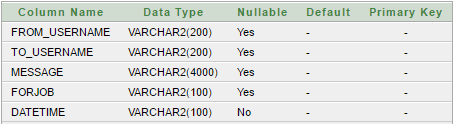
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
| ***Deployment Diagram: Job Recrutiement System*** | | | | | | |
|  | | | | | | |
| Machine | **Software Configuration** | **Hardware Configuration** | **Operating System** | **Compiler** | **Any other software** | **Software Modules and path** |
| Client | Google Chrome  /Internet Explorer/  any browser with JSP Support |  | Windows 7 and after | Java | NetBeans, | GUI |
| Application Server | Sun GlassFish 4.1.1. Server | * Sun GlassFish Enterprise Server: 35 MB minimum * SDK: 250 MB minimum * 1GB Memory | Windows 7 and after, Red Hat, MacOS | HTML, Java, JSP,  SQL | Internet Explorer, Chrome, Mozilla, Safari |  |
| Database Server | MySQL Connector/J Driver 5.1 |  | Windwos 7 & after, Unbuntu |  |  | tables |

* 1. ***Data Structure***

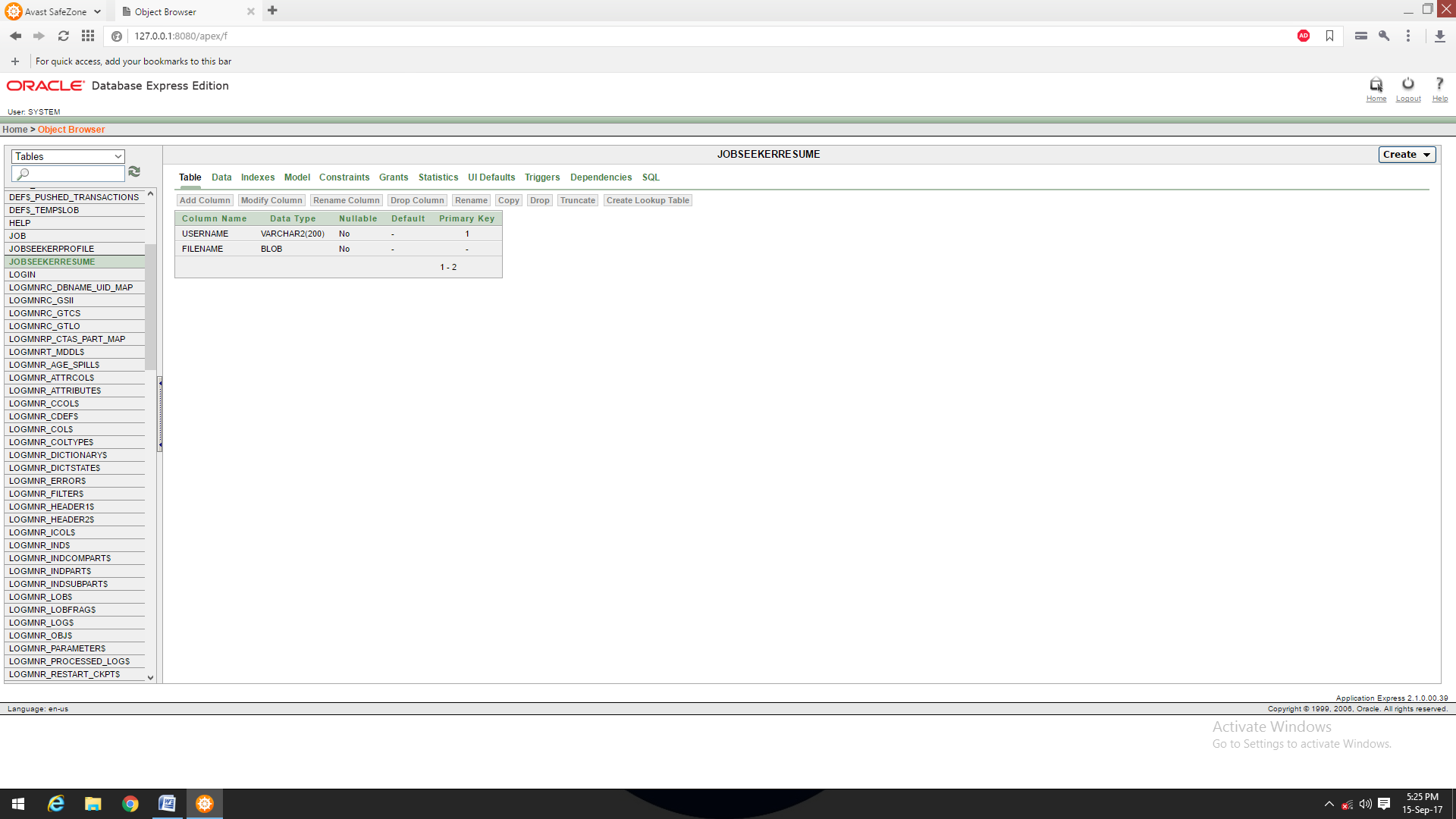
LOGIN

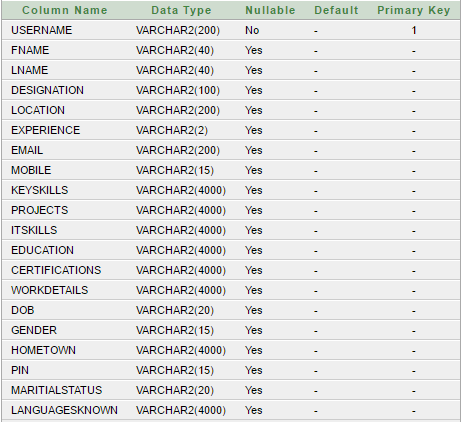


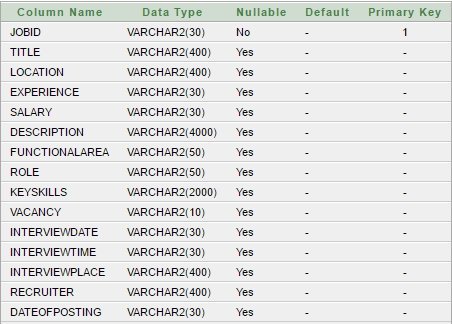
MESSAGE



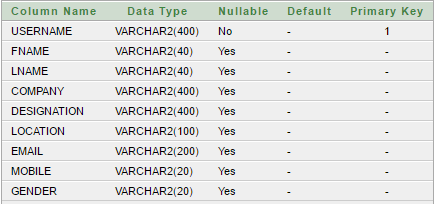
JOBSEEKERRESUME



JOBSEEKERPROFILE

JOB

RECRUITERPROFILE



* 1. ***ALGORITHM DESIGN***

## Use Case: Registration

|  |  |
| --- | --- |
| Name | Register |
| Summary | A new user (a user which doesn’t have an account) should first register into the system to use it. This feature will allow the user to enroll into the system if the user is a new user and doesn’t already have an existing account. |
| Rationale | The user can access the system |
| Actor | Job Seeker and Recruiter |
| Pre-Condition | None |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the registration. 3. User enters the valid details required to join the system 4. Submits the form |
| Post- Condition | User is registered |
| Alternate Flow | User goes back to the Login Page |
| Exception | Username already taken |

**Algorithm:**

**Name: Registration**

**Input:** username, password, usertype;

**Output:**  user added to the system

**Steps (Procedure)**

* + 1. Start
    2. Go to Registration Page
    3. Enter the username and usertype
    4. Enter password and confirm password
    5. If username!=NULL && usertype!=NULL &&password!=NULL && confirm\_password!=NULL then

If password==confirm\_password

If username already exists in table then enter another username

1. Else: Enter the entry into login table of database

Else

Alert: Password and confirm password do not match

Try again

1. Else:
   * + - 1. The field is required, do not leave it empty

## Use Case: Login

|  |  |
| --- | --- |
| Name | Login |
| Summary | Job seeker can change information into their profiles which include profile edit, delete, and update. |
| Rationale | After registration of the user, the valid user can login into the system |
| Actor | All Users |
| Pre-Condition | The login name and password should match with the login name and password provided while registering.  If the username and or password do not match, the user cannot login successfully into the system. |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the login form 3. The user writes its username, password and type 4. If valid, user can login into the system. If the username and or password do not match, the user cannot login successfully into the system. |
| Post- Condition | User goes to his/her dashboard |
| Alternate Flow | 1. The username and password is blank 2. The type of user is not specified 3. The username and password do not match |

**Algorithm:**

**Input:** username, password, usertype;

**Output:**  allow/denied access to home page

**Pseudo Code:**

1. Start
2. Login Page open
3. Enter the username, password, type input
4. If password==NULL | username==NULL|| type==NULL
   1. The field is required
5. Retrieve the entry from database for the given username
   1. If username=username(database) && password=password(database)&& type=usertype
   2. Go to Dashboard
6. Else
   1. Alert: Invalid entry, please try again
7. End

## Use Case: Manage Profile

|  |  |
| --- | --- |
| Name | Manage Profile |
| Summary | User can change information into their profiles which include profile edit, delete, and update. |
| Rationale | If there is unnecessary date and fake profiles that are running on the site then Admin will have rights to remove or delete the profile. |
| Actor | All Users |
| Pre-Condition | Profile and record must exist |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI of different pages. 3. User will input the necessary data into the fields. 4. System will update the record as per requirements. |
| Post- Condition | User receives the notification that profile has been approved or profile has been updated. |
| Alternate Condition | The details enter to not validate, e.g. age is <18, date of bitrth is out of range. |

**Algorithm:**

**Input:** username

**Output:** details

**Pseudo Code:**

1. Start
2. Go to Profile section
3. Update()
4. End

**Procedures**

*Update()*

1. Click on Edit

2. Choose the fields you want to edit

3. Click on Update

## Use Case: Upload CV

|  |  |
| --- | --- |
| Name | Upload CV |
| Summary | Job seeker can post his/her CV on job portal |
| Actor | Job seeker |
| Pre-Condition | New Jobs and other features should be displayed. |
| Basic course of Event | 1. Job seeker search for a specific job category. 2. The system displays GUI for the new jobs and features. 3. Job seeker clicks on the new jobs option and upload his/her CV there. 4. System will show all the CV on the admin and job seeker end. |
| Post- Condition | CV successfully uploaded. |
| Alternate Flow | * + - 1. The type of file uploaded does not match the required file type       2. The size of file is more than the maximum available size |

**Algorithm:**

**Input:** Resume File

**Pseudo Code:**

1. Start
2. Go to Upload Resume section
3. Click on choose file
4. Choose the file from your computer
5. Click on Upload
6. End

## Use Case: Job Application Status

|  |  |
| --- | --- |
| Name | Job application status |
| Summary | Job seeker can check its status either their application is accepted. |
| Rationale | If their application is accepted and rejected, then they must know about the status, so that they can further proceed. |
| Actor | Job seeker |
| Pre-Condition | Job should have been applied for |
| Basic course of Event | * + - 1. The job seeker login to system and check status of his application       2. The system displays GUI for the job application page.       3. Job seeker clicks on the page and check its status.       4. System will show all the results regarding their post on the jobs. |
| Post- Condition | Result successfully shown to the job seeker |

**Algorithm:**

**Pseudo Code:**

1. Start
2. Go to messages tab
3. Check for the application status
4. End

## Use Case: Apply/Search for Job

|  |  |
| --- | --- |
| Name | Search/Apply Jobs |
| Summary | Job seeker can search for jobs related to his interest using filters, if interested he can apply for the job |
| Rationale | Job Seeker is searching for a job |
| Actor | Job Seeker |
| Pre-Condition | Job Seeker must have an account |
| Basic course of Event | 1. Jobseeker access the required page to search the job. 2. System will show the appropriate GUI for searching and filtering the jobs 3. Job Seeker will fill all the required fields and describe the category and type of a job. 4. If, he/she is interested he will Apply for the job |
| Post- Condition | Apply for the job and wait for Interview notification. |
| Alternate Flow | * + - 1. The search criteria entered has no jobs listed |

**Algorithm:**

**Input:** keyword, location, experience, expected salary

**Output:** List of Matching Jobs

**Pseudo Code:**

* + - 1. Prompt the user for search query
      2. Check whether entry is valid or not
      3. If entry is invalid
  1. Then Information to the user & prompt again for entry
     1. Else
  2. check with database
  3. Show list of jobs
  4. Select the job of your choice
  5. The details of the same will be displayed
  6. Apply for the job
  7. If vacancy>0
     1. Successfully Applied
  8. Else
     1. Alert: “No vacancy, search other jobs”

5. End

## Use Case: Hire Jobseeker

|  |  |
| --- | --- |
| Name | Hire Job seeker |
| Summary | Recruiter can hire the employee for a specific job required |
| Rationale | Recruiter has a job and wants to search the employee for the project if found, Recruiter will hire schedule the interview. |
| Actor | Recruiter |
| Pre-Condition | Recruiter should have posted a job. |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results 3. System will show the entire list of job seeker. 4. System will send the interview notification to the job seeker. |
| Post- Condition | Job seeker will receive the notification that he or she has been asked for interview. |

**Algorithm:**

**Input:** list\_of\_applied\_candidate

**Pseudo Code:**

1. Start
2. Go to Messages tab
3. Check the list of applications
4. Accept the one, you want
5. End

## Use Case: Post jobs

|  |  |
| --- | --- |
| Name | Post Jobs |
| Summary | Recruiter can post the job |
| Rationale | Recruiter wants to find the right employees for specific job. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must have a task and reasonable price for that task |
| Basic course of Event | 1. Recruiter access the required page to post the job. 2. System will show the appropriate GUI for posting the project. 3. Recruiter will fill all the required fields and describe the category and type of a project. 4. System will post the job. |
| Post- Condition | Anyone can see the posted job and apply for it after registration. |
| Alternate Flow | * + - 1. The job details are not sufficient enough to be posted |

**Algorithm:**

**Input:** jobID, Title, Lactation, Experience, Salary, Description, KeySkills, Role, Vacancy, Recruiter, DateofPosting

**Pseudo Code:**

1. Start
2. Go to Post a Job Tab
3. Initialize: JobID
4. Initialize: Title
5. Initialize: Location
6. Initialize: Experience
7. Initialize: Salary
8. Initialize: Description
9. Initialize: Role
10. Initialize: KeySkills
11. Initialize: Vacancy
12. Initialize: Recruiter
13. Initialize: DateofPosting
14. Click on Post Job
15. End

## Use Case: View Job seeker profile

|  |  |
| --- | --- |
| Name | View Job seeker profile |
| Summary | Recruiter can search the job seeker by name and can view the whole portfolio of job seeker. |
| Rationale | If a Recruiter has a job then he can search the job seeker and see entire profile of all employees or jobseekers. |
| Actor | Recruiter |
| Pre-Condition | Recruiter must be existing in a list and Job Seeker has applied for the Job |
| Basic course of Event | 1. The user indicates that the system is to perform a search function. 2. System responds by the requesting the search term and shows the results. 3. System will show the entire job seeker in a relevant field in which Recruiter wants. |
| Post- Condition | Recruiter can see whole information about the job seeker. |

**Algorithm:**

**Input:** jobseekerName, JobseekerID

**Output:** Jobseeker Profile

**Pseudo Code:**

1. Start
2. Go to Search Jobseeker tab
3. Enter the name of jobseeker
4. If no entry
   1. Display: No such job seeker
5. Else  
   b. From the list of searches, chose the one you want to view profile of
6. End

## Use Case: Handle Users

|  |  |
| --- | --- |
| Name | Handle Users |
| Summary | Admin can view all the user’s proceedings. |
| Rationale | If admin wants to job seeker and recruiter to be added to the system or denied. |
| Actor | Admin |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. Admin verifies the details and credentials of the user 3. If valid, he admits them into the system, else removes them |
| Post- Condition | Job Seeker and Recruiter’s account will be created |

**Algorithm:**

**Input:** jobseekerID, RecrutierID

**Pseudo Code:**

1. Start
2. Login as Admin
3. Go to Jobseeker Report
   1. View or Remove Job Seeker
4. Go to Recruiter Report
   1. View or Remove Job Seeker
5. End

## Use Case: Feedback

|  |  |
| --- | --- |
| Name | Feedback |
| Summary | To get a feedback from the user of the system |
| Rationale | To improve the system and understand the bugs |
| Actor | Recruiter and Job Seeker |
| Pre-Condition | Job Seeker or Recruiter has registered through the website |
| Basic course of Event | 1. Job Seeker or Recruiter has registered through the website 2. GO to the feedback tab from dashboard 3. Enter and submit the feedback |
| Post- Condition | The developer receives the feedback |

**Algorithm:**

**Input:** username, message

**Pseudo Code:**

1. Start
2. Go to feedback section
3. Enter username, email, feedback
4. Click Send
5. End

***5. GUI DESIGN and IMPLEMENTATION (CODE)***

|  |
| --- |
| * + - 1. ***LOGIN PAGE*** |
| index.jsp |
| <html>  <head>  <link rel="stylesheet" href="style.css" type="text/css" />  <title>Login</title>  </head>  <body>  <jsp:include page="header.jsp" />  <center>  <fieldset>  <legend>Login</legend>  <form name="form1" method="post" action="loginaction.jsp">  <table>  <tr>  <td align='center' colspan=2>  <input type="radio" name="usertype" value="jobseeker" required>  JobSeeker <input type="radio" name="usertype" value="recruiter" required>  Recruiter <input type="radio" name="usertype" value="admin" required>Admin  </td>  </tr>  <tr>  <td>&nbsp</td>  </tr>  <tr>  <td>Email</td>  <td><input name="username" type="email" placeholder="Email" required></td>  </tr>  <tr>  <td>Password</td>  <td><input name="password" type="password" placeholder="Password" required></td>  </tr>  <tr>  <td></td>  <td align="center" colspan=2><input type="submit" name="Submit" value="Login"></td>  </tr>  </table>  </form>  <a href="registration.jsp">Forgot Password?</a><br>  <a href="registration.jsp">Click here to register an Account</a>  </fieldset>  <jsp:include page="footer.jsp" />  </center>  </body>  </html>  -----------------------------------------------------------------------------------  <header style="background:navy; color:white">  <center><h1>Job Recruitement System</h1></center>  </header>  <footer><hr></footer> |
| ***The login page must route to the Home Page of respective user, provided the details are correct***  **When Trying to Login with Wrong Details**  **When Trying to Login with Blank Fields** |
| LoginAction.jsp (on clicking Login) |
| <%@ page import="java.sql.\*" %>  <html>  <head> <title>Login</title></head>  <body>  <%  try {  String username=request.getParameter("username");  String password=request.getParameter("password");  String usertype=request.getParameter("usertype");  if(username.equals("admin") && password.equals("admin") && usertype.equals("admin"))  {  session.setAttribute("username",username);  session.setAttribute("usertype","admin");  response.sendRedirect("adminhome.jsp");  }  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from Login where username='"+username+"' and password='"+password+"' and usertype='"+usertype+"' ");  if(rs.next())  {  session.setAttribute("username",username);  rs.close();  stmt.close();  con.close();  if(usertype.equals("jobseeker"))  {  session.setAttribute("usertype","jobseeker");  response.sendRedirect("jobseekerhome.jsp");  }  else if(usertype.equals("recruiter"))  {  session.setAttribute("usertype","recruiter");  response.sendRedirect("recruiterhome.jsp");  }  else if(usertype.equals("admin"))  {  session.setAttribute("usertype","recruiter");  response.sendRedirect("adminhome.jsp"); }  }  else {  %>  Invalid Username or Password or User Type  <jsp:include page="index.jsp" />  <%} }  catch(NullPointerException e)  { out.print("Please Enter Login Details");  %>  <jsp:include page="index.jsp" />  <% }  catch(Exception e)  {out.print(e);}%>  </body>  </html> |
| * + - 1. ***NEW USER REGISTRATION PAGE*** |
| registration.jsp |
| <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title>New User Registration</title>  </head>  <body>  <jsp:include page="header.jsp" />  <fieldset>  <legend>Registration Form</legend>  <p style="color:red;" id="divCheckPasswordMatch"></p>  <form method="post" action="registrationaction.jsp" name="myForm">  <table align='center'>  <tr>  <td align='center' colspan=2><input type="radio" name="usertype" value="jobseeker" required>JobSeeker <input type="radio" name="usertype" value="recruiter" required>Recruiter </td>  </tr>  <tr><td>&nbsp;</td></tr>  <tr>  <td align='left'>Username</td>  <td><input type='text' name='username' placeholder="Username" required></td>  </tr>  <tr>  <td align='left'>Password</td>  <td> <input type="password" name="password" placeholder="Password" id="password" required></td>  </tr>  <tr><td align='left'>Confirm Password</td>  <td><input type="password" placeholder="Confirm Password" id="confirm\_password" onChange="checkPasswordMatch();" onClick="style.backgroundColor='white'" required></td>  </tr>  <script type="text/javascript">  function checkPasswordMatch() {  var password = document.getElementById("password").value;  var confirmPassword = document.getElementById("confirm\_password").value;  if (password != confirmPassword) {  document.getElementById("divCheckPasswordMatch").innerHTML = "Passwords do not match!"; document.getElementById("confirm\_password").style.backgroundColor = "red";  }  }  </script>  <table align='center'>  <tr><td align='center'><input type='submit' name='register' value="Register"></td>  </tr>  </table>  </form>  <a href="index.jsp">Click here to Login</a>  </fieldset>  <jsp:include page="footer.jsp" />  </body></html>  On click on ‘click here to register on index.js    When Passwords Don’t Match While Registration Process  When Registering Without Selecting Usertype |

|  |
| --- |
| Registrationaction.jsp (on clicking Register) |
| <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@ page import="java.sql.\*" %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>New User Registration</title>  </head>  <body>  <%  try  {  String usertype=request.getParameter("usertype");  String username=request.getParameter("username");  String password=request.getParameter("password");  String blank=" ";  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  ResultSet rs1=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  stmt.executeUpdate("insert into login values('"+usertype+"','"+username+"','"+password+"')");  session.setAttribute("username",username);  if(usertype=="jobseeker")  {  stmt.executeUpdate("insert into jobseekerprofile values('"+username+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"')");  rs.close();  con.close();  session.setAttribute("usertype","jobseeker");  response.sendRedirect("jobseekerhome.jsp");  }  else if(usertype=="recruiter"){  stmt.executeUpdate("insert into recruiterprofile values('"+username+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"','"+blank+"')");  rs.close();  con.close();  session.setAttribute("usertype","recruiter");  response.sendRedirect("recruiterhome.jsp");}  stmt.close();  con.close();  }  catch(NullPointerException e)  {out.print("Please Enter All Required Details");  %>  <jsp:include page="registration.jsp" />  <%}  catch(Exception e)  {  out.print(e);  }  %>  </body>  </html> |
| * + - 1. ***LOGOUT PAGE*** |
| Logout.jsp |
| <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  <title>Login</title>  </head>  <body>  <%  request.getSession();  session.removeAttribute("username");  session.removeAttribute("usertype");  session.invalidate();  %>  You are successfully logged out!  <jsp:include page="index.jsp" />  </body>  </html>  **----------------------------------------------------------------------------------------**  When User Do Logout From Session, he should be directed to the login page |

|  |
| --- |
| * + - 1. ***JOBSEEKER HOME PAGE*** |
| Jobseekerhome.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker - Home </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1><center> JobSeeker </center></h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <jsp:include page="footer.jsp" />  </body>  </html> |
| Jobseekermenu.jsp |
| <hr><center>  <pre style="font-family: sans-serif;font-size: 20px"> <a href="jobseekerprofile.jsp"> Profile </a> | <a href="jobseekerresume.jsp"> Upload Resume </a> | <a href="jobseekersearch.jsp"> Search Jobs </a> | <a href="jobseekermessagebox.jsp"> Messages </a> | <a href="jobseekerfeedback.jsp"> Feedback </a></pre>  </center><hr> |
|  |

|  |
| --- |
| * 1. ***PROFILE*** |
| Jobseekerprofile.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker - Profile </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Profile Details</legend>  <table border="1">  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+username+"'");  while(rs.next())  {  %>  <tr>  <th>Basic Details</th>  </tr>  <tr>  <td>Name</td>  <td><%=rs.getString("FNAME") %> <%=rs.getString("LNAME") %></td>  </tr>  <tr>  <td>Designation</td>  <td><%=rs.getString("DESIGNATION") %></td>  </tr>  <tr>  <td>Location</td>  <td><%=rs.getString("LOCATION") %></td>  </tr>  <tr>  <td>Experience</td>  <td><%=rs.getString("EXPERIENCE") %></td>  </tr>  <tr>  <th>Contact Details</th>  </tr>  <tr>  <td>Email</td>  <td><%=rs.getString("EMAIL") %></td>  </tr>  <tr>  <td>Mobile</td>  <td><%=rs.getString("MOBILE") %></td>  </tr>  <tr>  <th>Key Skills</th>  </tr>  <tr align=center>  <td colspan=2><%=rs.getString("KEYSKILLS") %></td>  </tr>  <tr>  <th>Projects</th>  </tr>  <tr align=center>  <td colspan=2><%=rs.getString("PROJECTS") %></td>  </tr>  <tr>  <th>IT Skills</th>  </tr>  <tr align=center>  <td colspan=2><%=rs.getString("ITSKILLS") %></td>  </tr>  <tr>  <th>Education</th>  </tr>  <tr align=center>  <td colspan=2><%=rs.getString("EDUCATION") %></td>  </tr>  <tr><th>Certifications</th></tr>  <tr align=center>  <td colspan=2><%=rs.getString("CERTIFICATIONS") %></td>  </tr>  <tr>  <td>Work Details</td>  </tr>  <tr align=center>  <td colspan=2><%=rs.getString("WORKDETAILS") %></td>  </tr>  <tr><td>Personal Details</td></tr>  <tr><td>DOB</td><td><%=rs.getString("DOB") %></td></tr>  <tr><td>Gender</td><td><%=rs.getString("GENDER") %></td></tr>  <td>Hometown</td>  <td><%=rs.getString("HOMETOWN") %></td>  </tr>  <td>PIN</td>  <td><%=rs.getString("PIN") %></td></tr>  <td>Maritial Status</td>  <td><%=rs.getString("MARITIALSTATUS") %></td>  </tr>  <td>Languages Known</td>  <td><%=rs.getString("LANGUAGESKNOWN") %></td>  </tr>  <tr>  <td>&nbsp</td>  </tr>  <%  }  rs.close();  stmt.close();  con.close();  %>  <tr>  <td colspan="2" align="center"><a href="jobseekerupdateprofile.jsp">Edit</a></td>  </tr>  </table>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When Jobseeker Updated Their Details  When Jobseeker Is NewBy clicking on ‘Edit’ Jobseeker can edit profile details. |
| Jobseekerupdate.jsp (on clicking edit) |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title>JobSeeker  Update Profile</title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Profile Details</legend>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+username+"'");  if(rs.next())  {  %>  <form method="post" action="jobseekerupdateprofileaction.jsp">  <table>  <tr>  <td>Basic Details</td>  </tr>  <tr>  <td>Name</td>  <td><input type="text" name="fname" placeholder="FirstName"></td>  </tr>  <tr>  <td></td>  <td><input type="text" name="lname" placeholder="LastName"></td>  </tr>  <tr>  <td>Designation</td>  <td><input type="text" name="designation"></td>  </tr>  <tr>  <td>Location</td>  <td><input type="text" name="location"></td>  </tr>  <tr>  <td>Experience</td>  <td>  <select name="experience" class="Form\_fields" required>  <option value="">- Select -</option>  <option value="0">0</option>  <option value="50">50</option>  </select>&nbsp;Years  </td>  </tr>  <tr>  <td>Contact Details</td>  </tr>  <tr>  <td>Email</td>  <td><input type="text" name="email"></td>  </tr>  <tr>  <td>Mobile</td>  <td><input type="text" name="mobile"></td>  </tr>  <tr>  <td>Key Skills</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="keyskills" placeholder="e.g. Hardworking, Time Management"></td>  </tr>  <tr>  <td>Projects</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="projects" placeholder="e.g. Online Job Search Portal in Java"></td>  </tr>  <tr>  <td>IT Skills</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="itskills" placeholder="e.g. Java,SQL,HTML,JSP"></td>  </tr>  <tr>  <td>Education</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="education" placeholder="e.g. BCA"></td>  </tr>  <tr>  <td>Certifications</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="certifications" placeholder="e.g. Certificate in HTML, JavaScript, JSP"></td>  </tr>  <tr>  <td>Work Details</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="workdetails" placeholder="e.g. Java Programmer in TCS"></td>  </tr>  <tr>  <td>Personal Details</td>  </tr>  <tr>  <td>DOB</td>  <td><input type="date" name="dob"></td>  </tr>  <tr>  <td>Gender</td>  <td><input type="radio" name="gender" value="male" required>Male<input type="radio" name="gender" value="female" required>Female</td>  </tr>  <td>Hometown</td>  <td><input type="text" name="hometown"></td>  </tr>  <td>PIN</td>  <td><input type="text" name="pin"></td>  </tr>  <td>Maritial Status</td>  <td><input type="radio" name="maritialstatus" value="single" required>Single<input type="radio" name="maritialstatus" value="married" required>Married<input type="radio" name="maritialstatus" value="divorced" required>Divorced</td>  </tr>  <td>Languages Known</td>  <td><input type="text" name="languagesknown"></td>  </tr>  <tr>  <td>&nbsp</td>  </tr>  <tr>  <td align="center" colspan=2><input type="submit" value="Update"></td>  </tr>  </table>  </form>  <%  }  while(rs.next())  {  %>  <form method="post" action="jobseekerupdateprofileaction.jsp">  <table>  <tr>  <td>Basic Details</td>  </tr>  <tr>  <td>Name</td>  <td><input type="text" name="fname" placeholder="FirstName" value=<%=rs.getString("fname") %>></td>  </tr>  <tr>  <td></td>  <td><input type="text" name="lname" placeholder="LastName" value=<%=rs.getString("lname") %>></td>  </tr>  <tr>  <td>Designation</td>  <td><input type="text" name="designation" value=<%=rs.getString("designation") %>></td>  </tr>  <tr>  <td>Location</td>  <td><input type="text" name="location" value=<%=rs.getString("location") %>></td>  </tr>  <tr>  <td>Experience</td>  <td>  <select name="experience" class="Form\_fields" required>  <option value="">- Select -</option>  <option value="0">0</option>  …….  <option value="24">24</option>  …..  <option value="50">50</option>  </select>&nbsp;Years  </td>  </tr>  <tr>  <td>Contact Details</td>  </tr>  <tr>  <td>Email</td>  <td><input type="text" name="email" value=<%=rs.getString("email") %>></td>  </tr>  <tr>  <td>Mobile</td>  <td><input type="text" name="mobile" value=<%=rs.getString("mobile") %>></td>  </tr>  <tr>  <td>Key Skills</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="keyskills" placeholder="e.g. Hardworking, Time Management" value=<%=rs.getString("keyskills") %>></td>  </tr>  <tr>  <td>Projects</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="projects" placeholder="e.g. Online Job Search Portal in Java" value=<%=rs.getString("projects") %>></td>  </tr>  <tr>  <td>IT Skills</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="itskills" placeholder="e.g. Java,SQL,HTML,JSP" value=<%=rs.getString("itskills") %>></td>  </tr>  <tr>  <td>Education</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="education" placeholder="e.g. BCA" value=<%=rs.getString("education") %>></td>  </tr>  <tr>  <td>Certifications</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="certifications" placeholder="e.g. Certificate in HTML, JavaScript, JSP" value=<%=rs.getString("certifications") %>></td>  </tr>  <tr>  <td>Work Details</td>  </tr>  <tr align=center>  <td colspan=2><input type="text" name="workdetails" placeholder="e.g. Java Programmer in TCS" value=<%=rs.getString("workdetails") %>></td> </tr>  <tr>  <td>Personal Details</td>  </tr>  <tr>  <td>DOB</td>  <td><input type="date" name="dob" value=<%=rs.getString("dob") %>></td>  </tr>  <tr>  <td>Gender</td>  <td><input type="radio" name="gender" value="male" required>Male<input type="radio" name="gender" value="female" required>Female</td>  </tr>  <td>Hometown</td>  <td><input type="text" name="hometown" value=<%=rs.getString("hometown") %>></td>  </tr>  <td>PIN</td>  <td><input type="text" name="pin" value=<%=rs.getString("pin") %>></td> </tr>  <td>Maritial Status</td>  <td><input type="radio" name="maritialstatus" value="single" required>Single<input type="radio" name="maritialstatus" value="married" required>Married<input type="radio" name="maritialstatus" value="divorced" required>Divorced</td>  </tr>  <td>Languages Known</td>  <td><input type="text" name="languagesknown" value=<%=rs.getString("languagesknown") %>></td>  </tr>  <tr>  <td>&nbsp</td>  </tr>  <tr><td align="center" colspan=2><input type="submit" value="Update"></td> </tr>  </table>  </form>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When Jobseeker Clicks On ‘Edit’  By clicking on ‘Update’ Button Jobseeker can update their profile details. |
| Jobseekerupdateprofileaction.jsp (On clicking Update) |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <title>JobSeeker – Update Profile Action</title>  </head>  <body>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <%  String fname=request.getParameter("fname");  String lname=request.getParameter("lname");  String designation=request.getParameter("designation");  String location=request.getParameter("location");  String experience=request.getParameter("experience");  String email=request.getParameter("email");  String mobile=request.getParameter("mobile");  String keyskills=request.getParameter("keyskills");  String projects=request.getParameter("projects");  String itskills=request.getParameter("itskills");  String education=request.getParameter("education");  String certifications=request.getParameter("certifications");  String workdetails=request.getParameter("workdetails");  String dob=request.getParameter("dob");  String gender=request.getParameter("gender");  String hometown=request.getParameter("hometown");  String pin=request.getParameter("pin");  String maritialstatus=request.getParameter("maritialstatus");  String languagesknown=request.getParameter("languagesknown");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  ResultSet rs1=null;  ResultSet rs2=null;  ResultSet rs3=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from login where username='"+username+"' and usertype='"+usertype+"' ");  if(rs.next())  {  rs1=stmt.executeQuery("select \* from jobseekerprofile where username='"+username+"'");  if(rs1.next())  {  stmt.executeUpdate("update jobseekerprofile set username='"+username+"', fname='"+fname+"', lname='"+lname+"', designation='"+designation+"', location='"+location+"', experience='"+experience+"', email='"+email+"', mobile='"+mobile+"', keyskills='"+keyskills+"', projects='"+projects+"', itskills='"+itskills+"', education='"+education+"', certifications='"+certifications+"', workdetails='"+workdetails+"', dob='"+dob+"', gender='"+gender+"', hometown='"+hometown+"', pin='"+pin+"', maritialstatus='"+maritialstatus+"', languagesknown='"+languagesknown+"' where username='"+username+"' "); // toLowerCase() is used to store data into table in LowerCase to remove case senstivity of fields except password. Password is case sensitive for security reasons.  }  else  {  stmt.executeUpdate("insert into jobseekerprofile values('"+username+"','"+fname+"','"+lname+"','"+designation+"','"+location+"','"+experience+"','"+email+"','"+mobile+"','"+keyskills+"','"+projects+"','"+itskills+"','"+education+"','"+certifications+"','"+workdetails+"','"+dob+"','"+gender+"','"+hometown+"','"+pin+"','"+maritialstatus+"','"+languagesknown+"')"); // toLowerCase() is used to store data into table in LowerCase to remove case senstivity of fields except password. Password is case sensitive for security reasons.  }  rs.close();  stmt.close();  con.close();  }  response.sendRedirect("jobseekerprofile.jsp");  %>  </body>  </html> |
| * 1. ***Upload Resume:-****This link is used to upload resume****.*** |
| Jobseekerresume.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker - Resume </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Upload Resume</legend>  <br>  <form method="post" action="/uploadServlet" enctype="multipart/form-data">  <input type="file" name="resume"><br><br>  <input type="submit" value="Upload">  </form>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html>  Jobseekersearch.jsp  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker – Search </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Search Jobs</legend>  <form method="get" action="jobseekersearchresult.jsp">  <table>  <tr><td>Keywords</td> </tr>  <tr><td><input type="text" name="keyword"></td> </tr>  <tr> <td>Location</td> </tr>  <tr><td><input type="text" name="location"></td> </tr>  <tr><td>Experience</td><td>Expected Salary</td> </tr>  <tr> <td><input type="number" name="experience"></td>  <td><input type="number" name="expectedsalary"></td> </tr> <tr> <td><input type="submit" name="submit" value="Search Jobs"></td></tr> </table>  </form>  </fieldset> </center><jsp:include page="footer.jsp" />  </body>  </html> |
| ‘UPLOAD RESUME’ PAGE ON JOBSEEKER  By Clicking on “Upload” Jobseeker can upload there resume on website. |
| Jobseekerupload.jsp (on clicking upload) |
| <%@ page contentType="text/html;charset=windows-1252"%>  <%@ page import ="java.sql.\*" %>  <%@ page import ="javax.sql.\*" %>  <%@ page import ="javax.naming.InitialContext" %>  <%@ page import ="java.sql.PreparedStatement.\*" %>  <html>  <head>link rel="stylesheet" href="layout.css" type="text/css" />  </head>  <body>  <jsp:include page="header.jsp" />  <form name="upload1" method="post">  <%  String username=(String)session.getAttribute("username");  Connection con=null;  PreparedStatement ps=null;  String up2=request.getParameter("resume");  java.io.File f=new java.io.File(up2);  java.io.FileInputStream fis= new java.io.FileInputStream(f);  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  ps=con.prepareStatement("insert into jobseekerresume values(?,?)");  ps.setString(1,username);  ps.setBinaryStream(2,fis,(int)f.length());  ps.executeUpdate();  ps.close();  con.close();%>  </form><jsp:include page="footer.jsp" />  </body></html> |
| * 1. ***Search Jobs:-****This link is used to search jobs posted by recruiters.* |
| Jobsekersearch.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker  Search </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Search Jobs</legend>  <form method="get" action="jobseekersearchresult.jsp">  <table>  <tr><td>Keywords</td>  </tr>  <tr><td><input type="text" name="keyword"></td></tr>  <tr><td>Location</td></tr>  <tr><td><input type="text" name="location"></td>  </tr>  <tr> <td>Experience</td><td>Expected Salary</td></tr>  <tr><td><input type="number" name="experience"></td>  <td><input type="number" name="expectedsalary"></td>  </tr>  <tr><td><input type="submit" name="submit" value="SearchJobs"></td></tr>  </table>  </form>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |
| Jobsekersearchresult.jsp (on clicking serach) |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker – Search Result </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Search Results</legend>  <form action="jobseekersearchresult.jsp">  <input type="search" name="keyword" size="50%"><input type="submit" value="Search">  </form>  <%  String keyword=request.getParameter("keyword");  String location=request.getParameter("location");  String experience=request.getParameter("experience");  String expectedsalary=request.getParameter("expectedsalary");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from job where title='"+keyword+"' or keyskills='"+keyword+"' or role='"+keyword+"' or recruiter='"+keyword+"' or location='"+location+"' or experience<='"+experience+"' or salary>='"+expectedsalary+"' ");  while(rs.next())  {  %>  <hr width=30%>  <div>  <table width=30%>  <tr>  <td><b><a href="jobseekerjobdetails.jsp?jobid=<%=rs.getString(" jobid") %>"><%=rs.getString("title") %></a></b></td>  <td style="text-align:right;">Posted on <%=rs.getString("dateofposting") %></td>  </tr>  <tr>  <td><%=rs.getString("experience") %>&nbsp;Years</td>  </tr>  <tr>  <td><%=rs.getString("location") %></td>  </tr>  <tr>  <td><%=rs.getString("salary") %> p.a.</td>  </tr>  <tr>  <td><%=rs.getString("keyskills") %></td>  </tr>  <tr>  <td>Vacancy-<%=rs.getString("vacancy") %></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| Search Result Page When Searched For “Keywords:Software, Location:Mumbai, Experience:0, Expected Salary:30000” |

|  |
| --- |
| Jobseekerjobdetails.jsp (on clicking on his/her name) |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker – Job Details </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Job Details</legend>  <%  String jobid=request.getParameter("jobid");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from job where jobid='"+jobid+"'");  while(rs.next())  {  %>  <div class="jobdetails">  <table width=30%>  <tr>  <td><b><%=rs.getString("title") %></b></td>  <td style="text-align:right;">Posted on <%=rs.getString("dateofposting") %></td>  </tr>  <tr><td><%=rs.getString("experience") %>&nbsp;Years</td>  <td style="text-align:right;">Posted by <%=rs.getString("recruiter") %></td>  </tr>  <tr><td><%=rs.getString("location") %></td>  </tr><tr><td><%=rs.getString("salary") %> p.a.</td>  </tr>  <tr><td>Vacancy-<%=rs.getString("vacancy") %></td></tr>  </table>  <hr width=30%>  <table><tr><td><%=rs.getString("description") %></td> </tr></table>  <hr width=30%>  <table>  <tr><td>Functional Area &nbsp; </td>  <td><%=rs.getString("functionalarea") %></td>  </tr>  <tr><td>Role &nbsp;</td>  <td><%=rs.getString("role") %></td>  </tr>  <tr><td>Key Skills &nbsp;</td>  <td><%=rs.getString("keyskills") %></td>  </tr>  <tr><td>Interview Date &nbsp;</td>  <td><%=rs.getString("interviewdate") %></td>  </tr>  <tr><td>Interview Time &nbsp;</td>  <td><%=rs.getString("interviewtime") %></td>  </tr>  <tr><td>Interview Place &nbsp;</td>  <td><%=rs.getString("interviewplace") %></td>  </tr>  </table>  </div>  <br>  <a href="jobseekersendmessageaction.jsp?user=<%=rs.getString(" recruiter") %>&jobid=<%=jobid %>">Apply for this Job</a>  <%}  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| By clicking on “Apply for this Job” JobSeeker can apply for selected job. |

|  |
| --- |
| Jobseekersendmessage.jsp (on applying, the system sends a message to recruiter) |
| <%@ page import="java.io.\*" %>  <%@ page import="java.sql.\*" %>  <%@ page import="java.util.Date" %>  <%@ page import="java.text.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker – Send Message </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <%  Date dnow=new Date();  SimpleDateFormat ft=new SimpleDateFormat("E yyyy.MM.dd 'at' hh:mm:ss");  String user=request.getParameter("user");  %>  <div class="message">  <center>  <fieldset>  <legend>Message</legend>  <form method="post" action="jobseekersendmessageaction.jsp" name="messageform">  <table>  <tr>  <td>To</td>  <td><input type="text" name="to\_username" value=<%=user %> disabled></td>  </tr>  <tr>  <td>From</td>  <td><input type="text" name="from\_username" value=<%=username %> disabled></td>  </tr>  <tr>  <td>Message</td>  <td><textarea name="message" form="messageform" required></textarea></td>  </tr>  <tr>  <td><input type="hidden" name="time" value=<%=ft.format(dnow) %>></td>  </tr>  <tr>  <td align="center" colspan=2><input type="submit" value="Send"></td>  </tr>  </table>  </form>  </fieldset>  </center>  </div>  <jsp:include page="footer.jsp" />  </body>  </html> |
| Jobseekersendmessageaction.jsp |
| <%@ page import="java.io.\*" %>  <%@ page import="java.sql.\*" %>  <%@ page import="java.util.Date" %>  <%@ page import="java.text.\*" %>  <html>  <head>  <title>JobSeeker- Send Message Action</title>  </head>  <body>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <%  Date dnow=new Date();  SimpleDateFormat ft=new SimpleDateFormat("E dd.MM.yyyy 'at' hh:mm:ss");  String user=request.getParameter("user");  String jobid=request.getParameter("jobid");  String message="Applied for ";  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  stmt.executeUpdate("insert into message values('"+username+"','"+user+"','"+message+"','"+jobid+"','"+ft.format(dnow)+"')");  rs.close();  stmt.close();  con.close();  response.sendRedirect("jobseekersearchresult.jsp");  %>  </body></html> |
| * 1. ***Messages:-*** *This link is used to view received messages about job selection.* |
| Jobseekermessagebox.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker - Message Box </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Messages</legend>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from message where to\_username='"+username+"' order by datetime desc ");  if(!rs.next())  {  %>  Empty Message Box  <%  }  while(rs.next())  {  %>  <hr width=30%>  <div class="inbox">  <table width=30%>  <tr>  <td><a href="jobseekerrecruiterprofile.jsp?user=<%=rs.getString(" from\_username") %>"><%=rs.getString("from\_username") %></a></td>  <td style="text-align:right;"><%=rs.getString("datetime") %></td>  </tr>  <tr>  <td><%=rs.getString("message") %><a href="jobseekerjobdetails.jsp?jobid=<%=rs.getString(" forjob")%>"><%=rs.getString("forjob") %></a></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When There Is No Message  When there are messages |

|  |
| --- |
| * 1. ***Feedback:-This link is used for feedback page to submit a feedback.*** |
| Jobseekerfeedback.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> JobSeeker - Feedback </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> JobSeeker </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="jobseeker") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="jobseekermenu.jsp" />  <center>  <fieldset>  <legend>Feedback</legend>  <form method="post" action="mailto:ry4761@gmail.com" name="messageform">  <table>  <tr>  <td>Username</td>  <td><input type="text" name="name" value="<%=username %>" disabled></td>  </tr>  <tr>  <td>E-Mail</td>  <td><input type="text" name="email"></td>  </tr>  <tr>  <td>Message</td>  <td><textarea name="message" form="messageform" required></textarea></td>  </tr>  <tr>  <td></td>  <td><input type="submit" value="Send Message"></td>  </tr>  </table>  </form>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |

|  |
| --- |
| 1. **RECRUITERS HOME PAGE** |
| Recrutierhome.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter - Home </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <jsp:include page="footer.jsp" />  </body>  </html> |
| Recrutiermenu.jsp |
| <hr>  <pre><a href="recruiterprofile.jsp"> Profile </a> | <a href="recruiterviewjob.jsp"> View Posted Jobs </a> | <a href="recruiteraddjob.jsp"> Post a Job </a> | <a href="recruitersearch.jsp"> Search Job Seeker </a> | <a href="recruitermessagebox.jsp"> Messages </a> | <a href="recruiterfeedback.jsp"> Feedback </a></pre>  <hr>  Recrutermessagebox.jsp  <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – Message Box </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  { response.sendRedirect("index.jsp");} %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Messages</legend>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from message where to\_username='"+username+"' order by datetime desc ");  if(!rs.next())  {%>  Empty Message Box  <%}  while(rs.next())  { %>  <hr width=30%>  <div class="inbox">  <table width=30%>  <tr><td><a href="recruiterjobseekerprofile.jsp?user=<%=rs.getString(" from\_username") %>"><%=rs.getString("from\_username") %></a></td>  <td style="text-align:right;"><%=rs.getString("datetime") %></td>  </tr>  <tr>  <td><%=rs.getString("message") %><a href="recruiterjobdetails.jsp?jobid=<%=rs.getString(" forjob")%>"><%=rs.getString("forjob") %></a></td>  <td style="text-align:right;"><a href="recruitersendmessageaction.jsp?user=<%=rs.getString(" from\_username")%>&jobid=<%=rs.getString("forjob")%>">Accept for Interview</a></td>  </tr>  </table>  </div>  <hr width=30%>  <% }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |

|  |
| --- |
| * 1. ***Profile*: -***This link is used to view/update profile of logged recruiter.* |
| Recrutierprofile.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter - Search </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  Search Job Seeker<br><br>  <form method="get" action="recruitersearchresult.jsp">  <input type="search" name="search">  <input type="submit" value="Search">  </form>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When Recruiter Updated Their Details    When Recruiter Is NewBy clicking on’Edit’ recruiter can edit their profile details |
| Recrutierupdateprofile.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – Updae Profile </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" /><hr>  <center>  <fieldset>  <legend>Profile Details</legend>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from recruiterprofile where username='"+username+"'");  if(!rs.next())  {  %>  <form method="post" action="recruiterupdateprofileaction.jsp">  <table>  <tr>  <td>Name</td>  <td><input type="text" name="fname" placeholder="FirstName"></td>  </tr>  <tr>  <td></td>  <td><input type="text" name="lname" placeholder="LastName"></td>  </tr>  <tr>  <td>Company Name</td>  <td><input type="text" name="company"></td>  </tr>  <tr>  <td>Designation</td>  <td><input type="text" name="designation"></td>  </tr>  <tr>  <td>Location</td>  <td><input type="text" name="location"></td>  </tr>  <tr>  <td>Email</td>  <td><input type="text" name="email"></td>  </tr>  <tr>  <td>Mobile</td>  <td><input type="text" name="mobile"></td>  </tr>  <tr>  <td>Gender</td>  <td><input type="radio" name="gender" value="male" required>Male<input type="radio" name="gender" value="female" required>Female</td>  </tr>  <tr>  <td>&nbsp;</td>  </tr>  <tr>  <td align="center" colspan=2><input type="submit" value="Update"></td>  </tr>  </table>  </form>  <%  }  while(rs.next())  {  %>  <form method="post" action="recruiterupdateprofileaction.jsp">  <table>  <tr>  <td>Name</td>  <td><input type="text" name="fname" placeholder="FirstName" value=<%=rs.getString("fname") %>></td>  </tr>  <tr>  <td></td>  <td><input type="text" name="lname" placeholder="LastName" value=<%=rs.getString("lname") %>></td>  </tr>  <tr>  <td>Company Name</td>  <td><input type="text" name="company" value=<%=rs.getString("company") %>></td>  </tr>  <tr>  <td>Designation</td>  <td><input type="text" name="designation" value=<%=rs.getString("designation") %>></td>  </tr>  <tr>  <td>Location</td>  <td><input type="text" name="location" value=<%=rs.getString("location") %>></td>  </tr>  <tr>  <td>Email</td>  <td><input type="text" name="email" value=<%=rs.getString("email") %>></td>  </tr>  <tr>  <td>Mobile</td>  <td><input type="text" name="mobile" value=<%=rs.getString("mobile") %>></td>  </tr>  <tr>  <td>Gender</td>  <td><input type="radio" name="gender" value="male" required>Male<input type="radio" name="gender" value="female" required>Female</td>  </tr>  <tr>  <td>&nbsp;</td>  </tr>  <tr>  <td align="center" colspan=2><input type="submit" value="Update"></td>  </tr>  </table>  </form>  <%  }  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When Recruiter Clicks On ‘Edit’By clicking on ‘Update’ Button Recruiter can update there profile details. |
| Recrutierupdateprofileaction.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <title> Recruiter – Update Profile </title>  </head>  <body>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  %>  <%  String fname=request.getParameter("fname");  String lname=request.getParameter("lname");  String company=request.getParameter("company");  String designation=request.getParameter("designation");  String location=request.getParameter("location");  String email=request.getParameter("email");  String mobile=request.getParameter("mobile");  String gender=request.getParameter("gender");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  ResultSet rs1=null;  ResultSet rs2=null;  ResultSet rs3=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from login where username='"+username+"' and usertype='"+usertype+"' ");  if(rs.next())  {  rs1=stmt.executeQuery("select \* from recruiterprofile where username='"+username+"'");  if(rs1.next())  {  stmt.executeUpdate("update recruiterprofile set username='"+username+"' , fname='"+fname+"' , lname='"+lname+"' , company='"+company+"' , designation='"+designation+"' , location='"+location+"' , email='"+email+"' , mobile='"+mobile+"' , gender='"+gender+"' where username='"+username+"' ");  }  else  {  stmt.executeUpdate("insert into recruiterprofile values('"+username+"','"+fname+"','"+lname+"','"+company+"','"+designation+"','"+location+"','"+email+"','"+mobile+"','"+gender+"')");  }  stmt.close();  con.close();  }  response.sendRedirect("recruiterprofile.jsp");  %>  </body>  </html>  Recrutieveiwjob.jsp  <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – View Job </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <fieldset>  <legend>Posted Jobs</legend>  <center>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from job where recruiter='"+username+"' ");  while(rs.next())  {  %>  <hr width=30%>  <div class="jobslist">  <table width=30%>  <tr>  <td><b><a href="recruiterjobdetails.jsp?jobid=<%=rs.getString(" jobid") %>"><%=rs.getString("title") %></a></b></td>  <td style="text-align:right;">Posted on <%=rs.getString("dateofposting") %></td>  </tr>  <tr>  <td><%=rs.getString("experience") %>Years</td>  </tr>  <tr>  <td><%=rs.getString("location") %></td>  </tr>  <tr>  <td><%=rs.getString("salary") %> p.a.</td>  </tr>  <tr>  <td><%=rs.getString("keyskills") %></td>  </tr>  <tr>  <td>Vacancy-<%=rs.getString("vacancy") %></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </center>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
| * 1. ***Post a Job:-*** |
| Recrutieraddjob.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title>Recruiter – Add Job</title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Add Job</legend>  <form action="recruiteraddjobaction.jsp" method="post" id="jobform">  <table>  <tr>  <td>JobId: </td>  <td><input type="text" name="jobid" value="" required></td>  </tr>  <tr>  <td>Job Title:</td>  <td><input type="text" name="title" required></td>  </tr>  <tr>  <td>Location: </td>  <td>  <select name="location" required>  <option> - Select -</option>  <option value="Delhi">DELHI</option>  <option value="Noida">NOIDA</option>  <option value="Gurgaon">GURGAON</option>  <option value="Faridabad">FARIDABAD</option>  <option value="Banglore">BANGLORE</option>  <option value="Chennai">CHENNAI</option>  <option value="Hyderabad">HYDERABAD</option>  <option value="Mumbai">MUMBAI</option>  <option value="Kolkata">KOLKATA</option>  <option value="Pune">PUNE</option>  <option value="Lucknow">LUCKNOW</option>  <option value="Kanpur">KANPUR</option>  </select>  </td>  </tr>  <tr>  <td>Minimum Experience</td>  <td>  </select>&nbsp;Years  </td>  </tr>  <tr>  <td>Salary:</td>  <td><input type="text" name="salary" required></td>  </tr>  <tr>  <td>Job Description:</td>  <td><textarea name="description" form="jobform" required></textarea></td>  </tr>  <tr>  <td>Functional Area: </td>  <td>  <select name="functionalarea" required>  <option> - Select - </option>  <option value="Any">Any</option>  <option value="Automotive">Automotive</option>  <option value="Banking">Banking</option>  <option value="Bio Technology">Bio Technology</option>  <option value="Chemicals">Chemicals</option>  <option value="Construction">Construction</option>  <option value="Consumer Goods">Consumer Goods</option>  <option value="Education">Education</option>  <option value="Entertainment">Entertainment</option>  <option value="Insurance">Insurance</option>  <option value="BPO">BPO</option>  <option value="Computer Hardware">Computer Hardware</option>  <option value="Computer Software">Computer Software</option>  </select>  </td> <tr>  <td>Role: </td>  <td>  <select name="role" required>  <option> - Select - </option>  <option value="Sales Executive">Sales Executive</option>  <option value="Teacher">Teacher</option>  <option value="Manager">Manager</option>  <option value="Accounting">Accounting</option>  <option value="Technician">Technician</option>  <option value="Software Professional">Software Professional</option>  <option value="Software Tester">Software Tester</option>  <option value="IT Professional">IT Professional</option>  </select>  </td>  </tr> <tr>  <td>Key Skills:</td>  <td><input type="text" name="keyskills" placeholder="Java,SQL,HTML,JSP etc." required></td>  </tr> <tr>  <td>Vacancy: </td>  <td><input type="number" name="vacancy" required></td>  </tr> <tr>  <td>Interview Date: </td>  <td><input type="date" name="interviewdate" required></td>  </tr>  <tr>  <td>Interview Time: </td>  <td><input type="time" name="interviewtime" required></td>  </tr>  <tr>  <td>Interview Place: </td>  <td><input type="text" name="interviewplace" required></td>  </tr>  <input type=hidden name="recruiter" value="<%=username %>">  <tr>  <td>&nbsp;</td>  </tr>  <tr>  <td colspan=2 align=center><input type="submit" value="ADD"></td>  </tr>  </table>  </form>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |

|  |
| --- |
| Recrutieraddjobaction.jsp |
| <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>  <%@ page import="java.io.\*" %>  <%@ page import="java.sql.\*" %>  <%@ page import="java.util.Date" %>  <%@ page import="java.text.\*" %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <title>Recruiter – Add Job</title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <hr>  <pre> <a href="recruiterprofile.jsp"> Profile </a> | <a href="recruiterviewjob.jsp"> View Your Jobs </a> | <a href="recruiteraddjob.jsp"> Add New Job </a> | <a href="recruitersearch.jsp"> Search Job Seeker </a> | <a href=""> Response Detail </a> | <a href="recruiterfeedback.jsp"> Feedback </a></pre>  <hr>  <%  Date dnow=new Date();  SimpleDateFormat ft=new SimpleDateFormat("E dd.MM.yyyy 'at' hh:mm:ss");  String jobid=request.getParameter("jobid");  String title=request.getParameter("title");  String location=request.getParameter("location");  String experience=request.getParameter("experience");  String salary=request.getParameter("salary");  String description=request.getParameter("description");  String functionalarea=request.getParameter("functionalarea");  String role=request.getParameter("role");  String keyskills=request.getParameter("keyskills");  String vacancy=request.getParameter("vacancy");  String interviewdate=request.getParameter("interviewdate");  String interviewtime=request.getParameter("interviewtime");  String interviewplace=request.getParameter("interviewplace");  String recruiter=request.getParameter("recruiter");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  stmt.executeUpdate("insert into job values('"+jobid+"','"+title+"','"+location+"','"+experience+"','"+salary+"','"+description+"','"+functionalarea+"','"+role+"','"+keyskills+"','"+vacancy+"','"+interviewdate+"','"+interviewtime+"','"+interviewplace+"','"+recruiter+"','"+ft.format(dnow)+"')");  rs.close();  stmt.close();  con.close();  response.sendRedirect("recruiterjobdetails.jsp?jobid="+jobid+"");  %>  <jsp:include page="footer.jsp" />  </body>  </html> |
| * 1. ***View Posted Jobs*** |
| Recrutierviewjob.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter  View Job </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <fieldset>  <legend>Posted Jobs</legend>  <center>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from job where recruiter='"+username+"' ");  while(rs.next())  {  %>  <hr width=30%>  <div class="jobslist">  <table width=30%>  <tr>  <td><b><a href="recruiterjobdetails.jsp?jobid=<%=rs.getString(" jobid") %>"><%=rs.getString("title") %></a></b></td>  <td style="text-align:right;">Posted on <%=rs.getString("dateofposting") %></td>  </tr>  <tr>  <td><%=rs.getString("experience") %>Years</td>  </tr>  <tr>  <td><%=rs.getString("location") %></td>  </tr>  <tr>  <td><%=rs.getString("salary") %> p.a.</td>  </tr>  <tr>  <td><%=rs.getString("keyskills") %></td>  </tr>  <tr>  <td>Vacancy-<%=rs.getString("vacancy") %></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </center>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
| Recrutierjobdetails.jsp (when a particular job is clicked) |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – Job Details </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Job Details</legend>  <%  String jobid=request.getParameter("jobid");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from job where jobid='"+jobid+"'");  while(rs.next())  {  %>  <div class="jobdetails">  <table width=30%>  <tr>  <td><b><%=rs.getString("title") %></b></td>  <td style="text-align:right;">Posted on <%=rs.getString("dateofposting") %></td>  </tr>  <tr>  <td><%=rs.getString("experience") %>&nbsp;Years</td>  <td style="text-align:right;">Posted by <%=rs.getString("recruiter") %></td>  </tr>  <tr>  <td><%=rs.getString("location") %></td>  </tr>  <tr><td><%=rs.getString("salary") %> p.a.</td>  </tr>  <tr>  <td>Vacancy-<%=rs.getString("vacancy") %></td>  </tr>  </table>  <hr width=30%>  <table>  <tr>  <td><%=rs.getString("description") %></td>  </tr>  </table>  <hr width=30%>  <table>  <tr>  <td>Functional Area &nbsp; </td>  <td><%=rs.getString("functionalarea") %></td>  </tr>  <tr>  <td>Role &nbsp;</td>  <td><%=rs.getString("role") %></td>  </tr>  <tr>  <td>Key Skills &nbsp;</td>  <td><%=rs.getString("keyskills") %></td>  </tr>  <tr>  <td>Interview Date &nbsp;</td>  <td><%=rs.getString("interviewdate") %></td>  </tr>  <tr>  <td>Interview Time &nbsp;</td>  <td><%=rs.getString("interviewtime") %></td>  </tr>  <tr>  <td>Interview Place &nbsp;</td>  <td><%=rs.getString("interviewplace") %></td>  </tr>  </table>  </div>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body></html> |
| When There Is No Job Posted By Recruiter  When There Are Jobs Posted By Rectuiters  Recruiter can open JobSeeker’s Profile Page by clicking on that JobSeeker’s name.  When Recruiter Views Jobs There Are No “Apply For Job” Option |
| * 1. ***Search JobSeeker:-*** |
| Recrutiersearch.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter - Search </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  Search Job Seeker<br><br>  <form method="get" action="recruitersearchresult.jsp">  <input type="search" name="search">  <input type="submit" value="Search">  </form>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |

|  |
| --- |
| Recrutiersearchresult.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – Search Result </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Search Results</legend>  <form action="jobseekersearchresult.jsp">  <input type="search" name="keyword" size="50%"><input type="submit" value="Search">  </form>  <%  String search=request.getParameter("search");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+search+"' or fname='"+search+"' or lname='"+search+"' or designation='"+search+"' or location='"+search+"' or experience='"+search+"' or email='"+search+"' or mobile='"+search+"' or keyskills='"+search+"' or projects='"+search+"' or itskills='"+search+"' or education='"+search+"' or certifications='"+search+"' or workdetails='"+search+"' or dob='"+search+"' or gender='"+search+"' or hometown='"+search+"' or pin='"+search+"' or maritialstatus='"+search+"' or languagesknown='"+search+"'");  while(rs.next())  {  %>  <hr width=30%>  <div class="recruitersearchresult">  <table width=30%>  <tr>  <td><b><a href="recruiterjobseekerprofile.jsp?user=<%=rs.getString(" username") %>"><%=rs.getString("fname") %> <%=rs.getString("lname") %></a></b></td>  </tr>  <tr>  <td><%=rs.getString("designation") %></td>  </tr>  <tr>  <td><%=rs.getString("experience") %>Years</td>  </tr>  <tr>  <td><%=rs.getString("location") %></td>  </tr>  <tr>  <td><%=rs.getString("keyskills") %></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| JOBSEEKER PROFILE PAGE ON RECRUITER |

|  |
| --- |
| Recrutierjobseekerprofile.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – JobSeeker Profile </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Profile Details</legend>  <%  String user=request.getParameter("user");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  ResultSet rs1=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+user+"'");  while(rs.next())  {  %>  <table>  <a href="recruitersendmessage.jsp?user=<%=user %>">Send Message</a>  <th>  <td>Basic Details</td>  </th>  <tr>  <td>Name</td>  <td><%=rs.getString("FNAME") %> <%=rs.getString("LNAME") %></td>  </tr>  <tr>  <td>Designation</td>  <td><%=rs.getString("DESIGNATION") %></td>  </tr>  <tr>  <td>Location</td>  <td><%=rs.getString("LOCATION") %></td>  </tr>  <tr>  <td>Experience</td>  <td><%=rs.getString("EXPERIENCE") %></td>  </tr>  <th>  <td>Contact Details</td>  </th>  <tr>  <td>Email</td>  <td><%=rs.getString("EMAIL") %></td>  </tr>  <tr>  <td>Mobile</td>  <td><%=rs.getString("MOBILE") %></td>  </tr>  <th>  <td>Key Skills</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("KEYSKILLS") %></td>  </tr>  <th>  <td>Projects</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("PROJECTS") %></td>  </tr>  <th>  <td>IT Skills</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("ITSKILLS") %></td>  </tr>  <th>  <td>Education</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("EDUCATION") %></td>  </tr>  <th>  <td>Certifications</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("CERTIFICATIONS") %></td>  </tr>  <th>  <td>Work Details</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("WORKDETAILS") %></td>  </tr>  <th>  <td>Basic Details</td>  </th>  <tr>  <td>DOB</td>  <td><%=rs.getString("DOB") %></td>  </tr>  <tr>  <td>Gender</td>  <td><%=rs.getString("GENDER") %></td>  </tr>  <td>Hometown</td>  <td><%=rs.getString("HOMETOWN") %></td>  </tr>  <td>PIN</td>  <td><%=rs.getString("PIN") %></td>  </tr>  <td>Maritial Status</td>  <td><%=rs.getString("MARITIALSTATUS") %></td>  </tr>  <td>Languages Known</td>  <td><%=rs.getString("LANGUAGESKNOWN") %></td>  </tr>  <%  }  rs.close();  stmt.close();  con.close();  %>  </table>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When Recruiter Is Accessing JobSeeker Profile There Is ‘Send Message’ Option By Which Recruiter Can Send Message To That Perticular JobSeeker. |

|  |
| --- |
| * 1. ***Message:-*** *contains information about jobseekers who have applied to jobs* |
| Recrutiermessagebox.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter  Message Box </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Messages</legend>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from message where to\_username='"+username+"' order by datetime desc ");  if(!rs.next())  {  %>  Empty Message Box  <% }  while(rs.next()) {  %>  <hr width=30%>  <div class="inbox">  <table width=30%>  <tr><td><a href="recruiterjobseekerprofile.jsp?user=<%=rs.getString(" from\_username") %>"><%=rs.getString("from\_username") %></a></td>  <td style="text-align:right;"><%=rs.getString("datetime") %></td></tr>  <tr><td><%=rs.getString("message") %><a href="recruiterjobdetails.jsp?jobid=<%=rs.getString(" forjob")%>"><%=rs.getString("forjob") %></a></td>  <td style="text-align:right;"><a href="recruitersendmessageaction.jsp?user=<%=rs.getString(" from\_username")%>&jobid=<%=rs.getString("forjob")%>">Accept for Interview</a></td>  </tr>  </table>  </div>  <hr width=30%>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| When There Is No Message For Recruiter  When There Are Messages |
| Recrutiersendmessage.jsp |
| <%@ page import="java.io.\*" %>  <%@ page import="java.sql.\*" %>  <%@ page import="java.util.Date" %>  <%@ page import="java.text.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter – Send Message </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <%  Date dnow=new Date();  SimpleDateFormat ft=new SimpleDateFormat("E yyyy.MM.dd 'at' hh:mm:ss");  String user=request.getParameter("user");  %>  <div class="message">  <center>  <fieldset>  <legend>Message</legend>  <form method="post" action="recruitersendmessageaction.jsp" name="messageform">  <table>  <tr><td>To</td>  <td><input type="text" name="to\_username" value=<%=user %> disabled></td> </tr>  <tr><td>From</td>  <td><input type="text" name="from\_username" value=<%=username %> disabled></td>  </tr>  <tr><td>Message</td><td><textarea name="message" form="messageform" required></textarea></td></tr>  <tr><td><input type="hidden" name="time" value=<%=ft.format(dnow) %>></td></tr>  <tr><td align="center" colspan=2><input type="submit" value="Send"></td></tr>  </table>  </form>  </fieldset>  </center>  </div>  <jsp:include page="footer.jsp" />  </body>  </html> |
| Recrutiersendmessageaction.jsp |
| <%@ page import="java.io.\*" %>  <%@ page import="java.sql.\*" %>  <%@ page import="java.util.Date" %>  <%@ page import="java.text.\*" %>  <html>  <head>  <title> Recruiter – Send Message </title>  </head>  <body>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <%  Date dnow=new Date();  SimpleDateFormat ft=new SimpleDateFormat("E dd.MM.yyyy 'at' hh:mm:ss");  String user=request.getParameter("user");  String jobid=request.getParameter("jobid");  String message="Selected for Interview for job ";  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  stmt.executeUpdate("insert into message values('"+username+"','"+user+"','"+message+"','"+jobid+"','"+ft.format(dnow)+"')");  rs.close();  stmt.close();  con.close();  response.sendRedirect("recruitermessagebox.jsp");  %>  </body>  </html> |
| **‘Send Message’ Page** |

|  |
| --- |
| * 1. ***Feedback*** |
| Recrutierfeedback.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Recruiter - Feedback </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1> Recruiter </h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="recruiter") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="recruitermenu.jsp" />  <center>  <fieldset>  <legend>Feedback</legend>  <form method="post" action="mailto:ry4761@gmail.com" name="messageform">  <table>  <tr><td>Username</td>  <td><input type="text" name="username" value="<%=username%>" disabled></td></tr><tr><td>E-Mail</td>  <td><input type="text" name="email"></td>  </tr>  <tr><td>Message</td>  <td><textarea name="message" form="messageform" required></textarea></td>  </tr>  <tr><td></td>  <td><input type="submit" value="Send Message"></td>  </tr>  </table>  </form>  </fieldset>  </center>  <jsp:include page="footer.jsp" />  </body>  </html> |
| 1. **ADMIN HOME PAGE MODULE** |
| Adminmenu.jsp |
| <center>  <pre style="font-family: sans-serif;font-size: 20px"><a href="adminhome.jsp"> Home </a> | <a href=""> Inbox </a> | <a href="adminjobseekerreport.jsp"> JobSeeker Reports </a> | <a href="adminrecruiterreport.jsp"> Recruiter Reports</a> </pre>  <hr>  Adminrecrutierprofile.jsp  <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Admin – Recruiter Profile</title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp" />  <fieldset>  <legend>Profile Details</legend>  <%  String user=request.getParameter("user");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from recruiterprofile where username='"+user+"'");  while(rs.next())  {  %>  <table width=30%>  <tr>  <td>Name</td>  <td><%=rs.getString("FNAME") %> <%=rs.getString("LNAME") %></td>  </tr>  <tr>  <td>Company Name</td>  <td><%=rs.getString("COMPANY") %></td>  </tr>  <tr>  <td>Designation</td>  <td><%=rs.getString("DESIGNATION")%></td>  </tr>  <tr>  <td>Location</td>  <td><%=rs.getString("LOCATION") %></td>  </tr>  <tr>  <td>Email</td>  <td><%=rs.getString("EMAIL") %></td>  </tr>  <tr>  <td>Mobile</td>  <td><%=rs.getString("MOBILE") %></td>  </tr>  <tr>  <td>Gender</td>  <td><%=rs.getString("GENDER") %></td>  </tr>  </table>  <%  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |

|  |
| --- |
| * 1. ***Home: -*** *This link is for administrator homepage.* |
| Adminhome.jsp |
| <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Admin - Home </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp" />  <jsp:include page="footer.jsp" />  </body>  </html> |
| * 1. ***JobSeeker Reports:***  administrator can delete or view JobSeekers profile. |
| Adminjobseekerreport.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Admin – JobSeeker Report </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp" />  <fieldset>  <legend>List of Registered JobSeekers</legend>  <div class="adminlist">  <table width=30% border=1>  <b>  <tr>  <td width=10%>S.No.</td>  <td width=50%>Username</td>  <td width=20%>View Profile</td>  <td width=20%>Remove Profile</td>  </tr>  </b>  </table>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  int count=1;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile");  while(rs.next())  {  %>  <table width=30% border=1>  <tr>  <td width=10%><%=count%>.</td>  <td width=50%><%=rs.getString("username") %></td>  <td width=20%><a href="adminjobseekerprofile.jsp?user=<%=rs.getString(" username") %>">View</a></td>  <td width=20%><a href="adminremoveaction.jsp?user=<%=rs.getString(" username")%>&type=jobseeker">Remove</a></td>  </tr>  </table>  </div>  <%  count++;  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
|  |
| * + 1. ***View: -*** *This link will open correspondence jobseeker’s profile page.* |
| adminjobseekerview.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Admin  JobSeeker Profile </title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp" />  <fieldset>  <legend>Profile Details</legend>  <%  String user=request.getParameter("user");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+user+"'");  while(rs.next())  {  %>  <table width=30%>  <th>  <td>Basic Details</td>  </th>  <tr>  <td>Name</td>  <td><%=rs.getString("FNAME") %> <%=rs.getString("LNAME") %></td>  </tr>  <tr><td>Designation</td>  <td><%=rs.getString("DESIGNATION") %></td>  </tr>  <tr>  <td>Location</td>  <td><%=rs.getString("LOCATION") %></td>  </tr>  <tr>  <td>Experience</td>  <td><%=rs.getString("EXPERIENCE") %></td>  </tr>  <th>  <td>Contact Details</td>  </th>  <tr>  <td>Email</td>  <td><%=rs.getString("EMAIL") %></td>  </tr>  <tr>  <td>Mobile</td>  <td><%=rs.getString("MOBILE") %></td>  </tr>  <th>  <td>Key Skills</td>  </th>  <tr align=center><td colspan=2><%=rs.getString("KEYSKILLS") %></td>  </tr>  <th>  <td>Projects</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("PROJECTS") %></td>  </tr>  <th>  <td>IT Skills</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("ITSKILLS") %></td>  </tr>  <th>  <td>Education</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("EDUCATION") %></td>  </tr>  <th>  <td>Certifications</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("CERTIFICATIONS") %></td>  </tr>  <th>  <td>Work Details</td>  </th>  <tr align=center>  <td colspan=2><%=rs.getString("WORKDETAILS") %></td>  </tr>  <th>  <td>Basic Details</td>  </th>  <tr>  <td>DOB</td>  <td><%=rs.getString("DOB") %></td>  </tr>  <tr>  <td>Gender</td>  <td><%=rs.getString("GENDER") %></td>  </tr>  <td>Hometown</td>  <td><%=rs.getString("HOMETOWN") %></td>  </tr>  <td>PIN</td>  <td><%=rs.getString("PIN") %></td>  </tr>  <td>Maritial Status</td>  <td><%=rs.getString("MARITIALSTATUS") %></td>  </tr>  <td>Languages Known</td>  <td><%=rs.getString("LANGUAGESKNOWN") %></td>  </tr>  <%  }  rs.close();  stmt.close();  con.close();  %>  </table>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
| * + 1. ***Remove:****to remove correspondence jobseeker’s profile from database.* |
| adminremove action.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head><title>Admin Remove Action</title>  </head>  <body>  <% String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <%  String user=request.getParameter("user");  String type=request.getParameter("type");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  ResultSet rs1=null;  ResultSet rs2=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1"; con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  if(type=="jobseeker")  {  rs=stmt.executeQuery("select \* from jobseekerprofile where username='"+user+"' ");  if(rs.next())  {  rs1=stmt.executeQuery("delete from jobseekerprofile where username='"+user+"' ");  rs2=stmt.executeQuery("delete from login where username='"+user+"' and usertype='"+type+"' ");  rs.close();  stmt.close();  con.close();  }  response.sendRedirect("adminjobseekerreport.jsp");  }  else if(type=="recruiter"){  rs=stmt.executeQuery("select \* from recruiterprofile where username='"+user+"' ");  if(rs.next())  {  rs1=stmt.executeQuery("delete from recruiterprofile where username='"+user+"' ");  rs2=stmt.executeQuery("delete from login where username='"+user+"' and usertype='"+type+"' ");  rs.close();  stmt.close();  con.close();  } response.sendRedirect("adminrecruiterreport.jsp");  }  %>  </body>  </html> |
| * 1. ***Recruiter Reports****: administrator can manage recruiters profile.* |
| adminrecruiterreport.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head><link rel="stylesheet" href="styles.css" type="text/css"/>  <title> Admin – Recruiter Report </title>  </head>  <body>  <jsp:include page="header.jsp"/>  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp"/>  <fieldset>  <legend>List of Registered Recruiters</legend>  <div class="adminlist">  <table width=30% border=1>  <b><tr>  <td width=10%>S.No.</td>  <td width=50%>Username</td>  <td width=20%>View Profile</td>  <td width=20%>Remove Profile</td>  </tr></b>  </table>  <%  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  int count=1;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from recruiterprofile");  while(rs.next())  {  %>  <table width=30% border=1>  <tr>  <td width=10%><%=count%>.</td>  <td width=50%><%=rs.getString("username") %></td>  <td width=20%><a href="adminrecruiterprofile.jsp?user=<%=rs.getString("username") %>">View</a></td>  <td width=20%><a href="adminremoveaction.jsp?user=<%=rs.getString("username")%>&type=recruiter">Remove</a></td>  </table>  </div>  <%  count++;  }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  <jsp:include page="footer.jsp"/>  </body>  </html> |
| **‘**RECRUITER REPORTS’ PAGE ON ADMINISTRATOR     * + 1. ***View: -*** *will open correspondence recruiter’s profile page*     2. ***Remove: -t****o remove correspondence recruiter’s profile from database.* |
| Adminrecruiterprofileview.jsp |
| <%@ page import="java.sql.\*" %>  <html>  <head>  <link rel="stylesheet" href="styles.css" type="text/css" />  <title> Admin  Recruiter Profile</title>  </head>  <body>  <jsp:include page="header.jsp" />  <h1>Administrator</h1>  <%  String username=(String)session.getAttribute("username");  String usertype=(String)session.getAttribute("usertype");  if(username==null || usertype!="admin") //to check user is logged in, if user is not logged username will be null and page will redirect to index.jsp  {  response.sendRedirect("index.jsp");  }  %>  <p style="text-align:right;"> Welcome <%=username %> <a href="logout.jsp"> Logout </a></p>  <jsp:include page="adminmenu.jsp" />  <fieldset>  <legend>Profile Details</legend>  <%  String user=request.getParameter("user");  Connection con=null;  Statement stmt=null;  ResultSet rs=null;  Class.forName("com.mysql.jdbc.Driver");  String url="jdbc:mysql://localhost:3306/ojp?characterEncoding=latin1";  con=DriverManager.getConnection(url,"root","4455surE");  stmt=con.createStatement();  rs=stmt.executeQuery("select \* from recruiterprofile where username='"+user+"'");  while(rs.next())  {  %>  <table width=30%>  <tr><td>Name</td>  <td><%=rs.getString("FNAME") %> <%=rs.getString("LNAME") %></td>  </tr>  <tr><td>Company Name</td>  <td><%=rs.getString("COMPANY") %></td>  </tr>  <tr><td>Designation</td>  <td><%=rs.getString("DESIGNATION")%></td>  </tr>  <tr><td>Location</td>  <td><%=rs.getString("LOCATION") %></td>  </tr>  <tr><td>Email</td>  <td><%=rs.getString("EMAIL") %></td>  </tr>  <tr><td>Mobile</td><td><%=rs.getString("MOBILE") %></td></tr>  <tr>  <td>Gender</td>  <td><%=rs.getString("GENDER") %></td>  </tr>  </table>  <% }  rs.close();  stmt.close();  con.close();  %>  </fieldset>  <jsp:include page="footer.jsp" />  </body>  </html> |
| ***CONCLUSION:*** Thus, from this experiment I have implemented object-oriented analysis and design for Job Recruitment System, along with specifying the software architecture and algorithm design for the system. |

*191071902*

*Srushti Shah*

*T. Y. B. Tech. C. S.*

***Experiment No. 3***

***AIM*:** OO System Testing

***THEORY:***

Apply object-oriented software testing to test your software. Use the following dimensions for testing

1. Unit Testing 2. Method Testing 3. Class Testing 4. Integration Testing

5. System Testing

You can also use the open source software testing tools, open source functional testing tools, open source web application testing tools, open source performance testing tools, , open source load testing tools, and any other open source testing tools of your choice.

Given below is the list of most popular open-source testing tools

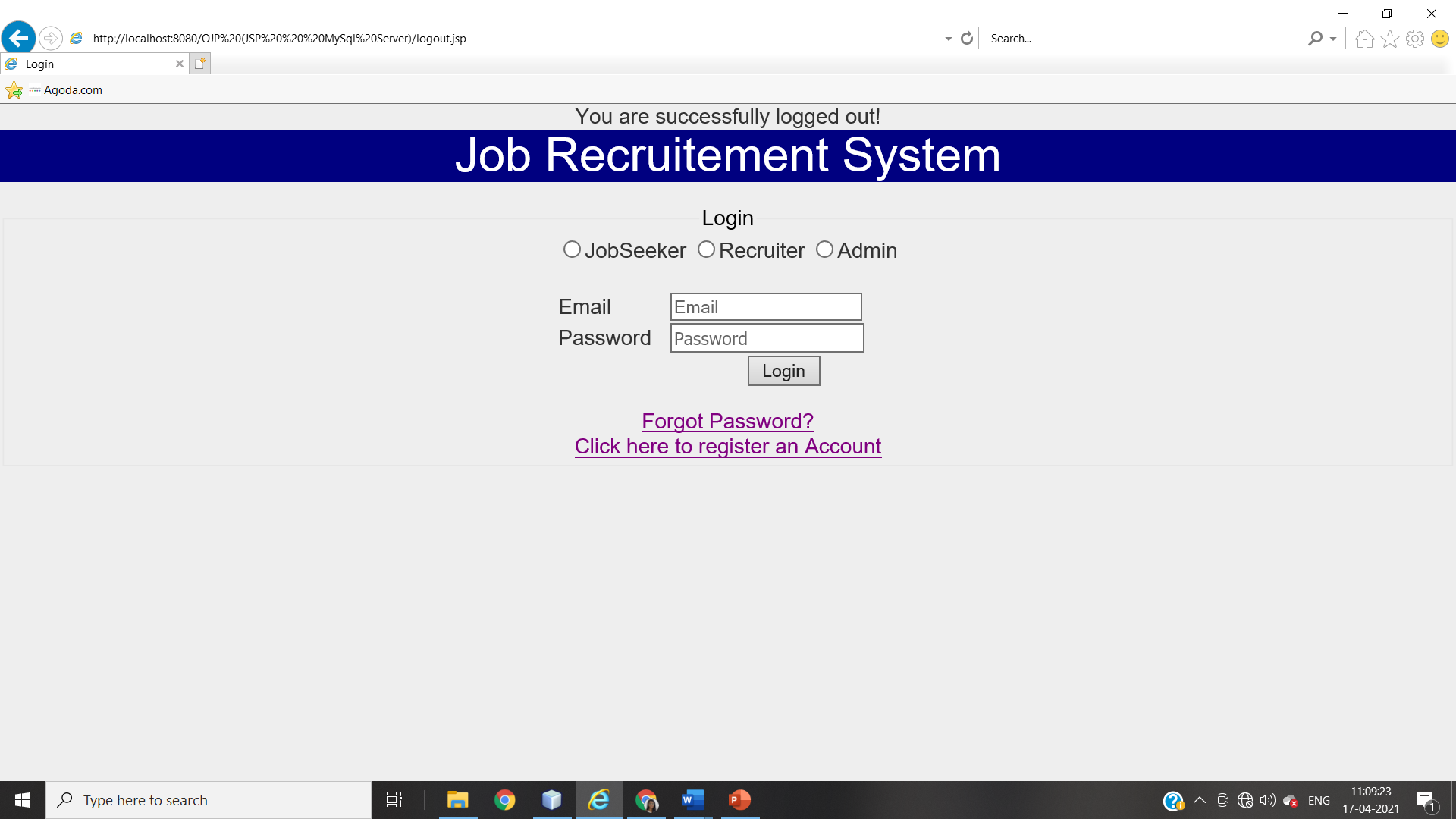
Katalon Studio , Selenium , Appium , Robotium ,Cucumber ,Watir , Sikuli

,JMeter ,WatiN ,SoapUI , Capybara ,Tarantula ,Testlink ,Windmill ,TestNG ,

Marathon ,httest , Xmind ,Wiremock ,Maven ,Espresso , FitNesse ,JUnit , Grinder ,Tsung ,Gatling ,Multi-mechanize , Selendroid ,KIF ,iMacros ,Linux Desktop Testing Tool , k6

**Unit Testing**

**Test Module: Login Page**



**Validation testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Datatype** | **Format** | **Length** | **Character**  **Set** | **Pattern** |
| Email | String | It should contain ‘@’ | Length should be >=6 | ASCII Character Set | Pattern should  \_ \_ @ \_ \_ . \_ \_ |
| It should contain ‘.’ |
| ‘@’ should be preceded and followed by a string |
| ‘.’ should preceded and followed by at least two letters |
| Before @ there should be at least one letter |
| **Equivalence Partitioning**   |  |  | | --- | --- | | Invalid | Valid | | <6 | 6, 7 and so on | | String without ‘@’ and ‘.’ | String with ‘@’ and ‘.’ | | String ending without a-z |A-Z | String ending with a-z |   **Boundary Value Analysis**   |  |  |  | | --- | --- | --- | | Test Scenario | Test Scenario Description | Expected Outcome | | 1 | Enter 0 to 5 characters in Email field | System should not accept | | 2 | Enter 6 to any characters in Email field | System should accept | | 3 | Input: srushti@gmail. | Invalid | | 4 | Input: srushti@gmail | Invalid | | 5 | Input: srushti@gmail.com | Valid | | 6 | Input: srushtigmail.com | Invalid | | | | | | |
| Password | String | It should contain lower case letter a-z | The length should be >=8 and <=50 | ASCII Character Set | At least one upper case, one lower case, one number and a special symbol |
| It should contain lower case letter A-Z |
| It should contain numeric letter |
| It should contain special symbol |
| **Equivalence Partitioning**   |  |  |  |  | | --- | --- | --- | --- | |  | Invalid | Valid | Invalid | | Length | <8 | 8,9,10…. to 49,50 | >50 | | Format | String without a number | String with a number |  | | String without a !,@,#,$,%,^,&,\*,(,), or any special sybmol | String with a !,@,#,$,%,^,&,\*,(,), or any special sybmol |  | | String without lower case | String with lower case |  | | String without upper case | String with upper case |  |   **Boundary Value Analysis**   |  |  |  | | --- | --- | --- | |  | Test Scenario Description | Expected Outcome | | 1 | Enter 0 to 7 characters in password field | System should not accept | | 2 | Enter 8 to 49, 50 any characters in password field | System should accept | | 3 | Enter 51 characters in password field | System should not accept | | 4 | Input: 4455$ure | Invalid (no upper case) | | 5 | Input: 44$5sure | Invalid (no upper case) | | 6 | Input: sure$urE | Invalid (no number) | | 7 | Input: 4455$URE | Invalid (no lower case) | | 8 | Input: 445$surE | Valid | | | | | | |
|  | | | | | |
| User-type | String | Radio Button | - | - | Admin, Jobseeker or Recruiter |

**Verification Testing**

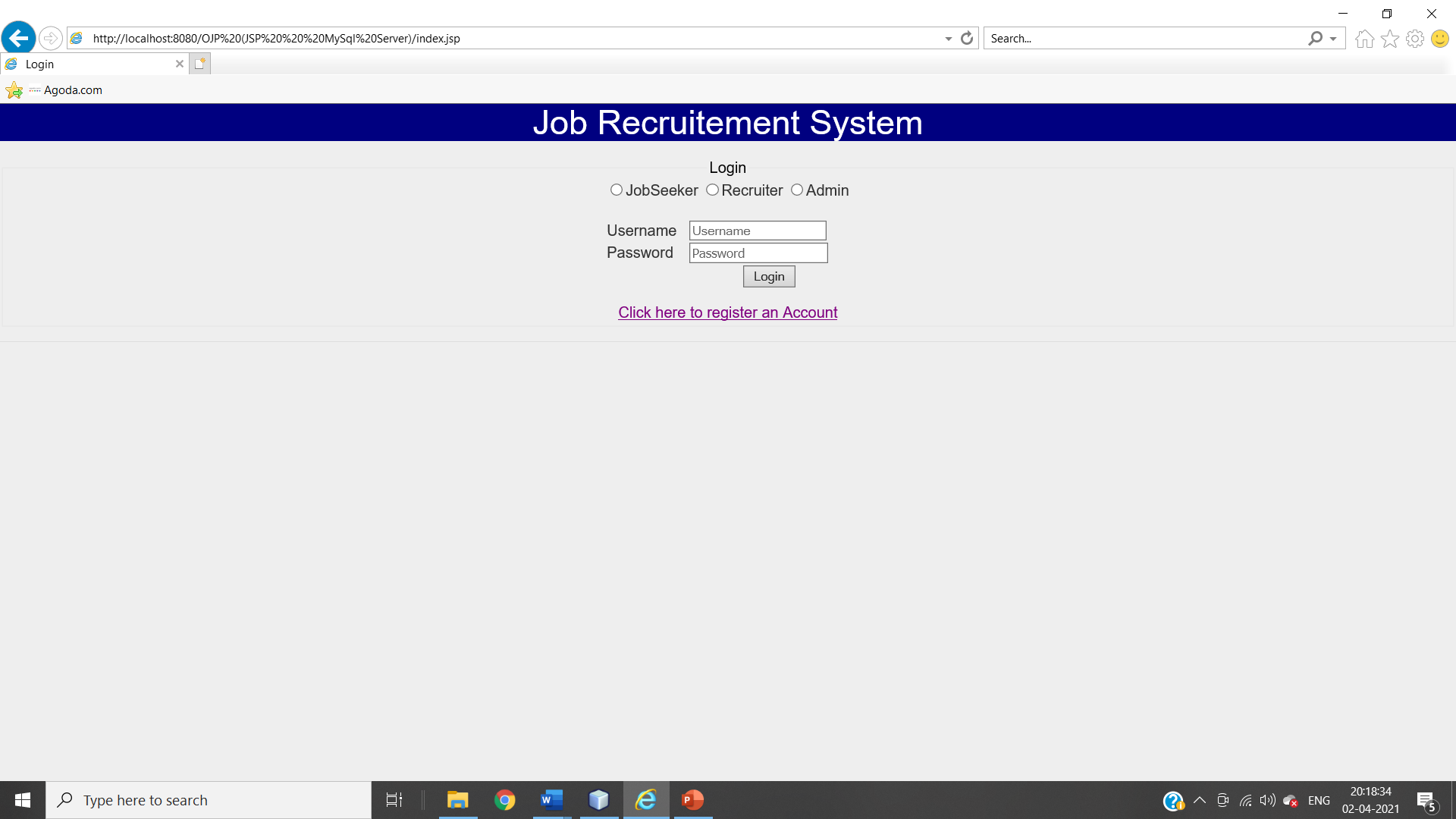
|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Case** | **Steps** | **Result** |
| 1 | Verify if the user will be able to login with a valid email and valid password | 1. Open the browser 2. Fill the form   Valid: email: [srushti@gmail.com](mailto:srushti@gmail.com)  Password: 445$surE   1. Click Submit | Positive  Message: ‘logged in successfully’ |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver();  driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("usertype")).click();  driver.findElement(By.name("username")).sendKeys("srushti@gmail.com");  driver.findElement(By.name("password")).sendKeys("445$urE");  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |
| 2 | Verify if a user cannot login with a valid email and invalid password | 1. Open the browser 2. Fill the form   Valid: email: [srushti@gmail.com](mailto:srushti@gmail.com)  Invalid: Password: 445$surr   1. Click Submit | Positive  Message: ‘cannot login, invalid password’ |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver(); driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("usertype")).click();  driver.findElement(By.name("username")).sendKeys("srushti@gmail.com");  driver.findElement(By.name("password")).sendKeys("445$urr");  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |
| 3 | Verify if the form can be submitted when blank | 1. Open the browser 2. Click Submit | Positive  Message: ‘cannot submit’ |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver();  driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |
| 4 | Verify if the user type if chosen | 1. Open the browser press submit | Positive  Message: ‘choose a user type’ |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver();  driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("username")).sendKeys("srushti@gmail.com");  driver.findElement(By.name("password")).sendKeys("445$urE");  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |
|  | | | |
| 5 | Verify if the email, password and user type match | 1. Open the browser 2. Choose a user type (Job Seeker) 3. Enter email and password   Email: [srushti@gmail.com](mailto:srushti@gmail.com)  Password: 445$surE   1. Click Submit | Positive  Message: ‘logged in successfully’ |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver();  driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("usertype")).click();  driver.findElement(By.name("username")).sendKeys("srushti@gmail.com");  driver.findElement(By.name("password")).sendKeys("445$surE");  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |
| 7 | Verify the message for invalid login | 1. Open the browser 2. Fill the form   Valid: email: [srushtigmail.com](mailto:srushti@gmail.com)  Invalid: Password: 445$surr   1. Click Submit | Positive:  Message: ‘Enter valid email and password’ |
|  | | | |
| 8 | Verify if the data in password field is visible | 1. Open Browser 2. Enter password | Positive:  ‘bullet points seen’ |
|  | | | |
| 9 | Verify if the user cannot login in the length doesn’t match |  |  |
| package javaapplication1;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.support.ui.Select;  import org.openqa.selenium.WebElement;  public class JavaApplication1 {  public static void main(String[] args) {  System.setProperty("webdriver.chrome.driver", "H:\\chromedriver\_win32\\chromedriver.exe");  WebDriver driver = new ChromeDriver();  driver.get("http://localhost:8080/OJP%20(JSP%20%20%20MySql%20Server)/index.jsp");  driver.manage().window().maximize();  driver.findElement(By.name("usertype")).click();  driver.findElement(By.name("username")).sendKeys("srushti@gmail.com");  driver.findElement(By.name("password")).sendKeys("445rE");  driver.findElement(By.name("Submit")).click();  String at = driver.getTitle();  String et = "Login";  if(at.equals(et)){  System.out.println("Test Successful");  }  else{  System.out.println("Test Failure");  }  }  } | | | |

**Integration Testing**

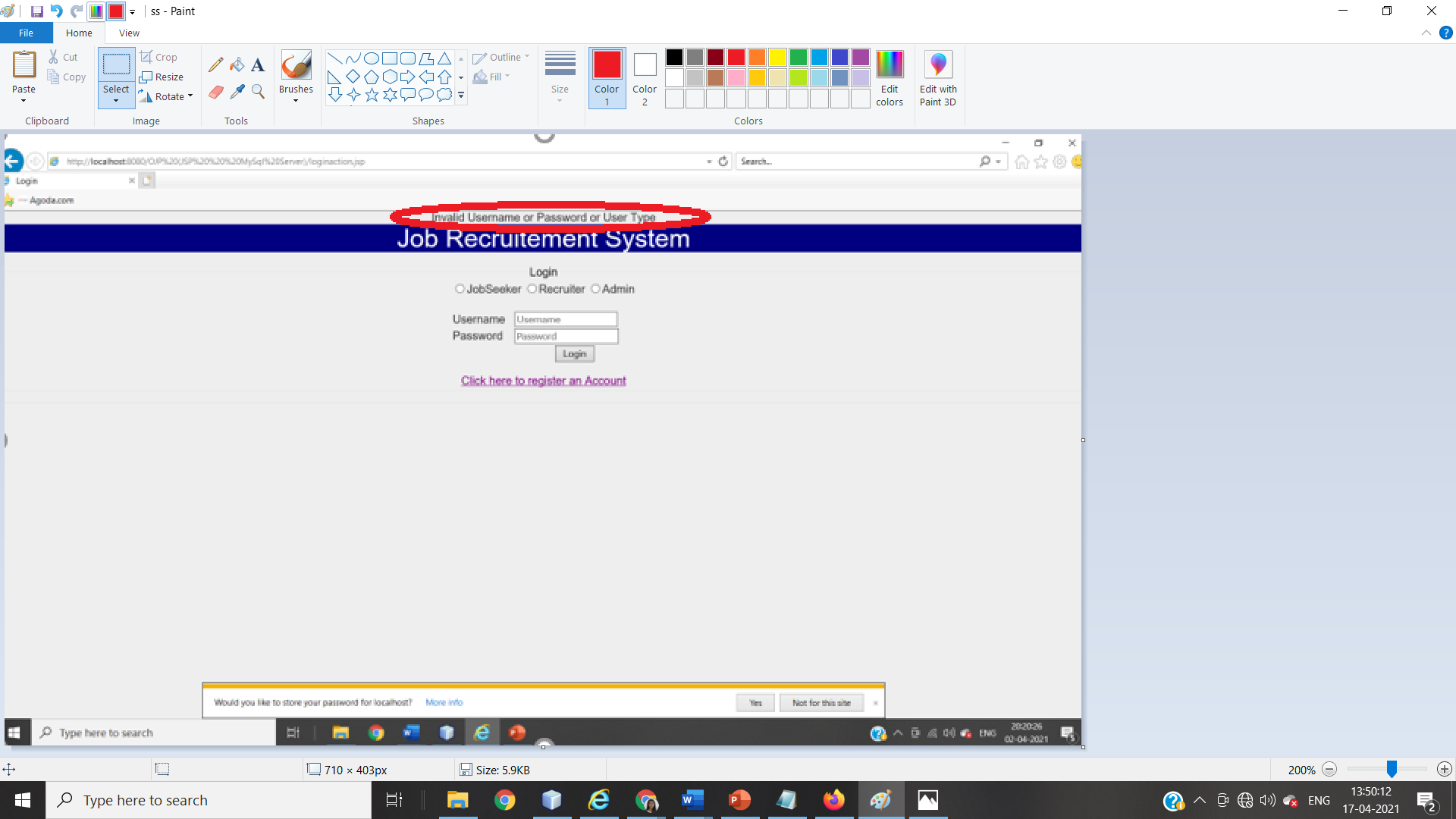
Integration testing is performed after unit testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. All the modules of the proposed system are tested through top-down Integration testing technique.

1. **LOGIN PAGE**

The login page must route to the Home Page of respective user, provided the details are correct

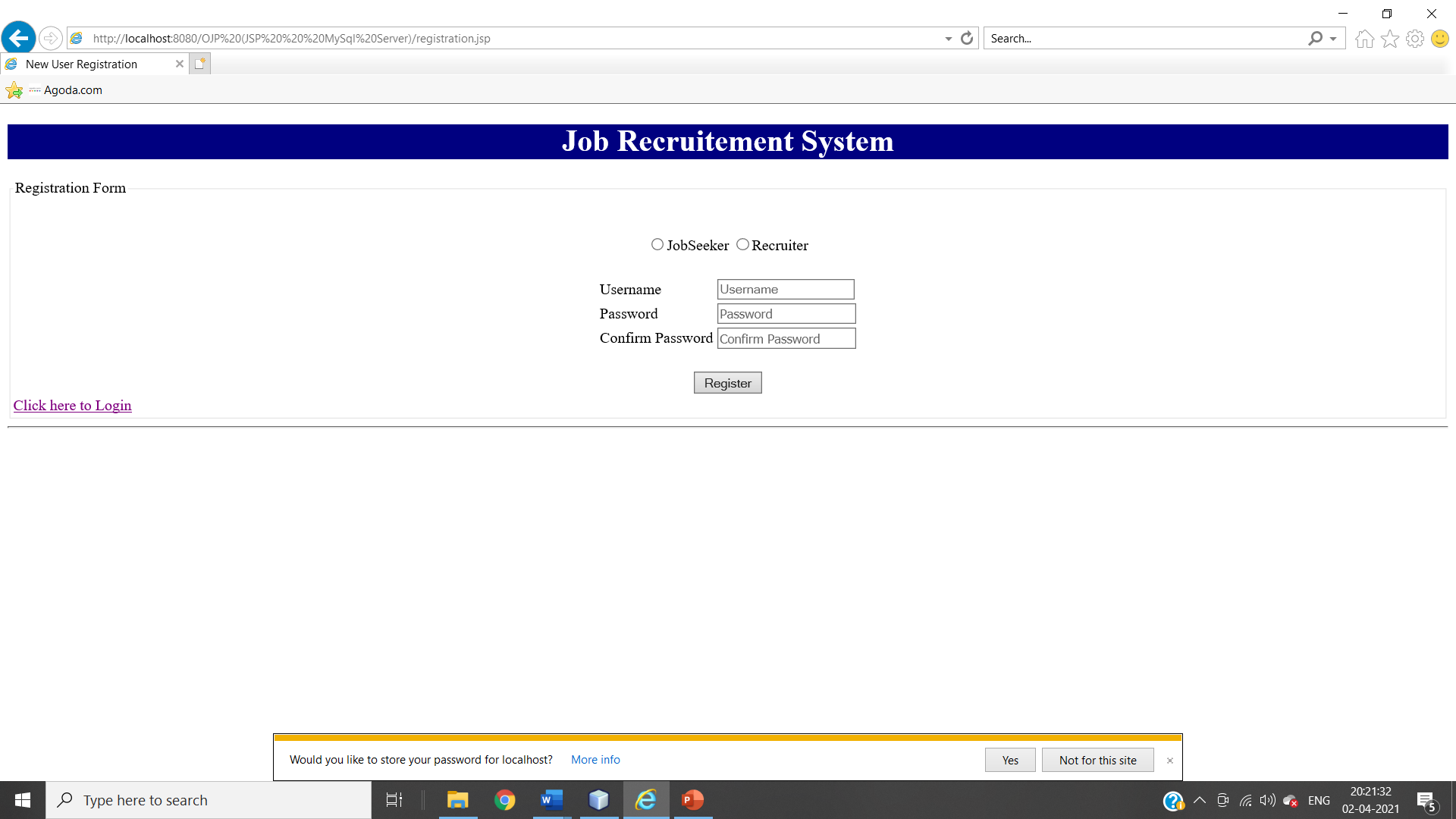


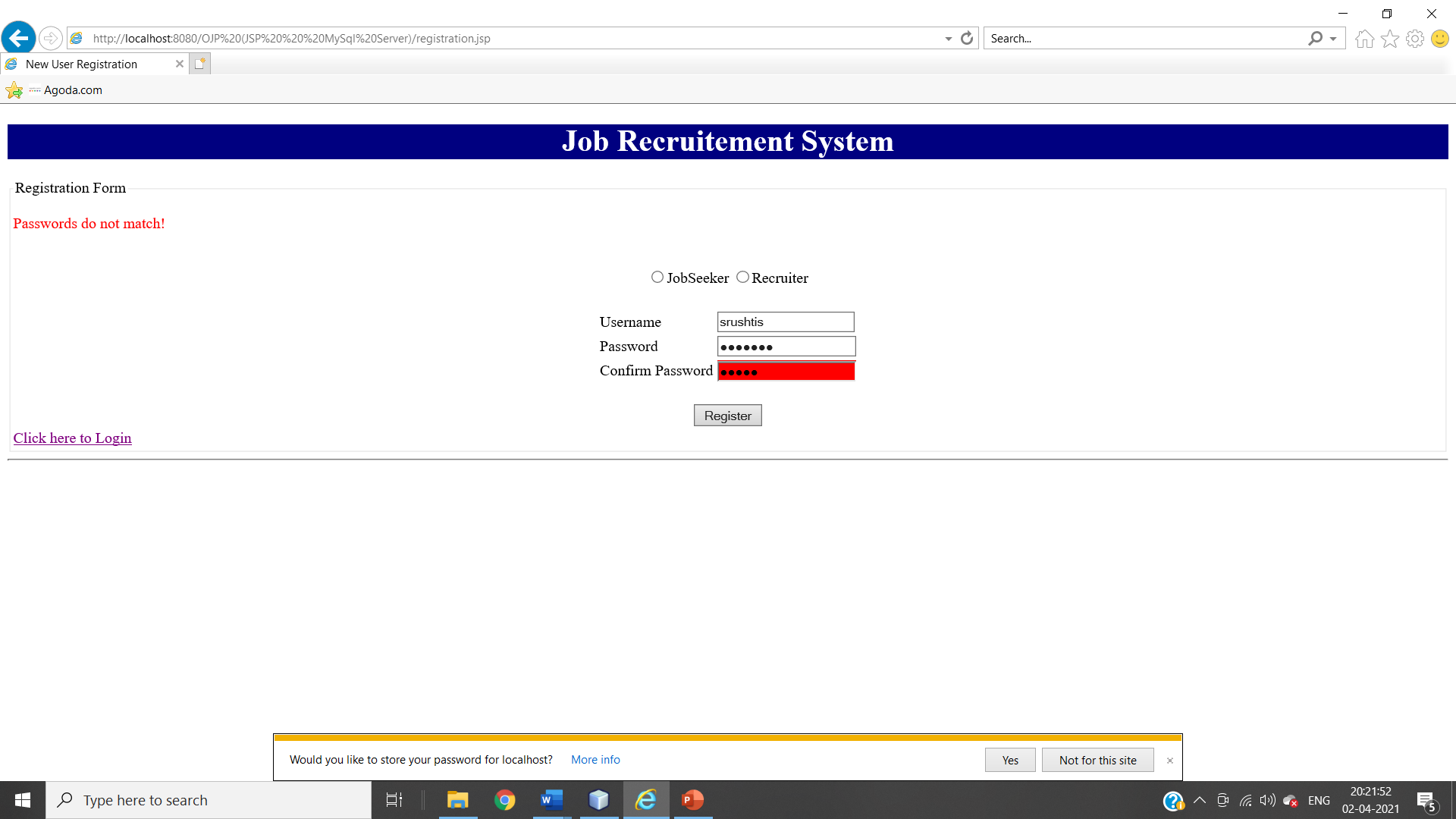
**When Trying to Login with Wrong Details**

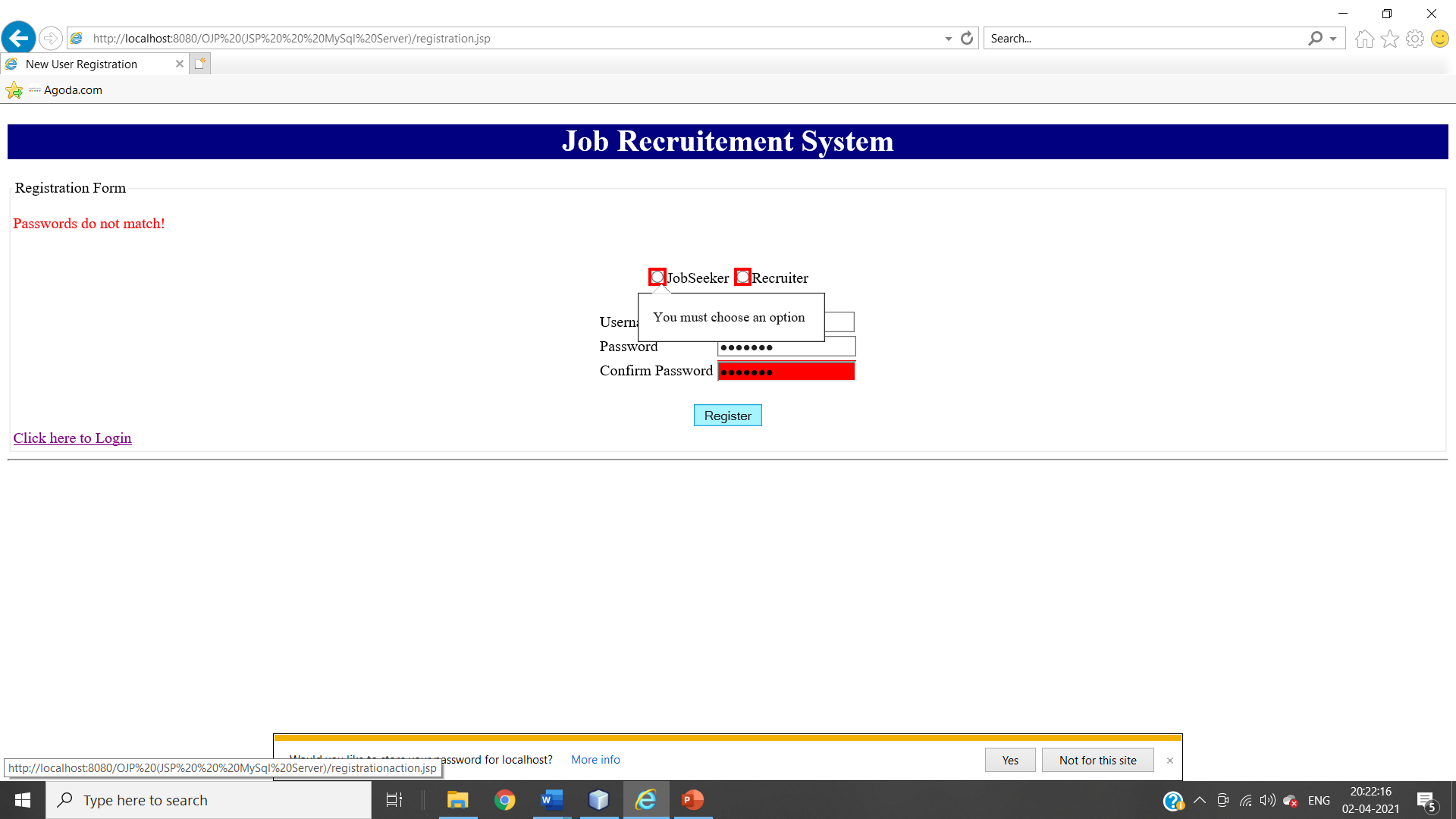


1. **NEW USER REGISTRATION PAGE**

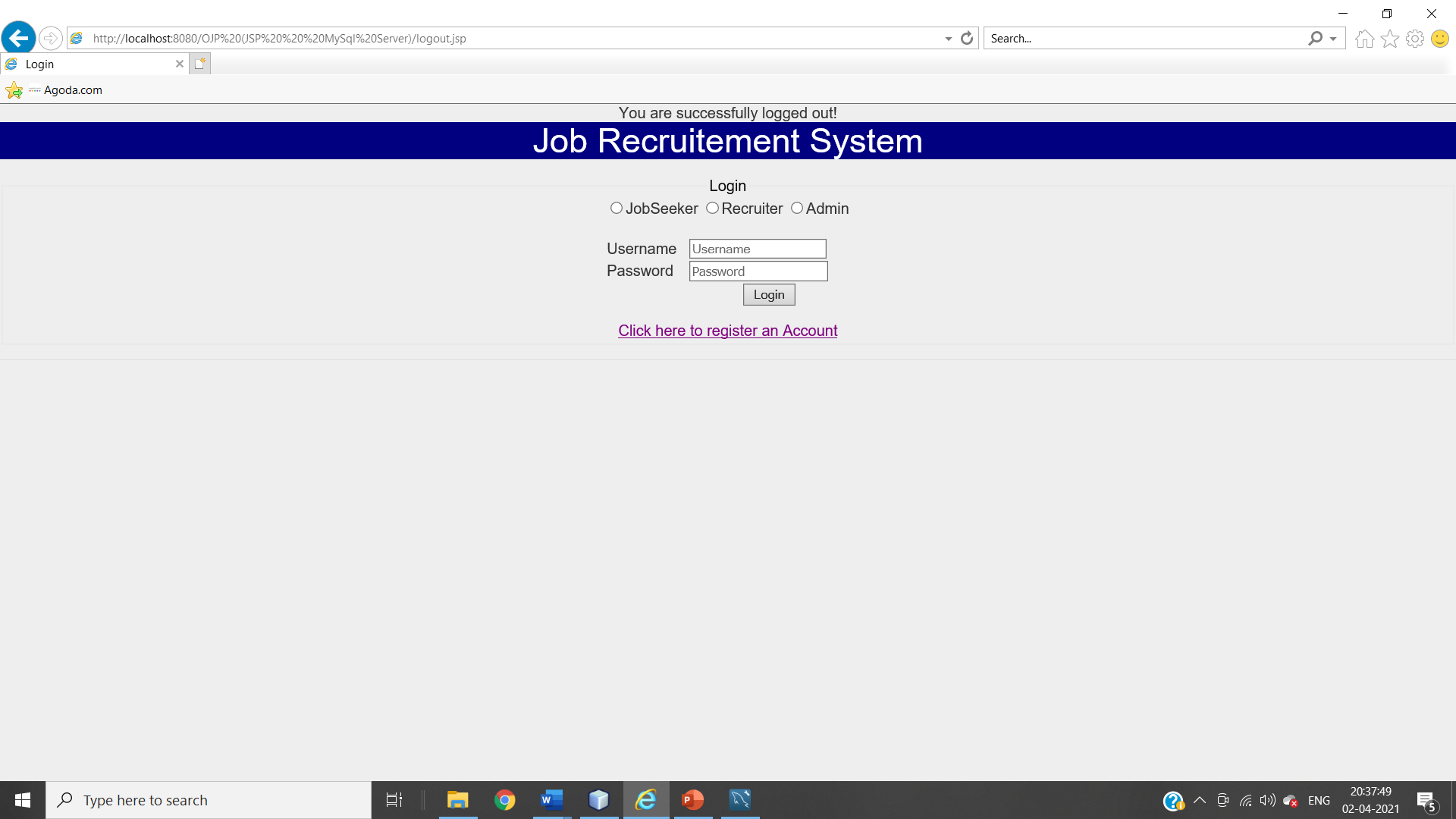
On click on ‘click here to login’, the page must be routed to Login page



**When Passwords Don’t Match While Registration Process** 

**When Registering Without Selecting Usertype** 

1. **LOGOUT PAGE**

When User Do Logout From Session, he should be directed to the login page

1. **JOBSEEKER HOME PAGE**

JOBSEEKER HOME PAGE

JOB STATUS CHECK

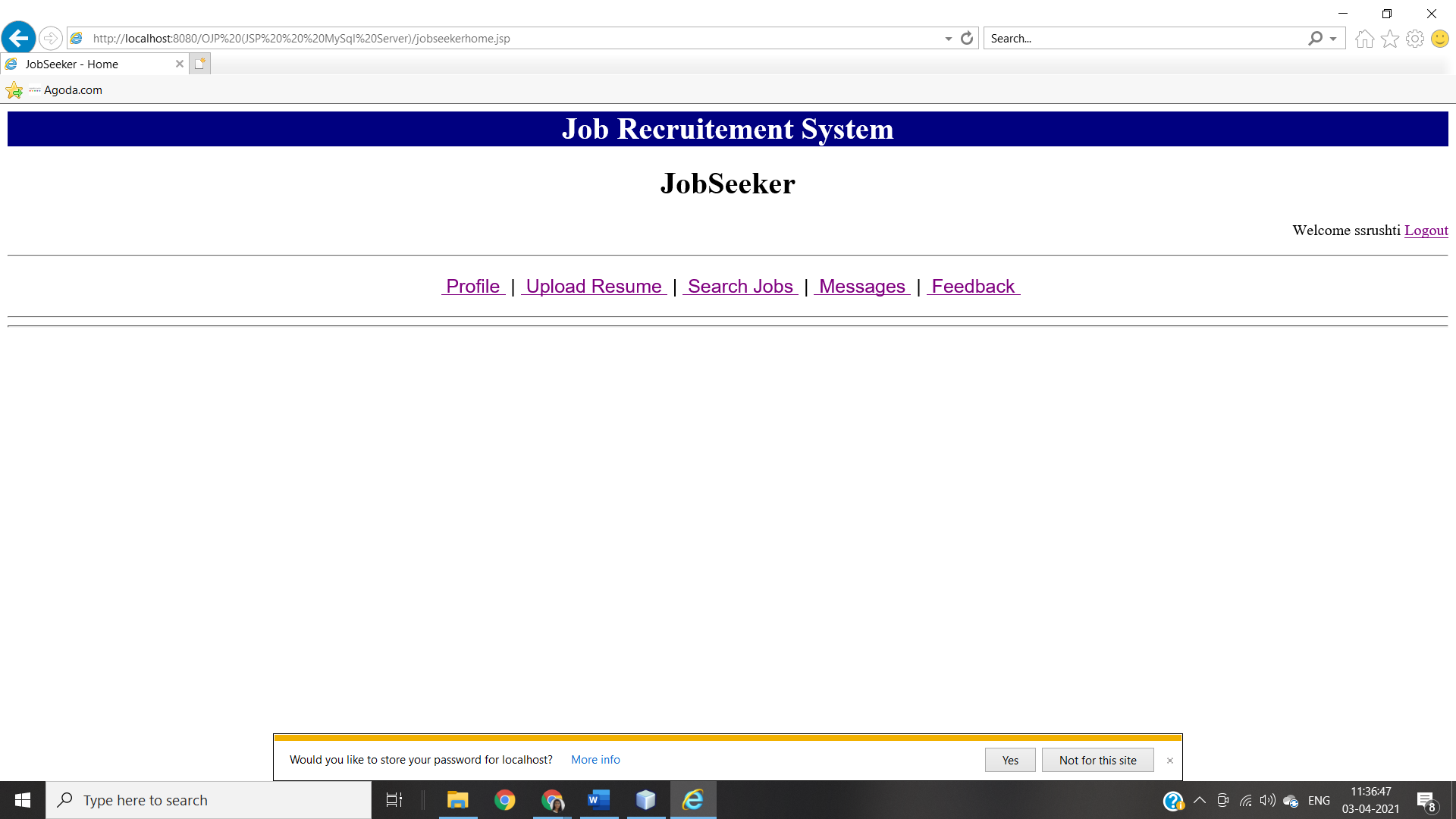
JOB SEARCH

PROFILE DETAILS

JOB ID GENERATION

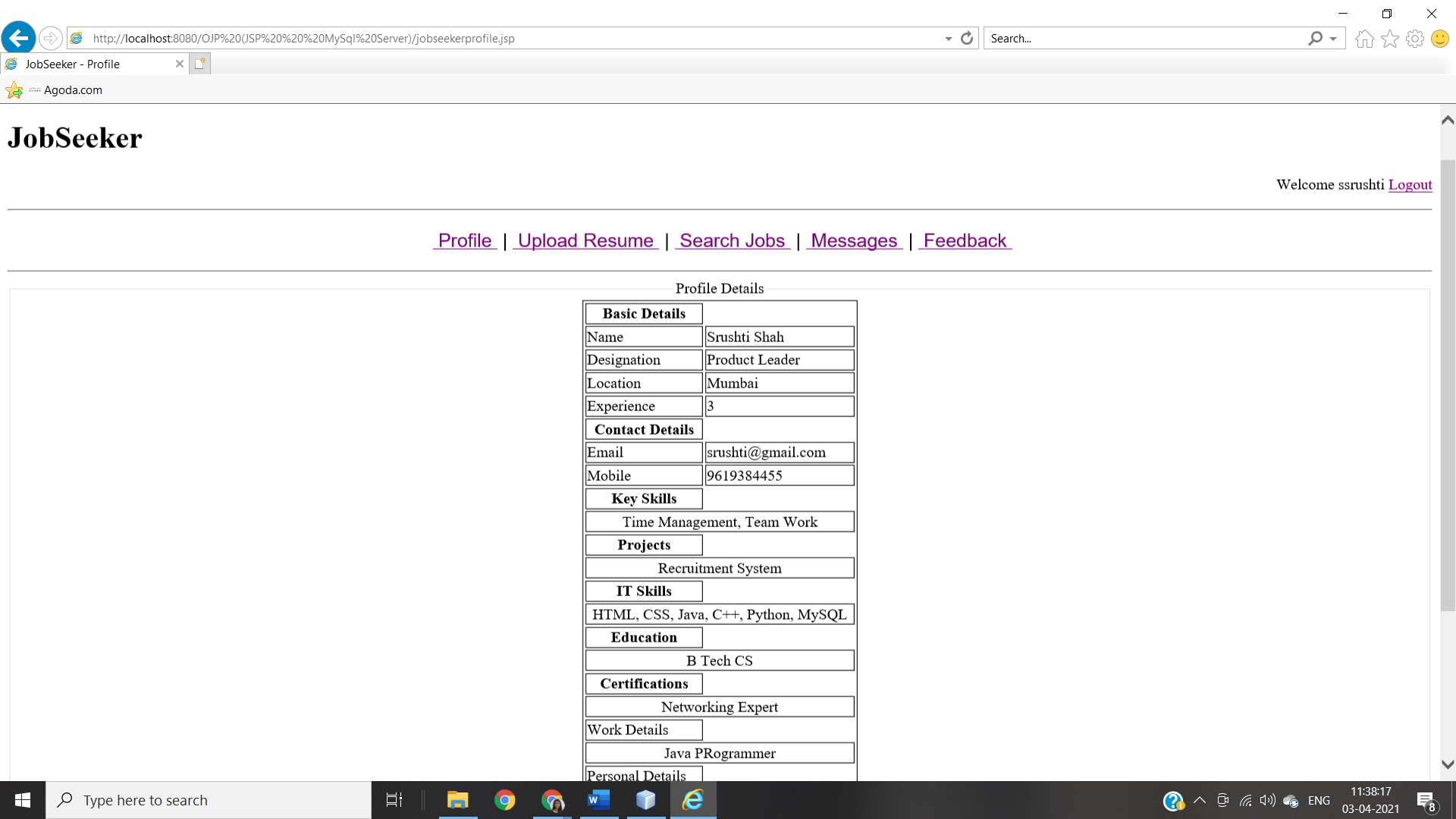
APPLYING FOR JOB

**JOBSEEKER HOME PAGE**



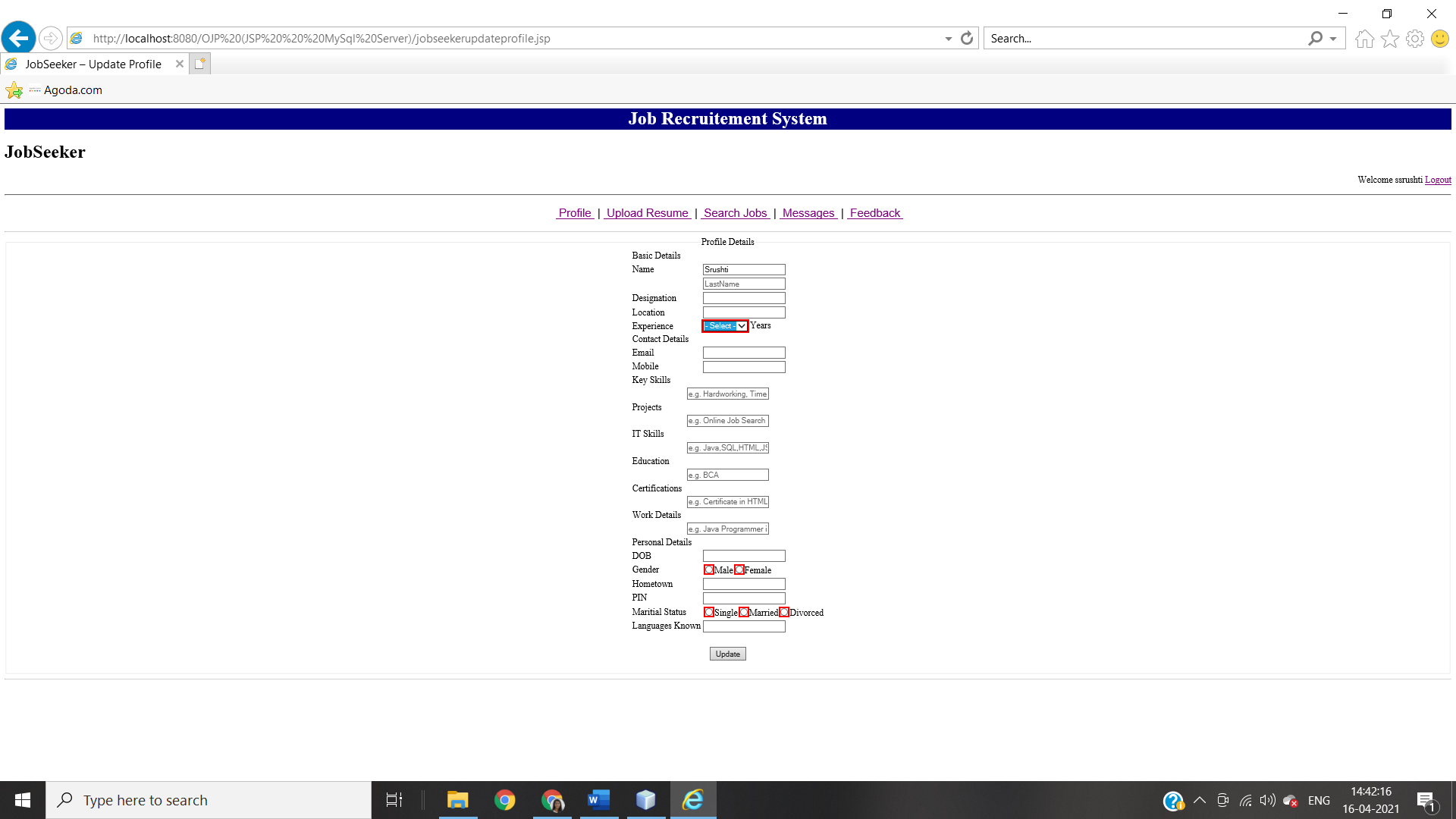
1. Profile: - This page will open profile page of current logged in user.
2. Upload Resume:-This link is used to upload resume.
3. Search Jobs:-This link is used to search jobs posted by recruiters.
4. Messages:-This link is used to view received messages about job selection.
5. Feedback:-This link is used for feedback page to submit a feedback.

**‘PROFILE’ PAGE ON JOBSEEKER**

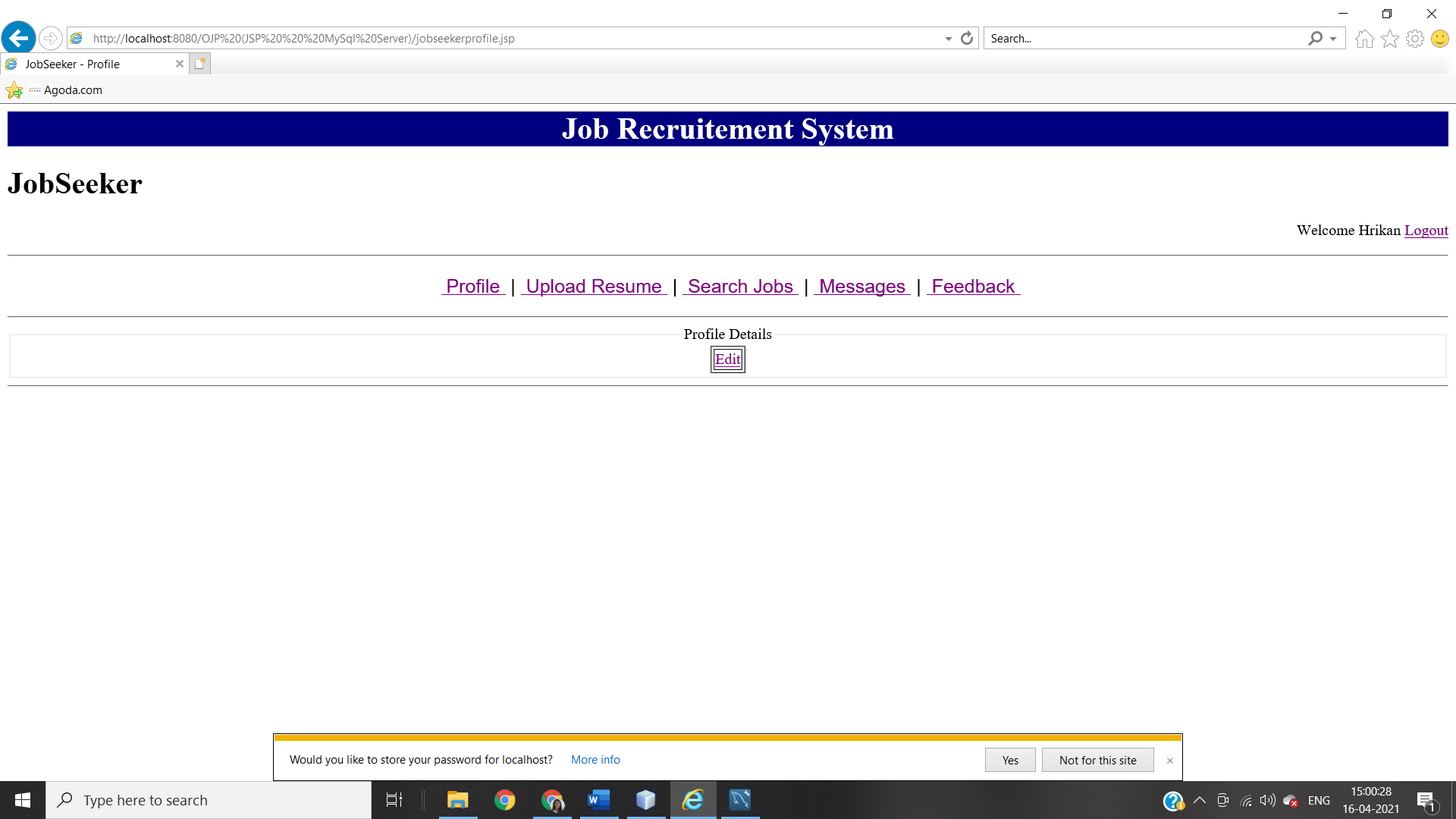
**When JobSeeker Updated There Details** This page is for jobseeker. All possible activities for the jobseeker are listed here.

1. Edit: -

This link is used to edit/update logged jobseeker’s profile details.

**When JobSeeker Clicks On ‘Edit’**

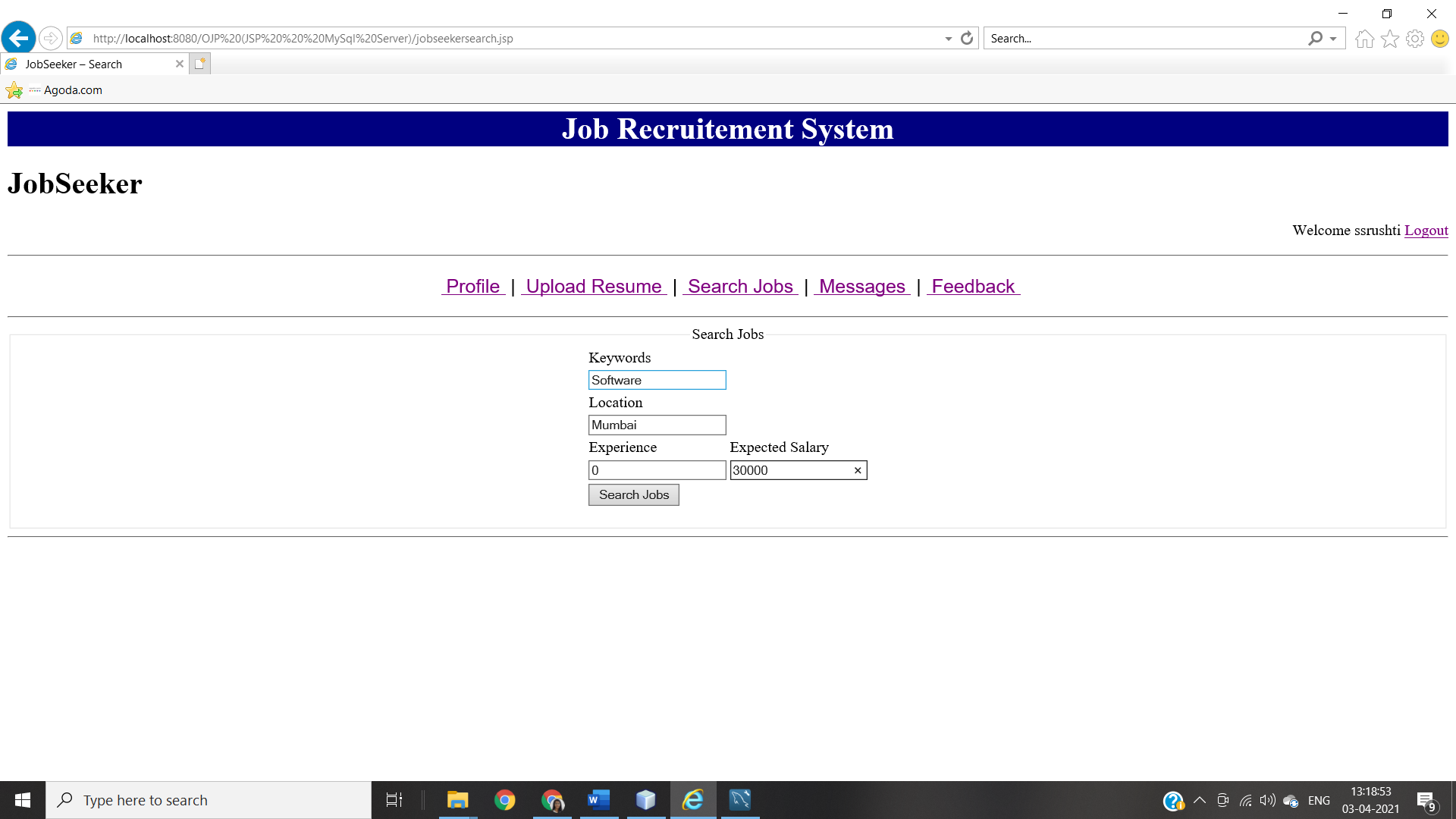
By clicking on ‘**Update**’ Button JobSeeker can update there profile details.

**When JobSeeker Is New**By clicking on **‘Edit’** JobSeeker can edit profile details.

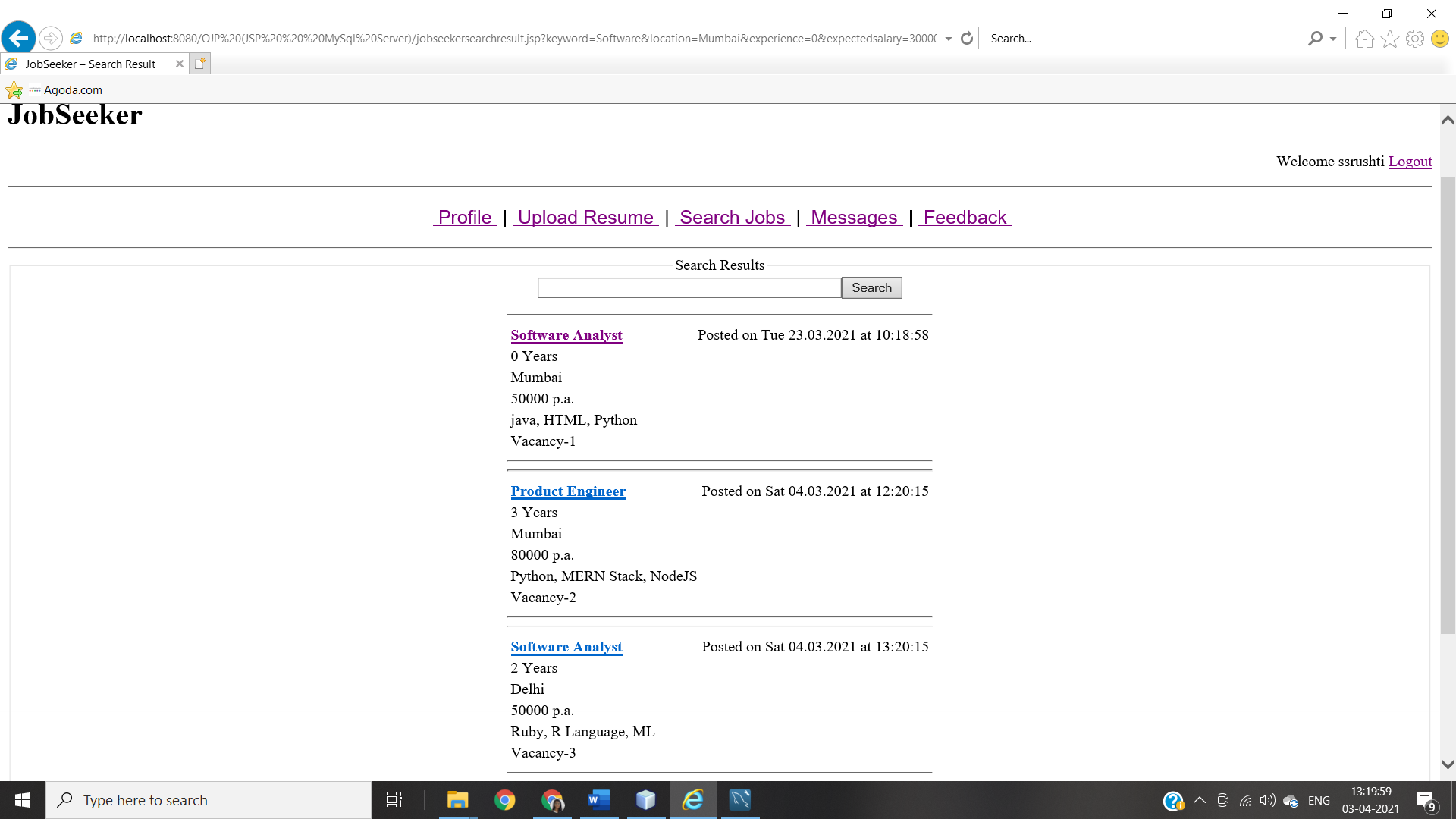
**‘UPLOAD RESUME’ PAGE ON JOBSEEKER**

By Clicking on **“Upload”** JobSeeker can upload there resume on website.

**‘SEARCH JOBS’ PAGE ON JOBSEEKER**



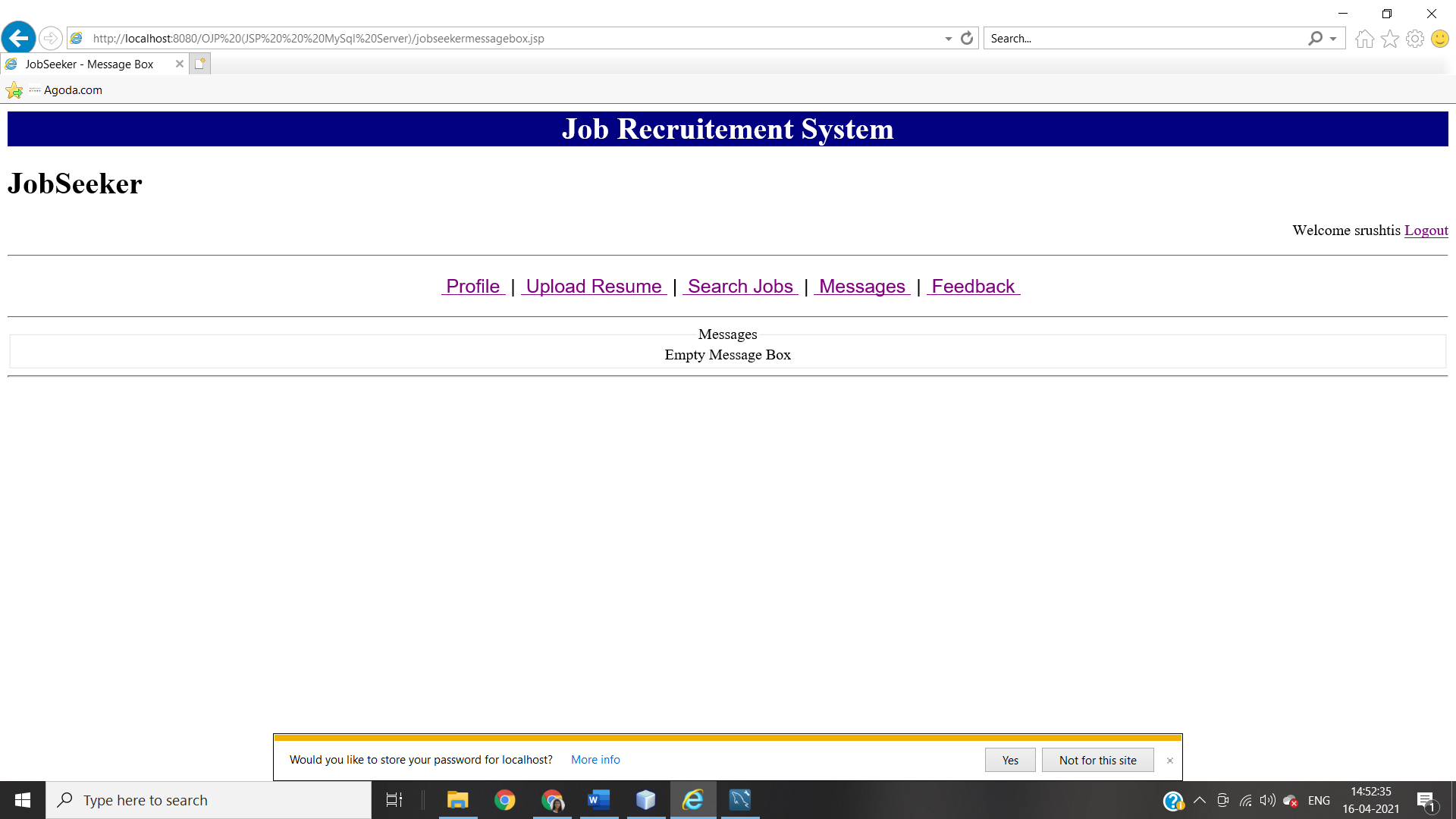
**Search Result Page When Searched For “Keywords:Software, Location:Mumbai, Experience:0, Expected Salary:30000”**



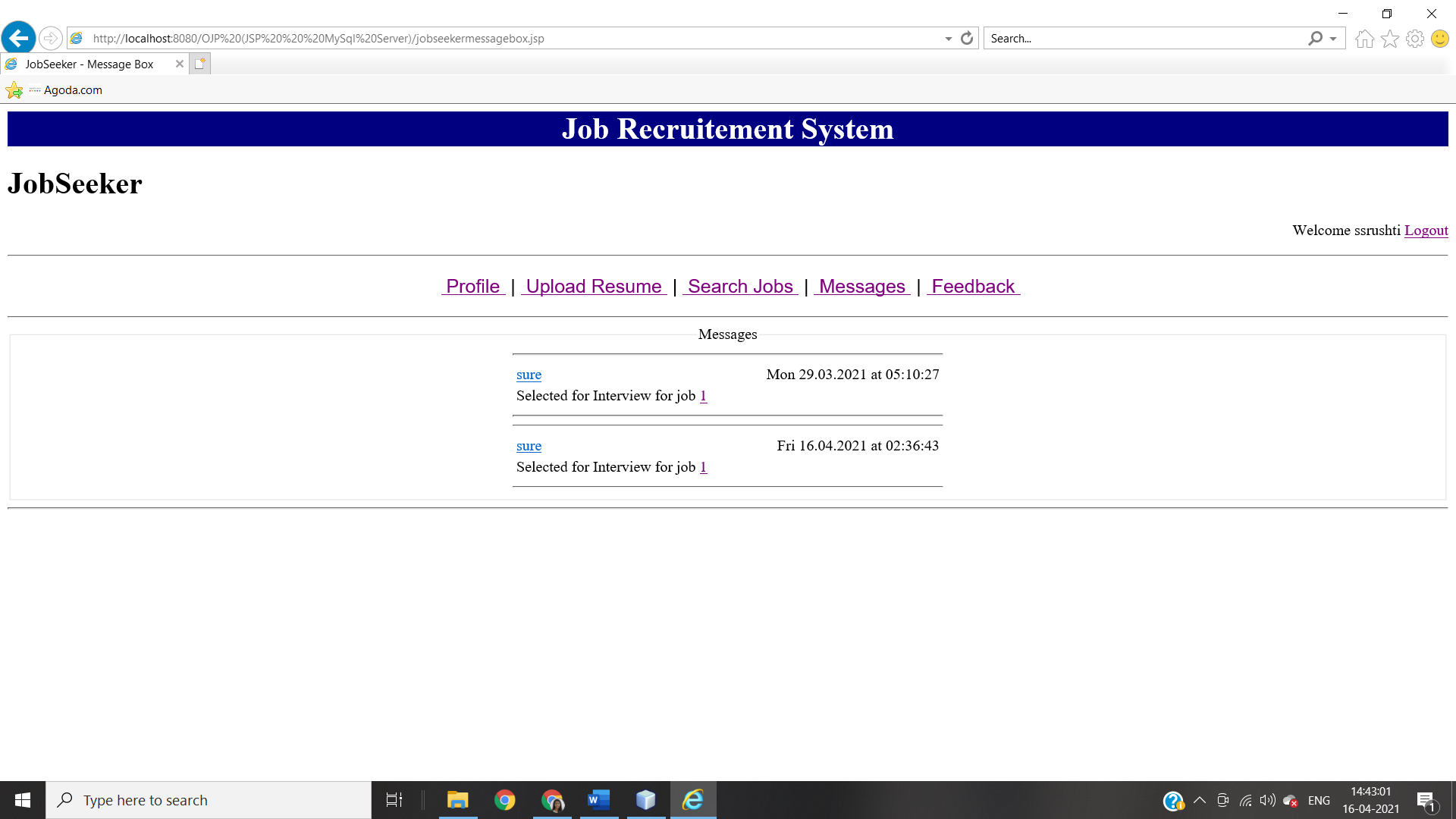
**JOB DETAILS PAGE ON JOBSEEKER**

By clicking on “**Apply for this Job**” JobSeeker can apply for selected job.

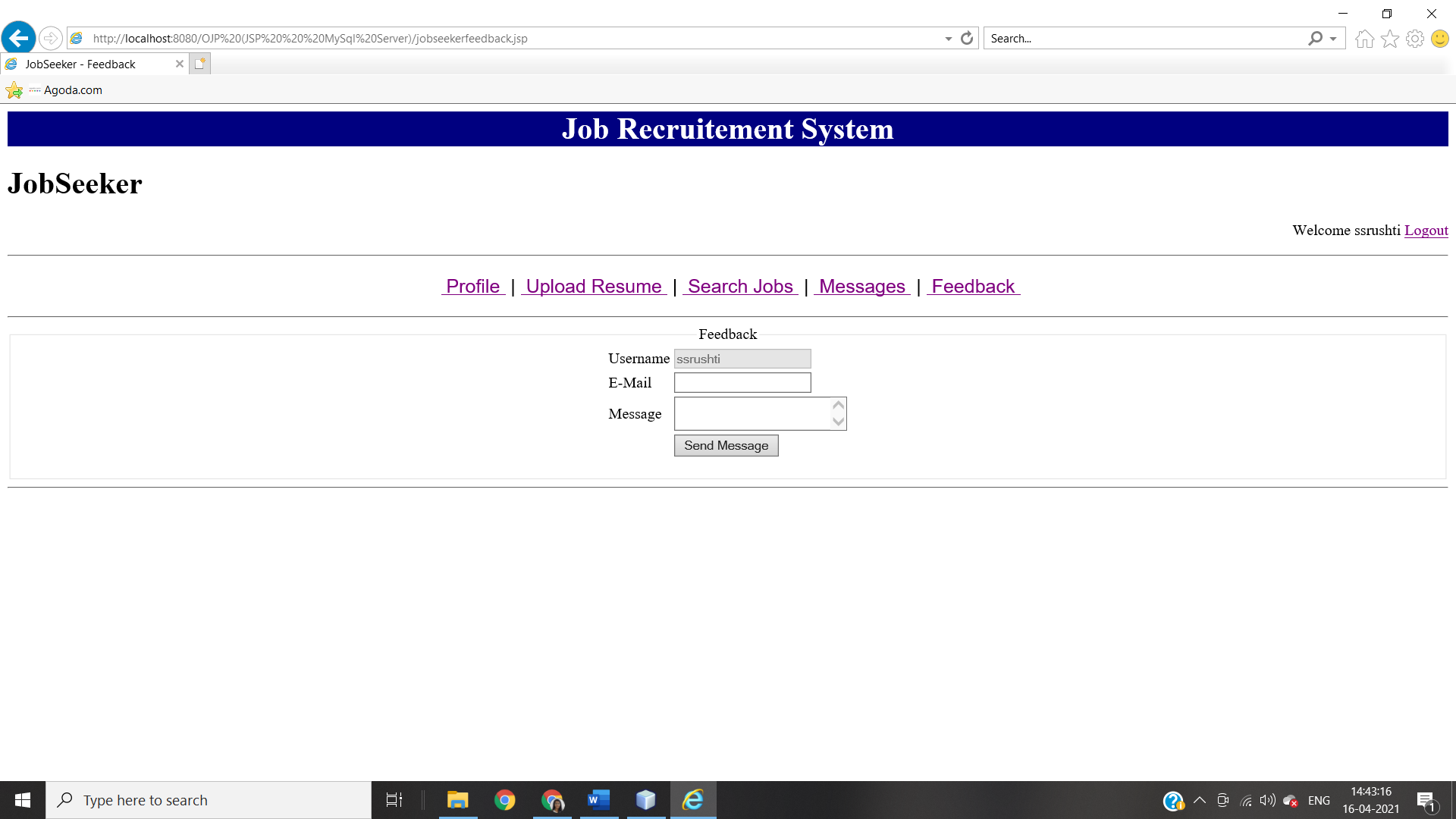
**‘MESSAGES’ PAGE ON JOBSEEKER**

**When There Is No Message**

**When there are messages**



**‘FEEDBACK’ PAGE ON JOBSEEKER**

**JobSeeker’s Feedback Page**

1. **RECRUITERS HOME PAGE MODULE INTEGRATION TEST**

RECRUITER HOME PAGE

JOBSEEKER SEARCH

COMPANY DETAILS

JOB POSTING

JOB POSTING

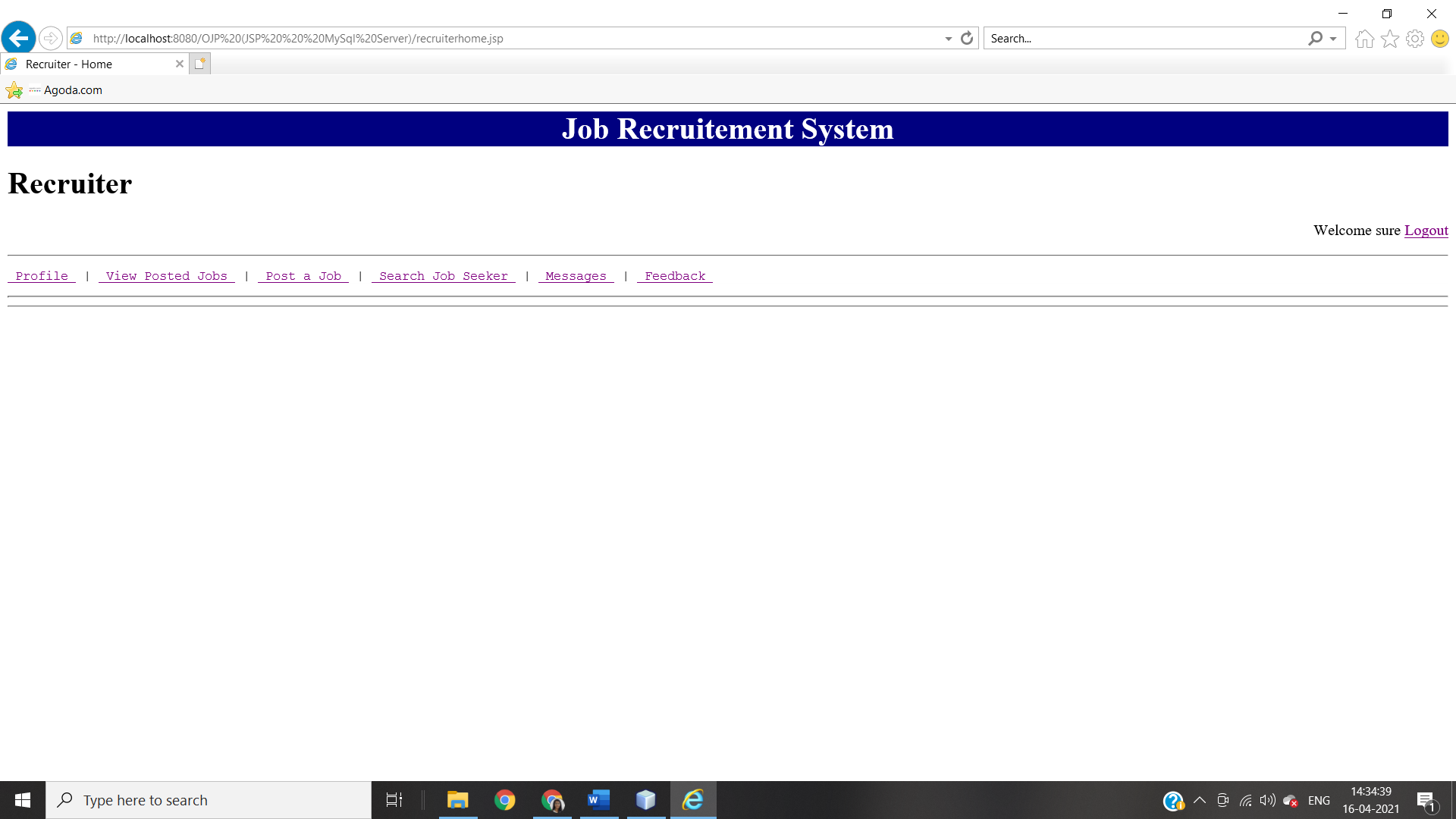
CONFIRMATION

REPORTS

JOBSEEKER PROFILE

SELECT FOR INTERVIEW

**RECRUITER HOME PAGE**



All possible activities for the recruiters are listed below.

1. Profile :-

This link is used to view/update profile of logged recruiter.

1. View Posted Jobs:-

This link will show jobs that is posted by current logged in user.

1. Post a Job:-

This link is used to post new job.

1. Search JobSeeker:-

This link is used to search registered jobseekers.

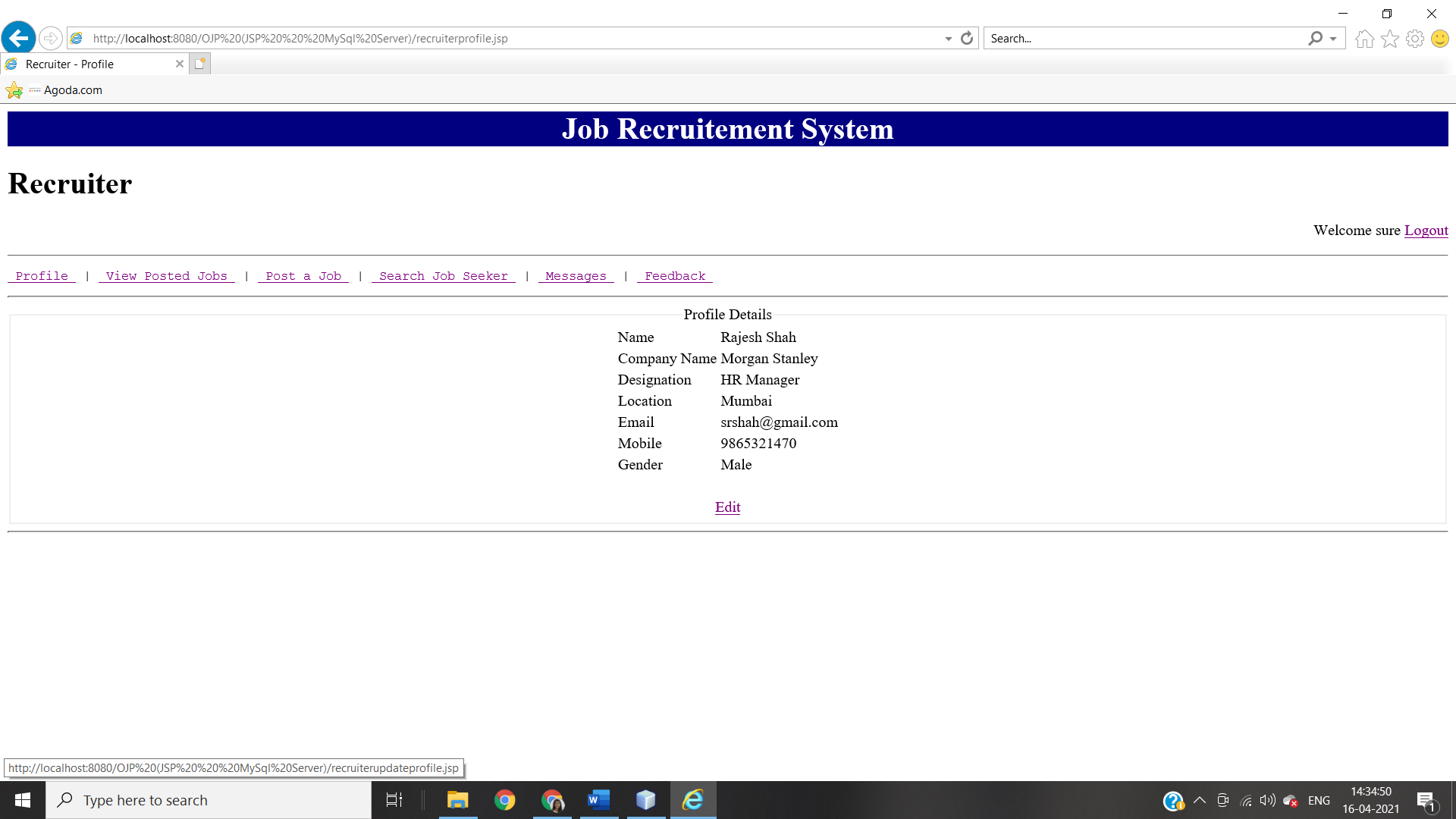
1. Message:-

This link is used to message box which contains information about jobseekers who have applied to jobs.

1. Feedback:-

This link is used to submit feedback about website.

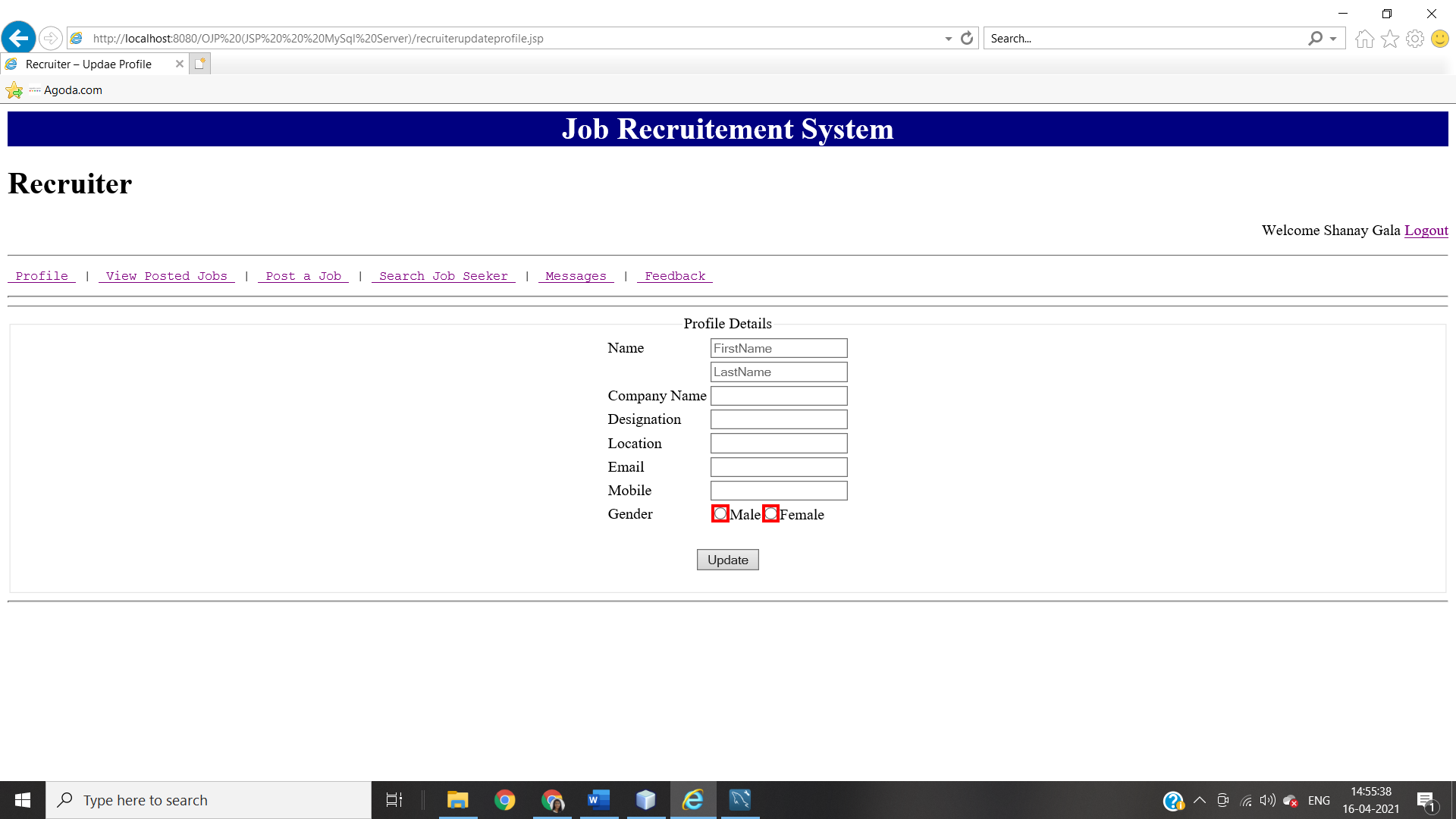
**‘PROFILE’ PAGE ON RECRUITER**

**When Recruiter Updated There Details**

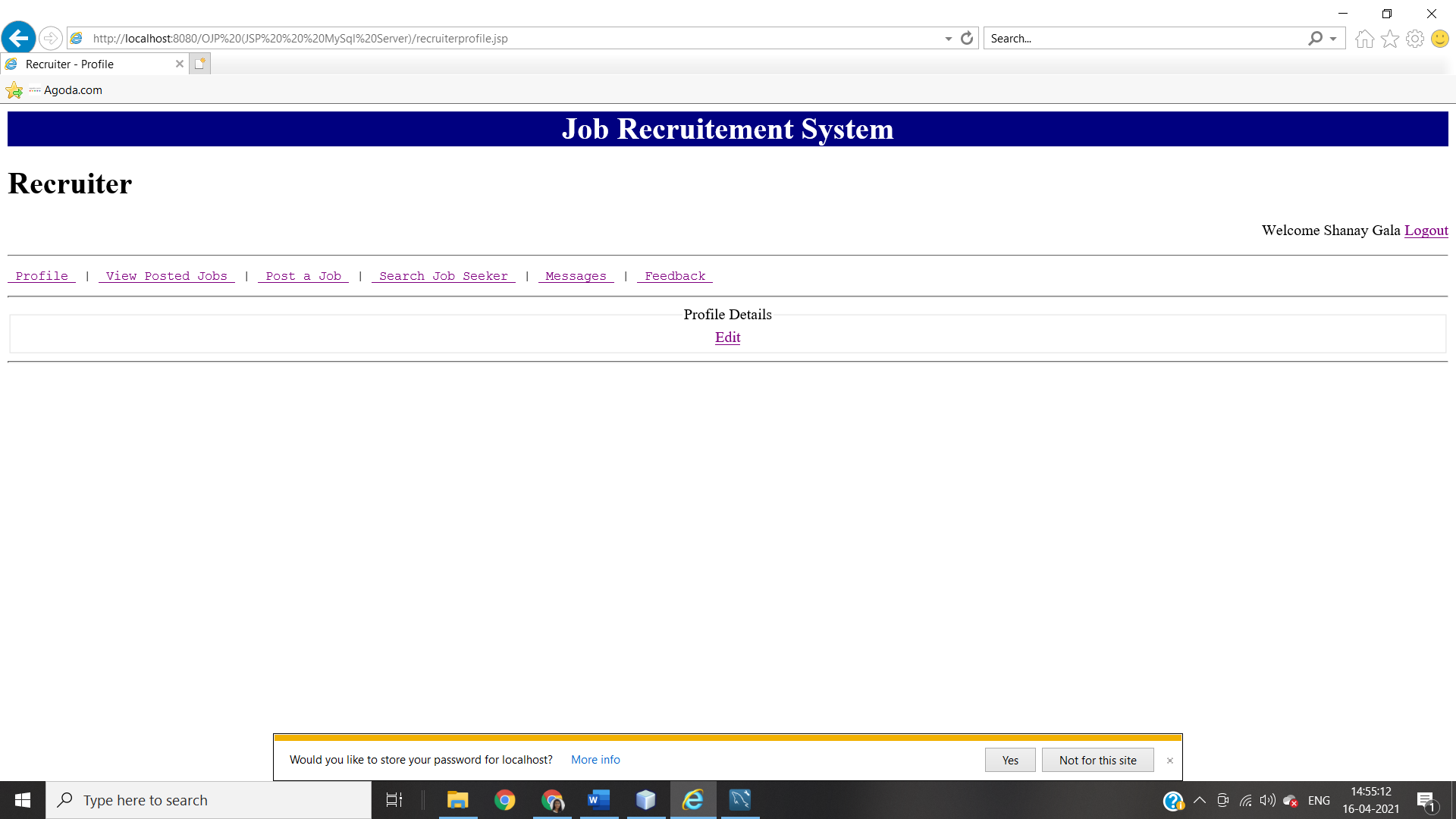
This page is for recruiter. All possible activities for the recruiter are listed here.

1. Edit: -

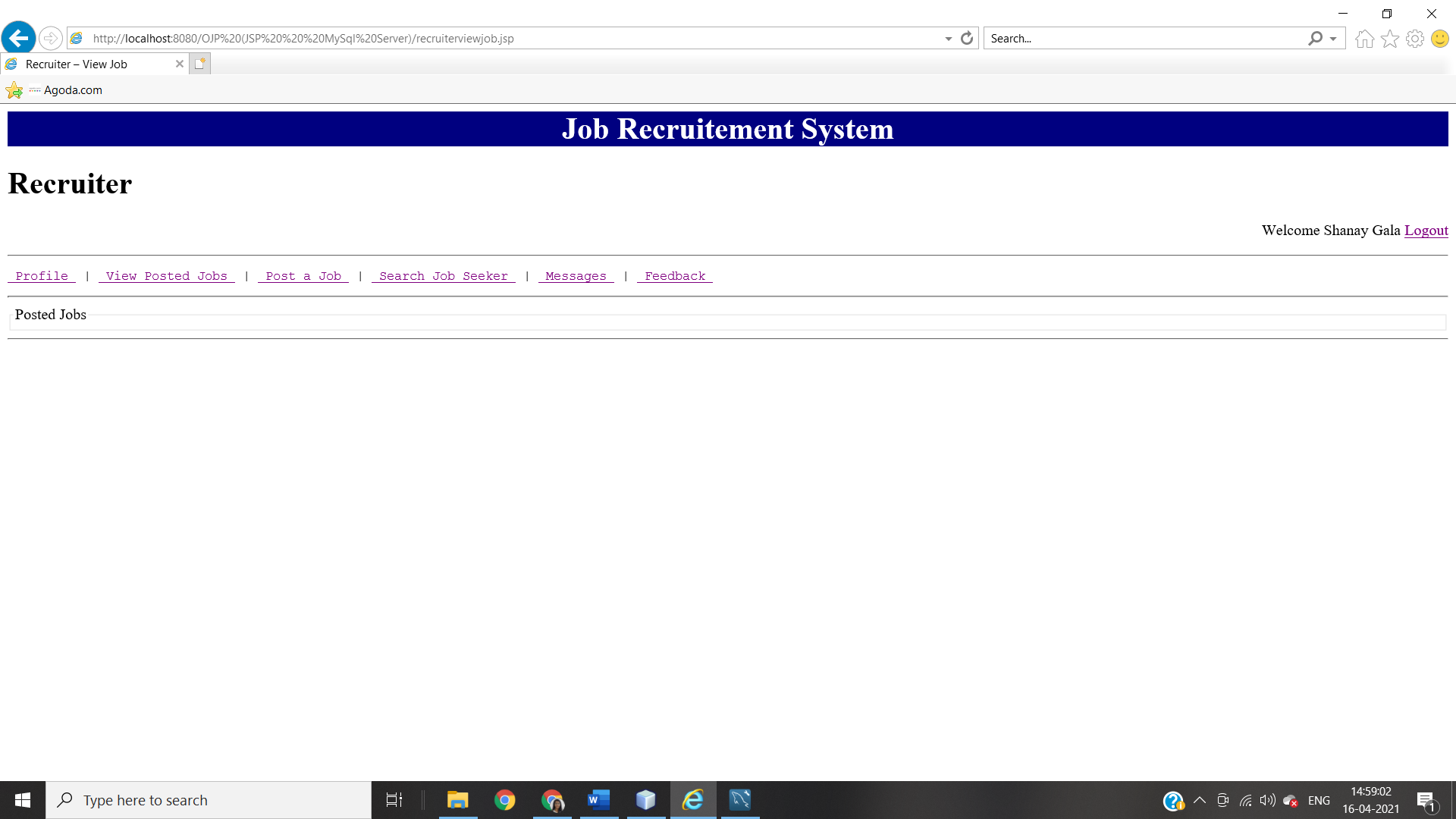
This link is used to edit/update logged recruiter’s profile details.

**When Recruiter Clicks On ‘Edit’**By clicking on **‘Update’** Button Recruiter can update there profile details.

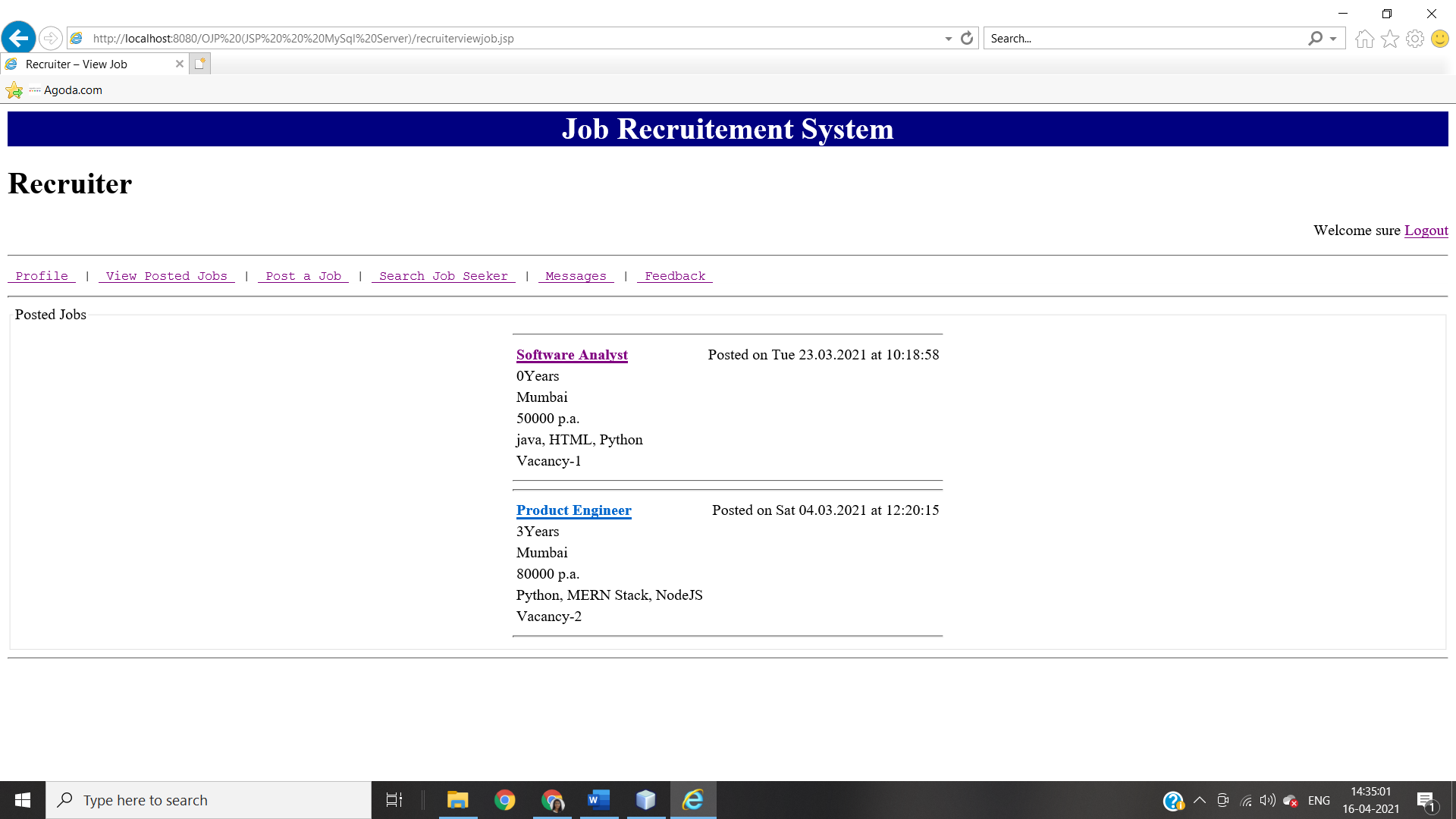
**When Recruiter Is New**

By clicking on’**Edit**’ recruiter can edit there profile details

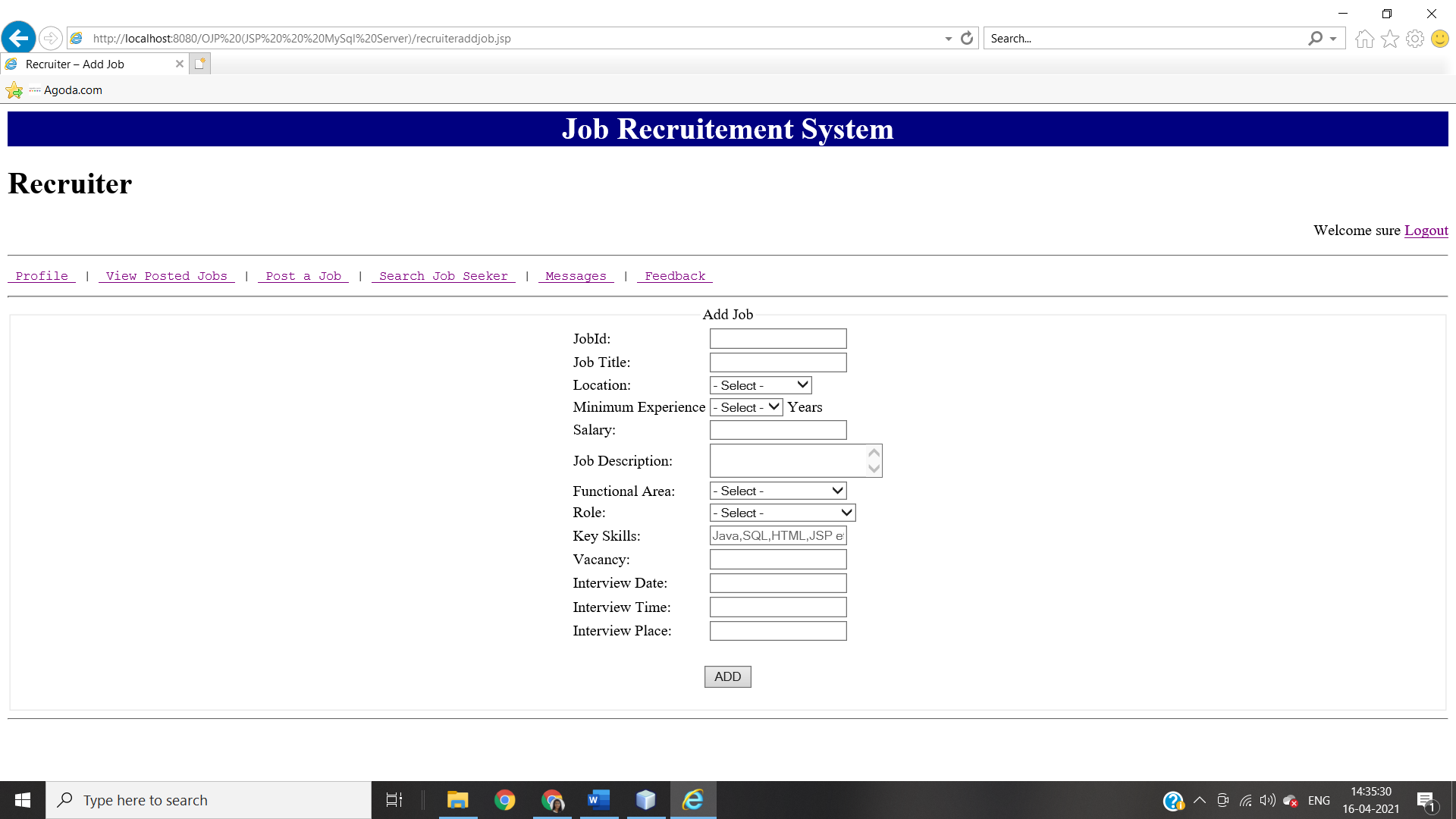
**‘VIEW POSTED JOBS’ PAGE ON RECRUITER**

**When There Is No Job Posted By Recruiter**

**When There Are Jobs Posted By Rectuiters**

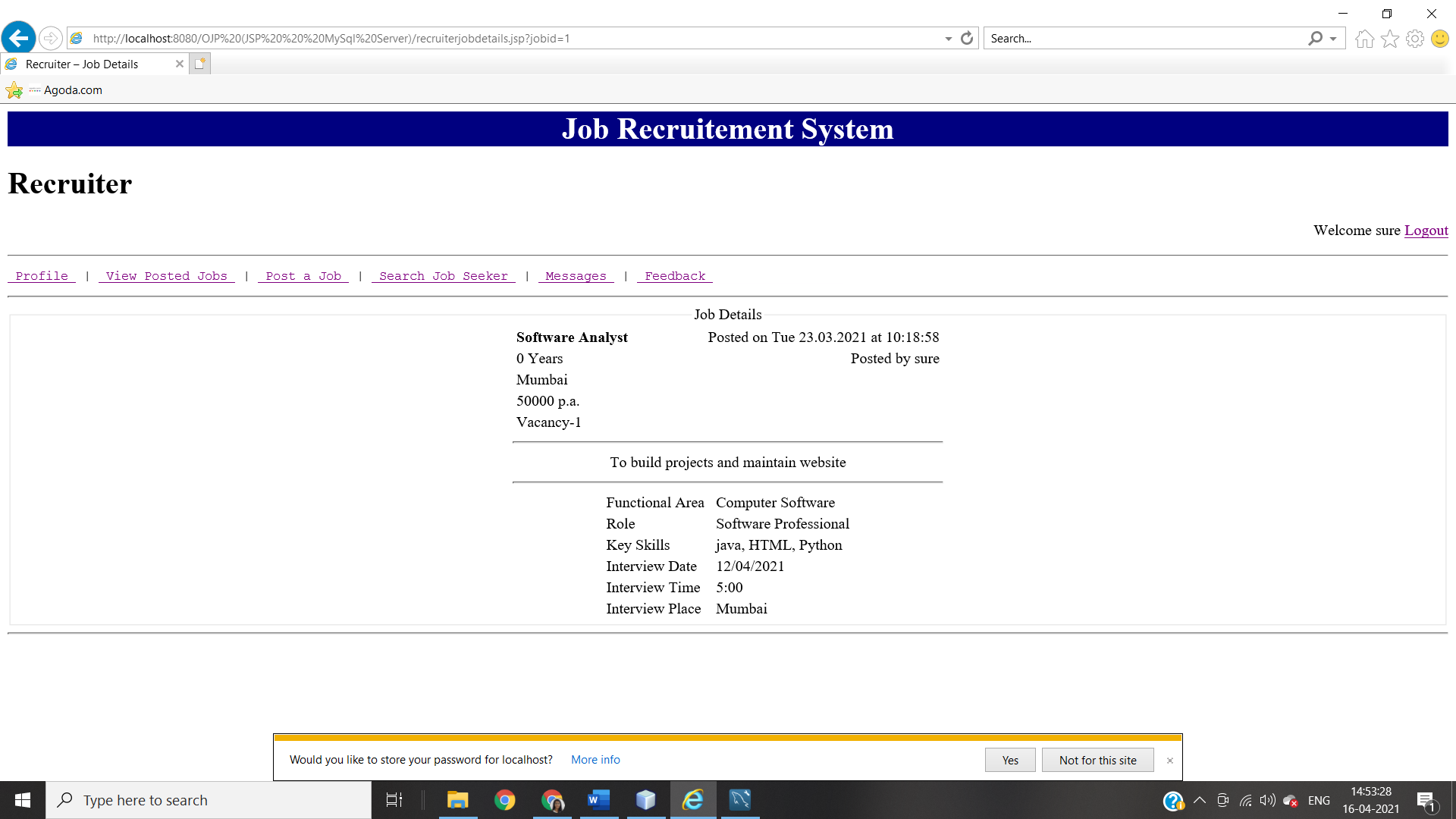
 Recruiter can open JobSeeker’s Profile Page by clicking on that JobSeeker’s name.

**‘POST A JOB’ PAGE ON RECRUITER**

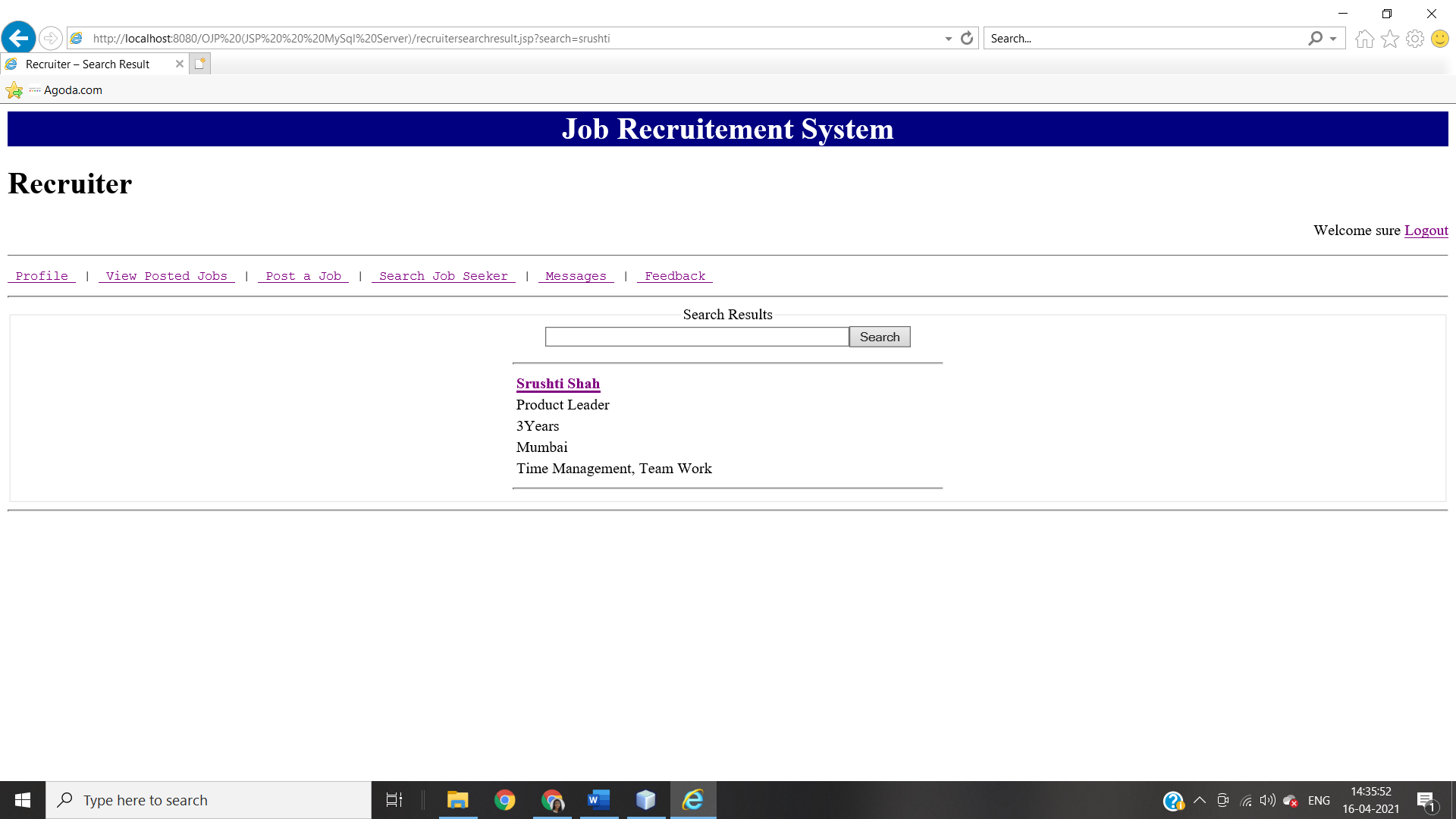


**JOB DETAILS PAGE ON RECRUITER**

**When Recruiter Views Jobs There Are No “Apply For Job” Option**

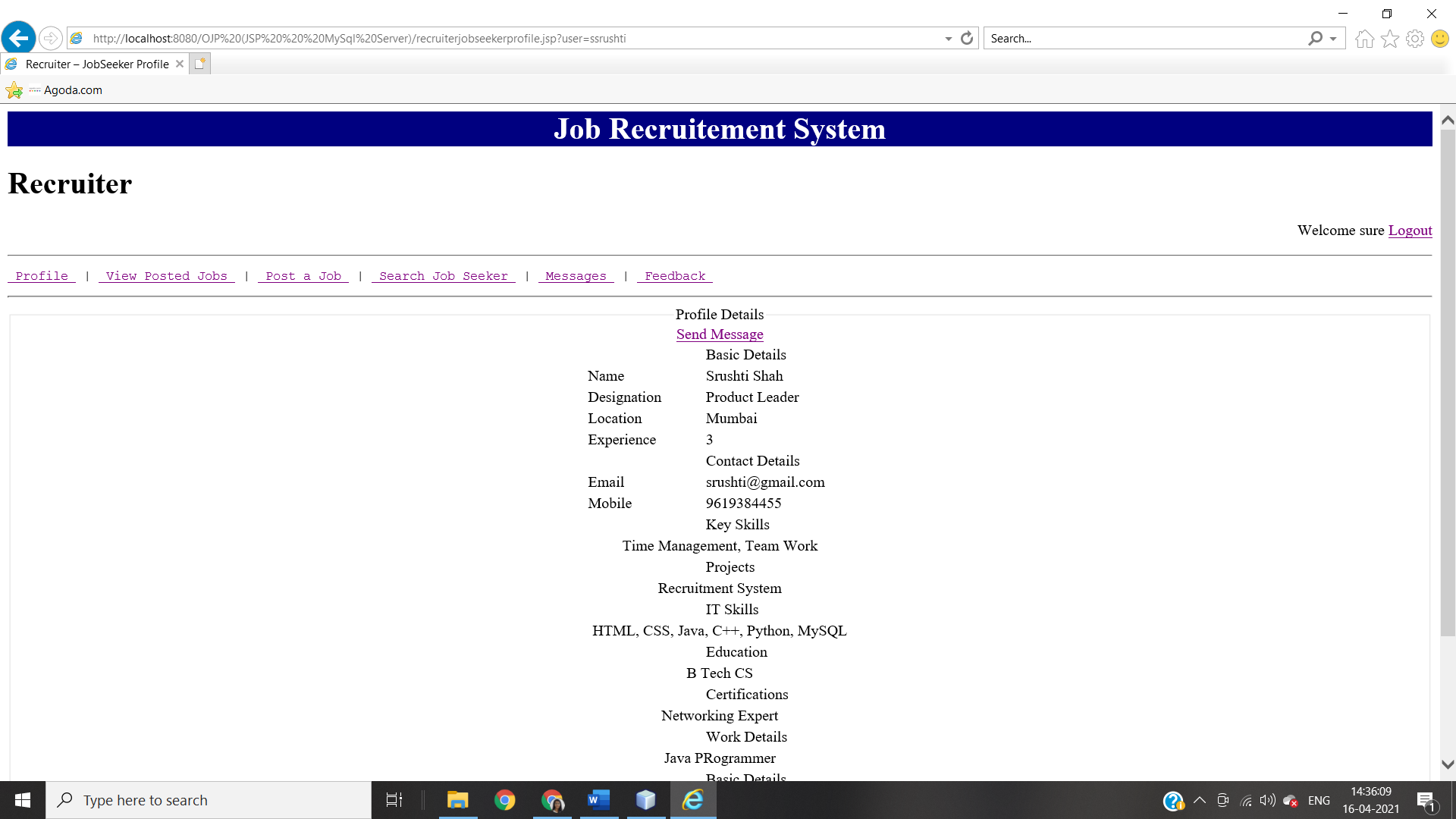


**‘SEARCH JOB SEEKER’ PAGE ON RECRUITER**

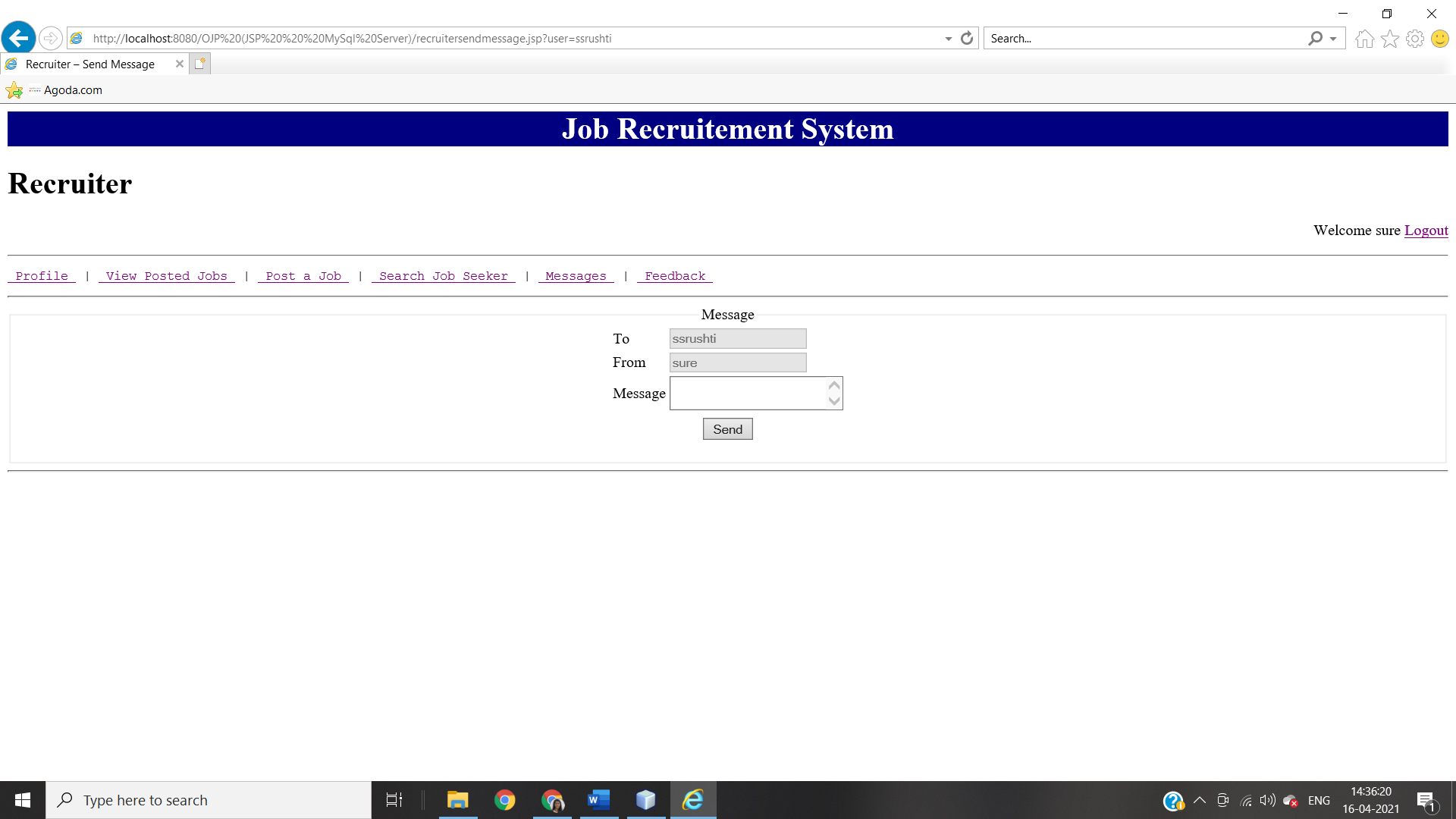


**JOBSEEKER PROFILE PAGE ON RECRUITER**

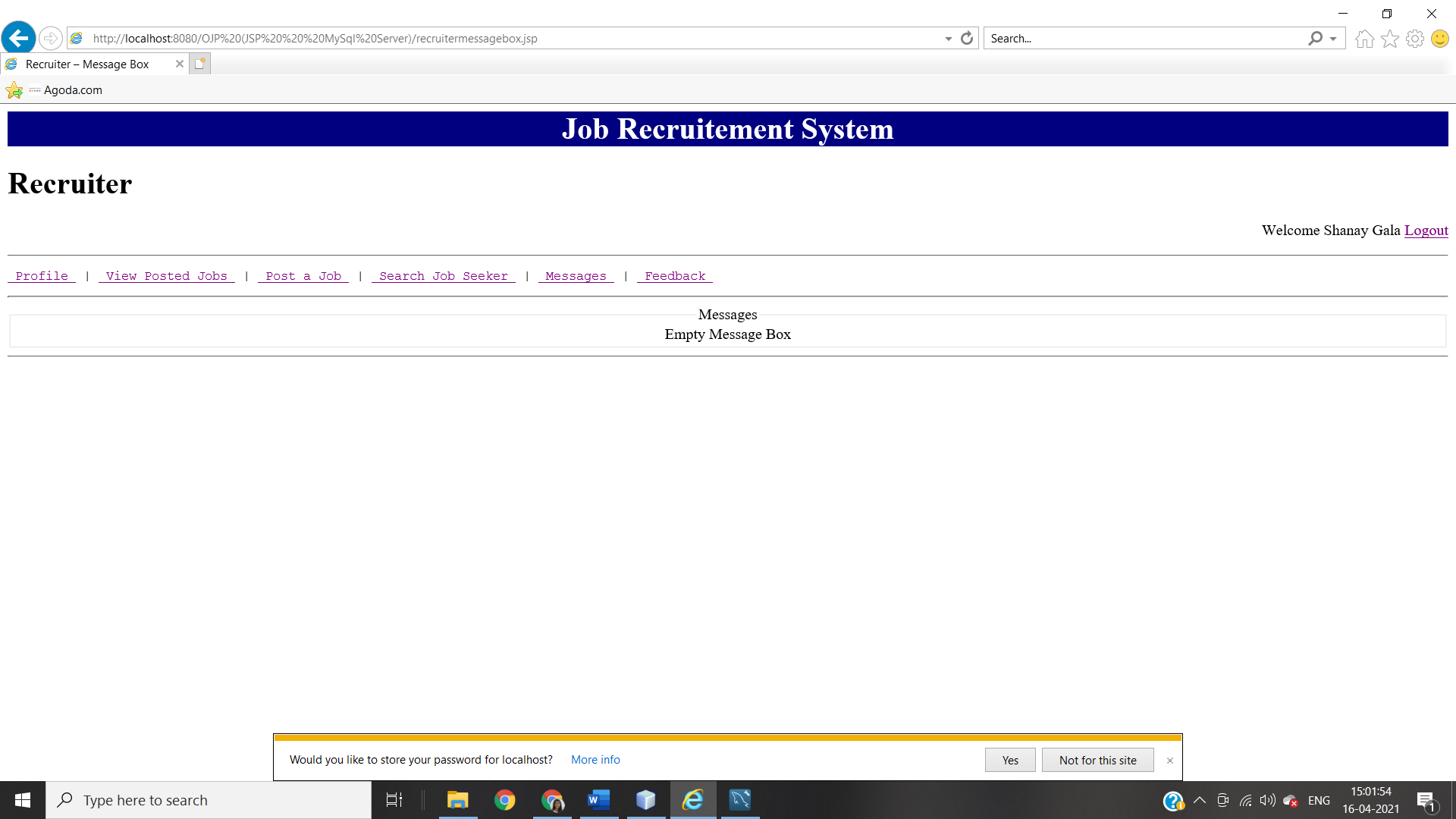
**When Recruiter Is Accessing JobSeeker Profile There Is ‘Send Message’ Option By Which Recruiter Can Send Message To That Perticular JobSeeker.**

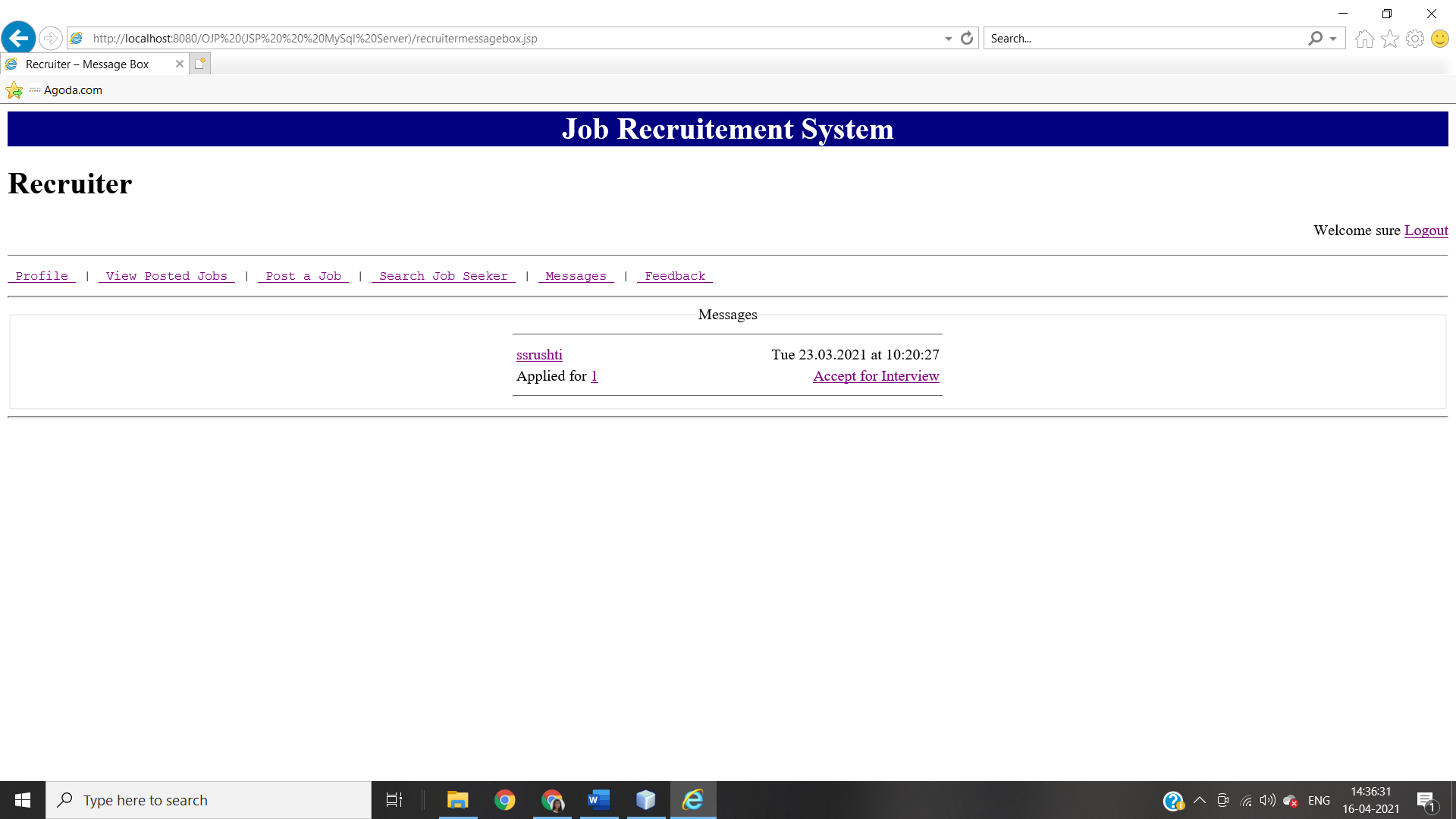


**‘Send Message’ Page**



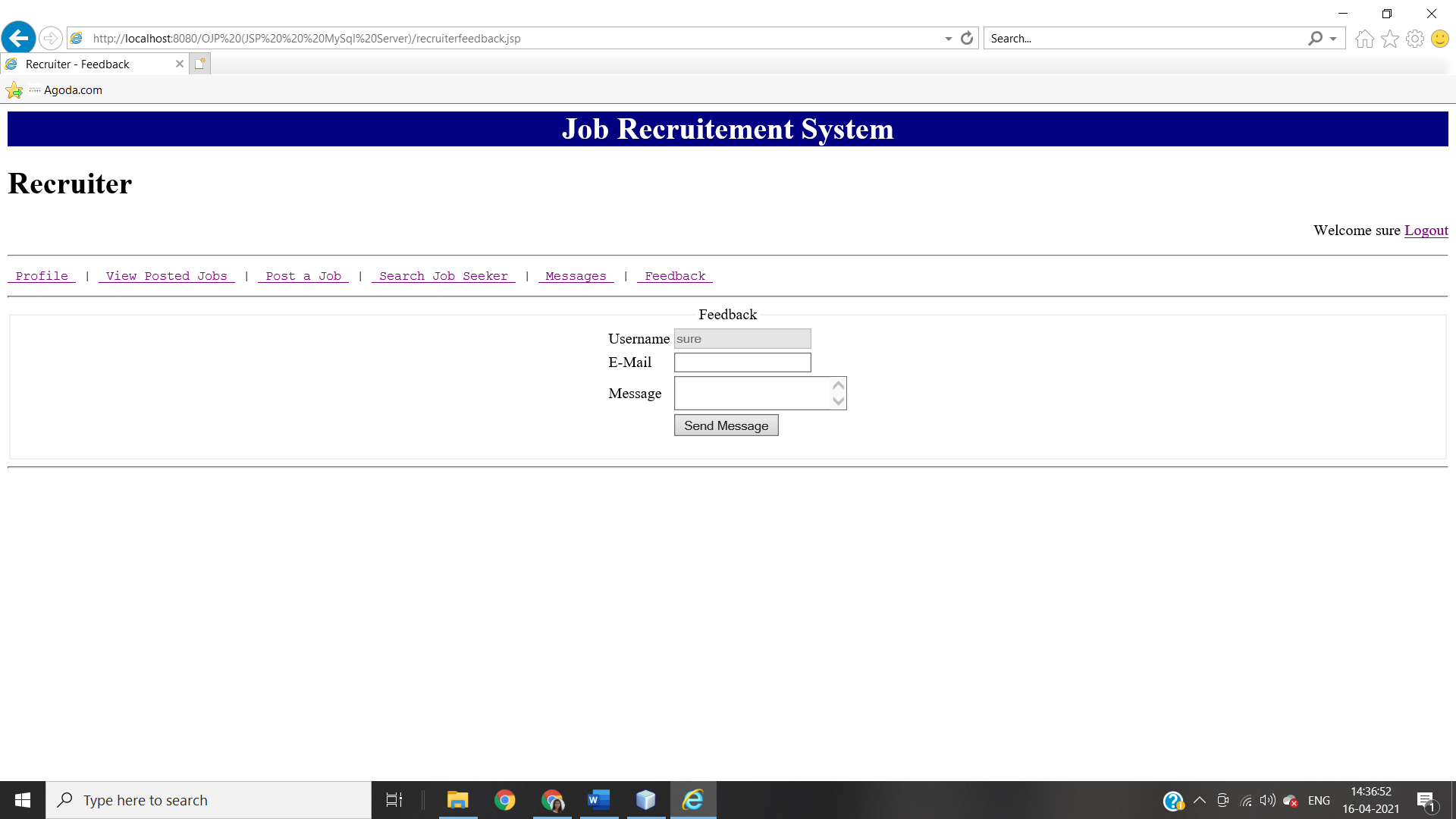
**‘MESSAGES’ PAGE ON RECRUITER**

**When There Is No Message For Recruiter**

**When There Are Messages**

**‘FEEDBACK’ PAGE ON RECRUITER**

**Recruiter’s Feedback Page**



1. **ADMIN HOME PAGE MODULE INTEGRATION TEST**

ADMINISTRATION

DELETION

LOGIN

REPORT

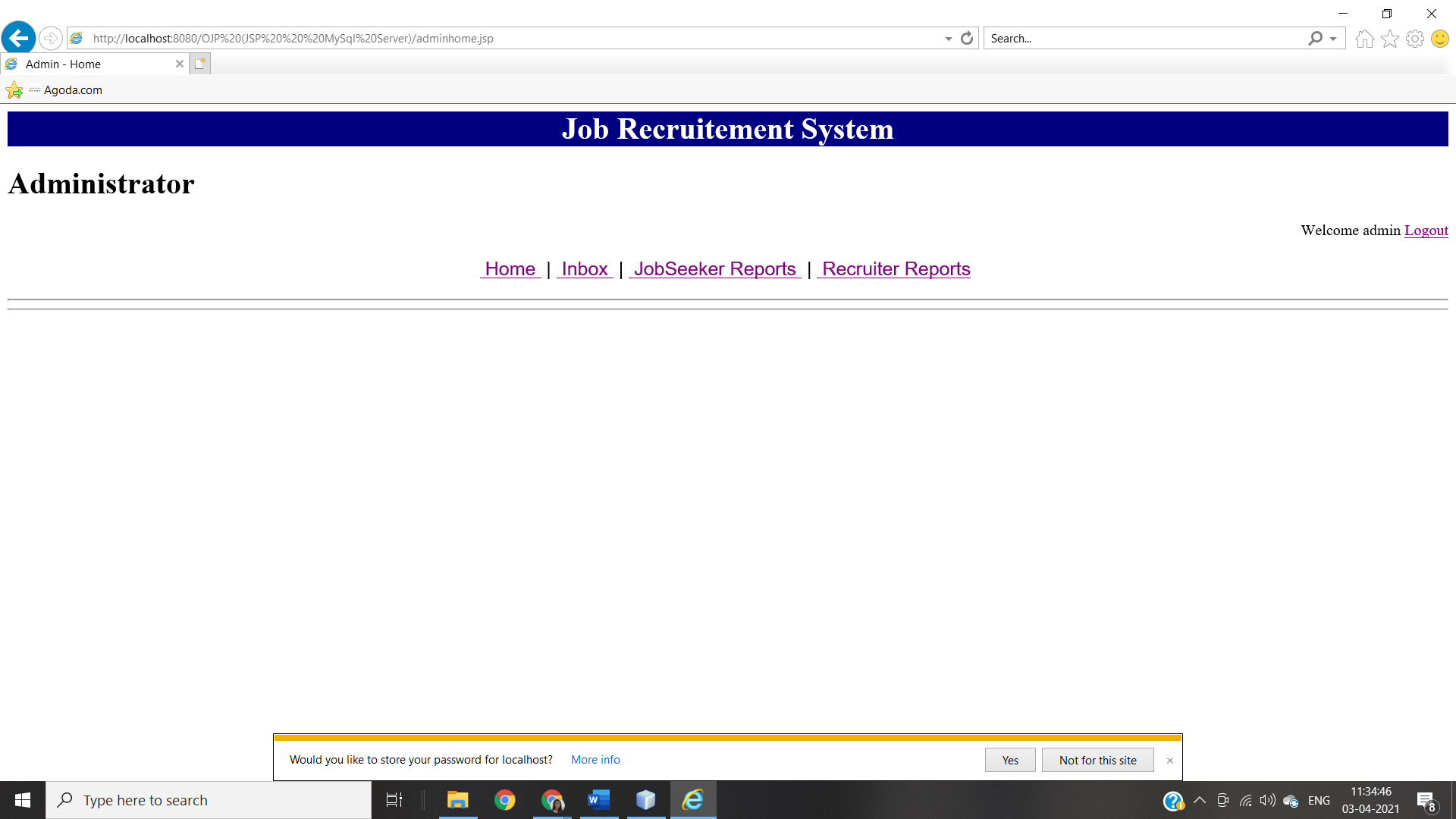
DELETION

LISTING

CREATION

RECRUITER

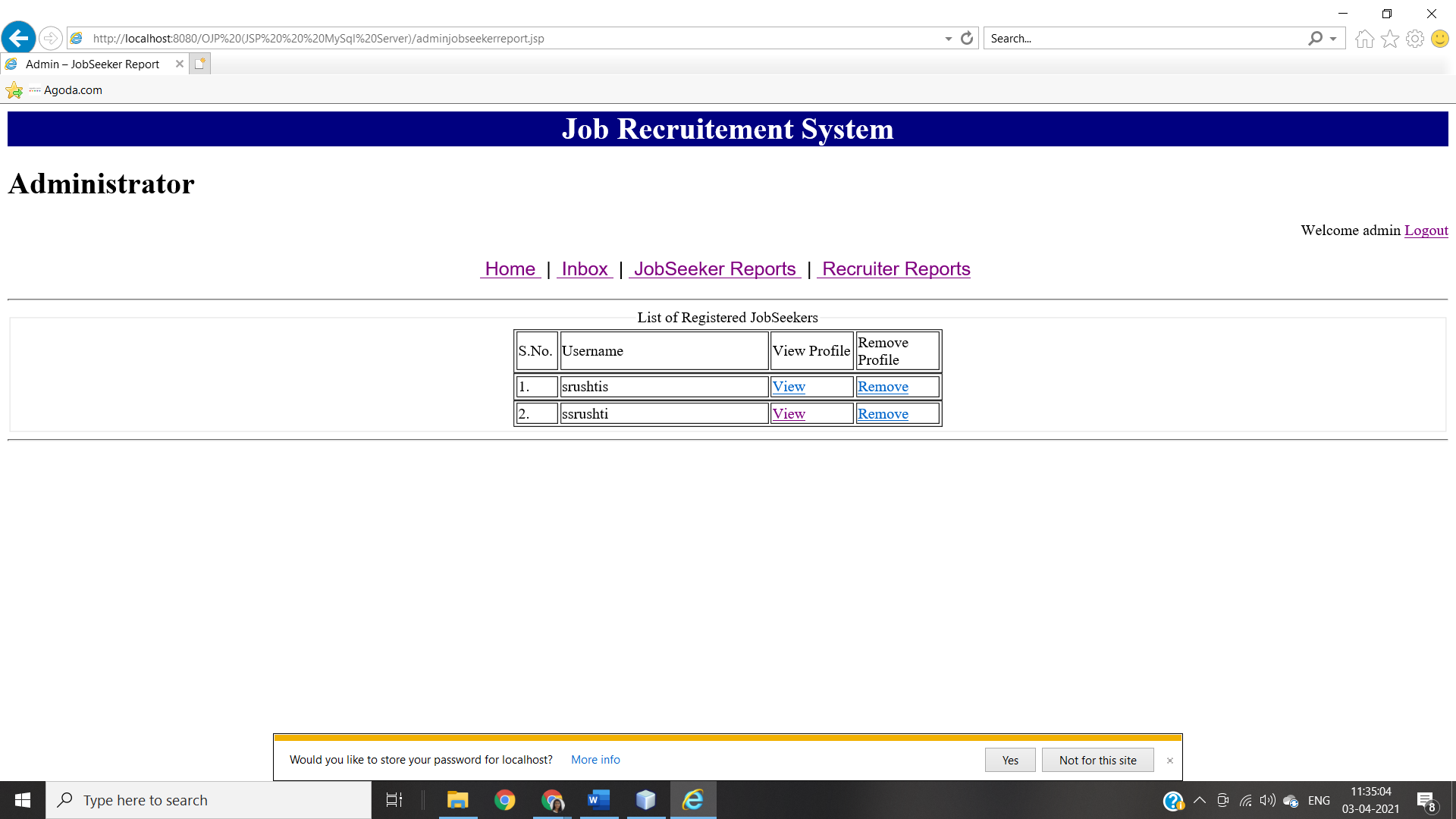
JOBSEEKER



All possible activities for the administrator are listed here.

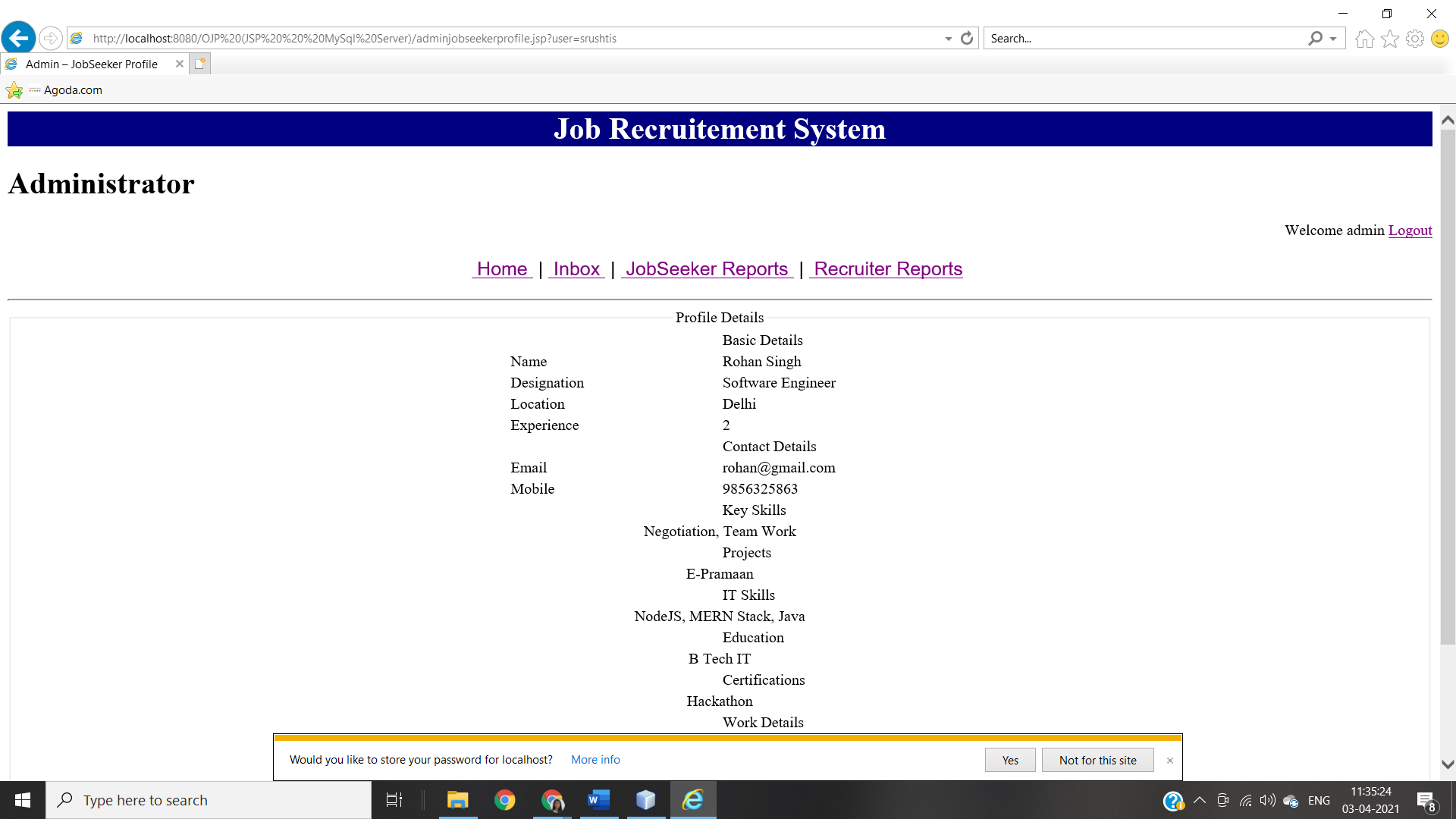
1. Home: - This link is for administrator homepage.
2. JobSeeker Reports.:- With the help of this link administrator can delete or view JobSeekers profile.
3. Recruiter Reports: From this link administrator can manage recruiters profile.

**‘JOBSEEKER REPORTS’ PAGE ON ADMINISTRATOR**

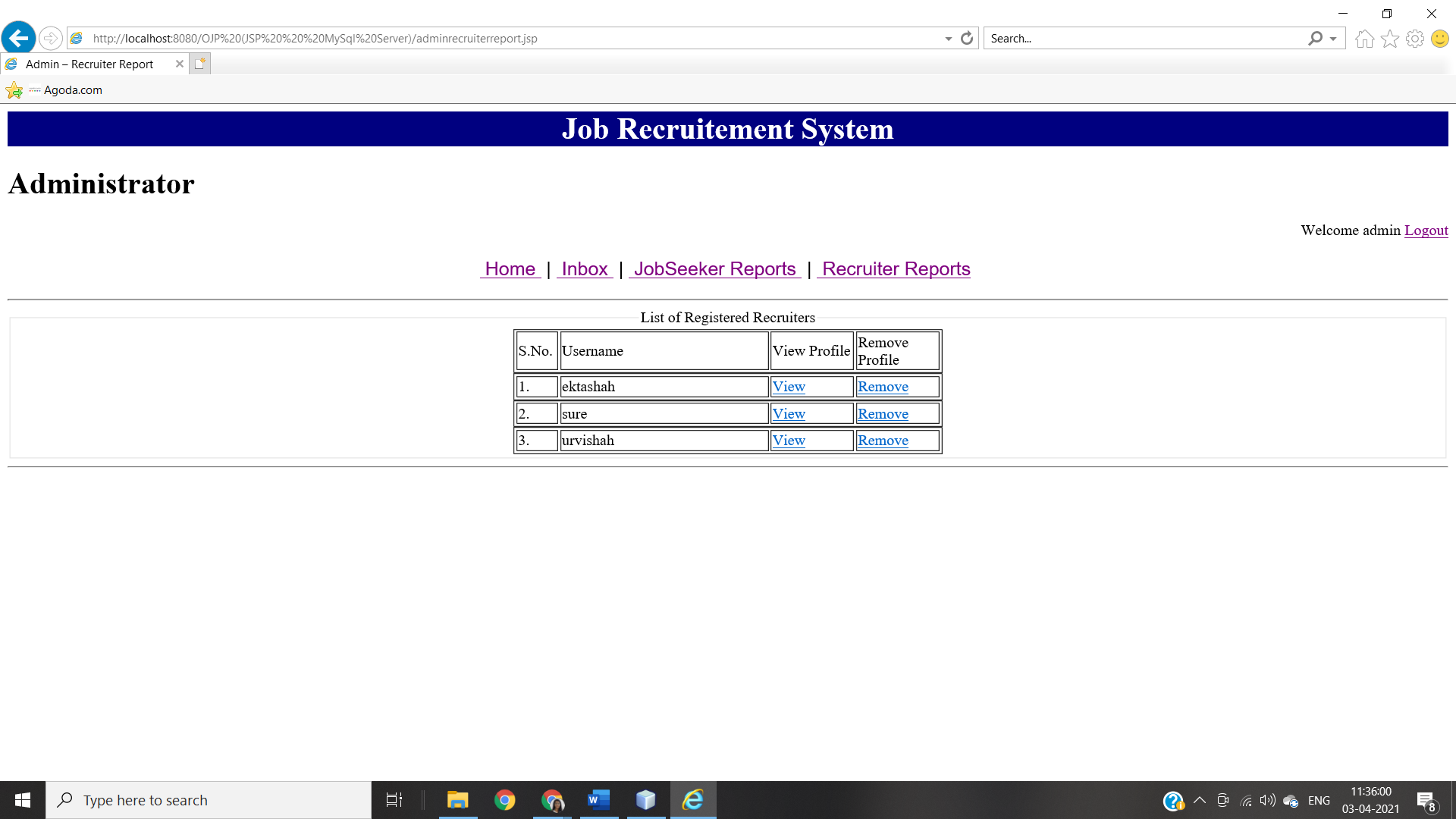


This page is for administrator. All possible activities for the administrator are listed here.

1. View: - This link will open correspondence jobseeker’s profile page.
2. Remove: - This link will remove correspondence jobseeker’s profile from database.



**‘RECRUITER REPORTS’ PAGE ON ADMINISTRATOR**



This page is for administrator. All possible activities for this page are listed here.

1. View: - This link will open correspondence recruiter’s profile page
2. Remove: - This link will remove correspondence recruiter’s profile from database.

**System Testing**

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

*Hardware:*

* The hardware used is Intel Code i3 and higher, on which the processing power is appropriate for the system and it works without ant delay
* Using Windows Server 2008, the database fetches details on time and correctly from the tables with minimum delay

*Software:.*

* Browser: The project works on all browsers efficiently without delay and all functionality are maintained
* Active internet connection: The project needs an active internet connection to run.

**Acceptance Testing**

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

***CONCLUSION:*** Thus, from this experiment I have implemented system testing using Selenium Web Driver for the project, along with unit and integration testing of the system.