**A Project Report on**

**“Jobugo: The Online Job Portal”**

**Submitted in partial fulfilment of requirement for the Award of degree**

**Bachelor of Computer Application**

**Of**

**Mangalore University**

****

**By**

**Chandan M Soorya Kumar M H Puneethraj K**

**186381929 186381997 186381963**

**Internal Guide**

**Mr. Prakash Kumar**

**Vivekananda Degree College**

**Puttur, D.K**

****

**VIVEKANANDA DEGREE COLLEGE PUTTUR, D.K**

**2020-2021**

**Phone: 08251230455**

**https://vcputtur.ac.in/**

**Declarations**

We here by declare that the project entitled "JOBUGO – The online job portal" has been proposed by us during the year 2020-2021 under guidance and supervision of our internal guide Mr. Prakash Kumar, HOD of computer science department, Vivekananda College, Puttur, Submitted to the Mangalore University in the partial fulfilment of the requirement of Bachelor Computer Application (BCA).

We also declare that this project is the result of our own efforts.

Place: Puttur

Date: By:

Chandan M

Soorya Kumar M H

Puneethraj K

**Acknowledgement**

We are grateful to all those, who made this project possible and helped me to complete within time.

During the course of project, we were supported by honorable principal **Prof. Vishnu Ganapathi Bhat**, Vivekananda College, Puttur to whom we should be grateful.

We extend our heart-full thanks to Mr. Prakash Kumar, head of the Computer Science Department, for his constant encouragement and support.

We take this opportunity to thank our internal guide Mr. Prakash Kumar for his valuable guidance and support.

This acknowledgement cannot be concluded without thanking our parents, friends and the God whose support had made us to accomplish tasks in hand.

By:

Chandan M

Soorya Kumar M H

Puneethraj K

**INDEX**

**1 Introduction-------------------------------------------------------------------------------1 – 4**

1.1 Project title

1.2.1 Introduction to the project

1.2.2 Objectives of the project

1.2.3 Scope of the project

1.3 Category

1.4 Overview

1.5 Structure of the system

1.6 System Architecture

1.7 End users

1.8 Software / Hardware used for the development

1.9 Software / Hardware required for the implementation

1.9.1 Hardware requirements

1.9.2 Software requirements

**2 Software Requirement Specification -------------------------------------------------5 – 8**

2.1 Introduction

2.2 Overall description

2.2.1 Product perspective

2.2.2 Product functions

2.2.3 User classes and Characteristics

2.2.4 General constraints

2.2.5 Assumptions and dependencies

2.3 Special requirements

2.3.1 External interface requirements

2.3.2 User interface

2.3.3 Hardware interface

2.3.4 Software interface

2.4 Functional requirements

2.4.1 Employer

2.4.2 Admin

2.4.3 Aspirant

2.5 Non – Functional requirements

2.6 Design constraints

2.6.1 Software language used

2.7 System attributes

**3 System design---------------------------------------------------------------------------------9 – 14**

3.1 Introduction

3.1.1 Purpose

3.2 Assumptions and constraints

3.2.1 Assumptions

3.2.2 Constraints

3.3 Functional decomposition

3.3.1 System software architecture

3.4 Description of the project

3.4.1 Context Flow diagrams

3.4.2 Data Flow diagram

3.4.3 Data flow diagram symbols

3.4.4 Constructing a DFD

3.5 Description of the components

3.5.1 Admin

3.5.2 Employer

3.5.3 Aspirants

**4 Database design-------------------------------------------------------------------------------15 – 22**

4.1 Introduction

4.2 Purpose and Scope

4.3 Database identification

4.4 Scheme information

4.5 Table definition

4.5.1 Table name: users

4.5.2 Table name: jobs

4.5.3 Table name: aspirants

4.5.4 Table name: feedback

4.6 Physical design

4.7 Data dictionary

4.7.1 Table name: users

4.7.2 Table name: jobs

4.7.3 Table name: aspirants

4.7.4 Table name: feedback

4.8 Entity Relationship diagram

4.8.1 Entity Relationship diagram symbols

4.9 Database Administration

4.9.1 System information

4.9.2 DBMS configuration

4.9.3 Support software requirements

4.9.4 Storage requirements

4.9.5 Backup and Recovery

**5 Detailed design-----------------------------------------------------------------------------23 – 42**

5.1 Introduction

5.1.1 Applicable documents

5.2 Structure of the software package

5.3 Modular decomposition components

5.3.1 Admin

5.3.1.1 Login

5.3.1.2 Add new admin

5.3.1.3 View all users and manage

5.3.1.4 Send response

5.3.1.5 View feedback

5.3.2 Aspirants

5.3.2.1 Login

5.3.2.2 View and apply

5.3.2.3 Send feedback

5.3.2.4 View response

5.3.2.5 Account

5.3.3 Employers

5.3.3.1 Login

5.3.3.2 View jobs posted by Employer

5.3.3.3 View aspirants

5.3.3.4 Send feedback

5.3.3.5 View response

5.3.3.6 Account

5.3.3.7 Add jobs

**6 Coding--------------------------------------------------------------------------43 – 70**

6.1Introduction

6.2 Database connection

6.3 Authorization / Authentication

6.4 Functions

6.5 Signup

6.5.1 Signup

6.5.2 Email verification

6.6 Login

6.7 Job listing

6.8 Job details

6.9 Add jobs (Employer)

6.10 View jobs (Employer)

6.11 View aspirants (Employer)

6.12 View aspirants (Admin)

6.13 View employers (Admin)

6.14 View jobs (Admin)

6.15 Feedback reply (Admin)

6.16 View new aspirant (Admin)

6.17 View new job (Admin)

6.18 Document viewer

6.19 Main Activity.java

6.20 Activity main.html

6.21 Android Manifest.xml

6.22 Testing

6.22.1 Example Instrument Test.java (JUNIT)

6.23 Build configuration

**7 User Interface------------------------------------------------------------------------------71 – 82**

7.1 Main page

7.2 User does not have account

7.3 About us

7.4 Team

7.5 Login page

7.6 Signup

7.6.1 Account type select

7.6.2 Account credentials

7.6.3 OTP verification

7.6.4 Account added

7.7 Job listing

7.8 Job description

7.9 Applied successfully

7.10 Account

7.11 Employer panel to add jobs

7.12 Employer panel to view jobs

7.13 View aspirants

7.14 Admin panel

7.15 Aspirant accounts (Admin panel)

7.16 Employer accounts (Admin panel)

7.17 Employer verification document viewer

7.18 Job list for admin

7.19 Add new admin

7.20 Mailer (Admin panel)

7.21 New jobs to notify

7.22 New aspirants to notify

7.23 New feedback to notify

7.24 Document viewer to view CV/Resume (Employer)

7.25 View admin accounts (Admin)

7.26 Reset password

7.26.1 Account verification

7.26.2 Email verification through OTP

7.26.3 Reset password

7.26.4 Password reset successful

7.27 Android app background image

7.28 Android app icon

**8 Testing---------------------------------------------------------------------------------------83 – 89**

8.1 Introduction

8.1.1 Purpose

8.1.2 Testing objectives

8.2 Testing types

8.2.1 Black box testing

8.2.2 White box testing

8.3 Levels of testing

8.3.1 Unit testing

8.3.2 Integrate testing

8.3.3 System testing

8.3.4 User acceptance testing

8.4 Test cases

8.4.1 Login form

8.4.2 Job details

8.4.3 Add jobs

8.4.4 Your jobs (Employer)

8.4.5 View aspirants (Employer)

8.4.6 Aspirant accounts (Admin)

8.4.7 Employer accounts (Admin)

8.4.8 View jobs (Admin)

8.4.9 View admin (Admin)

8.4.10 Add new admin

8.4.11 Reply to feedback

8.4.12 Forgot password (Account select)

8.4.13 Forgot password (Mailer)

8.4.14 Forgot password (Reset)

8.4.15 Logout

**9 Conclusion---------------------------------------------------------------------------------------90**

**10 Limitations-------------------------------------------------------------------------------------91**

**11 Future scope and Enhancements----------------------------------------------------------92**

**12 Abbreviations and Acronyms--------------------------------------------------------------93**

**13 Bibliography and References--------------------------------------------------------------94**

**List of Figures**

Figure 1: System Architecture-------------------------------------------------------------------------03

Figure 2: System Software Architecture-------------------------------------------------------------10

Figure 3: Context flow diagrams----------------------------------------------------------------------10

Figure 4: Data flow diagrams (Level 1) -------------------------------------------------------------12

Figure 5: Data flow diagrams (Level 2) -------------------------------------------------------------13

Figure 6: Data flow diagrams (Level 2 - Admin) ---------------------------------------------------13

Figure 7: Physical design-------------------------------------------------------------------------------17

Figure 8: Entity Relationship diagram----------------------------------------------------------------21

Figure 9: Admin module--------------------------------------------------------------------------------24

Figure 10: Aspirants module---------------------------------------------------------------------------24

Figure 11: Employer module---------------------------------------------------------------------------25

Figure 12: Admin login---------------------------------------------------------------------------------26

Figure 13: Add new admin-----------------------------------------------------------------------------27

Figure 14: View all users and manage----------------------------------------------------------------28

Figure 15: Send response-------------------------------------------------------------------------------29

Figure 16: View feedback------------------------------------------------------------------------------30

Figure 17: Aspirants login------------------------------------------------------------------------------31

Figure 18: View and apply-----------------------------------------------------------------------------32

Figure 19: Send feedback-------------------------------------------------------------------------------33

Figure 20: View response------------------------------------------------------------------------------34

Figure 21: Account--------------------------------------------------------------------------------------35

Figure 22: Employers login----------------------------------------------------------------------------36

Figure 23: View jobs posted by employer------------------------------------------------------------37

Figure 24: View aspirants------------------------------------------------------------------------------38

Figure 25: Send feedback------------------------------------------------------------------------------39

Figure 26: View response------------------------------------------------------------------------------40

Figure 27: Account--------------------------------------------------------------------------------------41

Figure 28: Add jobs-------------------------------------------------------------------------------------42

Figure 29: Main page 1---------------------------------------------------------------------------------71

Figure 30: Main page 2---------------------------------------------------------------------------------71

Figure 31: User does not have account---------------------------------------------------------------71

Figure 32: About us-------------------------------------------------------------------------------------72

Figure 33: Team-----------------------------------------------------------------------------------------72

Figure 34: Login page----------------------------------------------------------------------------------72

Figure 35: Sign up – Account type selection--------------------------------------------------------73

Figure 36: Account credentials------------------------------------------------------------------------73

Figure 37: OTP verification----------------------------------------------------------------------------73

Figure 38: Account added------------------------------------------------------------------------------74

Figure 39: Job listing------------------------------------------------------------------------------------74

Figure 40: Job description------------------------------------------------------------------------------74

Figure 41: Applied successfully-----------------------------------------------------------------------75

Figure 42: Account--------------------------------------------------------------------------------------75

Figure 43: Employer panel to add jobs---------------------------------------------------------------75

Figure 44: Employer panel to view jobs--------------------------------------------------------------76

Figure 45: View aspirants------------------------------------------------------------------------------76

Figure 46: Admin panel---------------------------------------------------------------------------------77

Figure 47: Aspirant account (Admin panel) ---------------------------------------------------------77

Figure 48: Employer accounts (Admin panel) ------------------------------------------------------77

Figure 49: Employer verification document viewer------------------------------------------------78

Figure 50: Job list for admin---------------------------------------------------------------------------78

Figure 51: Add new admin-----------------------------------------------------------------------------78

Figure 52: Mailer (Admin account) ------------------------------------------------------------------79

Figure 53: New jobs to notify--------------------------------------------------------------------------79

Figure 54: New aspirants to notify--------------------------------------------------------------------79

Figure 55: New feedback to notify--------------------------------------------------------------------80

Figure 56: Document viewer to view CV/Resume (Employer) ----------------------------------80

Figure 57: View admin accounts (Admin) ----------------------------------------------------------80

Figure 58: Reset password – Account verification--------------------------------------------------81

Figure 59: Email verification through OTP----------------------------------------------------------81

Figure 60: Reset password-----------------------------------------------------------------------------81

Figure 61: Password reset successful-----------------------------------------------------------------82

Figure 62: Android app background image----------------------------------------------------------82

Figure 63: Android app icon---------------------------------------------------------------------------82

**List of Tables**

Table 1: Data Flow diagram symbols-----------------------------------------------------------------11

Table 2: Table definition – users----------------------------------------------------------------------16

Table 3: Table definition – jobs-----------------------------------------------------------------------16

Table 4: Table definition – aspirants------------------------------------------------------------------16

Table 5: Table definition – feedback------------------------------------------------------------------16

Table 6: Data dictionary – users-----------------------------------------------------------------------18

Table 7: Data dictionary – jobs------------------------------------------------------------------------19

Table 8: Data dictionary – aspirants------------------------------------------------------------------19

Table 9: Data dictionary – feedback------------------------------------------------------------------20

Table 10: Entity Relationship diagram---------------------------------------------------------------21

Table 11: Test cases – Login form--------------------------------------------------------------------85

Table 12: Test cases – Job details---------------------------------------------------------------------86

Table 13: Test cases – Add jobs-----------------------------------------------------------------------87

Table 14: Test cases – Your jobs (Employer) -------------------------------------------------------87

Table 15: Test cases – View aspirants (Employer) -------------------------------------------------87

Table 16: Test cases – Aspirant accounts (Admin) -------------------------------------------------87

Table 17: Test cases – Employer accounts (Admin) -----------------------------------------------87

Table 18: Test cases – View jobs (Admin) ----------------------------------------------------------87

Table 19: Test cases – View admin (Admin) -------------------------------------------------------88

Table 20: Test cases – Add new admin---------------------------------------------------------------88

Table 21: Test cases – Reply to feedback------------------------------------------------------------88

Table 22: Test cases – Forgot password (Account select) -----------------------------------------88

Table 23: Test cases – Forgot password (Mailer) --------------------------------------------------89

Table 24: Test cases – Forgot password (Reset) ----------------------------------------------------89

Table 25: Test cases – Logout -------------------------------------------------------------------------89

**CHAPTER – 1**

**INTRODUCTION**

**1.1 Project title**

Jobugo: The Online Job Portal

**1.2.1 Introduction to the project**

Now a day’s everything is becoming online. Peoples are investing their valuable time in online platforms, from buying a pen to consulting doctors. Everything is online, because of this pandemic most of the peoples are struggling to find a suitable job.

So, our project will solve their problems, we provide a web application where aspirants can register themselves for free and apply their resume or curricular vitae for various jobs and employers accepts those CV and resume. This job portal system project is beneficial to all those.

Those aspirants who successfully registered with our website will get all the description about the job including all the necessary information’s like skills required, location, salary etc.

**1.2.2 Objectives of the project**

* conform and rejection of employer
* provided to submit CV and resume.
* online job portal project in php help job seekers.
* job portal project report.

**1.2.3 Scope of the project**

In this application the users can view many job profiles. Users who get registered to this web site can get information about all the sector wise jobs. The users can get the information about the job at a single place. Thus, prevents the users to visit individual website of each company. Here the user can check the number of subscribers who already referring our website. User can also give a feed about the website

Admin has access in all the part of the website. He has privilege to manage aspirants and Employees.

**1.3 Category**

Web based Relational Database Management System. (Web Application)

**1.4 Overview**

Our project provides a user-friendly platform in which the aspirant’s and employers can easily register themselves. The aspirants can find various jobs based on their interests.

There they can view multiple jobs with descriptions and apply CV or resume for free.

**1.5 Structure of the system**

* **Admin:** Admin will be in managing the website. he can perform the following activities: -

- Manage users

- Manage Employers

- Manage Accounts

- Manage Profile

* **Aspirant (USER):** User can login to this website by providing his Email ID and Password. He can perform following activities: -
* View Openings by companies
* Sort jobs based on categories
* Apply for job
* Fill details
* Upload CV
* Feedback
* Manage Profile and CV
* **Employers:** Employers can login to this website by providing his Email ID and Password. He can perform following activities: -
* Post vacancy details
* Update Details
* View Applicants
* Update Account Details
* Manage Profile
* Create Account / Login:
* Fill Account details
* Set Password and Username to your Account

**1.6 System Architecture**

Architectural design is the process of decomposing a large complex system into small subsystem. These sub systems are meant for providing some related services. The architectural design is basically a layout framework for the system to control and communicate with the subsystem.



* **Model :** Model represents the information of the application and rules to manipulate the data. The bulk of an application’s business logic will be concentrated in the model.
* **View :** The view is responsible for the generating a user interface normally based on data in the model
* **Controller:** Controller organizes the application. Controllers receive events from outside world, interact with the model and display the appropriate view to user.

**1.7 End users**

* Admin
* Aspirants
* Employers

**1.8 Software / Hardware used for the development**

* User Interface Design : HTML5, CSS3, Java script, Bootstrap
* Server : Apache
* Backend : MySQL, PHP (scripting)
* Tools : Sublime Text, XAMPP, EDraw Max9.4

**1.9 Software / Hardware required for the implementation**

**1.9.1 Hardware Requirements:**

* Processor : Intel Dual Core or any higher version.
* Processor speed : Minimum 1.83 GHz
* RAM : 256 MB DDR RAM
* Hard Disk : 40 GB or above

**1.9.2 Software Requirements:**

* Language : PHP 7.4.0
* Web components : HTML 5, CSS3
* Web Server : Apache
* Data base (backend) : MySQL

**CHAPTER – 2**

**SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 Introduction:**

A software Requirement Specification (SRS) – it is a complete description of the behaviour of a system to be developed and it includes a set of user cases that describe all the interactions that a user will have with the software and most of the requirements of the user.

The main advantages of SRS are:

* An SRS establishes the basis for agreement between the user and the provider on what the software product will do.
* An SRS provides a reference for validation of the final product.
* It acts like a main reference for System Design.

**2.2 Overall Description**

**2.2.1 Product perspective:**

The importance of placement system is increasing day by day. Thousands of applicants depend on placement cell. But the applicants are facing so many problems. This project is an attempt to minimize the problems of an applicant to find a correct job. If we can make it as a digital process. It will be so fast and convenient to the aspirants.

This software doesn’t require a dedicated computer and can be run on almost any pc running on windows OS.

* **Administrator:**

Administrators will get a notification whenever a new issue is raised by the end user through the web application. The issue from both the sides, will be solved as soon as possible. Administrator can then change the status of the issue so that the end users can get a notification saying that the work on the particular issue is in progress.

* **Aspirants**:

Aspirants can contact us at any time we will try to fix the problem shortly. He can search all the available jobs based on his interest.

* **Employers**:

They can easily post their job vacancies and get all the shared information about the aspirants.

**2.2.2 Product functions:**

The product has following functions:

* Create user.
* Update user details.
* Add employers.
* Sending mail.
* User verification.

**2.2.3 User classes and Characteristics:**

The end users use this application to post the job and then apply for the job, the end users will receive a notification via email. The end user can also track the status of the issue that is being resolved.

**2.2.4 General constraints:**

* The user has to give proper information about e-mail because the job messages are directly sent to Registered mail-id.
* The Employer should have a registered company certificate.
* Each section must be under the authorization of admin.

**2.2.5 Assumptions and dependencies:**

* Each user must have an email id and password.
* Each Admin must have a Username and password.
* Each company must have registered company name and password.
* Everything is maintained by admin.

**2.3 Special Requirements:**

**2.3.1 External Interface Requirements:**

All the interactions of the software with people, hardware and other software are specified here. The “Jobugo” web application is simple and easy to use, with easier UI.

**2.3.2 User Interface:**

* The system provides a user-friendly GUI to the users.
* Appropriate error messages are generated when a user performs an operation which is invalid.

**2.3.3 Hardware interface:**

* Operating system: Windows XP or above
* Hard Disk Utilization: 20 GB and above
* Speed: 1.60 GHz
* RAM: 2 GB and above
* Processor: Pentium® dual core or above
* Input Devices: Mouse, Keyboard
* Output Devices: Monitor, Printer

**2.3.4 Software interface:**

* Browser: Internet Explorer, Google chrome, Mozilla Firefox.
* Language: PHP 5.4
* Web components: HTML5, CSS
* Scripting Language: JAVA SCRIPT, PHP
* Operating system: Any Operating System Database with MySQL (Back End)
* Application server: XAMPP Server

**2.4 Functional requirements:**

**2.4.1 Employer:**

* Post jobs with detailed job description (job title, company name, company URL, location, salary, job type, work type, description, required skills...)
* Ability to create and edit a company profile.
* Employer registration.
* Information on number of vacancy viewings and applications.
* View Submitted job seeker CV

**2.4.2 Admin:**

* Manages job seekers and Employers all the information’s.
* Ability to change the look and feel of the site.
* Admin can Add / Edit / Delete / Approve / Disapprove any job.
* Can send mails to both aspirants and employers.

**2.4.3 Aspirant:**

* Apply for a job within a click of button.
* Resume/CV posting.
* Job Seeker Registration.
* Login and update profile.

**2.5 Non-Functional Requirements:**

* Secure access of confidential data (user’s details). SSL can be used.
* Better component design to get better performance at peak time.
* Flexible service-based architecture will be highly desirable for future extension.
* In case the user forgets Password, the repair functionality helps by choosing “forgot password” option in the main login window.

**2.6 Design Constraints:**

**2.6.1 Software Language Used**

Our project has three categories of users: admin and aspirants, employers and it runs in a web application environment. The system will be developed using PHP and MySQL database.

**2.7 System Attributes:**

The Quality of the website is maintained in such a way so that it can be very user friendly to all the users of the website.

* **Reliability:** Good validation of user inputs will be done to avoid entering incorrect username and password.
* **Maintainability:** During maintenance stage, the SRS can be referred for the validation.
* **Portability:** This system can be run in any operating system and browser.
* **Compatibility:** This System will be Compatible with almost all the web servers.
* **Flexibility:** The system keeps on updating the data according to the changes that takes place.
* **Timeliness:** The system carries out all the operations with consumptions of very less time.
* **Security:** Each time there is a security violation. System restricts the user from accessing that function.

**CHAPTER – 3**

**SYSTEM DESIGN**

**3.1 Introduction:**

System Design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

The resulrs of the system design process are recorded in the System Design Document(SDD). This document completely describes the system at the architecture level, including subsystems and their services, hardware mapping, data management, access control, global software control structure, and boundary conditions. This Document is similar to the blueprint or a plan for the solution and is used later during implement, testing and maintenance.

System Design aims to identify and modules that should be in the system, the specification of these modules, and how they interact with each other to produce the desired result. At the end of the System Design all the major data structures, file formats, and the major modules in the system specifications are decided.

**3.1.1 Purpose**

The purpose of the software design document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to build. The software design document provides information necessary to provide the description of the details for the software and the system to be built.

**3.2 Assumptions and constraints**

**3.2.1 Assumptions:**

The system should able to upload the files in the server. This web portal is user friendly and less cost. This is aimed to provide satisfactory result to the user by providing the required information. Administrator is created in the system manually. All the classes of user need to enter their login id and password for authentication during login.

**3.2.2 Constraints:**

* This system is provisioned to be built on the PHP.
* More Space is required to keep all the records
* The tables of the database are designed as normalized table.
* Database is not shared.
* The constraints of this project are the system must support the latest web browser and must be able to use all the forms.
* Login and Password is used for identification of user, admin and employer.

**3.3 Functional Decomposition**

**3.3.1 System Software Architecture:**

Architectural design is the process of decomposing a large complex system into small subsystem. These sub systems are meant for providing some related services. The architectural design is basically a layout framework for the system to control and communicate with the subsystem.



**3.4 Description of Programs**

**3.4.1 Context Flow Diagrams (CFD):**

The context diagram is used to establish the context and boundaries of the system to be modeled: which things are inside and outside of the system being modeled, and what is the relationship of the system with these external entities. A Context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software systems. It identifies the flows of information between the system and internal entities. The entire software system is shown as a single process.

EMPLOYER

Request

ASPIRANT

Response

ADMIN

**3.4.2 Data Flow Diagram:**

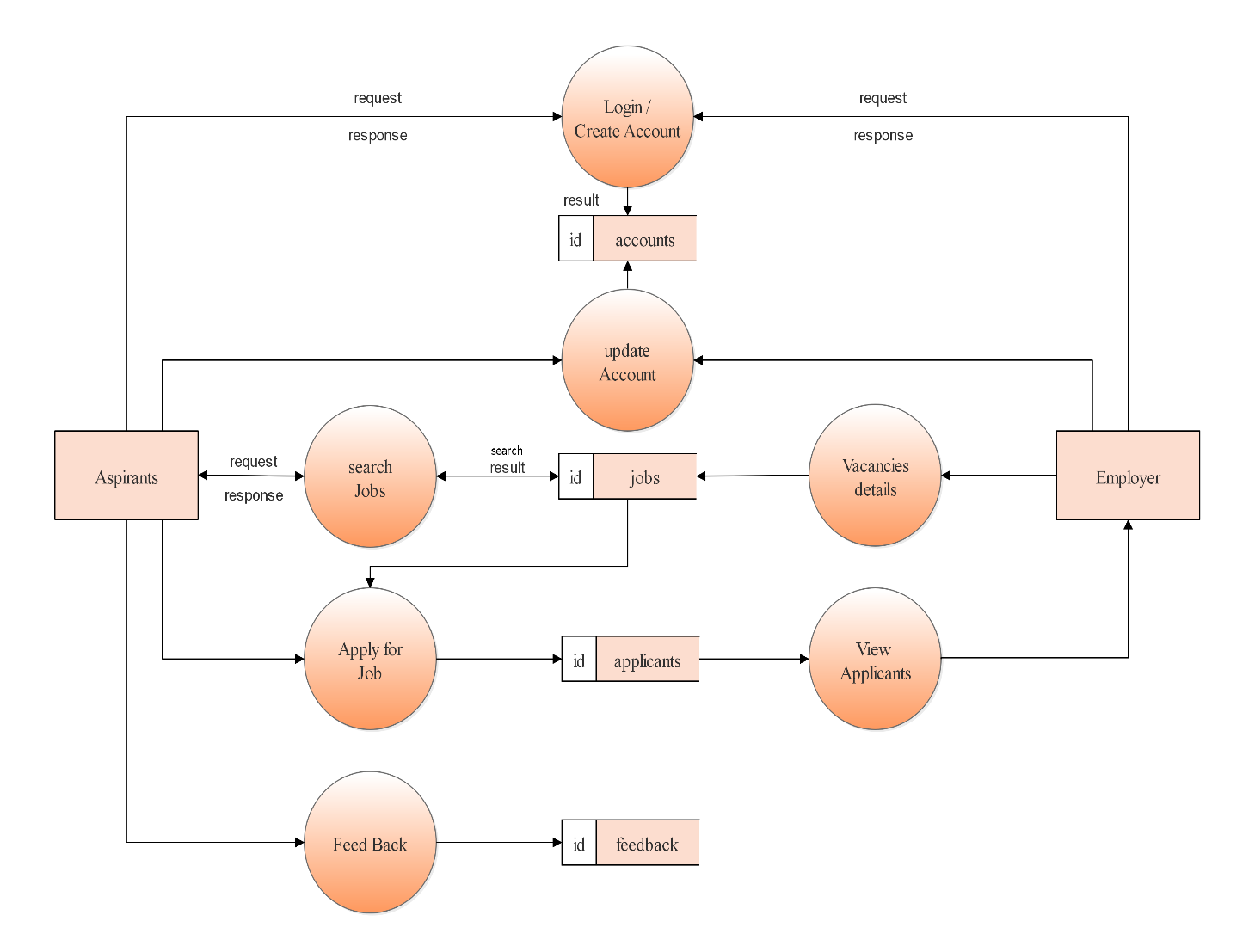
Data Flow Diagram is a graphical representation of a system or a portion of the system. It consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols. DFD is one of the most important modeling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system. DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output.

DFD is also known as bubble chart or Data Flow Graphs. DFD may be used to represents the system at any level of abstraction. DFD’s may partition into a level that represents increasing information flows and functional details.

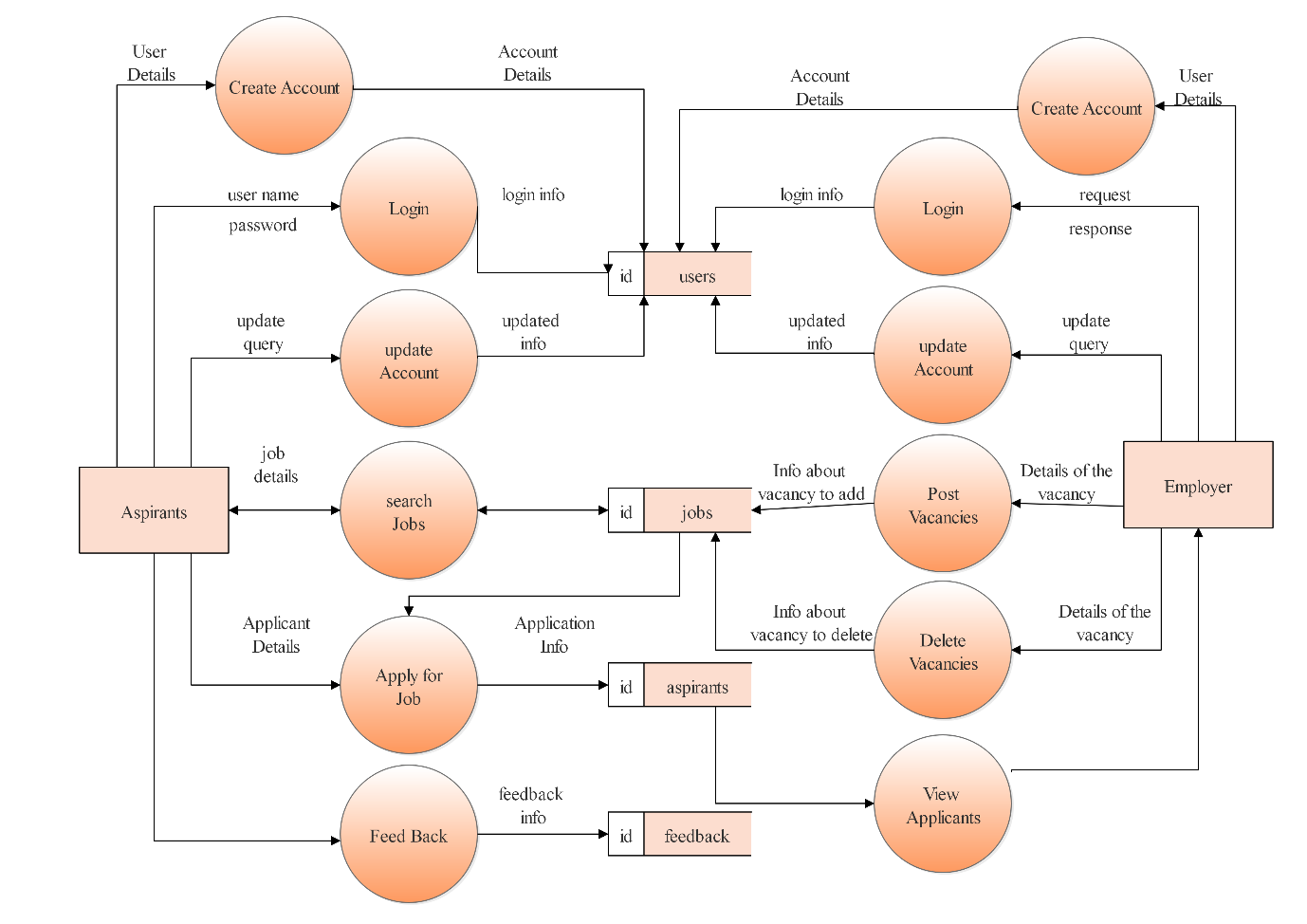
**3.4.3 DFD Symbols:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Notation** | **Description** |
| Data Store |  | The data store represents a logical file. A logical file can represent either a store symbol which represents either a data structure or a physical file on disk. |
| Data Flow |  | The data flow is used to describe the moment of information from one part of system to another part. Flow represents data in motion. It is a pipeline through which information flows. |
| Process |  | A circle or bubble represents a process that transforms incoming data to outgoing data. Process shows a part of the system that transforms inputs to outputs. |
| External Entities |  | A square defines a source or destination of system data. External entities represent any entity that supplies or receive information from the system but it is not a part of the system. |

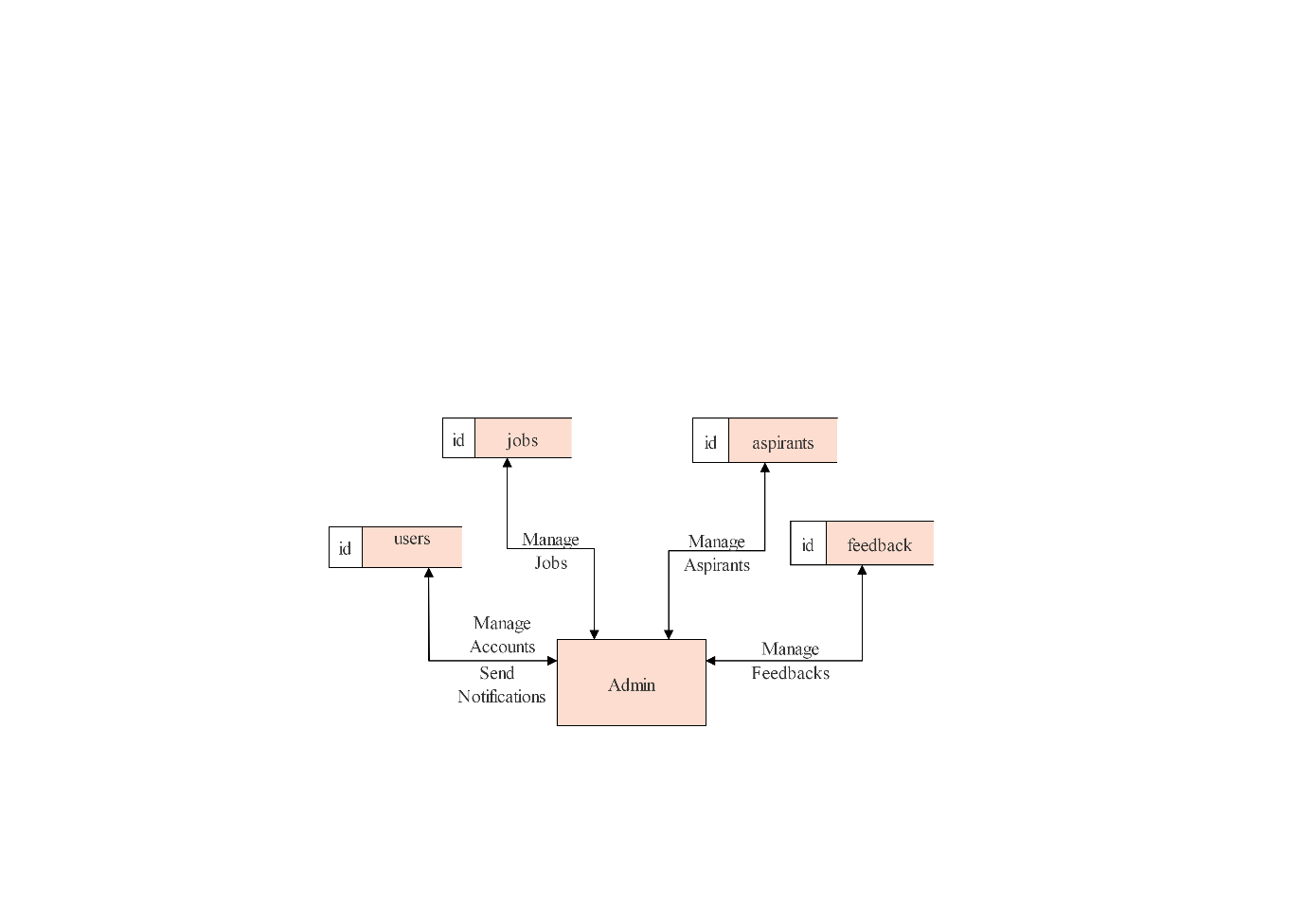
**Level 1:**



**Level 2:**



**Level 2(ADMIN):**



**3.4.4 Constructing a DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.

2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source.

3. When a process is exploded into lower-level details, they are numbered. 4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

**3.5 Description of components**

**3.5.1 Admin:** This component is to enable the logging in of the admin.

* **Input:** username, password
* **Process:** The system checks for the correctness of the username and password.
* **Output:** Admin will be taken to the home page on successful login.

**3.5.2 Employer:** This component to store the employee information,

employee job information.

* **Input:** username, password, new jobs
* **Process:** post job information
* **Output:** job information

**3.5.3 Aspirants:** This component to store the User details.

* **Input:** user id, user name, address, contact.
* **Process:** The entered details by the user during the registration will be stored in

the database.

* **Output:** Displays the user details in the admin form.

**CHAPTER – 4**

**DATABASE DESIGN**

**4.1 Introduction**

Database Design is a collection of process that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Database Design is the process of producing a detailed data model of a database. The main objectives of database designing are to produce logical and physical designs models of the proposed database system. Database design is the structure a database uses to plan, store and manage data. Data consistency is achieved when a database is designed to store only useful and required data.

The term database design can be used to describe many different parts of the design of an overall database system principally and most correctly. It can be thought of as the logical design of the database structures used to store the data. In the relational model, these are the tables and views. In an object database, the entities and relationship map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the data base structures, but also the forms and quires used as a part of the overall database application within the Data Base Management System (DBMS).

**4.2 Purpose and Scope**

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

The logical model: concentrates on the data requirements and data to be stored independent of physical considerations. It does not concern itself with how the data will be stored physically. The physical data design model involves translating the logical design of the database into physical media using hardware resources and software systems such as DBMS.

**4.3 Database Identification**

* Database table name and column names defined without leaving space.
* Lower case used to create database tables name columns names.
* Primary Key and foreign Key defined with same name.

**4.4 Schema Information**

A schema is the structure behind data organization. It is a visual representation of how different table relationships enable the schema’s underlying mission business rules for which the database is created. In a schema diagram, all database tables are designated with unique columns and special features, e.g., primary/foreign keys or not null, etc. Formats and symbols for expression are universally understood, eliminating the possibility of confusion. The table relationships also are expressed via a parent table’s primary key lines when joined with the child table’s corresponding foreign keys.

**4.5 Table Definition**

**4.5.1 Table name: users**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | user\_  id | user\_  name | | blog | email | password | acnt\_  type | conf\_  doc | mime | date | viewed | valid |
| mail\_  verified | | |

**4.5.2 Table name: jobs**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| job\_  num | eno | job\_  title | | job\_  details | | location | | salary | | job\_  type | qualifications | graduation | spex |
| about\_comp | | | last\_date | | selected | | viewed | |

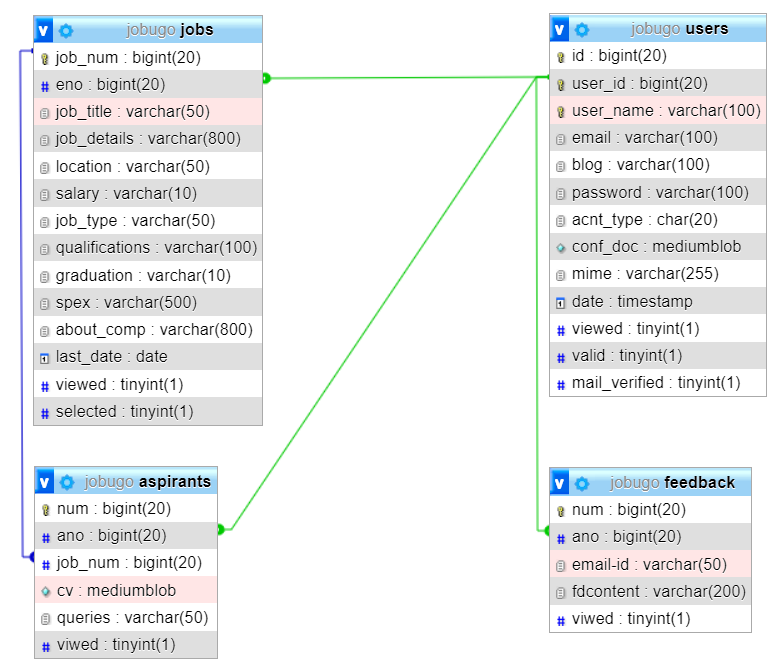
**4.5.3 Table name: aspirants**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| num | eno | job\_num | cv | queries | viewed |

**4.5.4 Table name: feedback**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| num | ano | email-id | fdcontent | viewed |

**4.6 Physical design**

****

**4.7 Data Dictionary**

**Table Structure**: The database JOBUGO is organized into following tables:

* users
* jobs
* aspirants
* feedback

**4.7.1 Table name: users**

This table is used to store users details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraints** | **Description** |
| Id | bigint | 20 |  | Identification Number |
| user\_id | bigint | 20 | PRIMARY KEY | User ID |
| user\_name | VARCHAR | 100 |  | User name of the user |
| blog | VARCHAR | 100 |  | Use to store website address |
| email | VARCHAR | 100 |  | Email id |
| password | VARCHAR | 100 |  | Password to Login |
| acnt\_type | char | 20 |  | Type of the account |
| conf\_doc | meduimblob |  |  | Use to store documents for confirmation/CV |
| mime | VARCHAR | 255 |  | Use to store the type of data stored in conf\_doc |
| date | timestamp |  |  | Date and time of the account creation |
| viewed | tinyint | 1 |  | Used to confirm the account has been viewed by admin |
| valid | tinyint | 1 |  | Used to confirm the account is valid |
| mail\_verified | tinyint | 1 |  | To confirm employer mail account is valid |

**4.7.2 Table name: jobs**

This table is used to store jobs details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **References** | **Data Type** | **Size** | **Constraints** | **Description** |
| job\_num |  | bigint | 20 | PRIMARY  KEY | Job number |
| eno | user\_id (users) | bigint | 20 | FOREIGN  KEY | Employer number |
| job\_title |  | VARCHAR | 50 |  | Title of job |
| job\_details |  | VARCHAR | 800 |  | Details of the job |
| location |  | VARCHAR | 50 |  | Location |
| salary |  | VARCHAR | 10 |  | Salary |
| job\_type |  | VARCHAR | 50 |  | What type of job |
| qualifications |  | VARCHAR | 100 |  | Type of the qualifications for the job |
| graduation |  | VARCHAR | 10 |  | What type of graduation |
| spex |  | VARCHAR | 500 |  | Any specialization of a job |
| about\_comp |  | VARCHAR | 800 |  | About company |
| last\_date |  | date |  |  | Last date for apply |
| viewed |  | tinyint | 1 |  | Used to confirm the account has been viewed by admin |
| selected |  | tinyint | 1 |  | Used while processing jobs |

**4.7.3 Table name: aspirants**

This table is used to store aspirants details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **References** | **Data Type** | **Size** | **Constraints** | **Description** |
| num |  | bigint | 20 | PRIMARY  KEY | Number |
| ano | user\_id (users) | bigint | 20 | FOREIGN  KEY | Aspirant number |
| job\_num | job\_num (jobs) | bigint | 20 | FOREIGN KEY | Employer number |
| cv |  | MEDIUMBLOB | 16MB |  | Curriculum vitae |
| queries |  | VARCHAR | 50 |  | Queries |
| viewed |  | tinyint | 1 |  | Used to confirm the account has been viewed by admin |

**4.7.4 Table name: feedback**

This table is used to store feedback details

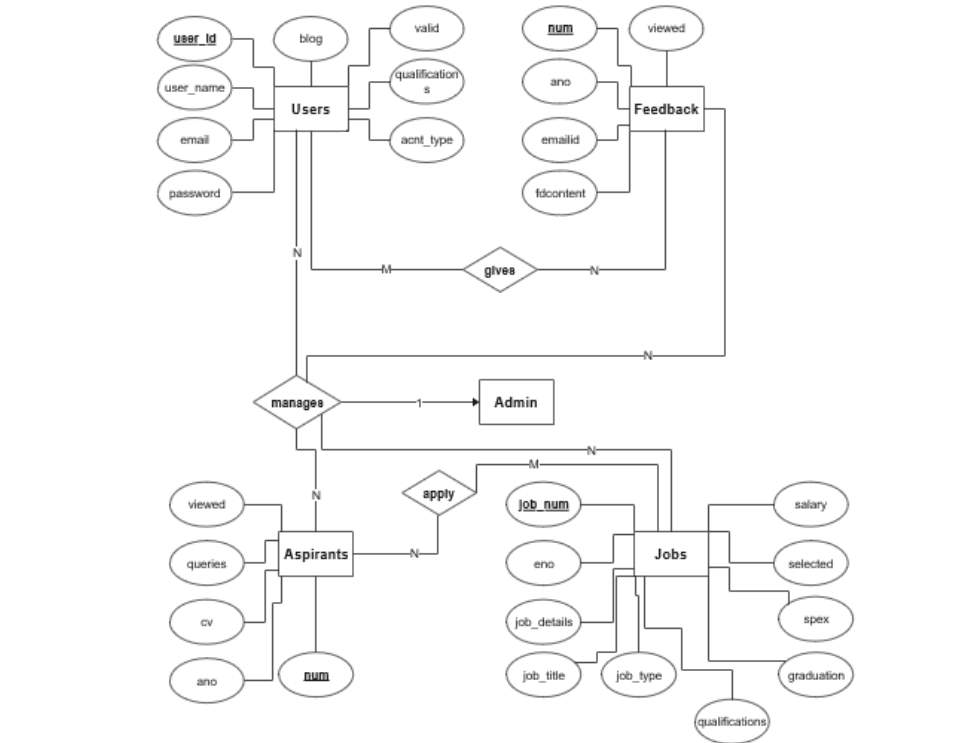
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **References** | **Data Type** | **Size** | **Description** |
| num  (PRM KEY) |  | bigint | 20 | Number |
| ano  (FOREIGN KEY) | user\_id (users) | bigint | 20 | Aspirant number |
| email-id |  | VARCHAR | 50 | Email-id |
| fdcontent |  | VARCHAR | 200 | Feedback content |
| viewed |  | tinyint | 1 | Used to confirm the account has been viewed by admin |

**4.8 ER Diagram**

**4.8.1 ER-Diagram Symbols**

|  |  |  |
| --- | --- | --- |
| **Name** | **Notation** | **Description** |
| Entity | 1Entity name | It may be an object with the physical existence or conceptual existence. It is represented by a Rectangle. |
| Attribute |  | The properties of the entity can be a Ellipse. |
| Relationship | Relation | Whenever an attribute of an entity refers to another entity, some relationship exists. It is represented by a diamond. |
| Link |  | Lines link attribute to entity sets to relation. |
| Key Attribute |  | An entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by a Underlined word in ellipse. |
| Cardinality Ratio | 1. 1:1 2. 1:M 3. M:1 4. M:M | It specifies the maximum number of relationship instances that an entity can participate in. There are four cardinality ratios. |

**ER Diagram**



**4.9 Database Administration**

**4.9.1 System Information**

* Server: XAMPP
* Database: MYSQL
* Operating System: Windows 10
* Programming languages: PHP, CSS, HTML, Java Script

**4.9.2 DBMS Configuration**

Part of the DBMS installation process is the connection of the DBMS to other system software components that must interact with the DBMS. Typical infrastructure software are that may need to be configured to work with DBMS includes networks, transaction processing monitors, message queues, other types of middleware, programming languages, Systems management software, Operations and job control software Web servers, and application servers.

**4.9.3 Support Software Requirements**

The system installs MYSQL server while installing XAMPP software. All the backup content stores in MySQL data folder.

**4.9.4 Storage Requirements**

* Storage for database continues a set of compatible software and hardware where database files are stored.
* Compatibility must be carefully tested to ensure eliminating bottlenecks and the possibility of data corruption.
* Failure free work in high load conditions and the redundancy of vulnerable components must be provided. Such a set must meet the highest standards in terms of performance and reliability, to ensure continuous and fast access to important data.

**4.9.5 Backup and Recovery**

A volatile storage like RAM stores all the active logs, disk buffers, and related data. In addition, it stores all the transactions that are being currently executed. If such a volatile storage crashes abruptly, it would take away all the logs and active copies of the database makes recovery almost impossible, as everything that is required to recover the data is lost.

Following techniques may be adopted in case of loss of volatile storage-

* We can have checkpoints at multiple stages so as to save the contents of the database periodically.
* A state of active database in the volatile memory can be periodically dumped onto a stable storage, which may also contain logs and active transactions and buffer blocks.
* When the system recovers from a failure, it can restore the latest dump.
* It can maintain a redo-list and an undo-list as checkpoints.
* It can recover the system by consulting undo-redo lists to restore the state of all transactions up to the last checkpoint.

**CHAPTER – 5**

**DETAILED DESIGN**

**5.1 Introduction**

The purpose of the design phase is to plan a solution of the problem specified by the requirements document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed; design takes us towards how to satisfy the needs. The design of the system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phases, particularly testing and maintenance.

The design activity often results in three separate outputs- Architecture Design, High Level Design and Detailed Design. Architecture focuses on looking at a system as a combination of many different components, and how they interact with each other to produce the desired results. The High-Level Design identifies the modules that should be built for developing the system and the specifications of these modules. At the end of system design all the major data structures, file formats, output formats etc., are also fixed. In detailed design, the internal logic of each of the modules are specified.

The design process for the software systems often has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules and how the modules should be interconnected. This is what is called the System Design or the Top-Level Design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied is decided. This design level is often called Detailed Design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

\

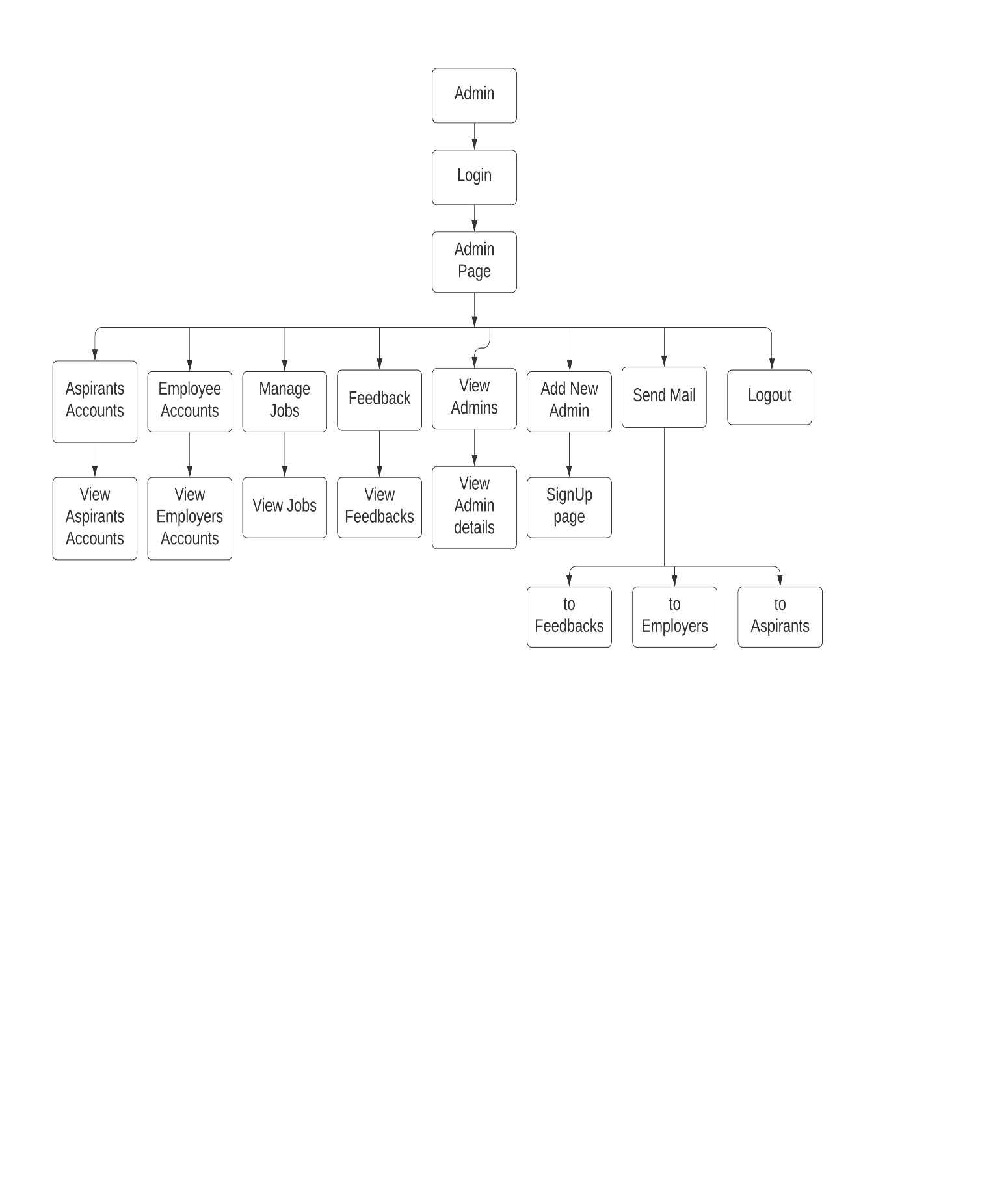
**5.1.1 Applicable Documents**

The Detailed Design refines the System Design document hence the first applicable document here is system design. We also refer the data structures. Hence the second applicable document is database design.

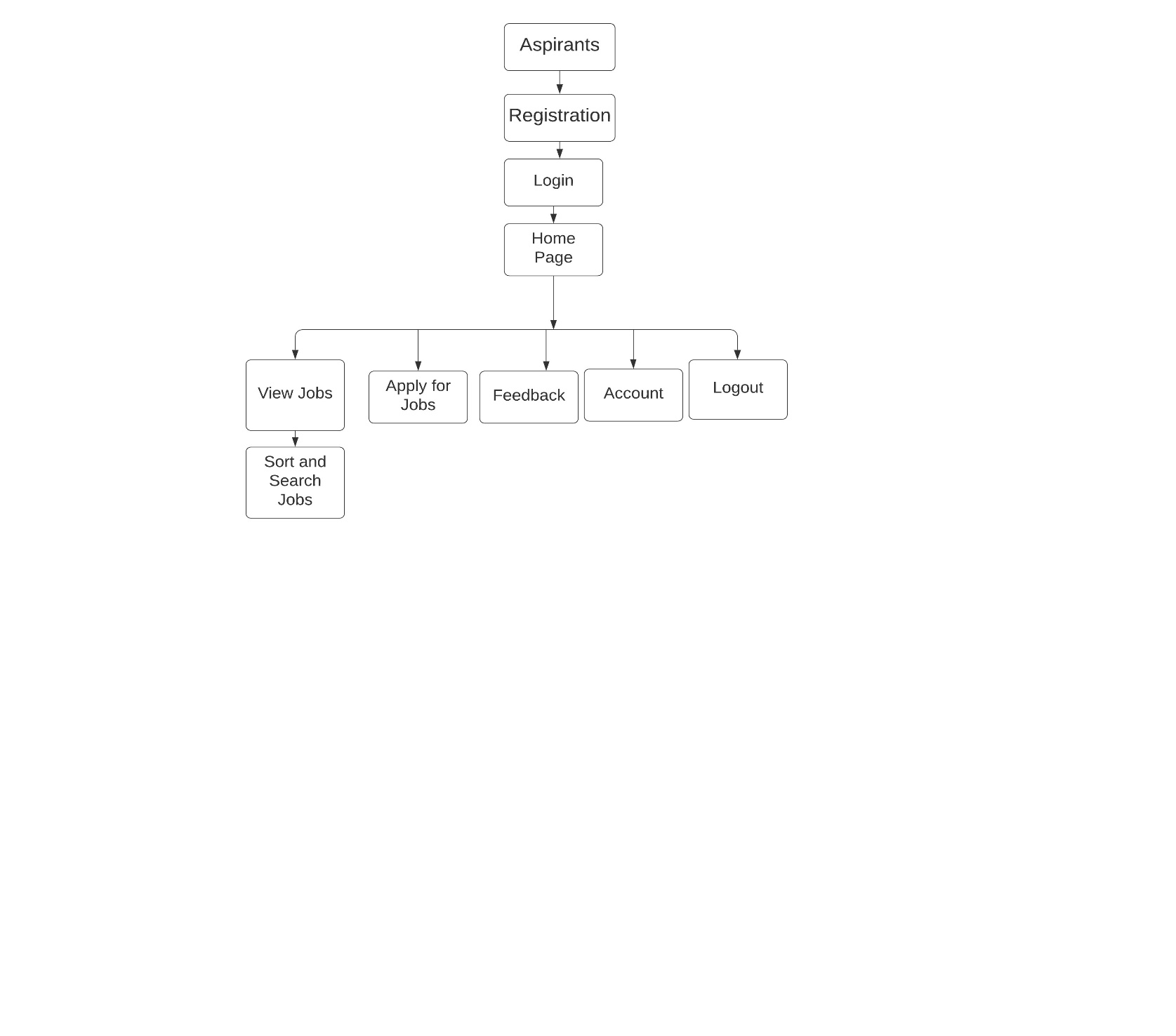
**5.2 Structure of the Software Package**

The various functional components used are

* Admin module

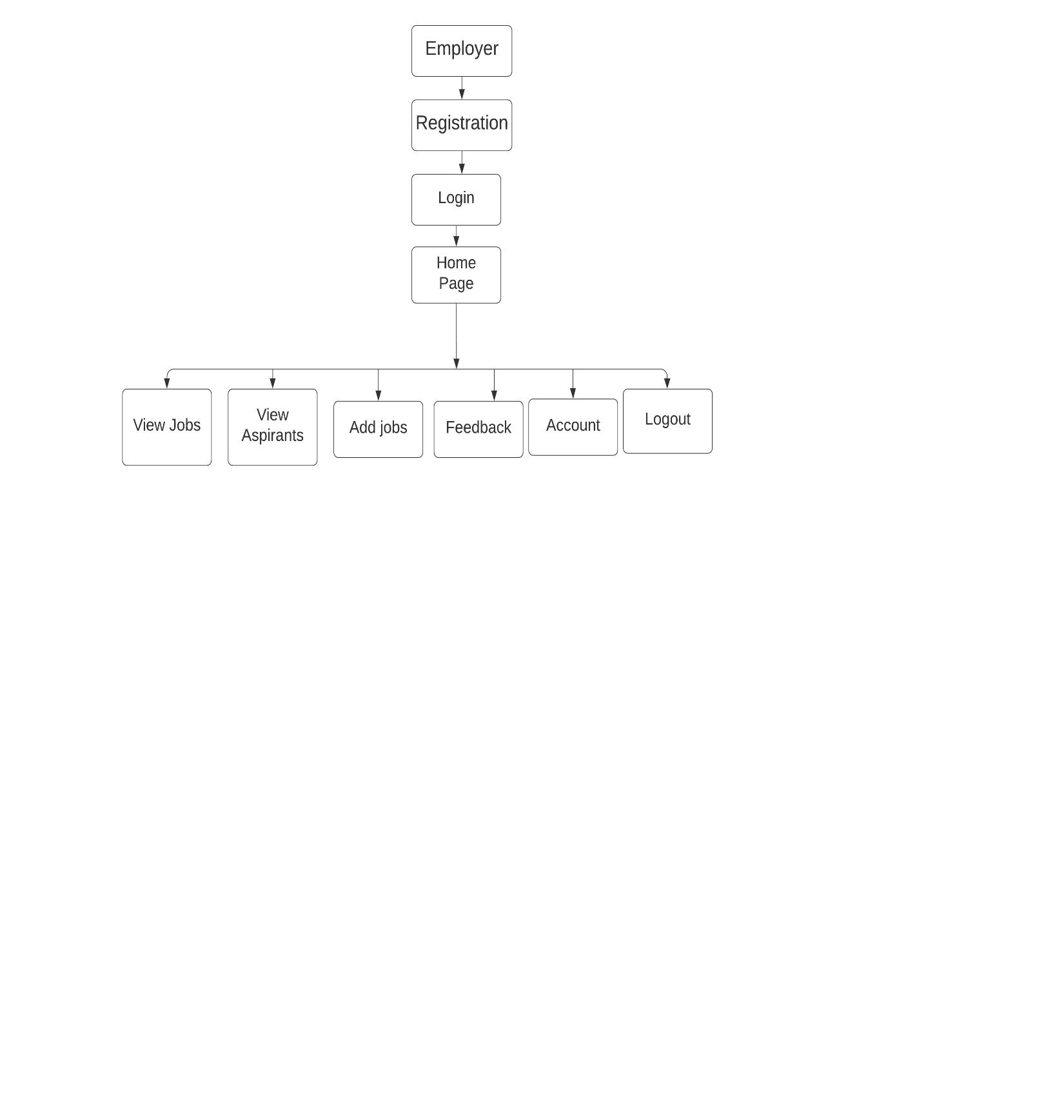


* Aspirants Module



Apply for job

* Employer Module



**5.3 Modular Decomposition Components**

**5.3.1 Admin**

**5.3.1.1 Login**

**a) Input:** Email ID, Password.

**b) Procedure details:**

Start

Enter email and password

Verification(Email,Psw)

Display Home Page

Stop

Incorrect Email/Password

True

False

**5.3.1.1 Login**

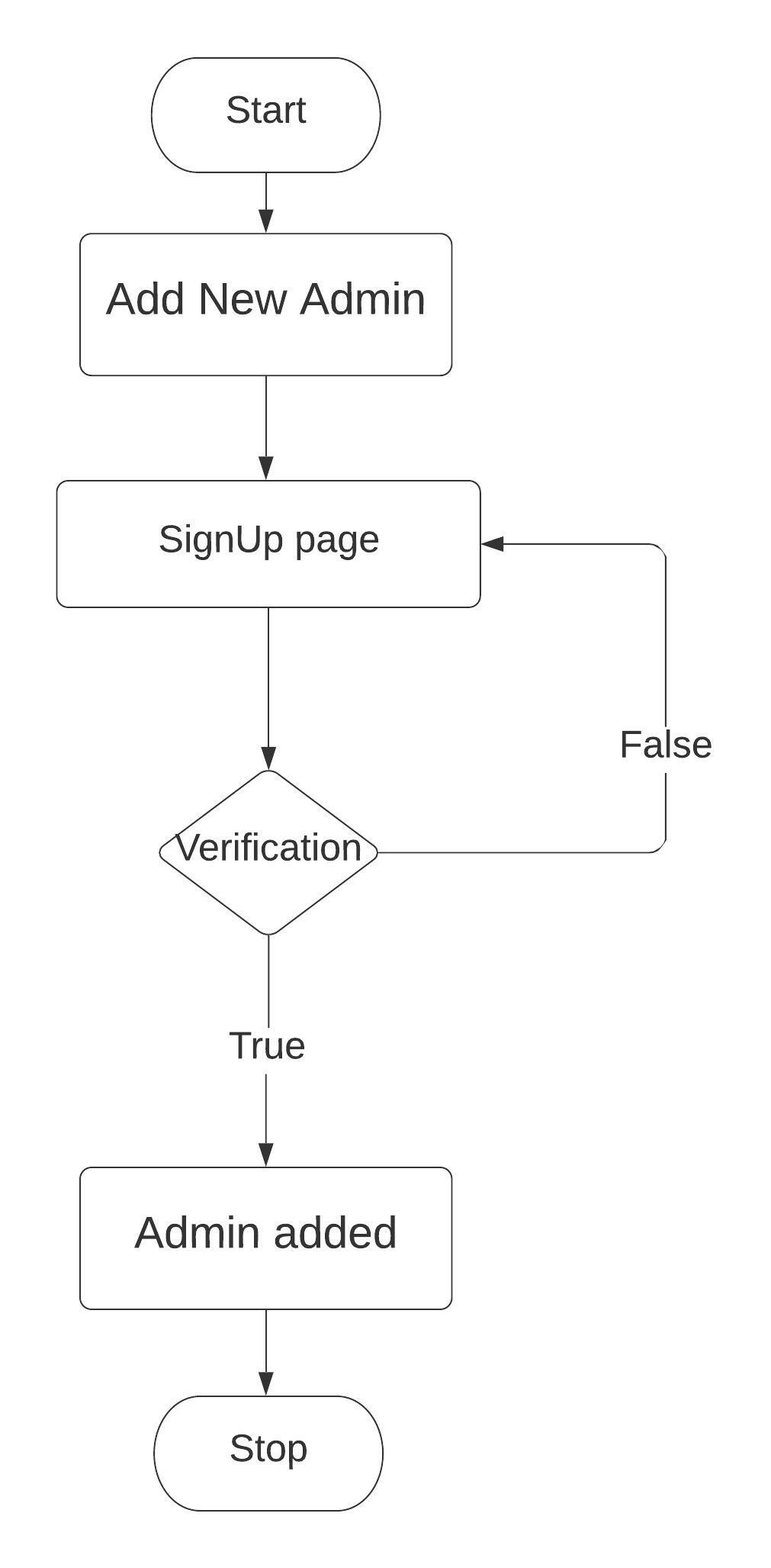
**c) I/O interface:** Graphical user interface(Form) for admin to enter the email and password.

**d) Output:** The email ID and Password is verified. If correct ,the admin gain authorization into the Admin Page.

**5.3.1.2 Add New Admin**

**a) Input:** Email, username, password.

**b) Procedure details:**



**5.3.1.2 Add New Admin**

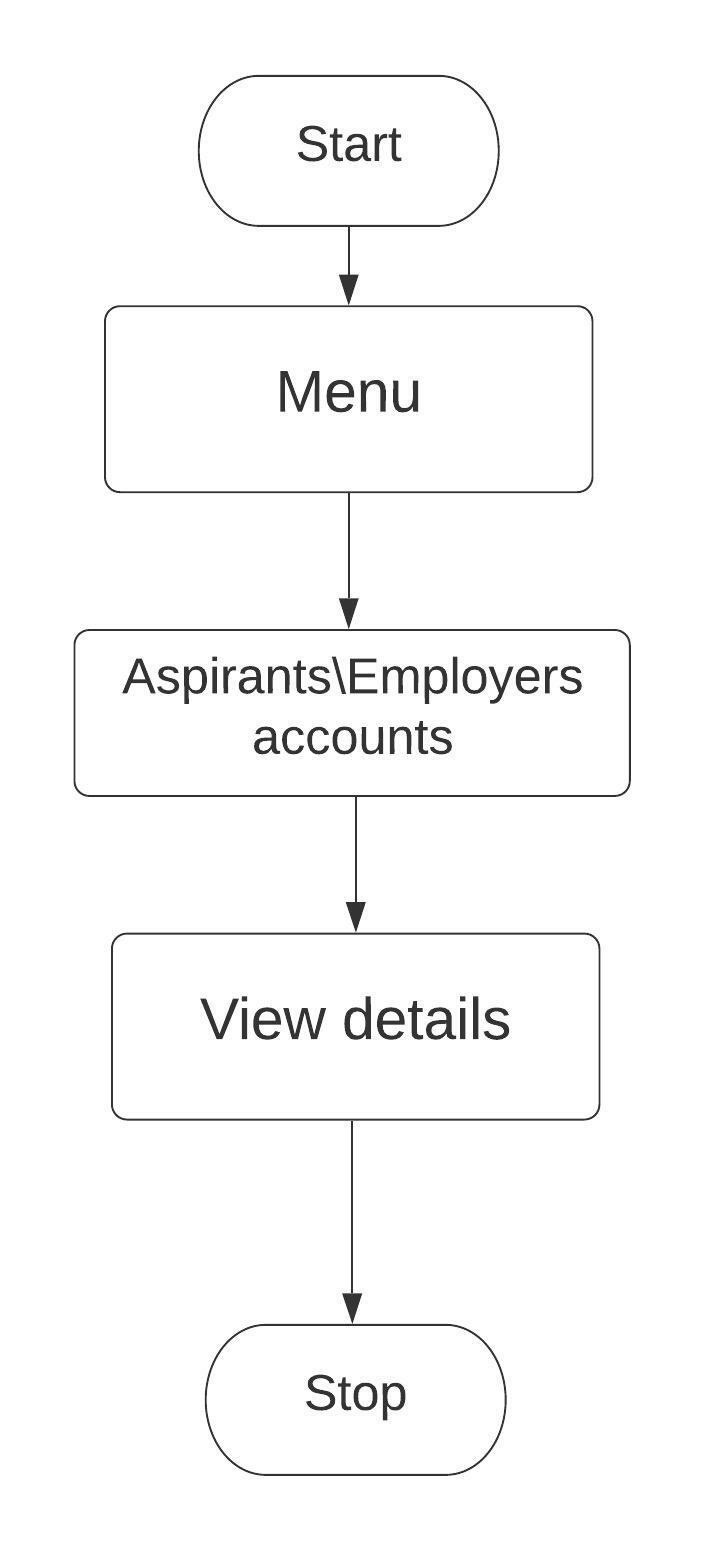
**c) I/O interface:** Graphical user interface(Form) for admin to enter the required details.

**d) Output:** The email ID and Password is verified. If correct, new admin will be created.

**5.3.1.3 View all users and Manage**

**a) Input:** After successful Login into admin panel.

**b) Procedure details:**



View account details

**5.3.1.3 View all users and Manage**

**c) I/O interface:** Graphical user interface.

**d) Output:** Admin can view the details of users and also, Admin can manage their accounts.

**5.3.1.4 Send Response**

**a) Input:** Email- id and feedback

**b) Procedure details:**

Start

Send Response

Validation

Stop

False

Sent successfully

**5.3.1.4 Send Response**

**c) I/O interface:** All the users of the application have interface to give feedback.

The admins can view the feedback and give Response if necessary.

**d) Output:** The Response for feedback Will be sent.

**5.3.1.5 View feedback**

**a) Input:** Email- id and feedback

**b) Procedure details:**

Start

Open Menu

Feedback

False

No feedback available

Feedback Display

Stop

True

**5.3.1.5 View feedback**

**c) I/O interface:** The admin can view the feedback.

**d) Output:** The feedback stored In the database is displayed.

**5.3.2 Aspirants**

**5.3.2.1 Login**

**a) Input:** Email, password

**b) Procedure details:**

Start

Enter email/password

Verification

Display Home Page

Stop

Incorrect Email/Password

True

False

**5.3.2.1 Login**

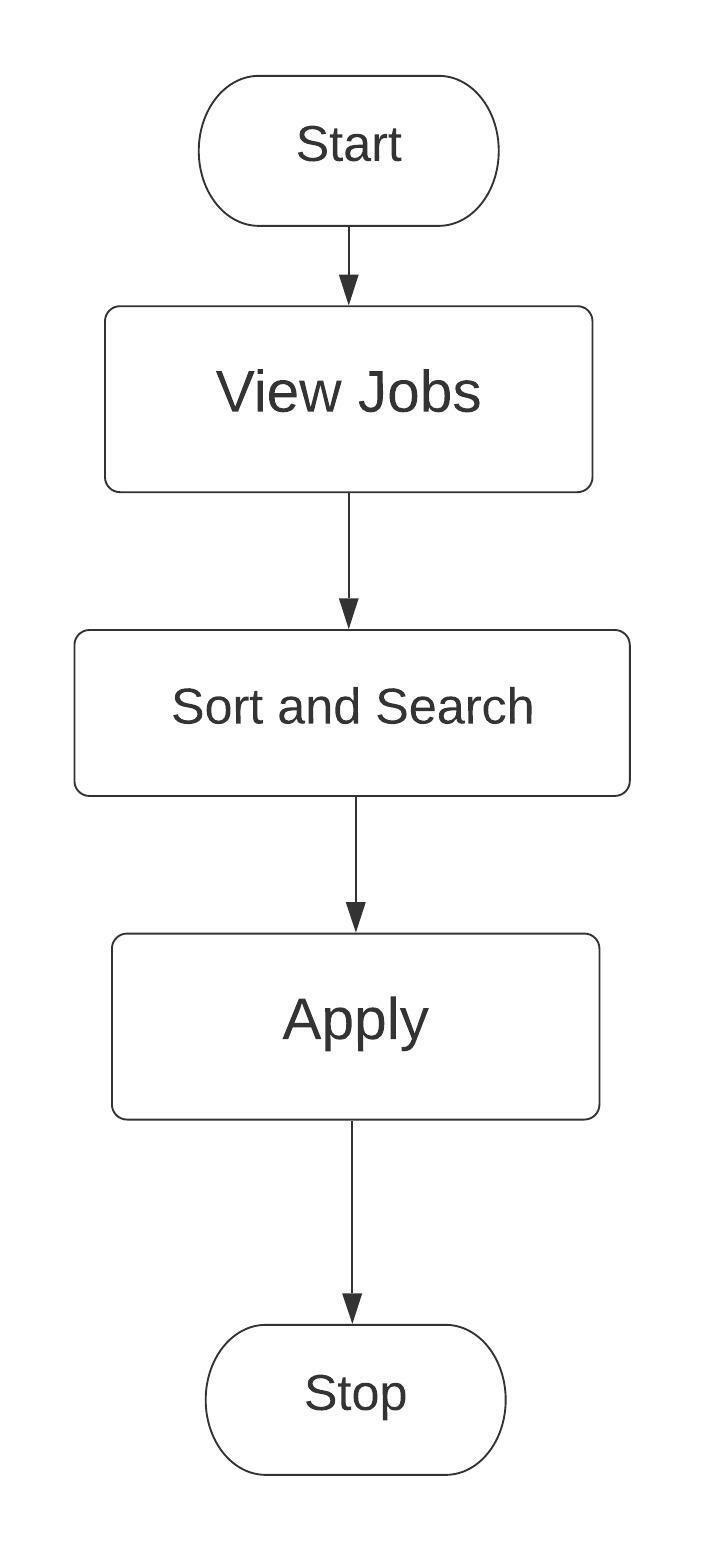
**c) I/O interface:** Graphical user interface for Aspirants to enter the email and password.

**d) Output:** The email and password are verified. If correct, the Aspirants gain authorization into the web application.

**5.3.2.2 View and Apply**

**a) Input:** Successful Login

**b) Procedure details:**



**5.3.2.2 View and Apply**

**c) I/O interface:** Graphical user interface for Aspirants to search and sort their dream jobs.

**d) Output:** Successfully Applied for dream job.

**5.3.2.3 Send feedback**

**a) Input:** The user needs to input feedback description and email.

**b) Procedure details:**

Start

Give feedback

Validation

Stop

False

Inserted successfully

**5.3.2.3 Send feedback**

**c) I/O interface:** All the users of the application have interface to give and view feedback. The admins can view the feedback

**d) Output:** The given feedback are stored in the database and Response will be sent back.

**5.3.2.4 View Response**

**a) Input:** Feedback and Email

**b) Procedure details:**

Start

Open Menu

Feedback

False

No feedback available

Display Response

Stop

True

**5.3.2.4 View Response**

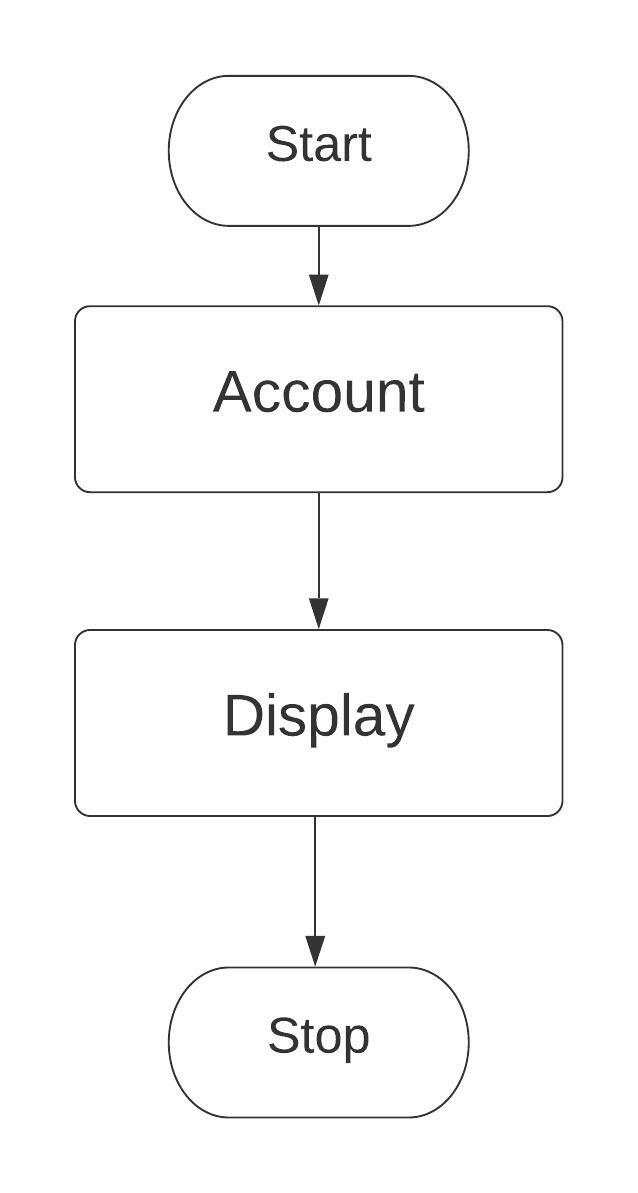
**c) I/O interface:** The admin will view the feedback and Give Response and Aspirants can view the responses.

**d) Output:** The Response is displayed.

**5.3.2.5 Account**

**a) Input:** Already Provided while Registering (Username, Email and other details).

**b) Procedure details:**



**5.3.2.5 Account**

**c) I/O interface:** The Aspirants can view their account details.

**d) Output:** The Account Details will be displayed.

**5.3.3 Employers**

**5.3.3.1 Login**

**a) Input:** Email, password

**b) Procedure details:**

Start

Enter email/password

Verification

Display Home Page

Stop

Incorrect Email/Password

True

False

**5.3.3.1 Login**

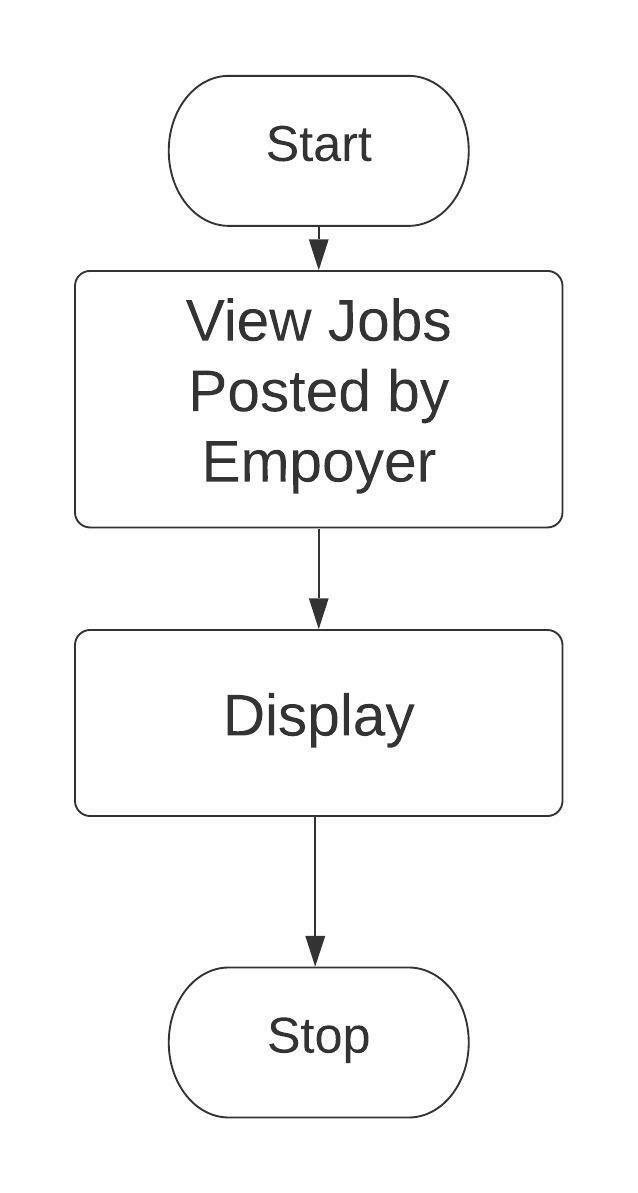
**c) I/O interface:** Graphical user interface for Employers to enter the email and password.

**d) Output:** The email and password are verified. If correct, the Employers gain authorization into the web application.

**5.3.3.2 View Jobs Posted by Employer**

**a) Input: New Jobs**

**b) Procedure details:**



**5.3.3.2 View Jobs Posted by Employer**

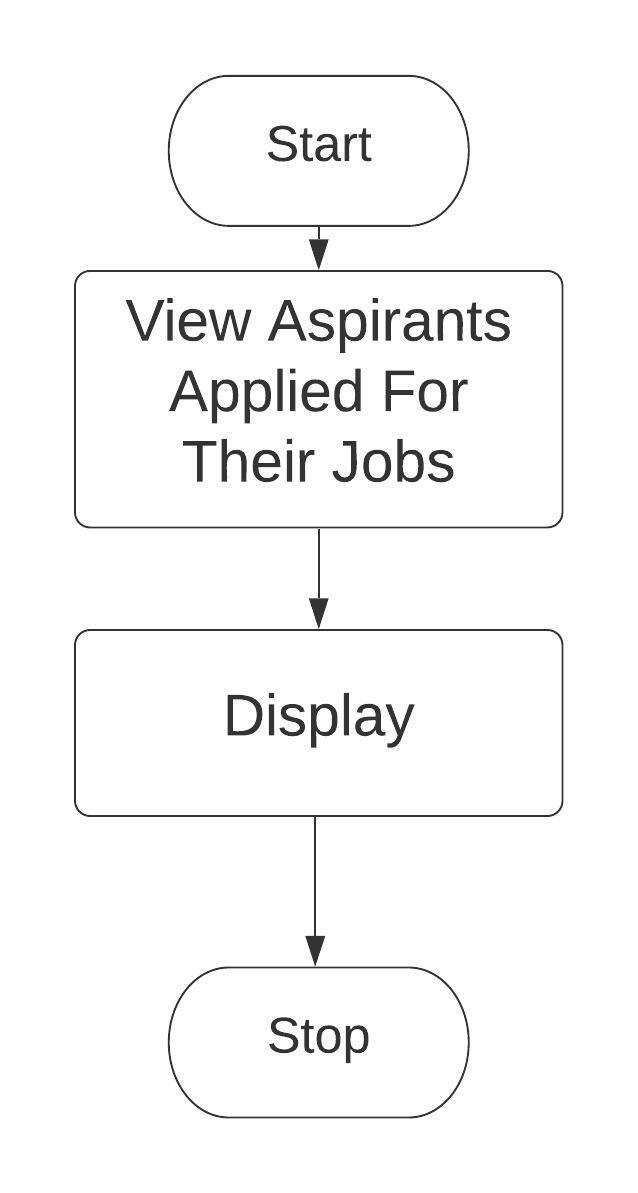
**c) I/O interface:** Graphical user interface for Employers to view the jobs.

**d) Output:** If there are Jobs then it will be displayed else an empty page will be displayed.

**5.3.3.3 View Aspirants**

**a) Input: Job Application**

**b) Procedure details:**



**5.3.3.3 View Aspirants**

**c) I/O interface:** Graphical user interface for Employers to view the aspirants who applied for their jobs.

**d) Output:** If there are Aspirants who applied for Jobs then it will be displayed else an empty page will be displayed.

**5.3.3.4 Send feedback**

**a) Input:** The user needs to input feedback description and email.

**b) Procedure details:**

Start

Give feedback

Validation

Stop

False

Inserted successfully

**5.3.3.4 Send feedback**

**c) I/O interface:** All the users of the application have interface to give and view feedback. The admins can view the feedback

**d) Output:** The given feedback are stored in the database and Response will be sent back.

**5.3.3.5 View Response**

**a) Input:** Feedback and Email

**b) Procedure details:**

Start

Open Menu

Feedback

False

No feedback available

Display Response

Stop

True

**5.3.3.5 View Response**

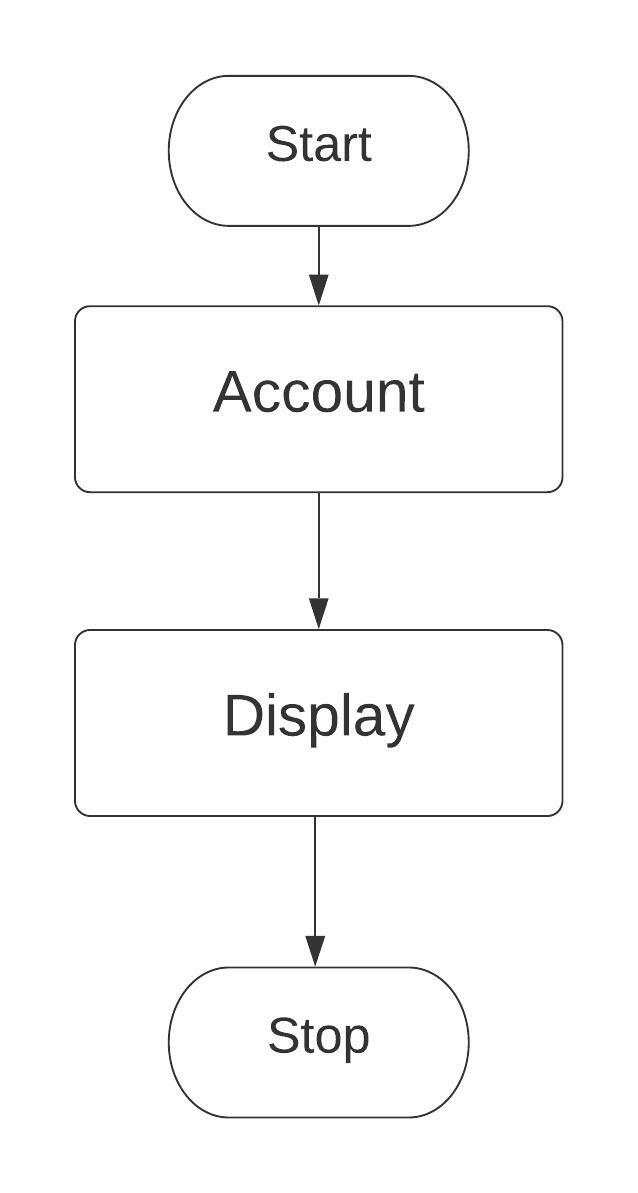
**c) I/O interface:** The admin will view the feedback and Give Response; The Employers can view the Response.

**d) Output:** The Response is displayed.

**5.3.3.6 Account**

**a) Input:** Already Provided while Registering (Username, Email and other details).

**b) Procedure details:**



**5.3.3.6 Account**

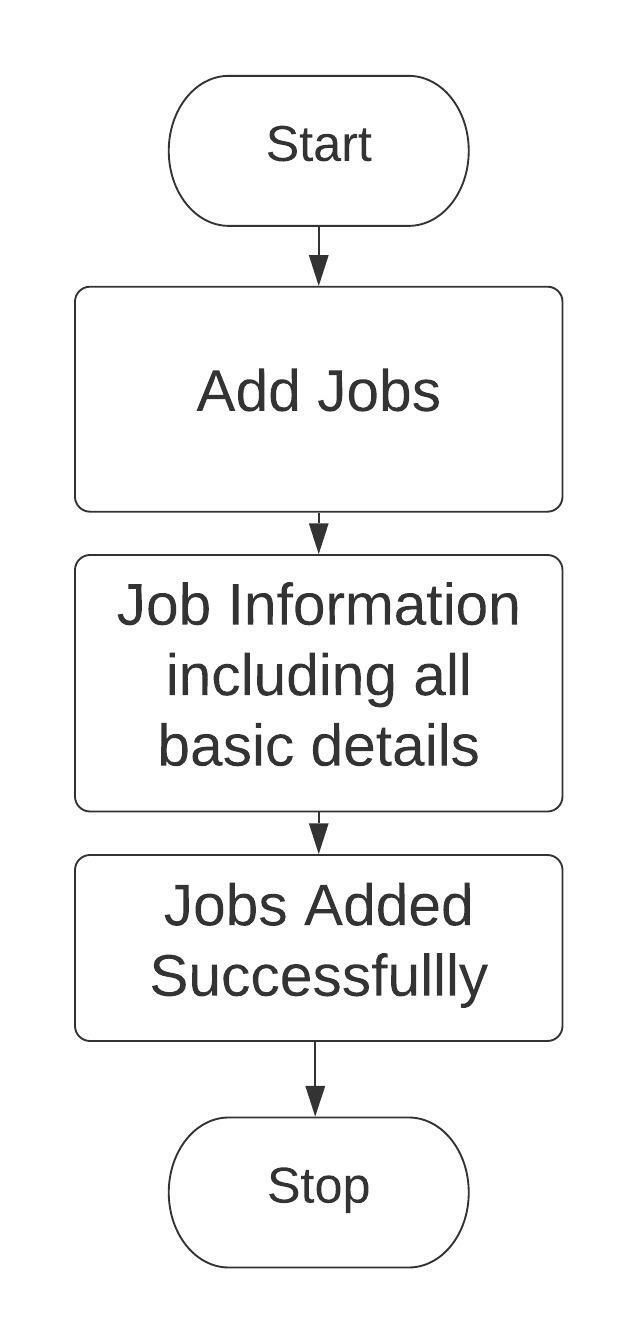
**c) I/O interface:** The Employers can view their account details.

**d) Output:** The Account Details will be displayed.

**5.3.3.7 Add Jobs**

**a) Input:** New Jobs

**b) Procedure details:**



**5.3.3.7 Add Jobs**

**c) I/O interface:** The Employers can add new Jobs along with basic description including eligibility, work life, salary and many more.

**d) Output:** New Job with all details will be displayed.

**CHAPTER – 6**

**CODING**

**6.1 Introduction**

The goal of coding or programming phase is to translate the design of the system, produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computations performed by the design.

**6.2 Database Connection:**

<?php

$dbhost = "localhost";

$dbuser = "root";

$dbpass = "";

$dbname = "jobugo";

if(!$con = mysqli\_connect($dbhost,$dbuser,$dbpass,$dbname))

{

die("failed to connect!");

}

?>

* 1. **Authorization / Authentication**

**Authentication** is the process of verifying who you are. When you log on to a PC with a user name and password you are authenticating.

**Authorization** is the process of verifying that you access to something gaining access to a resource

**6.4 Functions:**

<?php

function check\_login($con)

{

if(isset($\_SESSION['user\_id']))

{

$id = $\_SESSION['user\_id'];

$query = "select \* from users where user\_id = '$id' limit 1";

$result = mysqli\_query($con,$query);

if($result && mysqli\_num\_rows($result) > 0)

{

$user\_data = mysqli\_fetch\_assoc($result);

return $user\_data;

}

}

header("Location: noaccountalert.php");

die;

}

function random\_num($length)

{

$text = "";

if($length < 5)

{

$length = 5;

}

$len = rand(4,$length);

for ($i=0; $i < $len; $i++) {

$text .= rand(0,9);

}

return $text;

}

**6.5 Signup:**

**6.5.1 Signup:**

<?php

session\_start();

include("connection.php");

include("functions.php");

$error\_string = "";

$uid;

$query4 = "SELECT user\_id from users order by user\_id desc LIMIT 1";

$result4 = mysqli\_query($con,$query4);

if($result4){

do{

$row4=mysqli\_fetch\_assoc($result4);

$uid=(int)$row4['user\_id'];

}while(0);

}else{$error\_string = $con->error;}

$uid++;

mysqli\_report(MYSQLI\_REPORT\_ERROR | MYSQLI\_REPORT\_STRICT);

if($\_SERVER['REQUEST\_METHOD'] == "POST")

{

$user\_name = ($\_POST['uname']);

$password = ($\_POST['psw']);

$password\_rep = ($\_POST['psw-repeat']);

$mail = ($\_POST['email']);

$acnt\_type = "employer";

$blog = $\_POST['blog'];

$\_SESSION['mailverified'] = 0;

$\_SESSION['flag'] = 1;

$query = "SELECT user\_name FROM users WHERE user\_name='$user\_name'";

$junga = mysqli\_query($con, $query);

if (mysqli\_num\_rows($junga) != 0)

{

$error\_string = "Username already exists";

echo $error\_string;

} else if(!empty($user\_name) && !empty($password) && !is\_numeric($user\_name)){

if (filter\_var($mail, FILTER\_VALIDATE\_EMAIL)) {

if(strlen($password)>5){

if( preg\_match('([a-zA-Z].\*[0-9]|[0-9].\*[a-zA-Z])', $password) ) {

if($password == $password\_rep){

if(strlen($user\_name)>4){

$queryy = "SELECT email FROM users WHERE email='$mail'";

$resultt = mysqli\_query($con,$queryy);

if(mysqli\_num\_rows($resultt) == 0){

$data = file\_get\_contents($\_FILES['myfile']['tmp\_name']);

$mime = $\_FILES['myfile']['type'];

$stmt = $con->prepare("insert into users (user\_id,user\_name,email,password,blog,acnt\_type,conf\_doc,mime) values (?,?,?,?,?,?,?,?)");

$stmt->bind\_param('ssssssss',$uid,$user\_name,$mail,$password,$blog,$acnt\_type,$data,$mime);

if($stmt->execute()){

$\_SESSION['selected'] = $uid;

header("Location: confirm/emp\_conf.php");

} else{ $error\_string = $con->error; }

}else{ $error\_string = "There cannot be more than one accounts connected to the same Email-ID";}

}else{$error\_string = "Username must contain at least 5 characters!";}

}else{ $error\_string = "Passwords DoNot Match!"; }

}else{ $error\_string = "The password doesnot contain both alphabets and numbers!"; }

} else{ $error\_string = "Password less than 5 characters"; }

}else{

$error\_string = "Please Enter Valid Email ID!<br>";

if ($password != $password\_rep) {$error\_string ="Passwords DoNot Match!"; }

}

} else{ $error\_string = "Please enter some valid information!"; }

}

?>

**6.5.2 Email verification:**

<?php

session\_start();

include("../connection.php");

include("../functions.php");

include("../mail.php");

$length = 5;

$error\_string = "";

$otp\_send = random\_num($length);

$selected = $\_SESSION['selected'];

$query1 = "select \* from users where user\_id='$selected'";

$result1 = mysqli\_query($con,$query1);

if(!($result1)){

$error\_string = "SOme error Pardon us!";

}else{

$row = mysqli\_fetch\_assoc($result1);

$mailid = $row['email'];

$name = $row['user\_name'];

$sub = "Email Confirmation!";

$bdy = "This is the final step of your registration.<br>The OTP for the validation of your mail ID is: $otp\_send";

if($\_SESSION['flag'] == 1){

$\_SESSION['flag'] = 0;

$\_SESSION['otp'] = $otp\_send;

mailer($mailid,$name,$sub,$bdy);

}

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" && isset($\_POST['validate']))

{

$otp\_receive = $\_POST['otp\_val'];

if($otp\_receive === $\_SESSION['otp'])

{

$query2 = "UPDATE users set mail\_verified = b'1' where user\_id='$selected'";

if(mysqli\_query($con,$query2)){

$\_SESSION['source'] = 'acnt';

header("Location:../success.php");

}

}else{$error\_string="Wrong OTP!";}

}

?>

**6.6 Login:**

<?php

session\_start();

include("connection.php");

include("functions.php");

$error\_string = "";

$\_SESSION['loopcounter'] = 0;

$\_SESSION['flag'] = 8;

$\_SESSION['fcounter'] = 0;

$\_SESSION['filter'] = "0000";

$user\_data['acnt\_type'] = "Null";

if($\_SERVER['REQUEST\_METHOD'] == "POST")

{

$user\_name = $\_POST['uname'];

$password = $\_POST['psw'];

$acnt\_type = $\_POST['accnt\_select'];

$error\_string;

if(!empty($user\_name) && !empty($password) && !is\_numeric($user\_name))

{

$\_SESSION['accounttype'] = $acnt\_type;

if($acnt\_type == 'admin')

{

$query = "select \* from users where user\_name = '$user\_name' limit 1"; // ADMIN table name is users

$result = mysqli\_query($con, $query);

if($result)

{

if($result && mysqli\_num\_rows($result) > 0)

{

$user\_data = mysqli\_fetch\_assoc($result);

if($user\_data['password'] === $password && $user\_data['acnt\_type'] === $acnt\_type)

{

$\_SESSION['user\_id'] = $user\_data['user\_id'];

header("Location: admin.php");

die;

}else{ $error\_string = "Wrong Username or Password";}

}

}if ($user\_data['acnt\_type'] != $acnt\_type) {

$error\_string = "wrong type";

} else {

$error\_string ="Wrong User name or password";

}

} else if($acnt\_type == 'aspirant') {

$query = "select \* from users where user\_name = '$user\_name' limit 1";

$result = mysqli\_query($con, $query);

if($result)

{

if($result && mysqli\_num\_rows($result) > 0)

{

$user\_data = mysqli\_fetch\_assoc($result);

if($user\_data['password'] === $password && $user\_data['acnt\_type'] === $acnt\_type)

{

$\_SESSION['user\_id'] = $user\_data['user\_id'];

header("Location: index.php");

die;

}else{ $error\_string = "Wrong Username or Password";}

} }

if ($user\_data['acnt\_type'] != $acnt\_type) {

$error\_string = "wrong type";

} else {

$error\_string="Wrong User name or password!";

}

} else if($acnt\_type == 'employer') {

$query = "select \* from users where user\_name = '$user\_name' limit 1";

$result = mysqli\_query($con, $query);

if($result)

{

if($result && mysqli\_num\_rows($result) > 0)

{

$user\_data = mysqli\_fetch\_assoc($result);

if($user\_data['password'] === $password && $user\_data['acnt\_type'] === $acnt\_type)

{

$\_SESSION['user\_id'] = $user\_data['user\_id'];

header("Location: employerh.php");

die;

} else{ $error\_string = "Wrong Username or Password";}

} }

if ($user\_data['acnt\_type'] != $acnt\_type) {

$error\_string="Wrong type!";

} else {

$error\_string="Wrong User name or password!";

} } } }

?>

**6.7 Job listing:**

<?php

ini\_set('session.cache\_limiter','public');

session\_cache\_limiter(false);

session\_start();

include("connection.php");

include("functions.php");

$user\_data = check\_login($con);

$error\_string = "";

$count = 0;

$query4 = "update jobs set selected = b'0' where selected is true";

if(!(mysqli\_query($con,$query4))){

$error\_string = "Some error in updating";

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails1']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn1'])){

$jnum = $\_POST['btn1'];

$query2 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query2))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails2']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn2'])){

$jnum = $\_POST['btn2'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails3']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn3'])){

$jnum = $\_POST['btn3'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails4']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn4'])){

$jnum = $\_POST['btn4'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails5']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn5'])){

$jnum = $\_POST['btn5'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails6']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn6'])){

$jnum = $\_POST['btn6'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails7']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn7'])){

$jnum = $\_POST['btn7'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['viewdetails8']))

{

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['btn8'])){

$jnum = $\_POST['btn8'];

$query3 = "UPDATE jobs set selected = b'1' where job\_num='$jnum'";

if(mysqli\_query($con,$query3))

{

header("Location:job-details.php");

} } }

if(($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['sort\_btn']))||!($\_SESSION['filter'] == "0000"))

{

$check11 = 0; $check12 = 0;

$check31 = 0; $check32 = 0;

$jtype1 = "fu"; $jtype2 = "pa";

$edu\_level1 = "ug"; $edu\_level2 = "pg";

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['check11']))

{

$check11 = 1;

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['check12']))

{

$check12 = 1;

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['check31']))

{

$check31 = 1;

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['check32']))

{

$check32 = 1;

}

if(($check11 == 1 && $check12 == 1 && $check31 == 1 && $check32 == 1 )||($\_SESSION['filter'] == "1111")){

$sql = "SELECT \* FROM jobs";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1111";

$sql = "SELECT \* FROM jobs ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 1 && $check31 == 1 && $check32 == 0)||($\_SESSION['filter'] == "1110")){

$sql="select \* from jobs where qualifications = '$edu\_level1'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1110";

$sql="select \* from jobs where qualifications = '$edu\_level1' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 1 && $check31 == 0 && $check32 == 1)||($\_SESSION['filter'] == "1101")){

$sql="select \* from jobs where qualifications = '$edu\_level2'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1101";

$sql="select \* from jobs where qualifications = '$edu\_level2' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 1 && $check31 == 0 && $check32 == 0)||($\_SESSION['filter'] == "1100")){

$sql="select \* from jobs";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1100";

$sql="select \* from jobs ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 0 && $check31 == 1 && $check32 == 1)||($\_SESSION['filter'] == "1011")){

$sql="select \* from jobs where job\_type LIKE '%$jtype1%'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1011";

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 0 && $check31 == 1 && $check32 == 0)||($\_SESSION['filter'] == "1010")){

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' AND qualifications = '$edu\_level1'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1010";

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' AND qualifications = '$edu\_level1' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 0 && $check31 == 0 && $check32 == 1)||($\_SESSION['filter'] == "1001")){

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' AND qualifications = '$edu\_level2'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1001";

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' AND qualifications = '$edu\_level2' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 1 && $check12 == 0 && $check31 == 0 && $check32 == 0)||($\_SESSION['filter'] == "1000")){

$sql="select \* from jobs where job\_type LIKE '%$jtype1%'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "1000";

$sql="select \* from jobs where job\_type LIKE '%$jtype1%' ORDER BY job\_num desc";

$result = $con->query($sql);

}

else if(($check11 == 0 && $check12 == 1 && $check31 == 1 && $check32 == 1)||($\_SESSION['filter'] == "0111")){

$sql="select \* from jobs where job\_type LIKE '%$jtype2%'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0111";

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 1 && $check31 == 1 && $check32 == 0)||($\_SESSION['filter'] == "0110")){

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' AND qualifications = '$edu\_level1'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0110";

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' AND qualifications = '$edu\_level1' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 1 && $check31 == 0 && $check32 == 1)||($\_SESSION['filter'] == "0101")){

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' AND qualifications = '$edu\_level2'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0101";

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' AND qualifications = '$edu\_level2' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 1 && $check31 == 0 && $check32 == 0)||($\_SESSION['filter'] == "0100")){

$sql="select \* from jobs where job\_type LIKE '%$jtype2%'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0100";

$sql="select \* from jobs where job\_type LIKE '%$jtype2%' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 0 && $check31 == 1 && $check32 == 1)||($\_SESSION['filter'] == "0011")){

$sql="select \* from jobs";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0011";

$sql="select \* from jobs";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 0 && $check31 == 1 && $check32 == 0)||($\_SESSION['filter'] == "0010")){

$sql="select \* from jobs WHERE qualifications = '$edu\_level1'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$sql));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0010";

$sql="select \* from jobs WHERE qualifications = '$edu\_level1' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

else if(($check11 == 0 && $check12 == 0 && $check31 == 0 && $check32 == 1)||($\_SESSION['filter'] == "0001")){

$query = "select \* from jobs WHERE qualifications = '$edu\_level2'";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$query));

$maxcounter = ceil($cnt/8);

$\_SESSION['filter'] = "0001";

$sql="select \* from jobs WHERE qualifications = '$edu\_level2' ORDER BY job\_num desc";

$result = $con->query($sql);

if(!($result)){

$error\_string = $con->error;

} }

}else

{

$query = "SELECT \* FROM jobs";

$cnt = mysqli\_num\_rows(mysqli\_query($con,$query));

$maxcounter = ceil($cnt/8);

$sql = "SELECT \* FROM jobs ORDER BY job\_num desc";

$result = $con->query($sql);

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['nxt\_btn'])){

if(!isset($\_SESSION['already\_refreshed'])){

$\_SESSION['already\_refreshed'] = true;

header("refresh: 0;");

} }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['reset\_btn'])){

$\_SESSION['filter'] = "0000";

header("Location: job-listing.php");

}

?>

**6.8 Job details:**

<?php

session\_start();

include("connection.php");

include("functions.php");

$error\_txt = "";

$user\_data = check\_login($con);

$title = "No Data"; $loc = "No Data"; $sal = "0"; $details = "No Data"; $id = 0;

$query1 = "select \* from jobs where selected is true";

$result1 = mysqli\_query($con,$query1);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['apply\_btn']))

{

$uname = $user\_data['user\_name'];

$query3 = "select user\_id from users where user\_name='$uname'";

$result3 = mysqli\_query($con,$query3);

$row3 = mysqli\_fetch\_assoc($result3);

$uid = $row3['user\_id'];

$query4= "select job\_num from jobs where selected is true";

$result4 = mysqli\_query($con,$query4);

$row4 = mysqli\_fetch\_assoc($result4);

$jno = $row4['job\_num'];

$query7 = "SELECT \* from aspirants where ano='$uid' AND job\_num='$jno'";

$result7 = mysqli\_query($con,$query7);

if(mysqli\_num\_rows($result7) != 0){ $error\_txt = "You have already applied for this job";

}else{

$query5 = "insert into aspirants (ano,job\_num) values ('$uid','$jno')";

$result5 = mysqli\_query($con,$query5);

if($result5){

echo "DONE!";

$\_SESSION['source'] = 'japplied';

header("Location:success.php");

}else{

$error\_txt = "Some error in inserting!";

} } }

?>

**6.9 Add jobs (Employer):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$error\_string = "";

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='employer'))

{

$\_SESSION['error\_src'] = 'emp';

header("Location:noaccountalert.php");

}

if($user\_data['valid'] == false){

header("Location:empnotverified.php");

}

if($\_SERVER['REQUEST\_METHOD'] == "POST")

{

$jtitle = $\_POST['jtitle'];

$location = $\_POST['loc'];

$jtype;

if($\_POST['jtype'] == "full\_time"){$jtype="full\_time";}else if($\_POST['jtype'] == "part\_time"){$jtype="part\_time";}

$qual = $\_POST['qualification'];

$graduation = $\_POST['degree'];

$specialisation = $\_POST['spex'];

$salary = $\_POST['sal'];

$jdetails = $\_POST['jdetails'];

$date = $\_POST['ldate'];

$comp\_details = $\_POST['comp\_details'];

if(!empty($jtitle) && !empty($location) && !empty($jdetails) && is\_numeric($salary)){

$eno = $\_SESSION['user\_id'];

$query = "insert into jobs (ENO,JOB\_TITLE,JOB\_DETAILS,LOCATION,SALARY,job\_type,qualifications,graduation,spex,last\_date,about\_comp) values ('$eno','$jtitle','$jdetails','$location','$salary','$jtype','$qual','$graduation','$specialisation','$date','$comp\_details')";

if($res = mysqli\_query($con, $query)){

echo "DONE";

$\_SESSION['source'] = 'jadded';

header("Location: success.php");

}else{echo $con->error;}

}else{$error\_string = "Please enter some valid information!";}

}

?>

**6.10 View jobs (Employer):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='employer'))

{

$\_SESSION['error\_src'] = 'emp';

header("Location:noaccountalert.php");

}

if($user\_data['valid'] == false){

header("Location:empnotverified.php"); }

$username = "";

$id = "";

$mail = "";

$qualification= "";

$select\_id = "";

$errortxt = "";

$eno = $user\_data['user\_id'];

$sql = "SELECT eno,job\_title,job\_num,job\_details,location,salary,job\_type,qualifications,last\_date FROM jobs where eno='$eno'";

$result = $con->query($sql);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_id']))

{ $select\_id = $\_POST['del\_id']; }

if($

\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_acnt']))

{

$sql2 = "SELECT \* from aspirants where job\_num = '$\_POST[del\_id]'";

$result2 = mysqli\_query($con,$sql2);

if(mysqli\_num\_rows($result2) != 0){

$sql3 = "DELETE FROM aspirants WHERE job\_num = '$\_POST[del\_id]'";

if (mysqli\_query($con, $sql3)) {

$sql4 = "SELECT \* from jobs WHERE job\_num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql = "DELETE FROM jobs WHERE job\_num='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

}else{ $errortxt = "Error deleting record: " . $con->error; }

}else{$errortxt= "No Such Records!";}

}else{echo "error in deleting aspirants..".$con->error;}

}else{

$sql4 = "SELECT \* from jobs WHERE job\_num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql = "DELETE FROM jobs WHERE job\_num='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

}else{ $errortxt = "Error deleting record: " . $con->error; }

}else{$errortxt= "No Such Records!";} } }

?>

**6.11 View aspirants (Employer):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$ano;

$job\_num;

$count = 1;

$errortxt = "";

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='employer'))

{

$\_SESSION['error\_src'] = 'emp';

header("Location:noaccountalert.php");

}

if($user\_data['valid'] == false){

header("Location:empnotverified.php");

}

$eno = $user\_data['user\_id'];

$sql1 = "SELECT job\_num,job\_title from jobs where eno='$eno'";

$result1 = mysqli\_query($con,$sql1);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_id'])) {

$select\_id = $\_POST['del\_id'];

$\_SESSION['sel\_id'] = $select\_id;

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_acnt'])) {

$sql4 = "SELECT \* from aspirants WHERE num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql = "DELETE FROM aspirants WHERE num='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) { $errortxt = "Record deleted successfully";

} else { $errortxt = "Error deleting record: " . $con->error; }

}else{$errortxt= "No Such Records!";}

}

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['view\_doc']))

{

$sql4 = "SELECT \* from aspirants WHERE num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

header("Location:view.php");

}else{$errortxt= "No Such Records!";}

}

?>

**6.12 View aspirants (Admin):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:noaccountalert.php"); }

$username = "";

$id = "";

$mail = "";

$count = 1;

$qualification= "";

$select\_id = "";

$errortxt = "";

$sql = "SELECT user\_name,user\_id,email FROM users where acnt\_type='aspirant'";

$result = $con->query($sql);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_id']))

{ $select\_id = $\_POST['del\_id']; }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_acnt']))

{

$sql4 = "SELECT \* from aspirants WHERE ano='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql6 = "SELECT \* from aspirants where ano='$\_POST[del\_id]'";

$result6 = mysqli\_query($con,$sql6);

if(mysqli\_num\_rows($result6) != 0){

$errortxt = "Error! Delete the Job posted to this account and then delete account.";

}else{

$sql = "DELETE FROM users WHERE user\_id='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

} else { $errortxt = "Error deleting record: "; } }

}else{$errortxt= "No Such Records!";} }

?>

**6.13 View employers (Admin):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:noaccountalert.php"); }

$username = "";

$id = "";

$mail = "";

$count = 1;

$qualification= "";

$select\_id = "";

$errortxt = "";

$sql = "SELECT user\_name,user\_id,email,valid FROM users where acnt\_type='employer' AND viewed = 1";

$result = $con->query($sql);

$sql2 = "SELECT user\_name,user\_id,email,valid FROM users where acnt\_type='employer' AND viewed = 0";

$result2 = $con->query($sql2);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_id']))

{ $select\_id = $\_POST['del\_id']; }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_acnt'])) {

$sql6 = "SELECT \* from jobs where eno='$\_POST[del\_id]'";

$result6 = mysqli\_query($con,$sql6);

if(mysqli\_num\_rows($result6) != 0){

$errortxt = "Error! Delete the Job posted to this account and then delete account.";

}else{

$sql4 = "SELECT \* from users WHERE user\_id='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql = "DELETE FROM users WHERE user\_id='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

} else { $errortxt = "Error deleting record: "; }

}else{$errortxt= "No Such Records!";} } }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['view\_btn'])){

$\_SESSION['sel\_id'] = $select\_id;

$sql4 = "SELECT \* from users WHERE user\_id='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

header("Location:viewdocemp.php");

$sql5 = "UPDATE users set viewed = b'1' where acnt\_type='employer' AND viewed = 0";

if (mysqli\_query($con,$sql5)) { $errortxt = "Updated";

} else

{ $errortxt = "Error updating: "; }

}else{$errortxt= "No Such Records!";}

}

?>

**6.14 View jobs (Admin):**

<?php

session\_start();

include("connection.php");

include("functions.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:noaccountalert.php"); }

$username = "";

$id = "";

$mail = "";

$qualification= "";

$select\_id = "";

$errortxt = "";

$sql = "SELECT job\_title,job\_num,eno FROM jobs";

$result = $con->query($sql);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_id']))

{ $select\_id = $\_POST['del\_id']; }

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['del\_acnt'])){

$sql2 = "SELECT \* from aspirants where job\_num = '$\_POST[del\_id]'";

$result2 = mysqli\_query($con,$sql2);

if(mysqli\_num\_rows($result2) != 0){

$sql4 = "SELECT \* from jobs WHERE job\_num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql3 = "DELETE FROM aspirants WHERE job\_num = '$\_POST[del\_id]'";

if (mysqli\_query($con, $sql3)) {

$sql = "DELETE FROM jobs WHERE job\_num='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

}else{ $errortxt = "Error deleting record: " . $con->error; }

}else{$errortxt= "No Such Records!";}

}else{echo "error in deleting aspirants..".$con->error;}

}else{

$sql4 = "SELECT \* from jobs WHERE job\_num='$\_POST[del\_id]'";

$result4 = mysqli\_query($con,$sql4);

if(mysqli\_num\_rows($result4) != 0) {

$sql = "DELETE FROM jobs WHERE job\_num='$\_POST[del\_id]'";

if (mysqli\_query($con, $sql)) {

$errortxt = "Record deleted successfully";

}else{ $errortxt = "Error deleting record: " . $con->error; }

}else{$errortxt= "No Such Records!";} } }

?>

**6.15 Feedback reply (Admin):**

<?php

session\_start();

include("../connection.php");

include("../functions.php");

include("../mail.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:../login.php");

}

if(!$\_SESSION['accounttype'] === 'admin'){

header("Location: noaccountalert.php");

}

$username = "";

$id = "";

$mail = "";

$qualification= "";

$select\_id = "";

$errortxt = "";

$ano;

$deftxt = "Your response means a lot.. Thank You!";

$sql = "SELECT \* FROM feedback";

$result = $con->query($sql);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['send\_tq'])){

if(isset($\_POST['emailer'])) {

$num = $\_POST['emailer'];

$query3="SELECT ano,emailid FROM feedback where num='$num'";

$result3=mysqli\_query($con,$query3);

$row3 = mysqli\_fetch\_assoc($result3);

if(mysqli\_num\_rows($result3) != 0) {

$ano = $row3['ano'];

$mailid = $row3['emailid'];

$query4 = "select user\_name,email from users where user\_id ='$ano'";

$result4 = mysqli\_query($con,$query4);

if($result4) {

$row4 = mysqli\_fetch\_assoc($result4);

$name = $row4['user\_name'];

$sub = "Thank You for your Feedback!";

if(isset($\_POST['reply\_content'])){

$bdy = $\_POST['reply\_content'];

}

mailer($mailid,$name,$sub,$bdy);

$query5 = "update feedback set viewed = b'1' where num='$num'";

$result5 = mysqli\_query($con,$query5);

if($result5){

header("Location:feedbackreply.php");

}else{ $errortxt = "Some error in updating!"; }

}else{ $errortxt = "No Resords with provided ID"; }

}else{$errortxt= "No Such Records!";}

} }

?>

**6.16 View new aspirant (Admin):**

<?php

session\_start();

include("../connection.php");

include("../functions.php");

include("../mail.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:../login.php");

}

$username = "";

$id = "";

$mail = "";

$qualification= "";

$select\_id = "";

$errortxt = "";

$query6 = "SELECT num,job\_num from aspirants where viewed is false";

$result6 = $con->query($query6);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['notify\_emp']))

{

while($row6=mysqli\_fetch\_assoc($result6)) {

$jnum = $row6['job\_num'];

$query7 = "SELECT eno from jobs where job\_num='$jnum'";

$result7 = mysqli\_query($con,$query7);

$row7 = mysqli\_fetch\_assoc($result7);

$eno = $row7['eno'];

$query8 = "SELECT user\_name,email from users where user\_id = '$eno'";

$result8 = mysqli\_query($con,$query8);

$row8 = mysqli\_fetch\_assoc($result8);

$mailid = $row8['email'];

$name =$row8['user\_name'];

$sub = "You Have new Applications!";

$bdy = "Visit our website to view the new applications for your job posting!";

mailer($mailid,$name,$sub,$bdy);

$query9 = "Update aspirants set viewed = b'1'";

$result9 = mysqli\_query($con,$query9);

if($result9)

{

$errortxt= "Updated";

}else{

$errortxt = "I believe some error in updation!";

} } }

?>

**6.17 View new job (Admin):**

<?php

session\_start();

include("../connection.php");

include("../functions.php");

include("../mail.php");

$user\_data = check\_login($con);

if(!($user\_data['acnt\_type']==='admin'))

{

$\_SESSION['error\_src'] = 'admin';

header("Location:../login.php");

}

$username = "";

$id = "";

$mail = "";

$qualification= "";

$select\_id = "";

$errortxt = "";

$sql = "SELECT job\_title,job\_num,eno FROM jobs where viewed is false";

$result = $con->query($sql);

if($\_SERVER['REQUEST\_METHOD'] == "POST" and isset($\_POST['notify']))

{

$query4 = "select user\_name,email from users where acnt\_type='aspirant'";

$result4 = $con->query($query4);

if($result4){

$row4 = mysqli\_fetch\_assoc($result4);

$mailid = $row4['email'];

$name = $row4['user\_name'];

$sub = "New Jobs Available!";

$bdy = "Visit our website to view the new jobs available!";

mailer($mailid,$name,$sub,$bdy);

$query5 = "select job\_num from jobs";

$result5 = $con->query($query5);

while($row5 = mysqli\_fetch\_assoc($result5)) {

$jno = $row5['job\_num'];

$query6 = "update jobs set viewed = b'1' where job\_num = '$jno'";

if($con->query($query6))

{

$errortxt = "Updated Table";

}else{

$errortxt = "I believe some error in updating!!";

} }

}else{echo $con->error;} }

?>

**6.18 Document viewer:**

<?php

session\_start();

include("connection.php");

$num = $\_SESSION['sel\_id'];

$query = "SELECT ano from aspirants where num='$num'";

$result= mysqli\_query($con,$query);

if($result){

$row = mysqli\_fetch\_assoc($result);

$id = $row['ano'];

$query2 = "SELECT \* from users where user\_id='$id'";

$result2 = mysqli\_query($con,$query2);

if($result2){

$row2 = mysqli\_fetch\_assoc($result2);

?>

<html>

<body>

<object data="data:application/pdf;base64,<?php echo base64\_encode($row2['conf\_doc']) ?>" type="application/pdf" style="height:720px;width:100%"></object>

</body>

</html>

<?php

}else{echo $con->error;}

}

?>

**6.19 Main Activity.java**

package com.example.job\_u\_go;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

getUrlFromIntent("http://192.168.43.28/jobugo/index.html");

} }); }

public void getUrlFromIntent(String url) {

Intent i = new Intent(Intent.ACTION\_VIEW);

i.setData(Uri.parse(url));

startActivity(i);

} }

**6.20 Activity main.xml**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<Button

android:id="@+id/button"

android:layout\_width="166dp"

android:layout\_height="41dp"

android:background="#FF6D00"

android:text="@string/explore"

android:textColor="#C51162"

android:textSize="20sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="@+id/imageView7"

app:layout\_constraintVertical\_bias="0.973" />

<ImageView

android:id="@+id/imageView7"

android:layout\_width="916dp"

android:layout\_height="1269dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.348"

app:srcCompat="@drawable/background"

android:contentDescription="@string/todo" />

<TextView

android:id="@+id/textView2"

android:layout\_width="208dp"

android:layout\_height="48dp"

android:fontFamily="sans-serif-smallcaps"

android:text="@string/welcome\_to"

android:textColor="#FFFFFF"

android:textSize="30sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toTopOf="@+id/button"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.945"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.083" />

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/new\_world\_of\_oppurtunities"

android:textColor="#FFFFFF"

android:textSize="20sp"

app:layout\_constraintBottom\_toTopOf="@+id/button"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/textView2"

app:layout\_constraintVertical\_bias="0.869" />

</androidx.constraintlayout.widget.ConstraintLayout>

**6.21 Android Manifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.job\_u\_go">

<uses-permission android:name="android.permission.INTERNET"/>

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**6.22 Testing:**

**6.22.1 Example Instrument Test.java (JUNIT)**

package com.example.job\_u\_go;

import android.content.Context;

import androidx.test.platform.app.InstrumentationRegistry;

import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;

import org.junit.runner.RunWith;

import static org.junit.Assert.\*;

/\*\*

\* Instrumented test, which will execute on an Android device.

\* @see <a href="http://d.android.com/tools/testing">Testing documentation</a>

\*/

@RunWith(AndroidJUnit4.class)

public class ExampleInstrumentedTest {

@Test

public void useAppContext() {

// Context of the app under test.

Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();

assertEquals("com.example.job\_u\_go", appContext.getPackageName());

} }

**6.23 Build Configuration**

package com.example.job\_u\_go;

public final class BuildConfig {

public static final boolean DEBUG = Boolean.parseBoolean("true");

public static final String APPLICATION\_ID = "com.example.job\_u\_go";

public static final String BUILD\_TYPE = "debug";

public static final int VERSION\_CODE = 1;

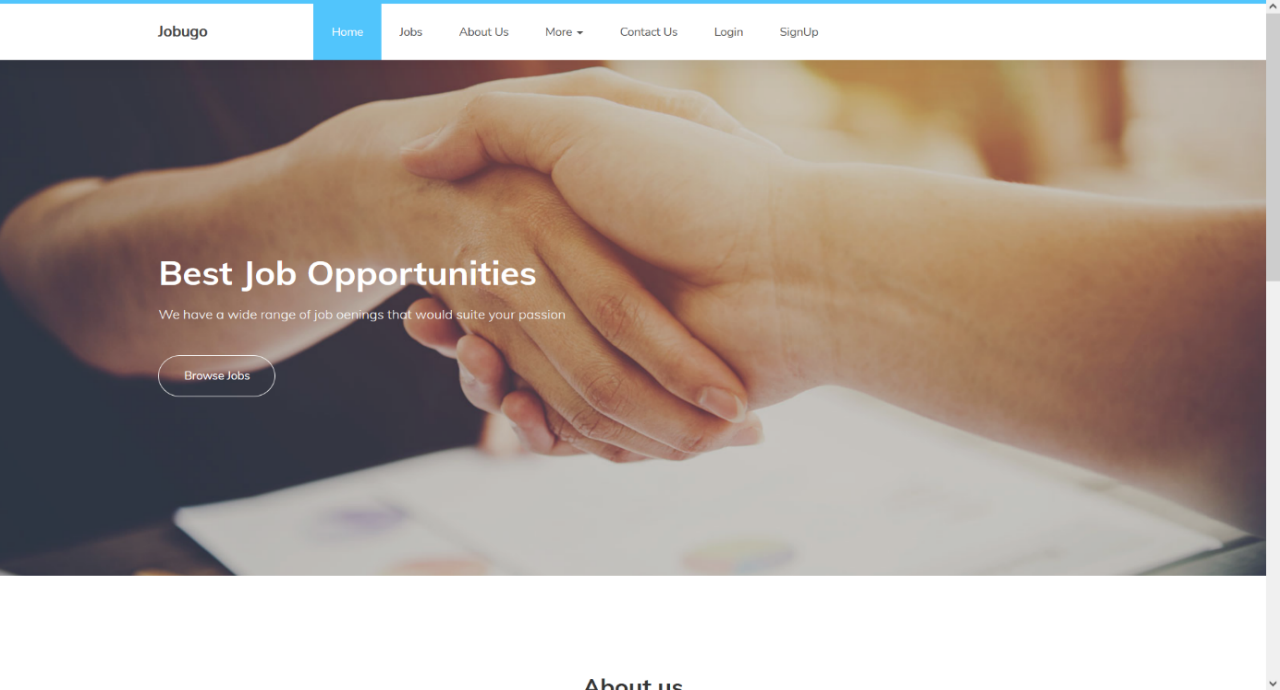
public static final String VERSION\_NAME = "1.0";

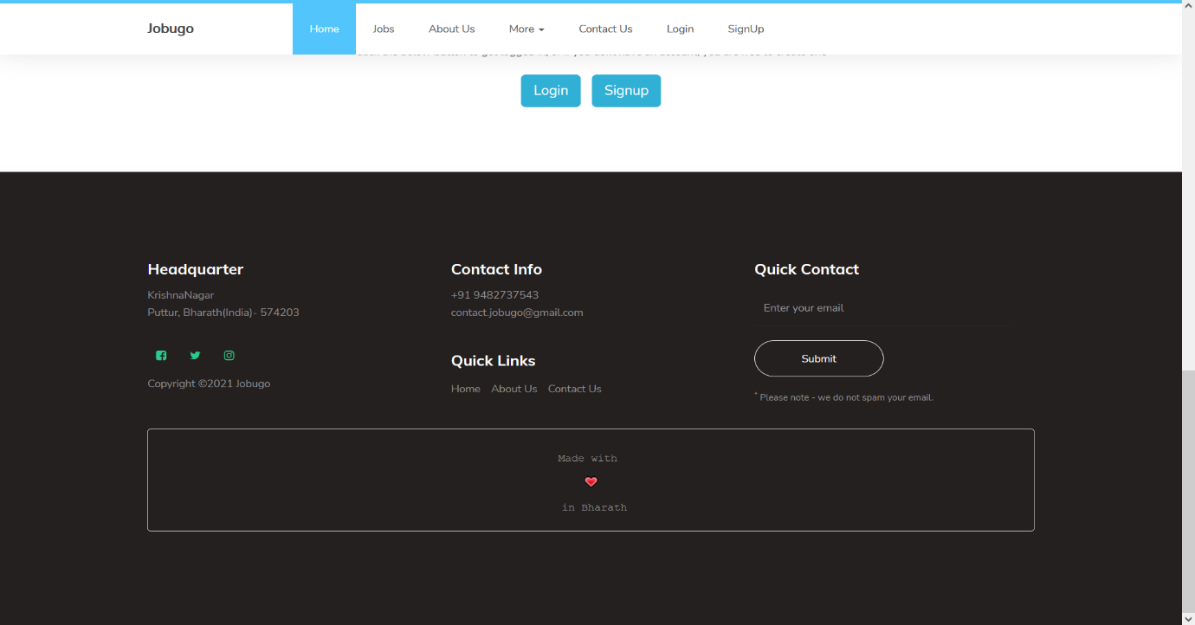
}

**CHAPTER – 7**

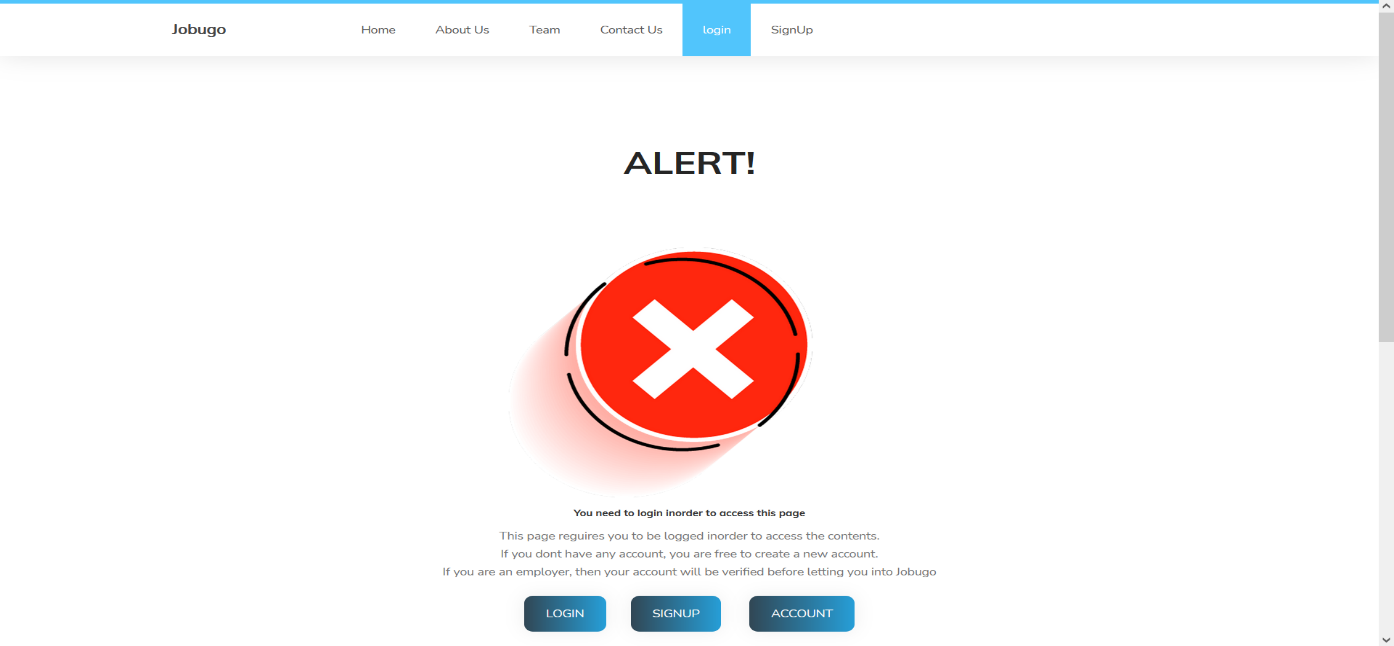
**USER INTERFACE**

**7.1 Main page**

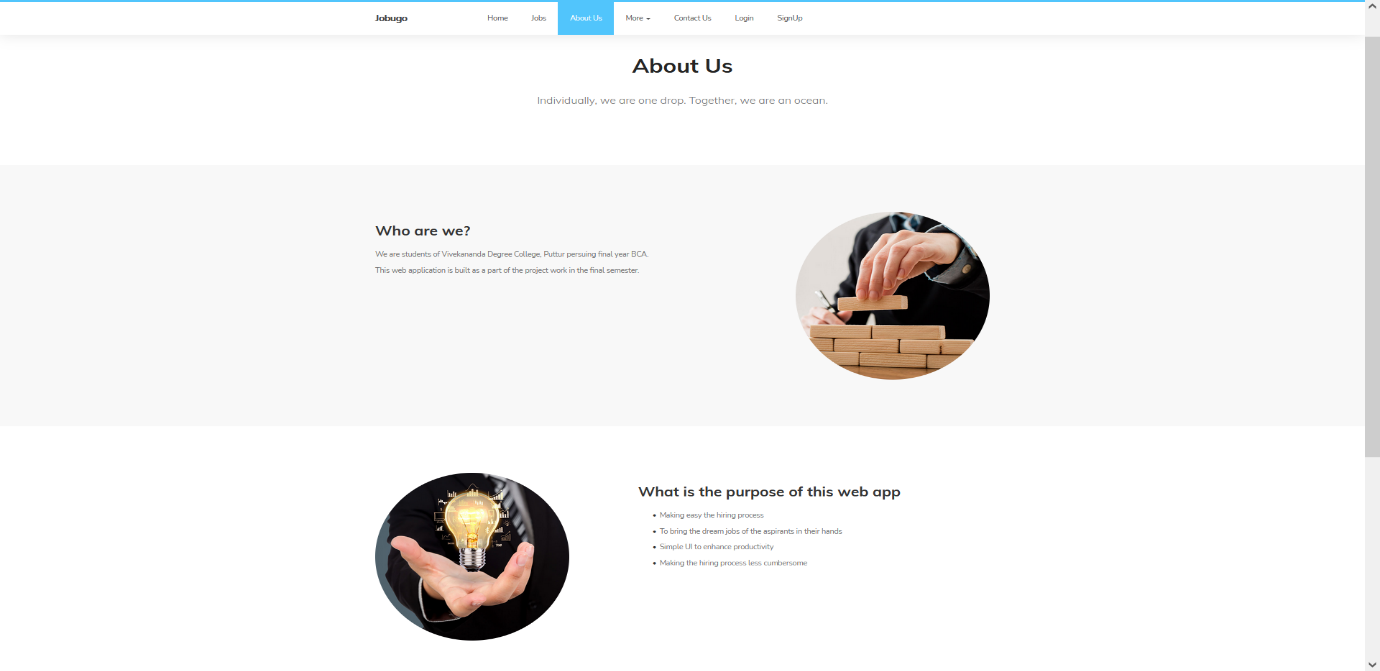




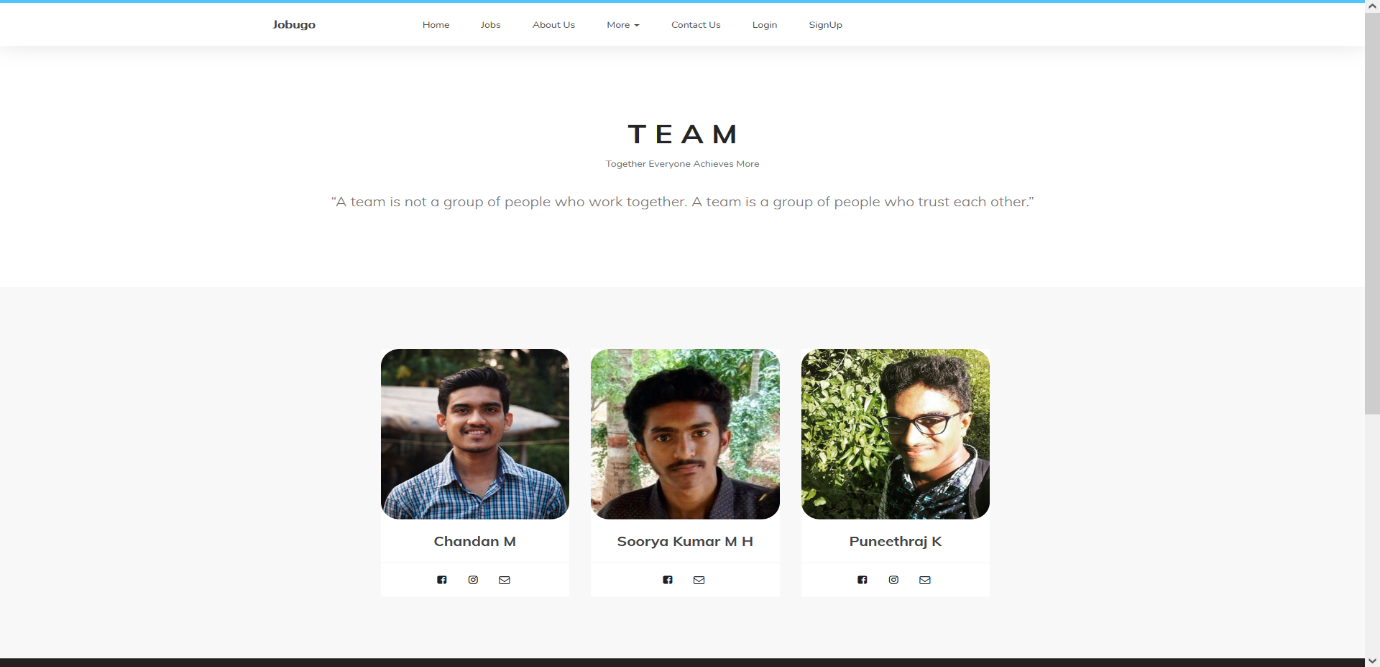
**7.2 User does not have account**



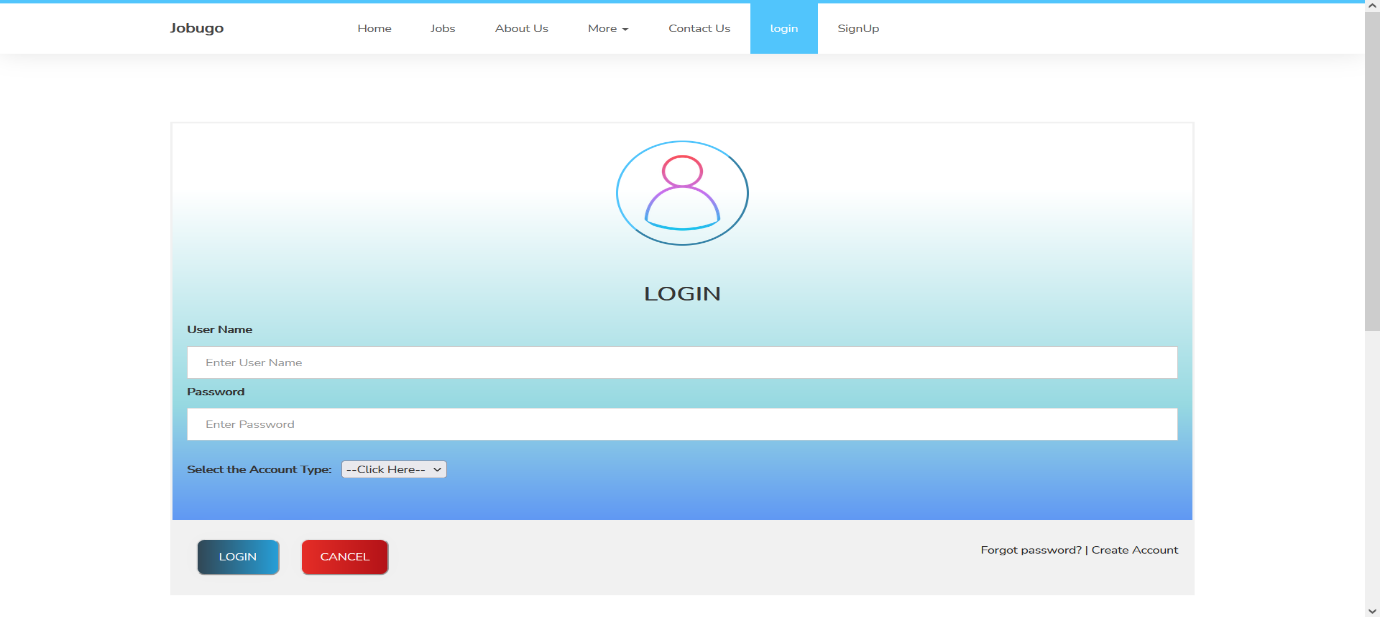
**7.3 About us**



**7.4 Team**

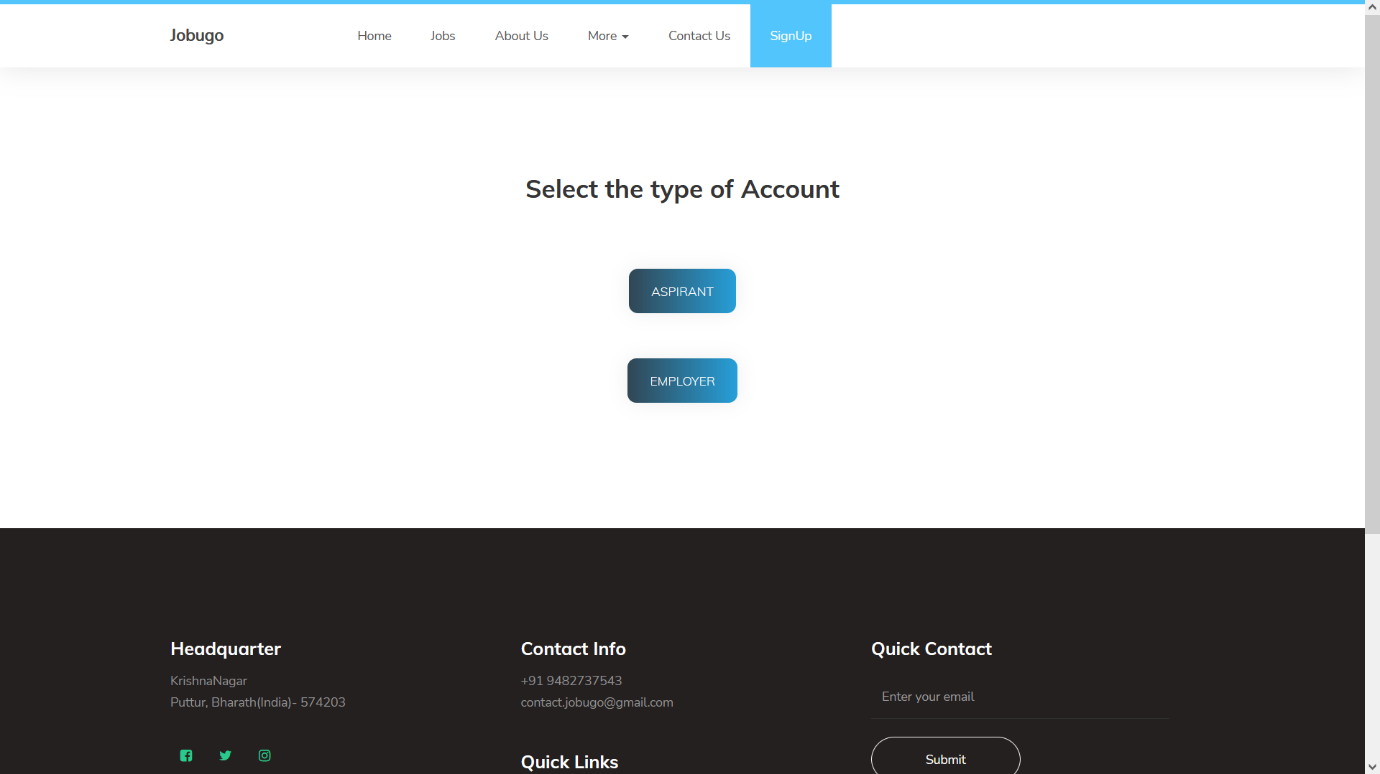


**7.5 Login page**

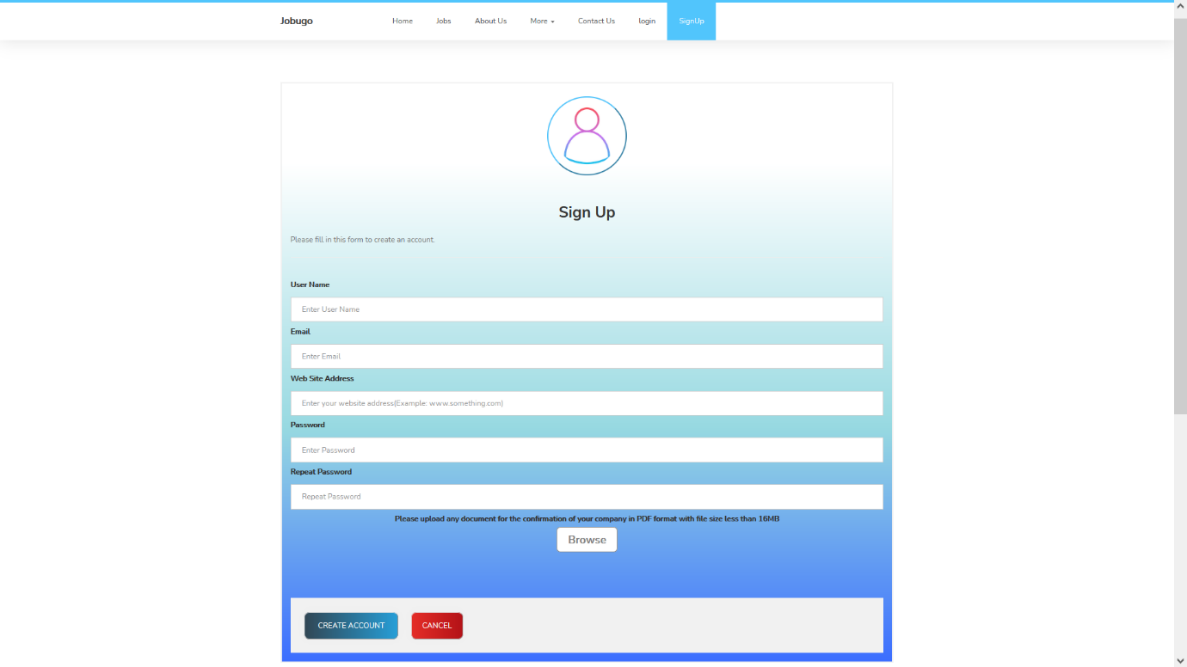


**7.6 Sign up**

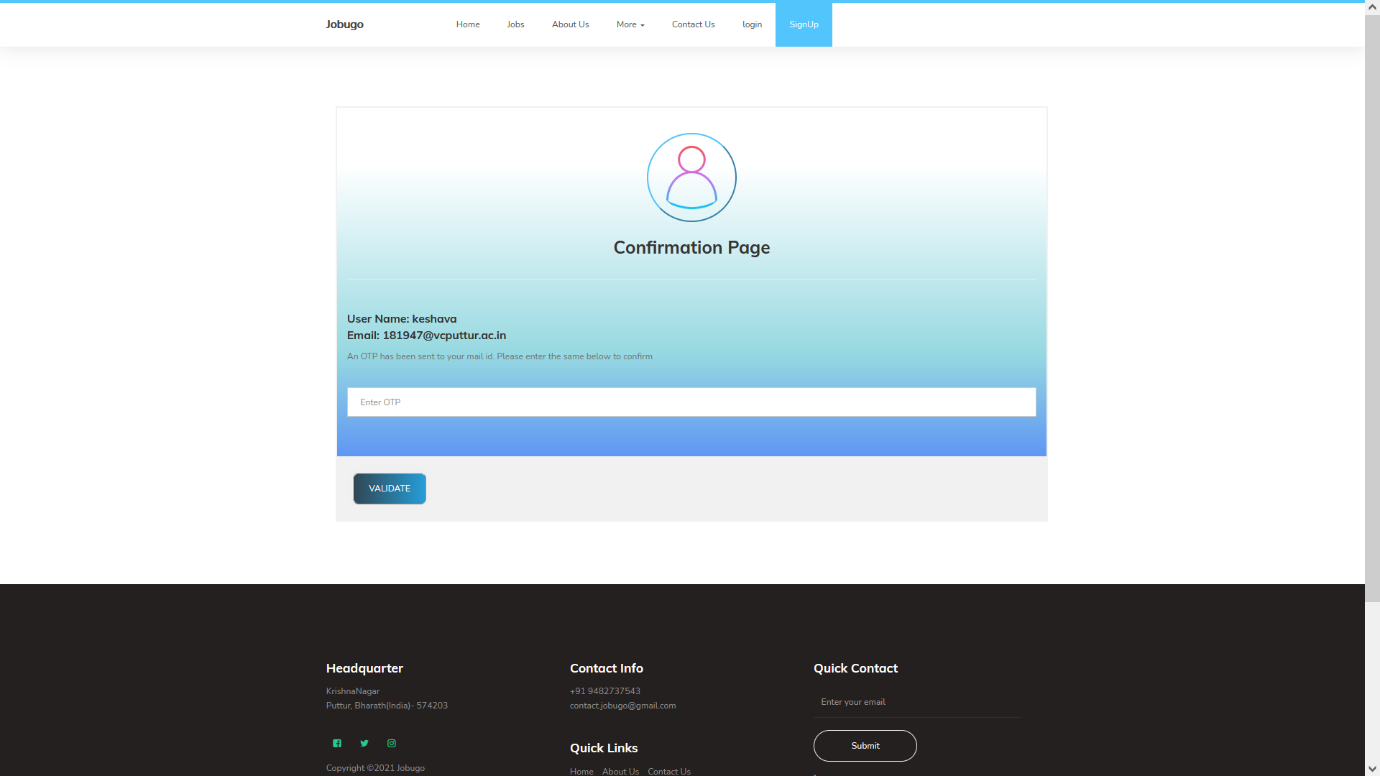
**7.6.1 Account type select**



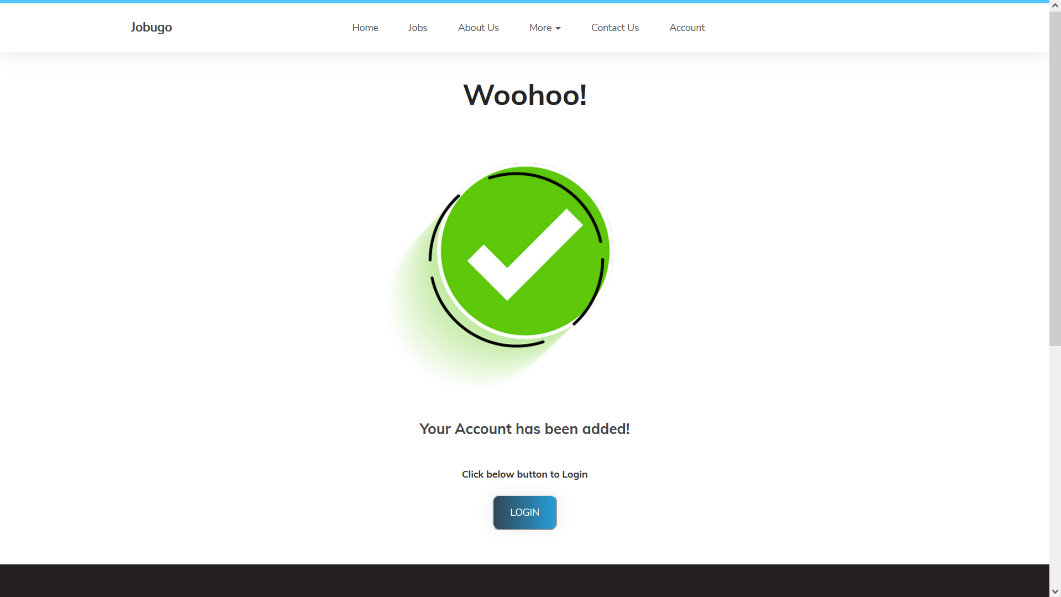
**7.6.2 Account credentials**



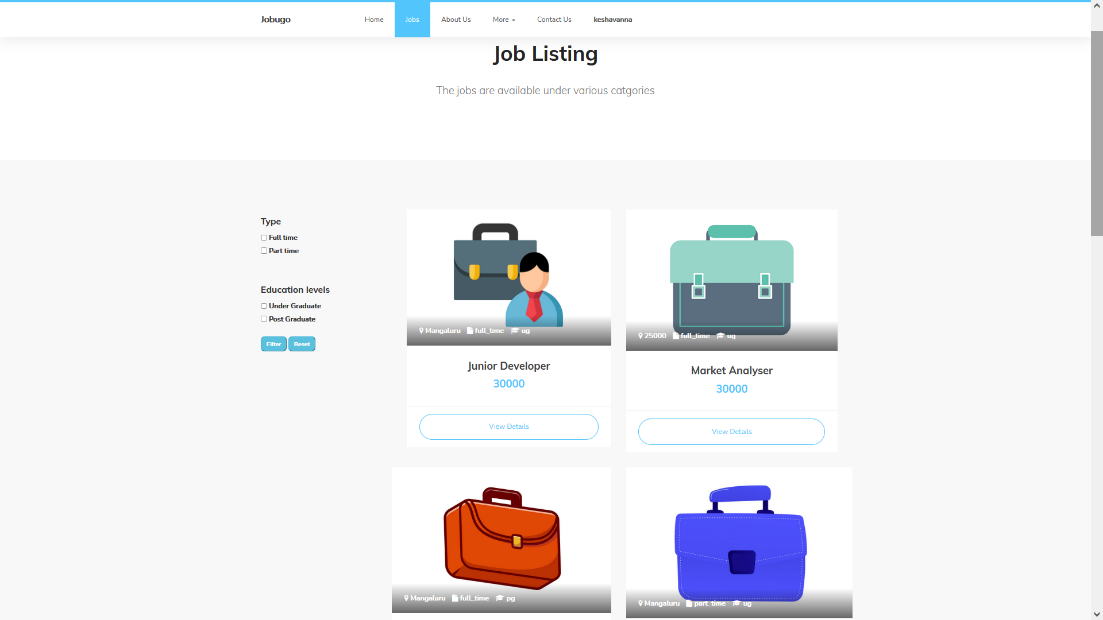
**7.6.3 OTP verification**



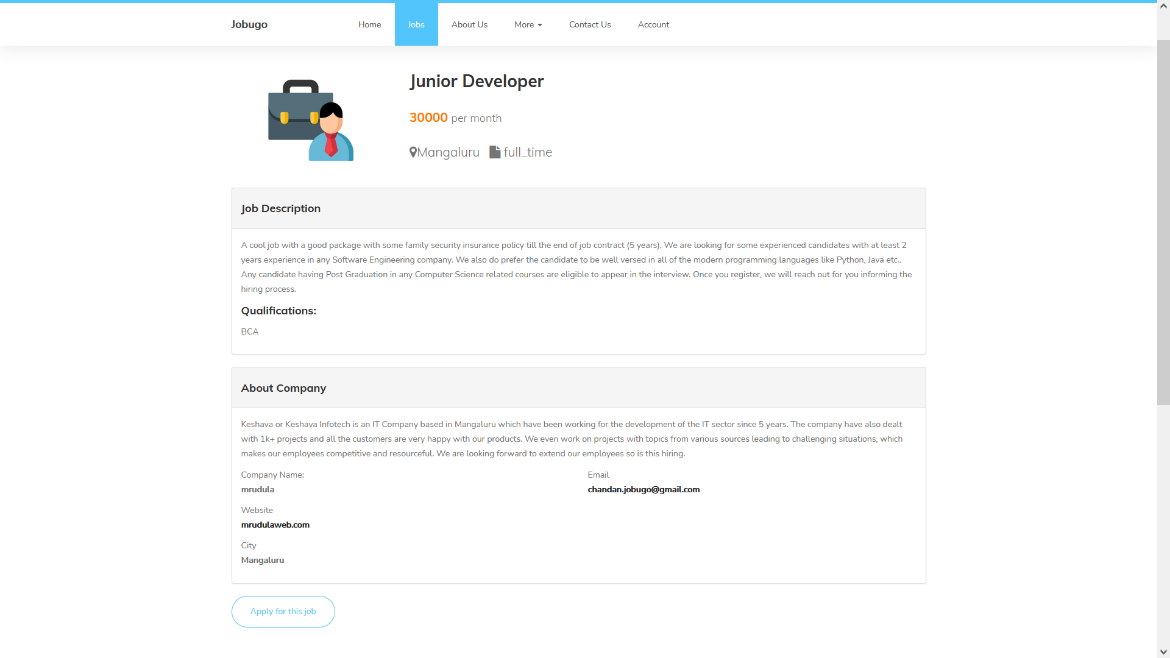
**7.6.4 Account added**



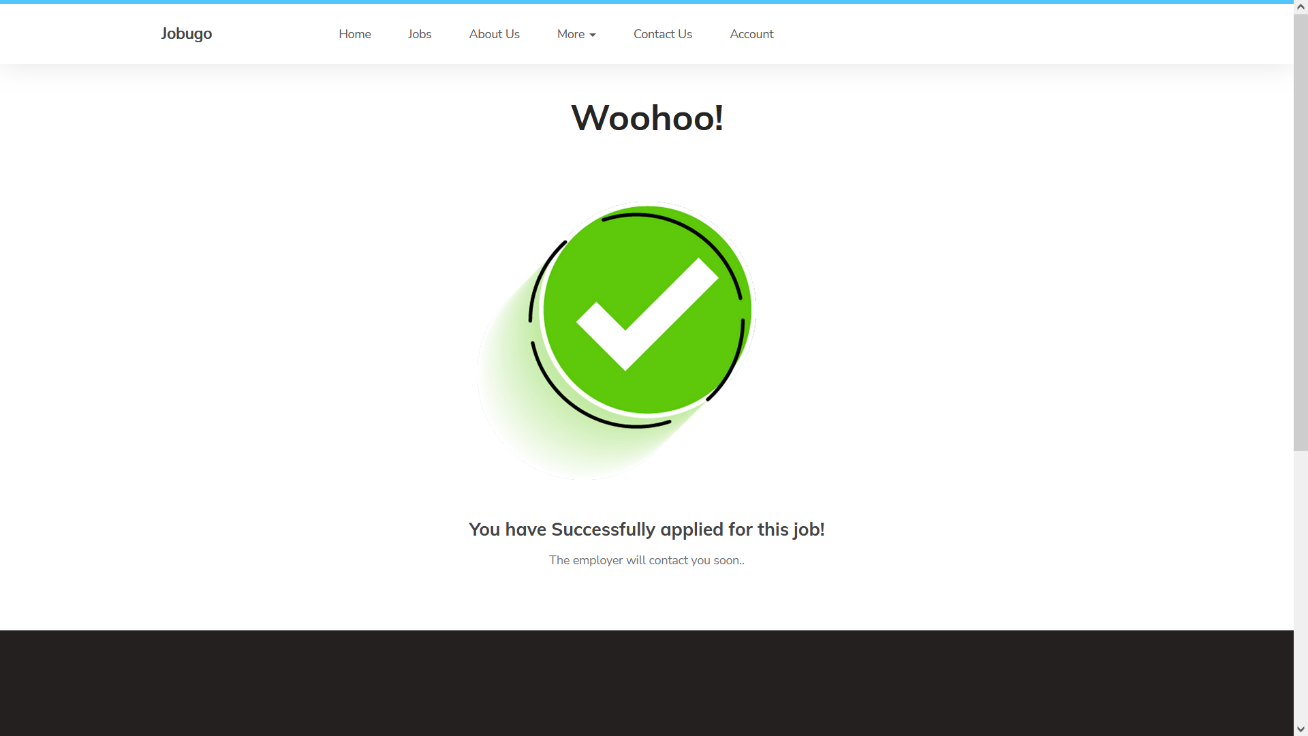
**7.7 Job listing**



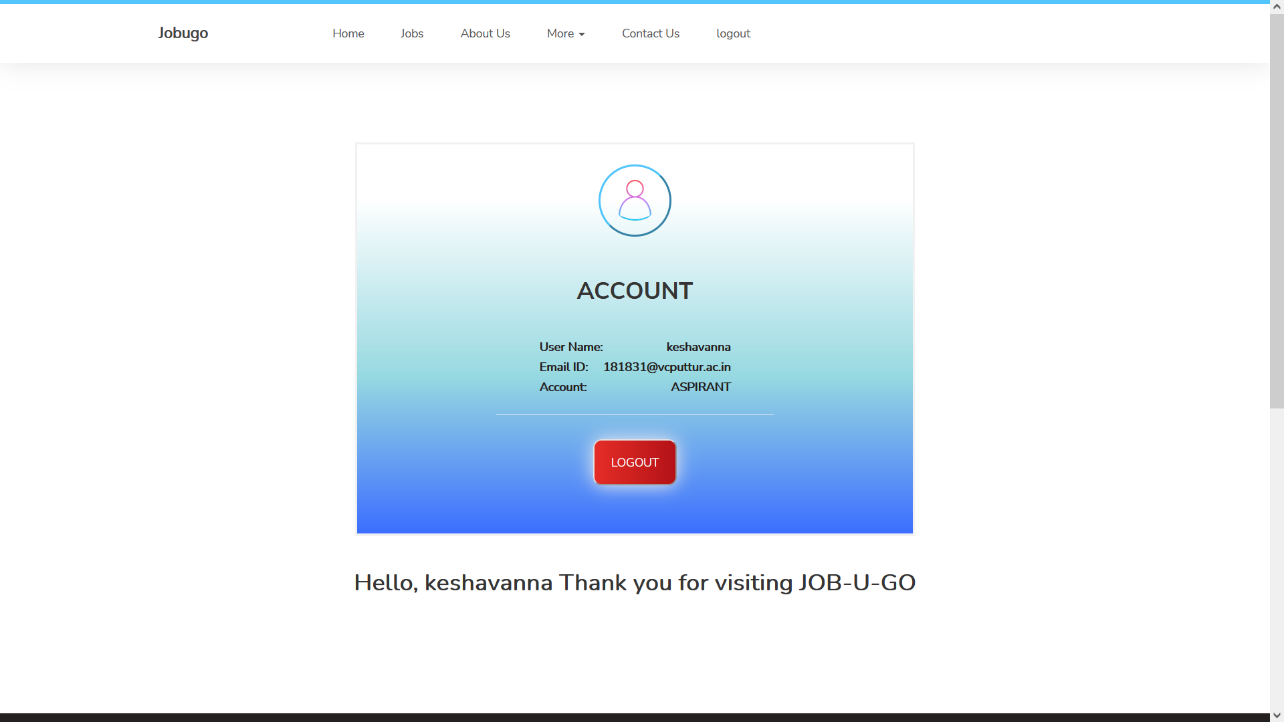
**7.8 Job description**



**7.9 Applied successfully**



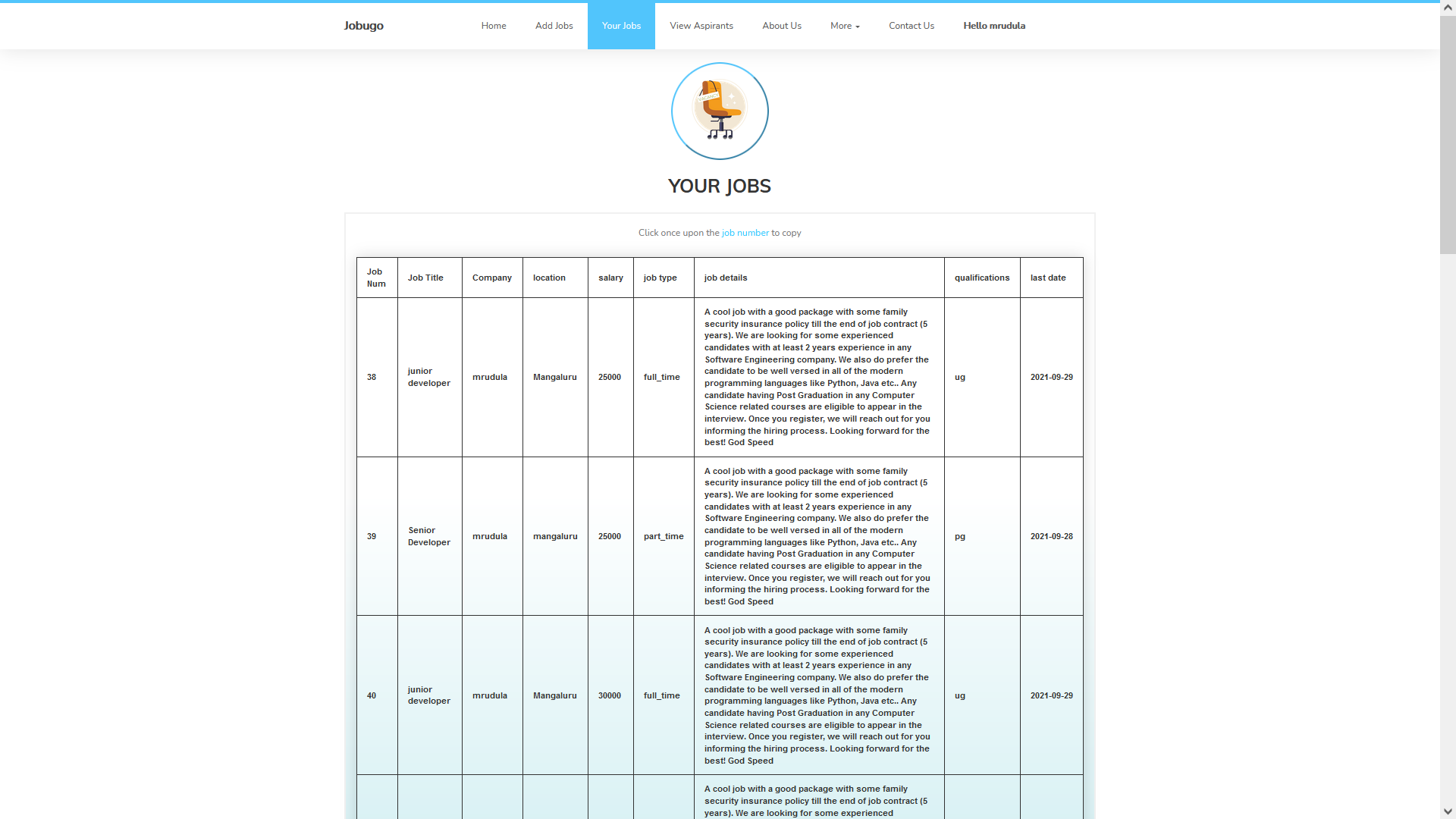
**7.10 Account**



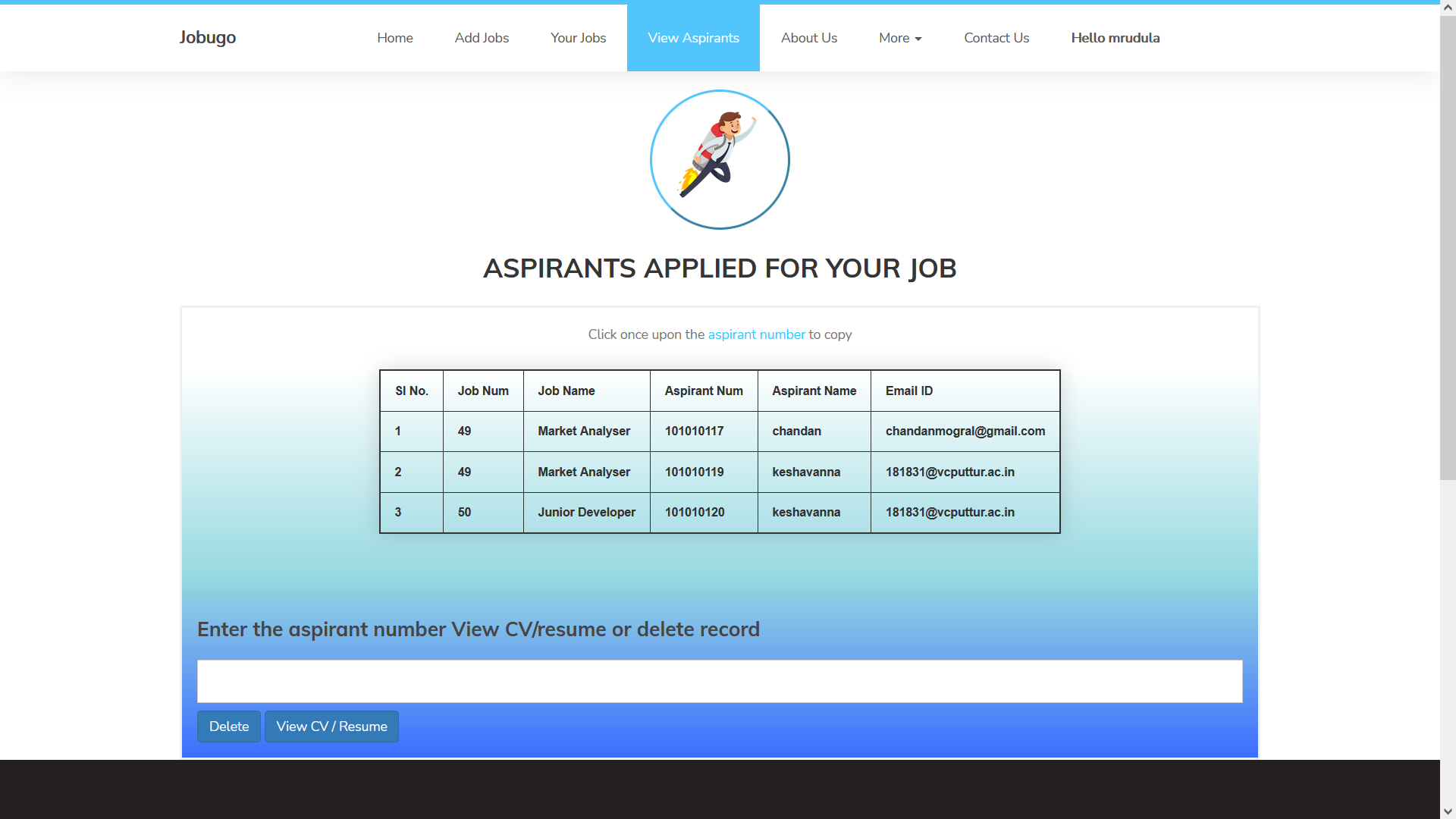
**7.11 Employer panel to add jobs**



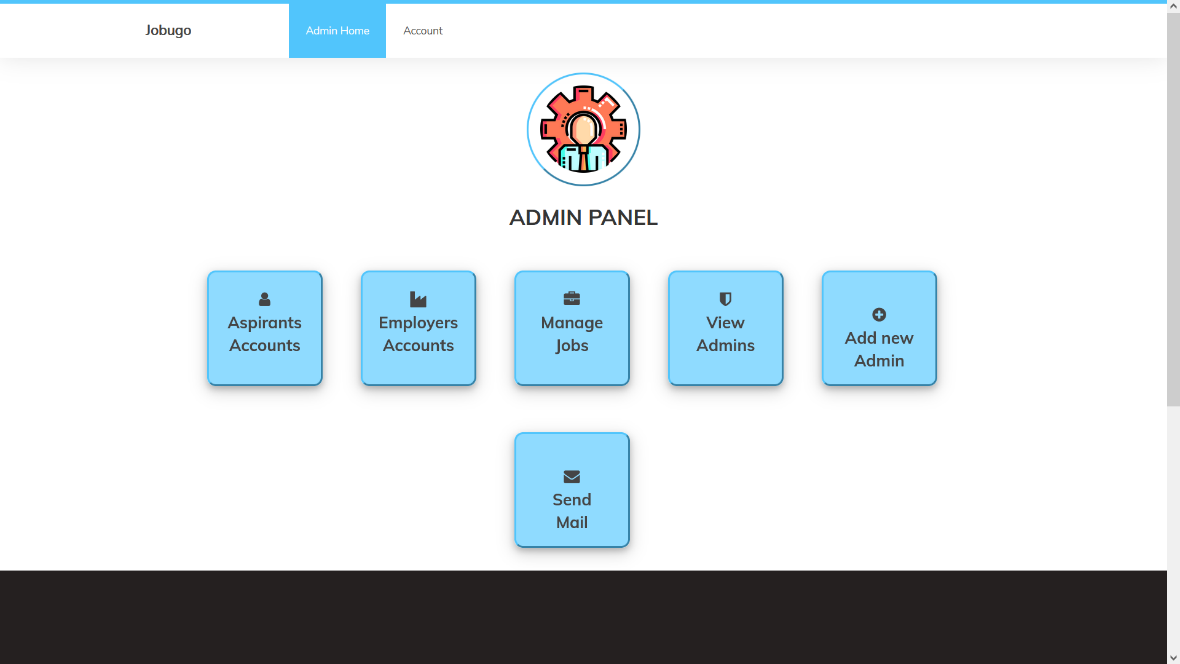
**7.12 Employer panel to view jobs**



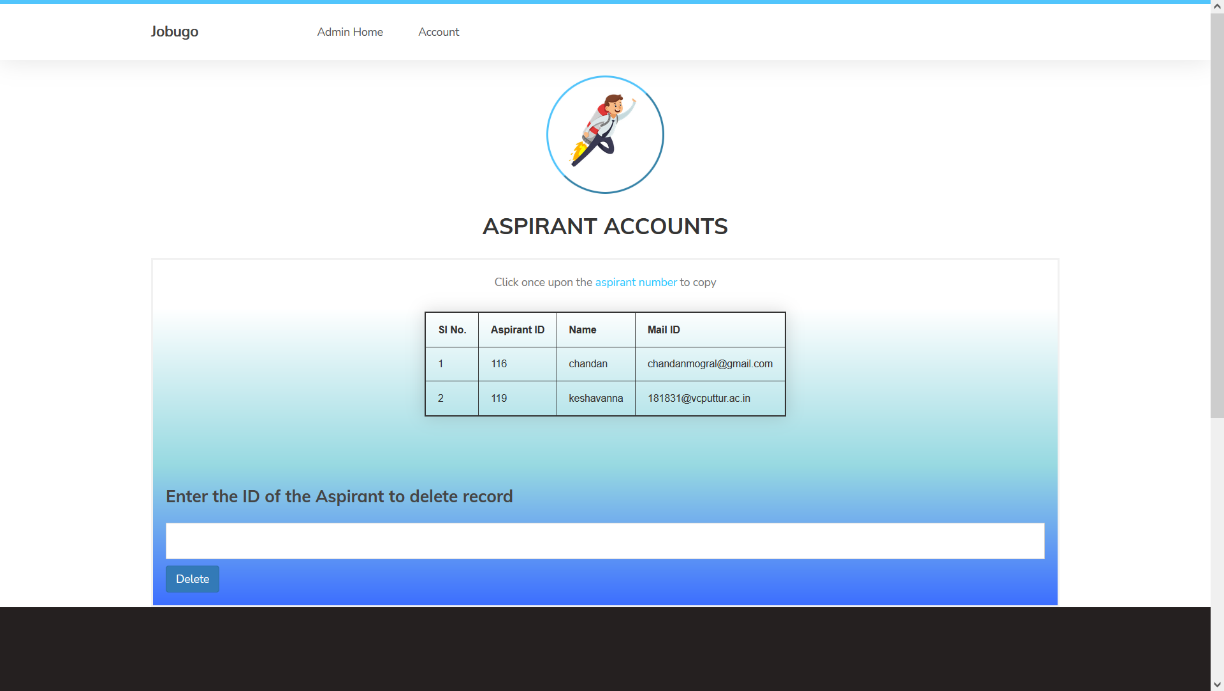
**7.13 View aspirants**



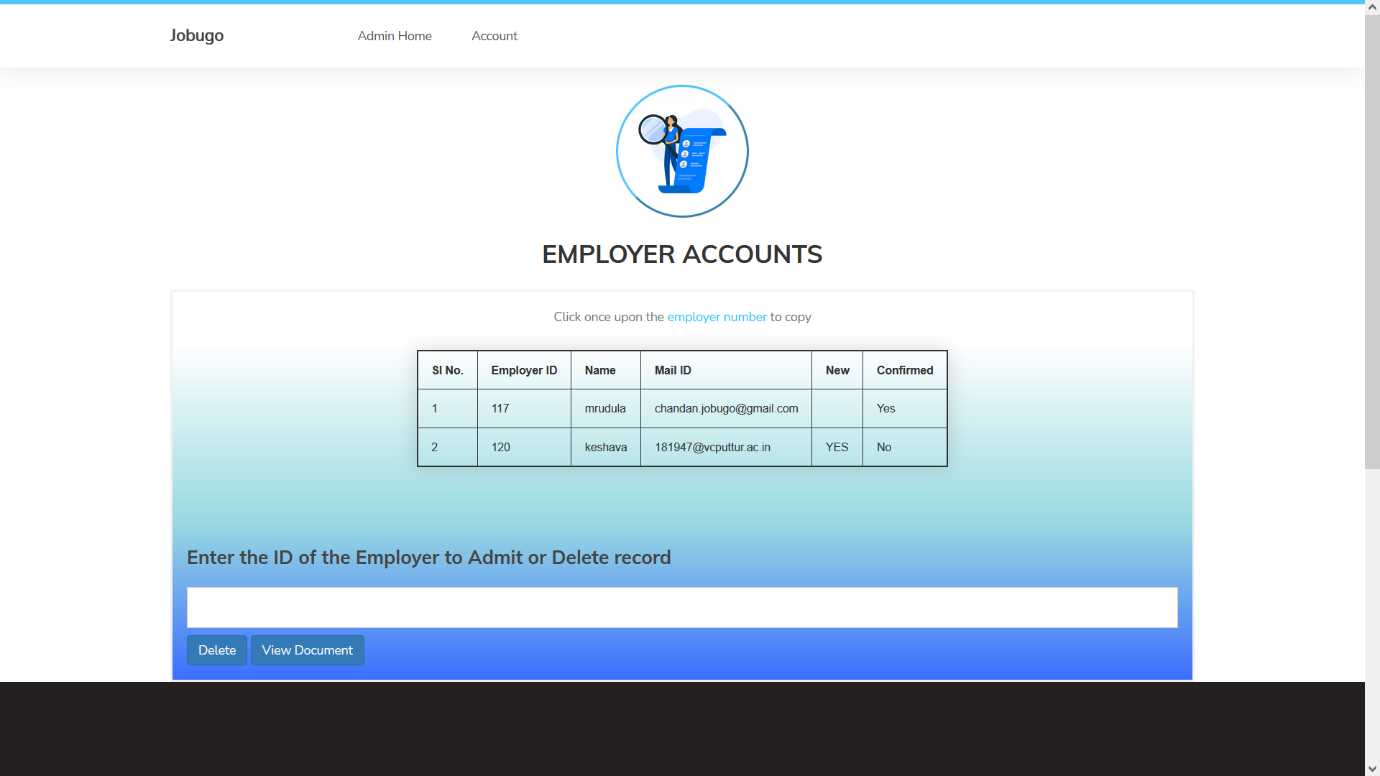
**7.14 Admin panel**



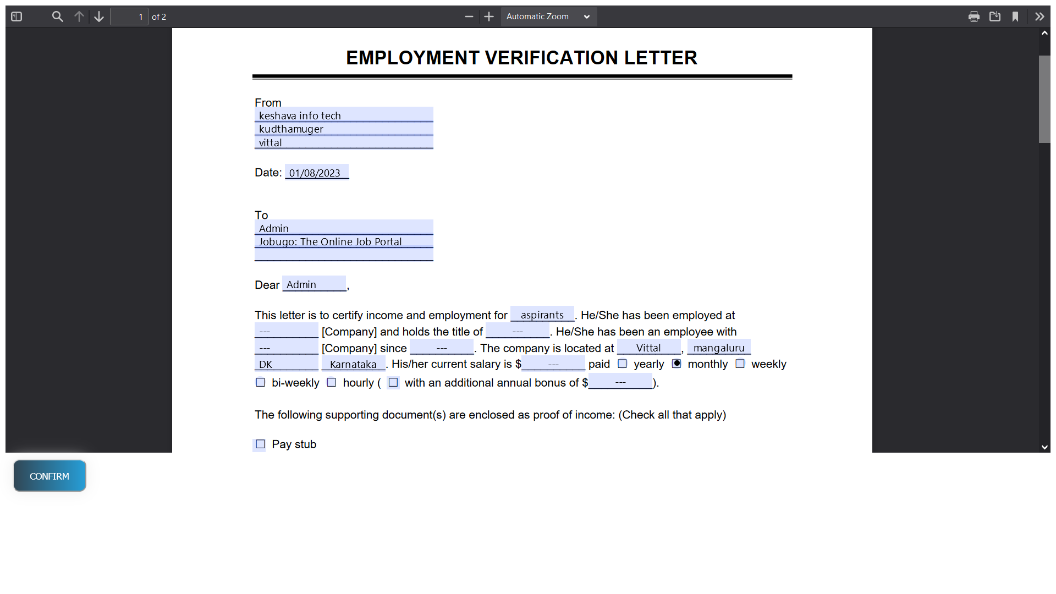
**7.15 Aspirant accounts (Admin panel)**



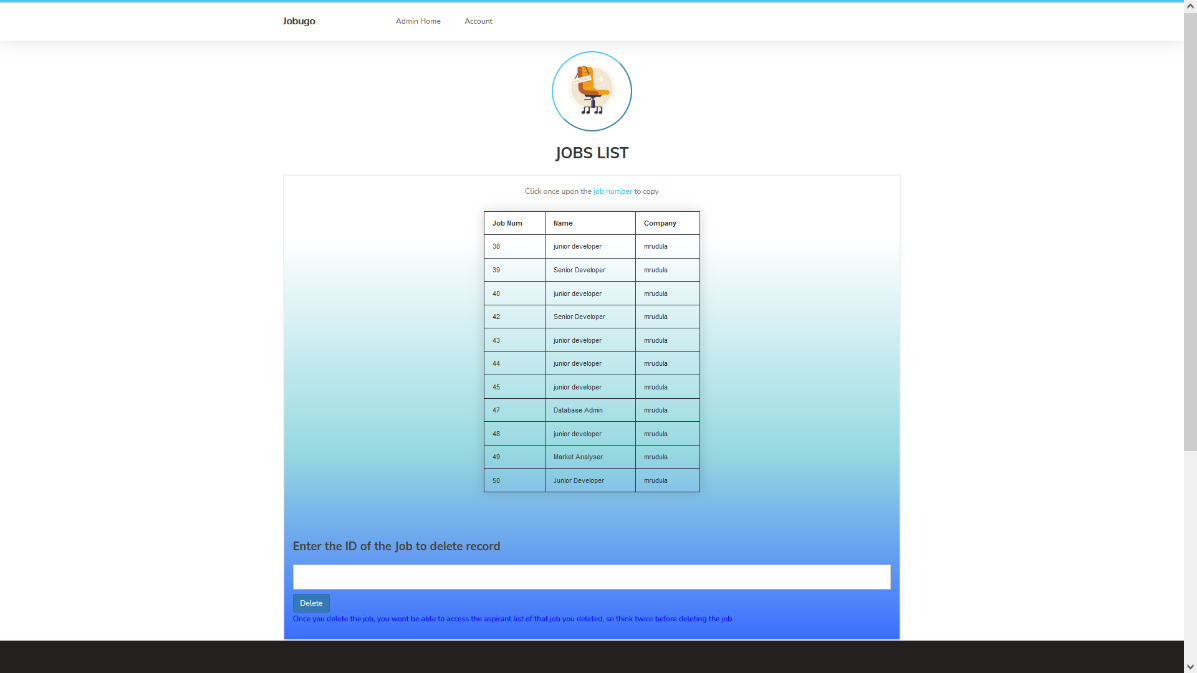
**7.16 Employer accounts (Admin panel)**



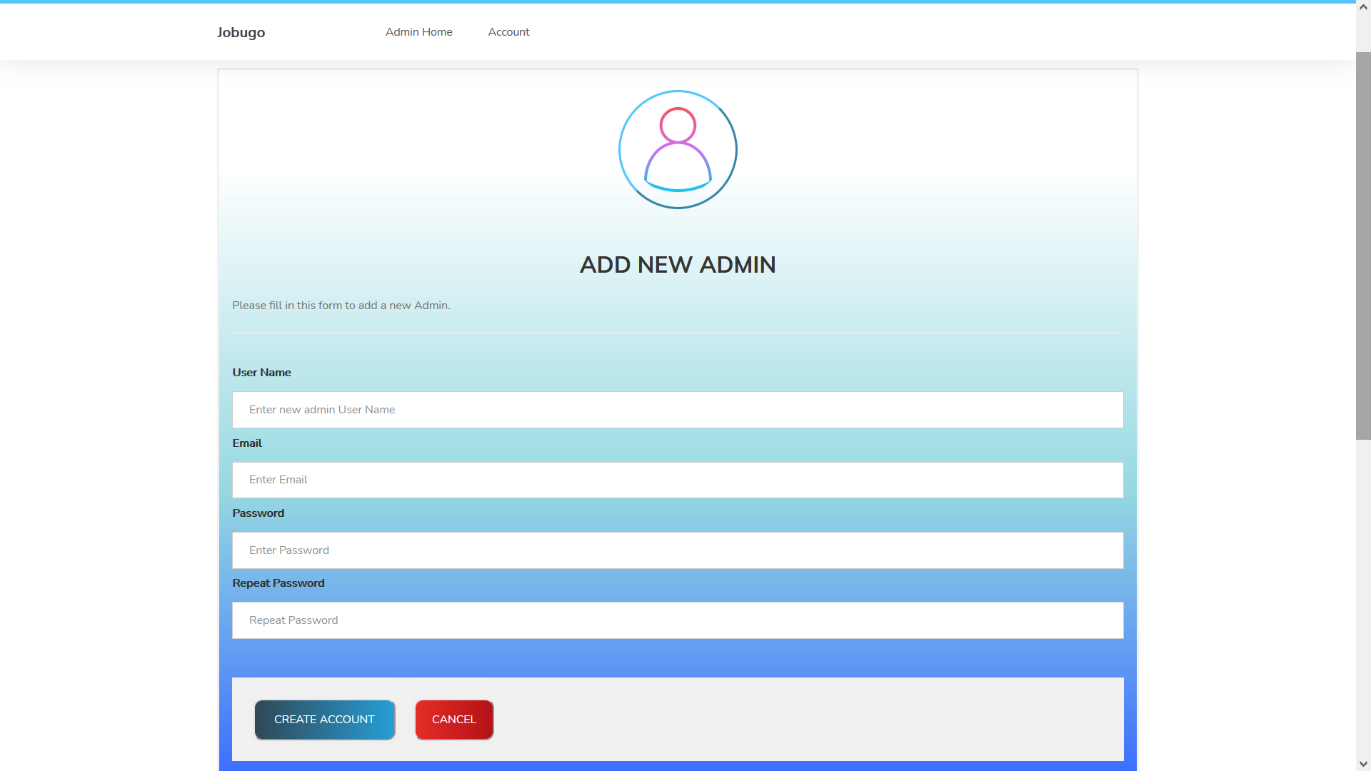
**7.17 Employer verification document viewer**



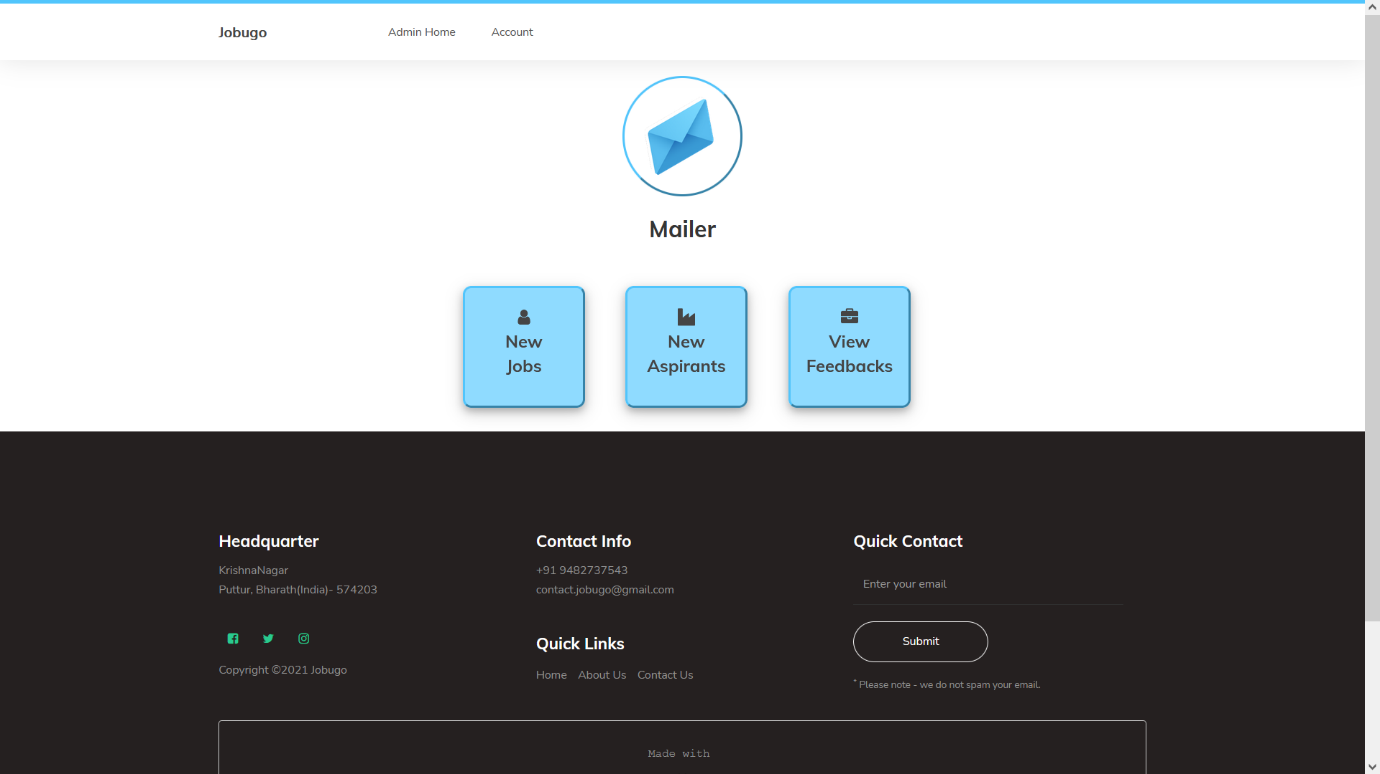
**7.18 Jobs list for admin**



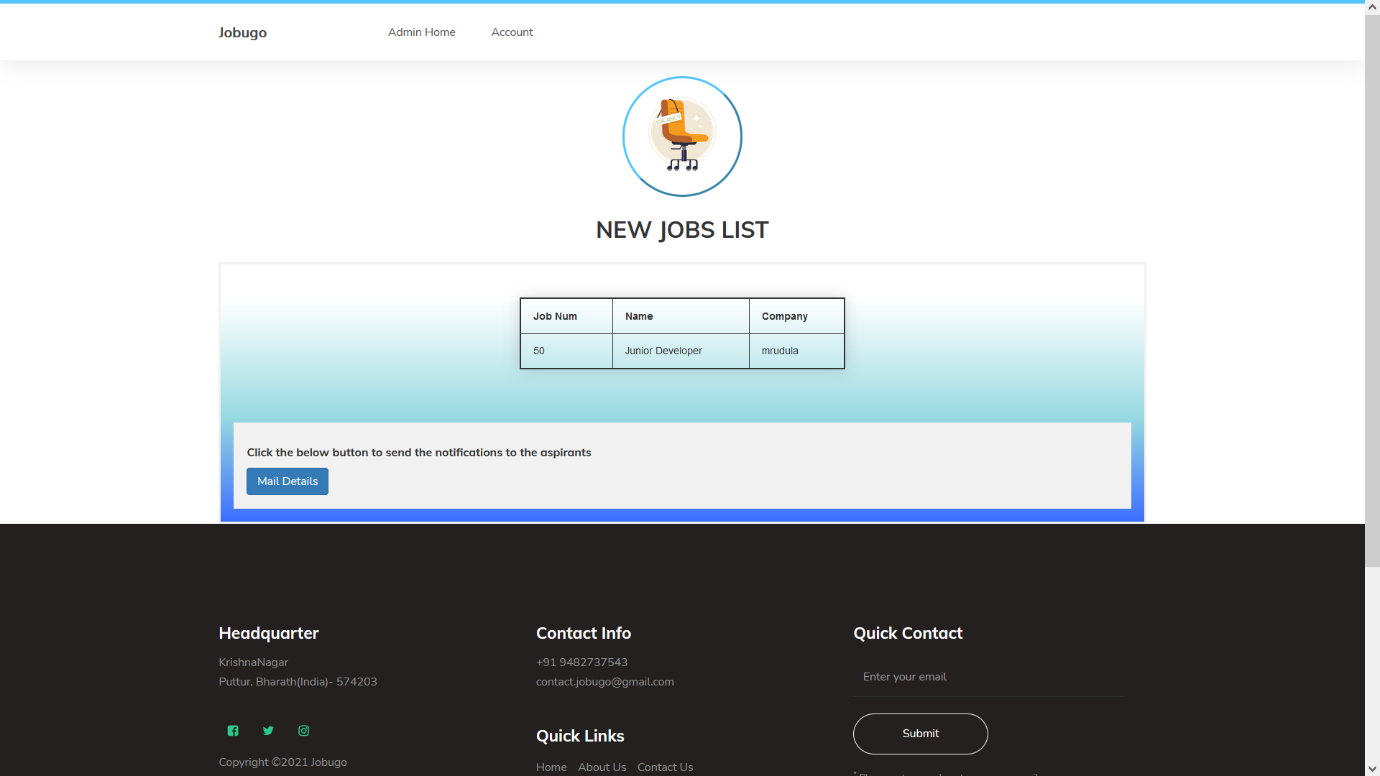
**7.19 Add new admin**



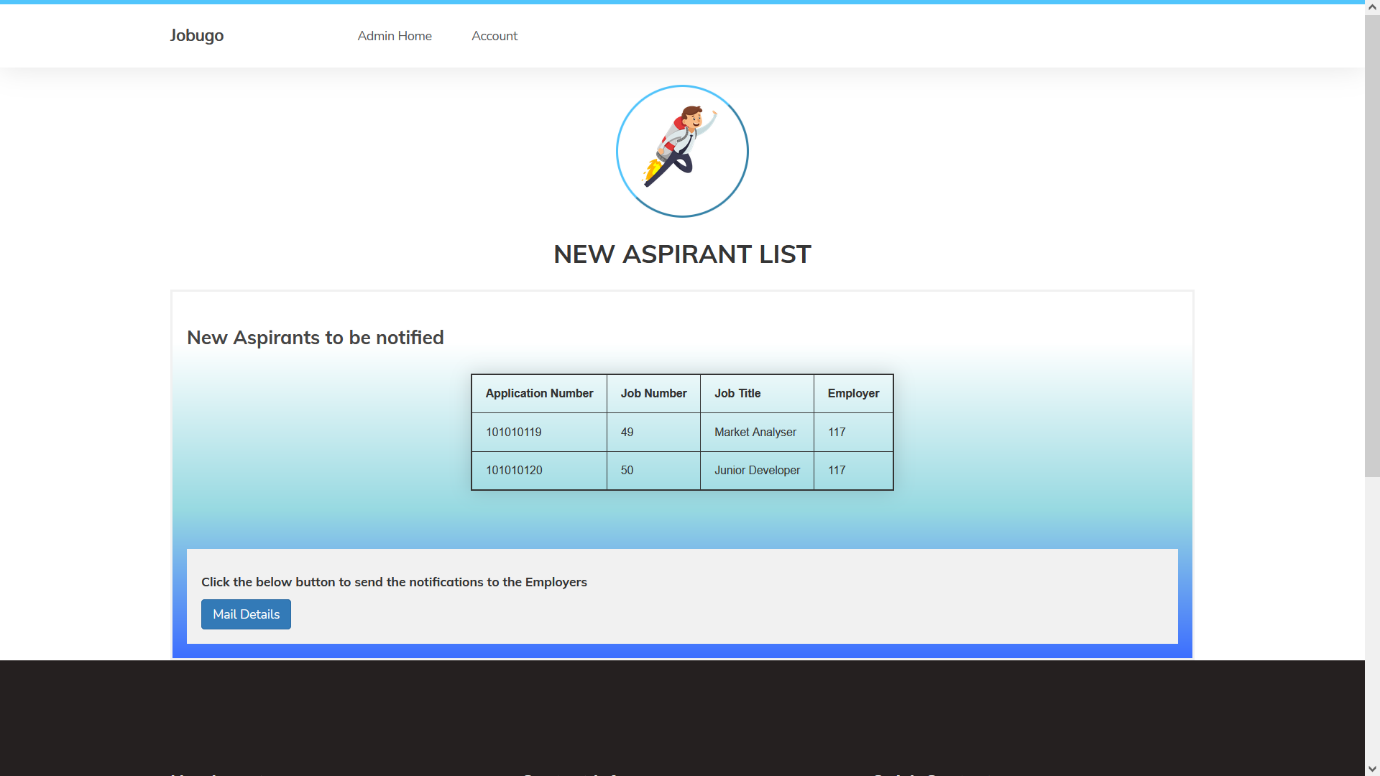
**7.20 Mailer (Admin account)**



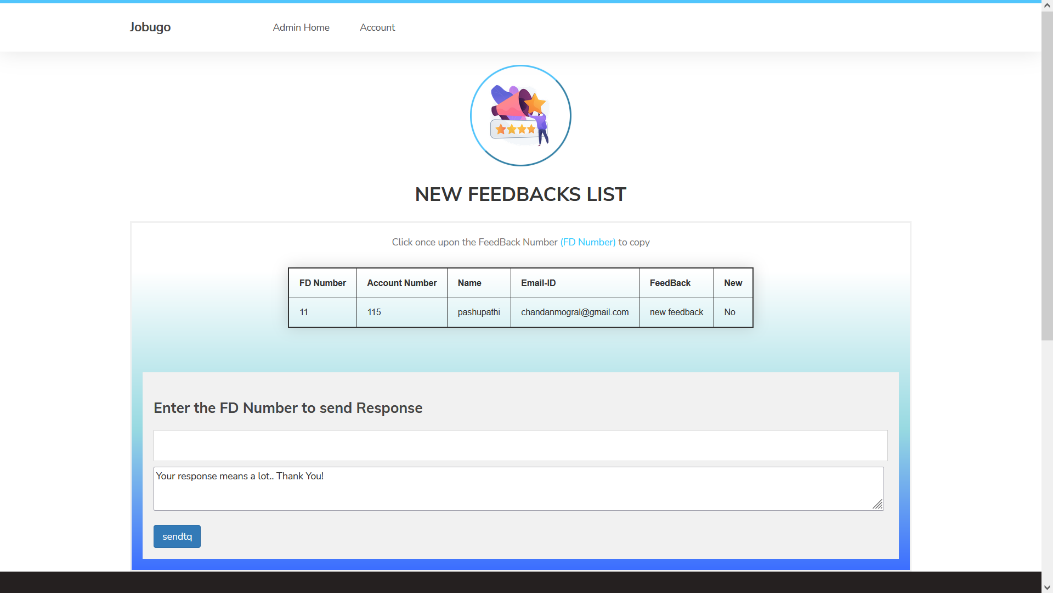
**7.21 New jobs to notify**



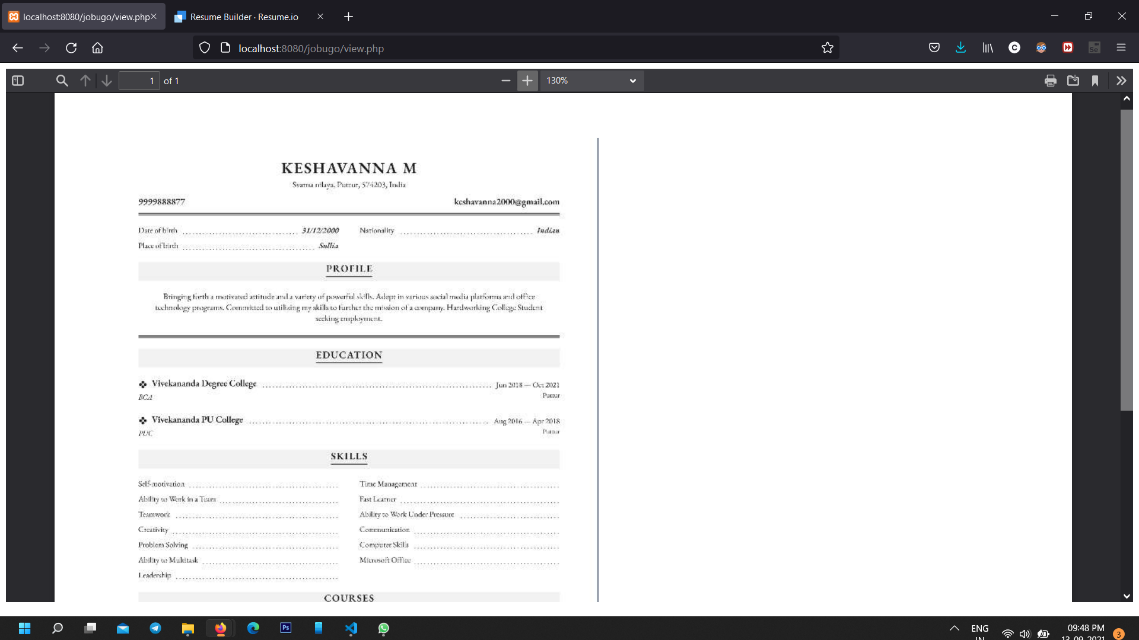
**7.22 New aspirants to notify**



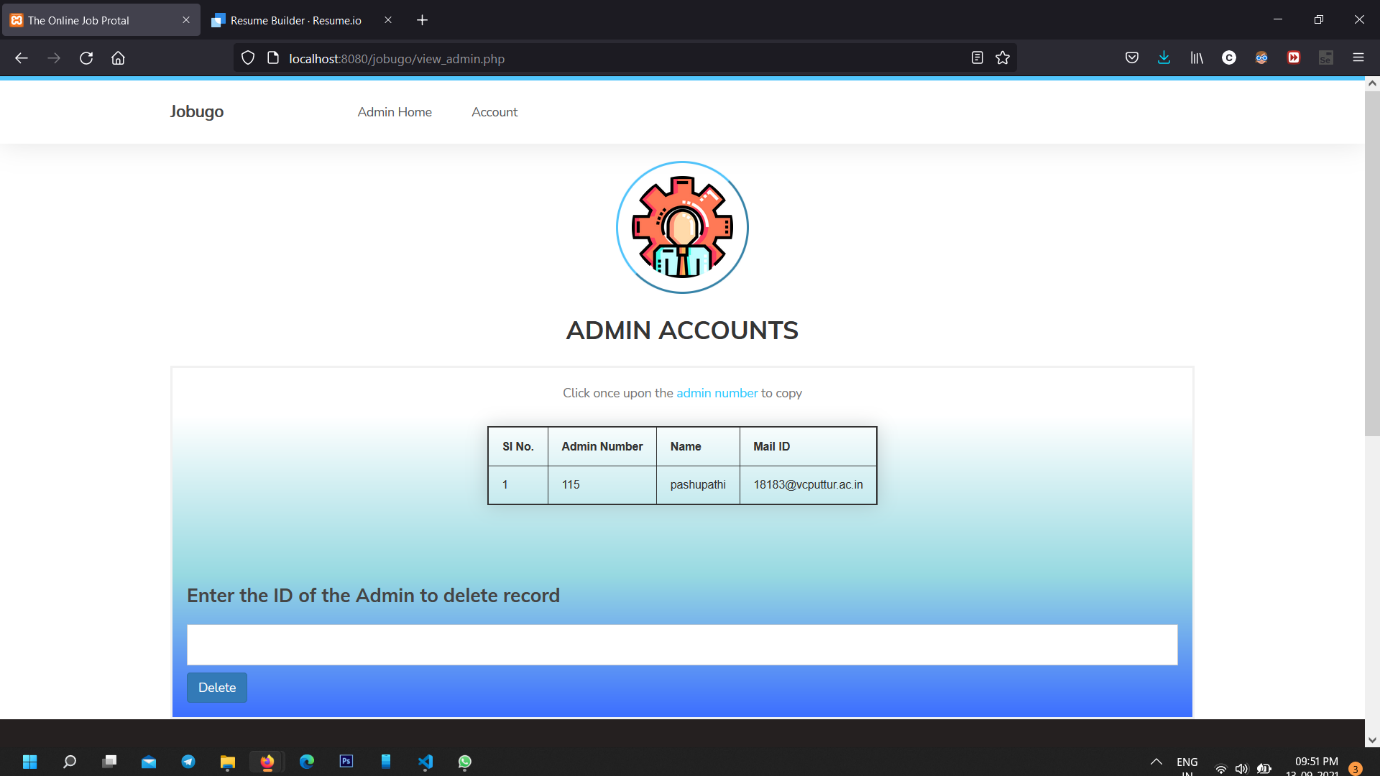
**7.23 New feedback to notify**



**7.24 Document viewer to view CV/Resume (Employer)**

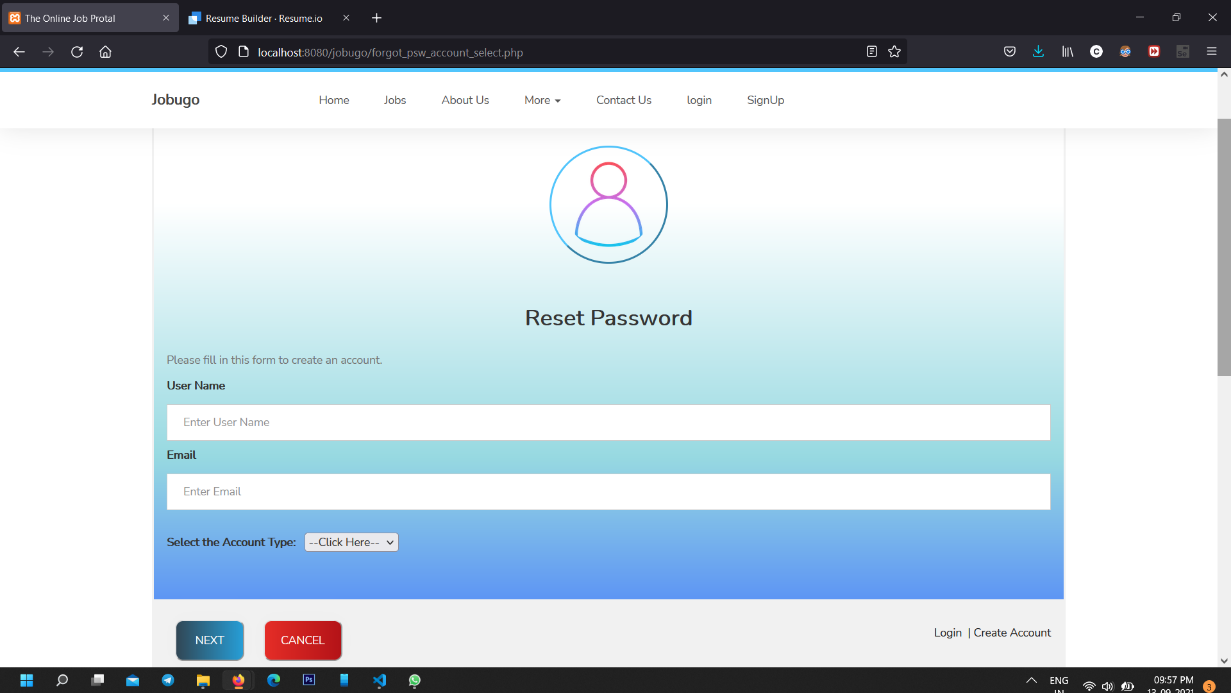


**7.25 View admin accounts (Admin)**



**7.26 Reset password**

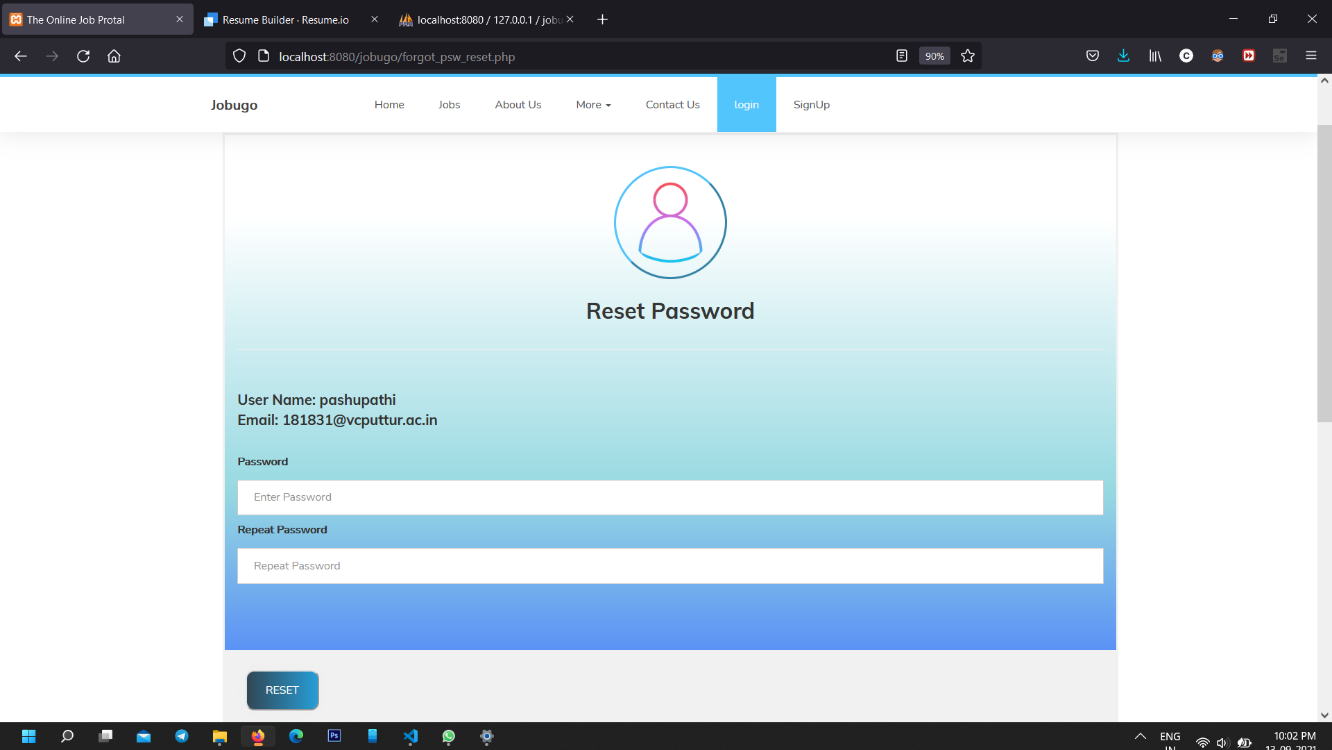
**7.26.1 Account verification**



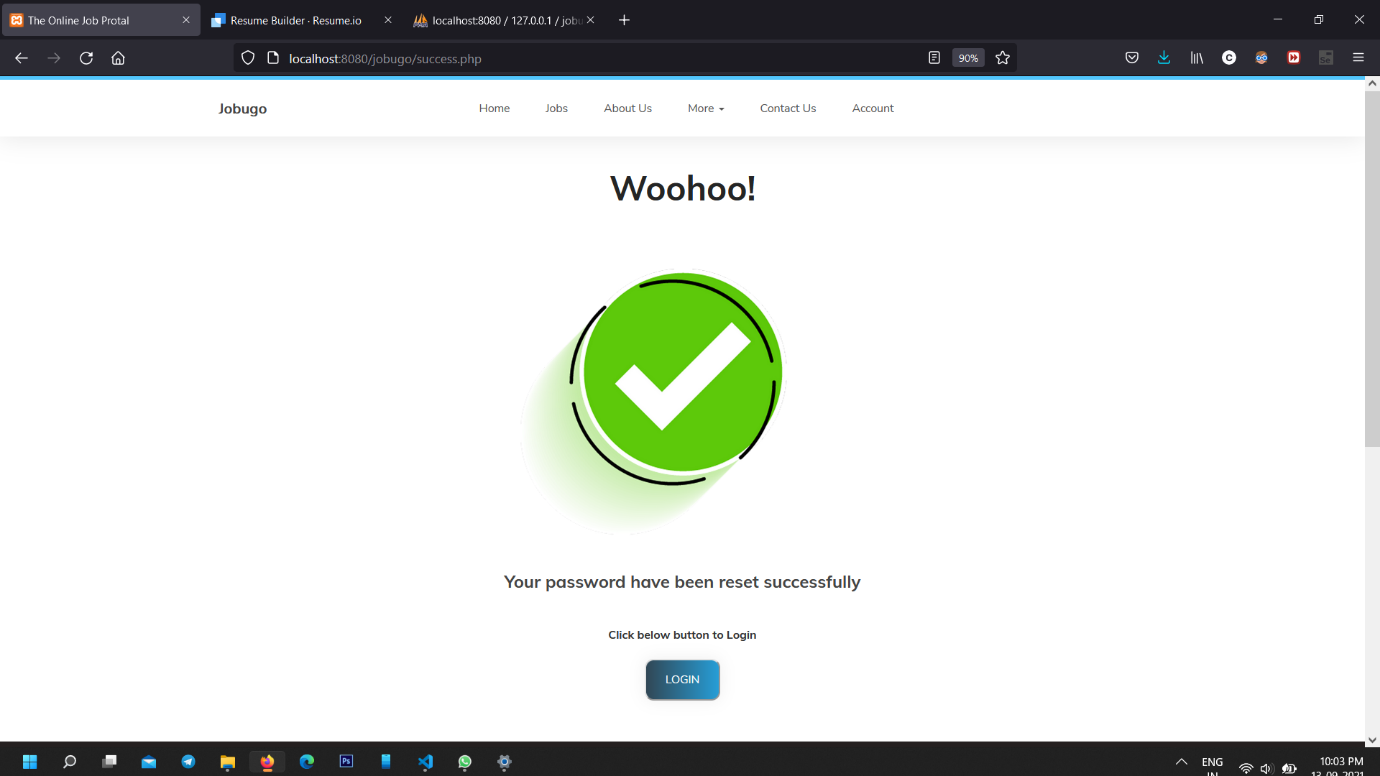
**7.26.2 Email verification through OTP**



**7.26.3 Reset password**



**7.26.4 Password reset successful**



**7.27 Android app background image**



**7.28 Android app icon**



**CHAPTER – 8**

**TESTING**

**8.1** **Introduction**

Software testing is an investigation conducted to provide stack holders with information about the quality of the product or service under test. Testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Software testing is the process used to assess the quality of computer software.

It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should. It is oriented to ‘detection’.

**8.1.1 Purpose**

* The verification process confirms that the software meet its technical specifications. A “specification” is a description of a function in terms of a measurable output value given a specific input value under specific preconditions.
* The validation process confirms that the software meets the business requirements.
* A defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development phases. Not all the defects will necessarily result in failures.

**8.1.2 Testing objectives**

* Finding defects which may get created by the programmer while developing the software
* Gaining confidence in and providing information about the level of quality
* To prevent defects.
* To make sure that the end results meets the business and user requirements.
* To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification.

**8.2 Testing types**

**8.2.1 Black box testing:**

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Tests are based on requirements and functionality.

**8.2.2 White box testing:**

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass box testing.** In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. Here tests ara based on coverage of code statements, branches, paths and conditions. A **test case** is a software testing document, which consists of event, action, input, output, expected results and actual results. White box testing is applicable at the unit, integration and system levels of the software testing process.

**8.3 Levels of testing**

**8.3.1 Unit testing:**

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

**8.3.2** **Integrate testing:**

In integration testing the different units of the system are integrated together to form the complete system. This type of testing checks the system as a whole to ensure that it is doing what it is supposed to do. The testing of an integrated system can be carried out top-down, bottom-up or big-bang. I

n this type testing some parts are tested with white-box testing and some with black-box testing techniques. This type of testing plays a very important role in increasing systems productivity.

**8.3.3 System testing:**

Apart from testing the system to validate the functionality of software against requirements, it is also necessary to test the non-functional aspect of the system. Some examples of non-functional tools include tests to check the performance, data, security, usability, volume, load and stress.

**8.3.4 User acceptance testing**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done in regard to the following point:

* Input screen design.
* Output screen design.
* Online message should be guide to the user.
* Format of reports and other outputs.

**8.4** **Test cases:**

**8.4.1 Login form**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If user clicks on login button without entering username and password. | Please Fill out this Field. | Successful |
| 2 | If username is blank but password is entered. | Please Fill Out this Field. | Successful |
| 3 | If password is blank but username is entered. | Please Fill out this Field. | Successful |
| 4 | If the username or password is incorrect. | Wrong user Name or password. | Successful |
| 5 | If the account type is not selected | Please select an item in the list | Successful |
| 6 | If the selected account type is incorrect | Wrong type | Successful |
| 7 | If the e-mail is not specified | Please Fill out this Field. | Successful |
| 8 | If the password is not specified | Please Fill out this Field. | Successful |
| 9 | If the repeat pass  word is not specified | Please Fill out this Field. | Successful |
| 10 | If the CV / Resume / Confirmation document is not selected to upload | Please fill in this form to create an account | Successful |
| 11 | If the website address is not specified (employer signup) | Please Fill out this Field. | Successful |
| 12 | If the username is already exists | Username already exists | Successful |
| 13 | If the password is less than 5 characters | Password less than 5 characters | Successful |
| 14 | When combination of alphabet and number not used in password | The password does not contain both alphabets and numbers | Successful |
| 15 | If the password and repeat password is not identical | Passwords do not match | Successful |
| 16 | If the same e-mail id is used to create an account more than once | There cannot be more than one accounts connected to the same e-mail id | Successful |
| 17 | When wrong OTP is inserted | Wrong OTP | Successful |
| 18 | When no OTP is typed by user | Please fill out this field | Successful |
| 19 | If the valid username and valid password is entered | System displays main page. | Successful |

**8.4.2 Job details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | When same aspirant applies for same job more than once | You have already applied for this job | Successful |
| 2 | If the job title is not specified | Please Fill out this Field. | Successful |
| 3 | If the job location is not specified | Please Fill out this Field. | Successful |
| 4 | If the job type is not selected | Please select an item in the list | Successful |
| 5 | If the qualification and graduation course is not selected | Please select an item in the list | Successful |
| 6 | If the salary is not specified | Please fill out this field | Successful |
| 7 | If the job detail is not specified | Please fill out this field | Successful |
| 8 | If the company detail is not specified | Please fill out this field | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If the last date is not specified | Please fill out this field | Successful |

**8.4.3 Add jobs**

**8.4.4 Your jobs (Employer)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is inserted | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.5 View aspirants (Employer)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is inserted | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.6 Aspirant accounts (Admin)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is selected | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.7 Employer accounts (Admin)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is inserted | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.8 View jobs (Admin)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is inserted | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.9 View admin (Admin)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data is selected | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.10 Add new admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If the username is mot inserted | Please fill out this field | Successful |
| 2 | If the e-mail id is not specified | Please fill out this field | Successful |
| 3 | If the password is not specified | Please fill out this field | Successful |
| 4 | If the repeat password is not specified | Please fill out this field | Successful |
| 5 | If the password is less than 5 characters | Password less than 5 characters | Successful |
| 6 | When password is not alpha-numeric | The password does not contain both alphabets and numbers | Successful |
| 7 | If the password and repeat password do not match | Passwords do not match | Successful |

**8.4.11 Reply to feedback**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If no data selected for id of feedback to reply | Please fill out this field | Successful |
| 2 | If wrong value inserted | No such records | Successful |

**8.4.12 Forgot password (Account select)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If the username is not specified | Please fill out this field | Successful |
| 2 | If the e-mail id is not specified | Please fill out this field | Successful |
| 3 | If the account type is not selected | Please select an item in the list | Successful |
| 4 | If the wrong type is selected | Account not found: wrong type | Successful |
| 5 | If the wrong username or e-mail id specified | Account not found: Wrong username or  e-mail id | Successful |

**8.4.13 Forgot password (Mailer)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If the OTP is not specified | Please fill out this field | Successful |
| 2 | If the entered OTP is incorrect | Wrong OTP | Successful |

**8.4.14 Forgot password (Reset)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1 | If the password is not specified | Please fill out this field | Successful |
| 2 | If the repeat password is not specified | Please fill out this field | Successful |
| 3 | If the new password is same as old password | The new password is same as old password | Successful |
| 4 | If the password is less than 5 characters | Password is less than 5 characters | Successful |
| 5 | If the specified password is not  alpha-numeric | The password does not contain both alphabets and numbers | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Test Condition** | **Expected Result** | **Result** |
| 1. | Clicks on “logout” button | Client should be logged out, should be redirected to main page of the website | Successful |

**8.4.15 Logout**

**CHAPTER – 9**

**CONCLUSION**

JOBUGO is an online job portal where the aspirants search for their dream jobs and employers will insert the job details into website and get the aspirant details. In the same way the aspirant can find set of job details to apply for. The admin has all the controls and can control the transaction in the web app.

The project developed using Php and MySQL is based on the requirements specification and the analysis of the existing system, with future enhancement. The expanded functionality of today’s software requires an appropriate approach towards software development.

**CHAPTER – 10**

**LIMITATIONS**

1) Less number of filters available in searching jobs

2) Sending mail is not automatic (Except OTP)

3) Admin doesn't have access to the CV of the aspirants, so the admin cannot validate the

Aspirant accounts. The CV can only be viewed by the Employer

4) All the records related to the aspirants of the job will be deleted and cannot be retrieved

again, once that particular job is removed

**CHAPTER – 11**

**FUTURE SCOPE AND ENHANCEMENTS**

* Completely independent android app
* Reduce the waiting time
* Conduct the first level of selection
* Guide to participating and cracking the interviews
* More flexible approach with employers

**CHAPTER – 12**

**ABBREVIATIONS AND ACRONYMS**

* **CSS:** Cascading Style Sheet
* **RDBMS:** Relational database management system.
* **PHP:** Hypertext Pre-processor.
* **MySQL:** My Structured Query Language.
* **HTML:** Hyper Text Mark-up Language.
* **CFD:** Context Flow Diagram.
* **DFD:** Data Flow Diagram.
* **ER diagram:** Entity Relationship diagram.
* **GUI:** Graphical User Interface.
* **RAM:** Random Access Memory

**CHAPTER – 13**

**BIBILIOGRAPHY AND REFERENCES**

**Web references**:

https://stackoverflow.com/

<https://www.W3schools.com/>

<https://www.tutorialspoint.com/>

<https://google.com>

<https://youtube.com>

**Book Reference:**

Programming PHP by Rasmus Lerdrof.