# **CSE2004 DATABASE MANAGEMENT SYSTEM**

# 

# PROJECT REVIEW REPORT

# PHASE - 3

# REGISTER No. 1: 19BCE0162

# NAME 1: Apurva Sharma

# REGISTER No. 2: 19BCE0914

# NAME 2: Khushee Jain

# EPJ SLOT: L13+14

# Project Title: ONLINE RECRUITMENT NETWORK

# Mobile Number: 83199 43063

# Project Type: Application

# Application Name : RecruitEASE

**ACKNOWLEDGEMENT**

We take this opportunity to express our profound gratitude and deep regards to our project guide **Prof. Saravanakumar K** for his exemplary guidance, monitoring and constant encouragement throughout the course of this subject **CSE2004: Database Management Systems** that helped us to complete this project .

The blessing, help and guidance given by him from time to time shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to the management of **VIT UNIVERSITY** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

Lastly, we thank the almighty, **our parents, brothers, sisters and friends** for their constant encouragement without which this project would not be possible.

**CONTENTS (Click on the contents):**

I. [Introduction](#_nqykz8m3cw5h)

1. [Aim](#_8ba6j6299hfu)
2. [Problem Statement](#_v625naj3vpzi)

II. [Proposed System Overview](#_nbh12txarjam)

III. [Phase 1](#_jwhvowd4u70w)

1. [Data Collection and Identification](#_k16cypbfb1mu)
2. [ER Diagram](#_tc2qmfg80ryc)
3. [Relationship Sets](#_t6j02ig9j0zi)
4. [Schema](#_amkjwnqmqqds)

IV. [Phase 2](#_c6zzpboopndt)

1. [Normalisation](#_5w1po5er4uwj)
2. [Final Schema](#_bvehl9bu5dmk)

V. [Phase 3](#_lzm3vni7pilo)

1. [Hardware and Software Requirements](#_5n9nvnbwek5)
2. [Help File](#_io1pn5bb55nt)
3. [Frontend Implementation](#_ehc504h4w9kx)
4. [Backend Implementation](#_3h2vy6p7mt0o)
5. [Flow of control](#_Flow_of_Control)
6. [Screenshots of the working project](#_b7x3v2owkbld)

# **INTRODUCTION:**

# **AIM:**

This project has been done by the students of B.Tech Computer Science program for the course of “Database Management Systems” with course code CSE2004. The basic approach of this project is to create a application for job seekers and companies where they can post job opportunities and look for jobs at the portal, the application will be connected to a database, and hence a fully functional application is to be made.

# **PROBLEM STATEMENT:**

The main aim of this project is to prepare an online recruitment system where applicants and companies can find jobs and post vacancies and opportunities.

# **Proposed System Overview**

1. **Home Page:** The landing page where the user can navigate to login or signup. The job seekers can also browse for job opportunities without logging in.
2. **Finding Jobs:** Navigate through jobs posted by companies.
3. **Sign Up Page:** Signing up as either a job seeker or a company.
4. **Login:** Logging into your account
5. **Account Page:** Interface for the user to see application status as a job seeker, look for applications from the job seekers and send interview schedules and offer letters to the applicant.

# **PHASE 1 DOCUMENTATION:**

**Note: Some tables were added/changed**

# **Data Collection Stage:**

**List of Entity Sets:**

1. Company

2. Offer Letter

3. User

4. Job Seeker

5. Requisition

6. Interview

7. Skill

8. Interview\_Type

9. Result

# **Data Identification Stage:**

1. Company (Comp\_id , Comp\_name, Email\_id, Comp\_Type,Vacany,Post )

2. Offer Letter (Off\_Id, Date, Incharge)

3. User(Username, Password, Login\_Time, Logout\_Time)

4. Job Seeker (Can\_Id, Can\_Name, Pesonal\_Info, Experience,Contact\_details)

5. Requisition (Package, Criteria, Req\_Id)

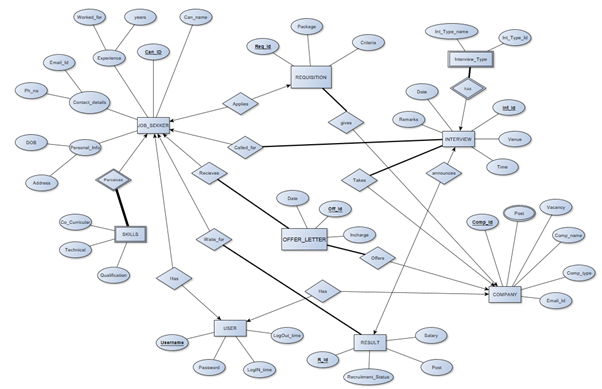
6. Skill ( Qualification, Co\_Curricular, Technical)

7. Interview (int\_Id, Remarks, Date, Time,Venue )

8. Interview\_Type(int\_Type\_Id, int\_Type\_Name)

9. Result (R\_Id, Recruitment\_status, Post, Salary)

# **E-R Diagram:**

****

# **Relationship Sets:**

1. Company (Comp\_id , Comp\_name, Email\_id,Comp\_Type,Vacany, Username)

2. Offer Letter (Off\_Id, Comp\_Id, Can\_Id, Date, Incharge)

3. User(Username, Password, Login\_Time, Logout\_Time)

4. Job Seeker (Can\_Id, Can\_Name, DOB, Address, Email\_Id, Ph\_no, Worked\_for,years, Offer\_Id, Username)

5. Requistion (Package , Criteria, Req\_Id, Comp\_Id,Can\_Id)

6. Skill (Can\_Id, Qualification, Co\_Curricular, Technical)

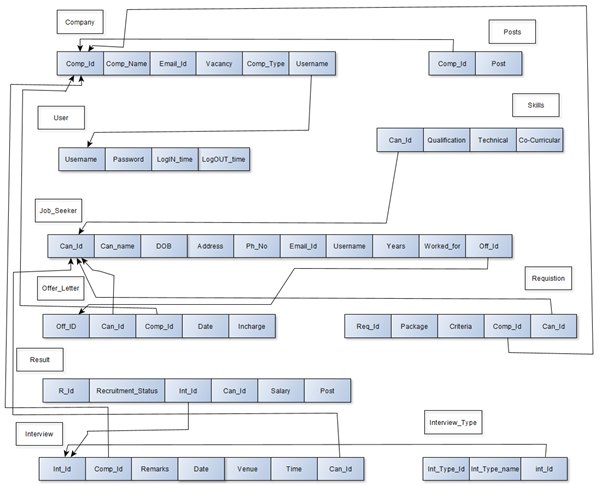
7. Interview (int\_Id, Comp\_Id, Remarks, Date, Time, Venue, Can\_Id)

8. Interview\_Type(int\_Type\_Id, int\_Type\_Name, int\_Id)

9. Result(R\_Id,Recruitment\_status, Post, Salary, Can\_Id, int\_Id)

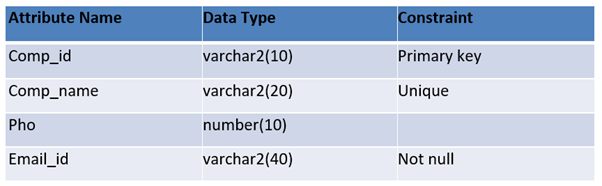
10.Posts(Comp\_Id,Post)

# **Schema:**

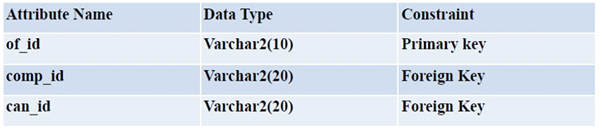
****

**Addition of Constraint on the Conceptual Schema**

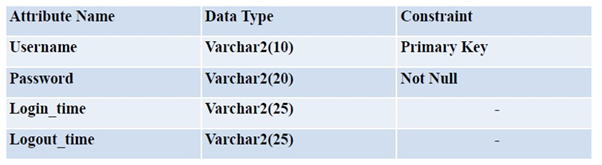
COMPANY:



Offer Letter:



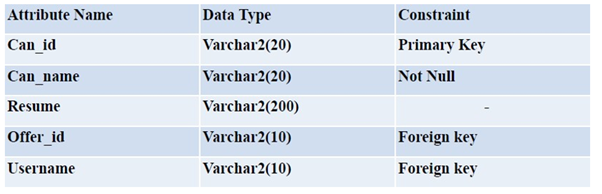
ADMIN:

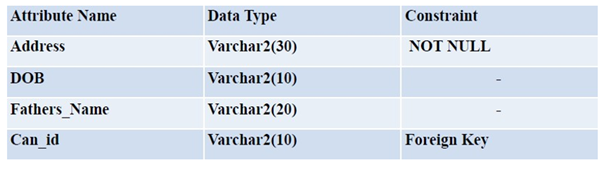


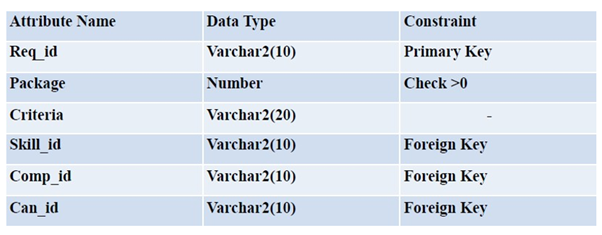
Experience:

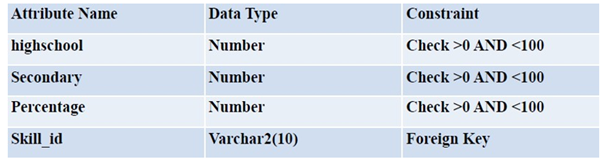
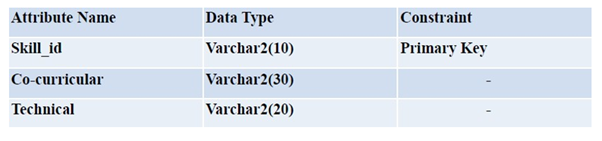
|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Constraint |
| Exp\_detail | Varchar2(30) | - |
| Exp\_org | Varchar2(10) | - |
| Comp\_id | Varchar2(20) | Foreign key |
| Can\_id | Varchar2(20) | Foreign key |

Job Seeker:

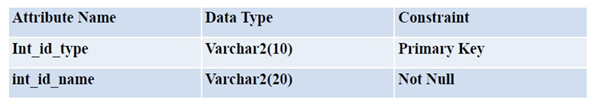
Personal Info:



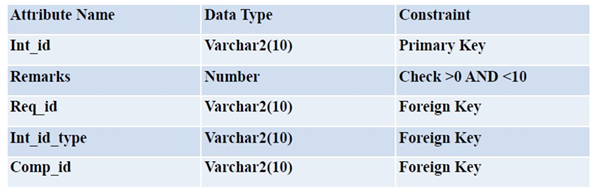
Requisition:Academics:

SKILL:

Interview Type:



Interview:



# **PHASE 2 DOCUMENTATION:**

## **Normalization:**

**Job Seeker**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Can\_id | Can\_name | DOB | Address | Email\_id | Ph\_no | Worked\_for | Years | Off\_id | Username |
| J1 | Flynn Rider | 29-4-1990 | B104 Street A, California | flyn@ymail.com | 12654836 | Trello | 3 | O1 | Flynn\_02 |
| J2 | Jake Harper | 25-5-1995 | C22 Street B, New Jersey | jke@ymail.com | 31177434 | Accenture | 4 | O2 | Jake\_45 |
| J3 | Rodger S | 29-4-1997 | D23  Street A, Tawain | rodger@ymail.com | 39675369 | Hopper Technologies | 5 | O2 | Rodger\_09 |
| J4 | Linda martin | 30-8-1990 | X3  Street A, India | linm@ymail.com | 34567889 | Safety  Travels | 2 | O3 | Linda\_DJ02 |

**FD: Can\_id→Can\_name,DOB,Address,Email\_id,Ph\_no,Username**

**Email\_id → Can\_id,Can\_name,DOB,Address,Ph\_no, Username**

**Username → Can\_id,Can\_name,DOB,Address,Email\_id,Ph\_no**

**Can\_id, Worked\_for → Years**

**Email\_id, Worked\_for→ Years**

**Username,Worked\_for →Years**

**Candidate keys:** Since Off\_id is not dependent on any attribute it must be present in candidate key.

(Can\_id, Worked\_for,Off\_id)+= Can\_id, Worked\_for, Off\_id, Can\_name, DOB, Address,Username,Email\_id,Ph\_no,Years = R

(Email\_id,Worked\_for,Off\_id)+= Can\_id, Worked\_for, Off\_id, Can\_name, DOB, Address,Username,Email\_id,Ph\_no,Years = R

(Username,Worked\_for,Off\_id)+ = Can\_id, Worked\_for, Off\_id, Can\_name, DOB, Address,Username,Email\_id,Ph\_no,Years = R

(Ph\_no,Worked\_for,Off\_id)= Can\_id, Worked\_for, Off\_id, Can\_name, DOB, Address,Username,Email\_id,Ph\_no,Years = R

Candidate keys are : { (Can\_id, Worked\_for,Off\_id), (Email\_id,Worked\_for,Off\_id), (Username,Worked\_for,Off\_id), (Ph\_no,Worked\_for,Off\_id) }

Prime Attributes are: {Can\_id,Email\_id , Ph\_no,

Worked\_for, Off\_id, Username}

Non-prime attributes are: {Can\_name, years, DOB, Address}

**Normalisation:**

1. Since all the attributes of this relation are atomic. The table is in 1 Normal Form.

2. **For 2NF:**

1. It should be in 1NF.

2. Elimination of partial key functional dependency.

Minimal Cover of FD’s : can\_id → Username Email\_id→Username Username→can\_name, Username→Address Username→ DOB Username → Email\_id Username→ph\_no (Email\_id,Worked\_for) → years

Now in Username→ Can\_name,DOB, Address, Ph\_no partial dependency is present.

So, we will decompose this in a separate table as:

**Decomposition:**

R1→ Username, can\_name, DOB, Address,ph\_no

with FD’s: Username → can\_name, DOB, Address,ph\_no

R2→ Can\_id,Email\_id,Worked\_for,Years,Off\_id,Username

with FD’s: Can\_id→ Username Email\_id→Username Username→Email\_id,Can\_id Worked\_for,Username→ Years

R1 is in 2NF as it does not contain any partial dependency and it is in 1NF.

And candidate key for R2 is Username as (Username)+ = Username,can\_name, DOB, Address,ph\_no = R1

In R2 we have ( Worked\_for,Username) → Years as partial dependency.

So, we will decompose the table as

R3→ Worked\_for,Username,Years with FD’s: (Worked\_for,Username) → Years

R4→ Can\_id,Email\_id, Worked\_for,Off\_id, Username with FD’s: Cab\_id→Username, Email\_id→Username, Username→Email\_id,Can\_id

Here both R3 and R4 do not contain any partial dependency and are in 1NF thus they are in 2NF.

Candidate key of R3 is Worked\_for,Username as (Worked\_for,Username)+= Worked\_for,Username,Years = R3

Candidate keys of R4 are (Can\_id, Worked\_for,Off\_id), (Email\_id,Worked\_for,Off\_id), (Username,Worked\_for,Off\_id) as their closure gives R4.

Final Tables are

R1→ Username, can\_name, DOB, Address,ph\_no R3 → Worked\_for,Username,Years

R4→ Can\_id,Email\_id, Worked\_for,Off\_id, Username

**Checking lossless decomposition:**

· R1 Ո R3 = Username

And Username is the candidate key for R1. Hence, Decomposition into R1 and R3 is lossless.

· R3 Ո R4 = Worked\_for, Username

And Worked\_for, Username is the candidate key for R3. Hence, Decomposition into R3 and R4 is lossless.

· R1 Ո R4 = Username

And Username is the candidate key for R1. Hence, Decomposition into R1 and R4

is lossless.

3. **For 3NF:**

1. It should be in 2NF.

2. It should not contain any transitive dependency.

R1 do not contain any Transitive Dependency

Thus it is in 3NF.

R3 do not contain any Transitive Dependency.

Thus it is in 3NF.

R4 do not contain any Transitive Dependency.

Thus it is in 3NF.

**4. For BCNF:**

1. It should be in 3NF.

2. LHS of each FD should be candidate key or super key.

R1→ Username, can\_name, DOB, Address,ph\_no

It is in BCNF as LHS is a candidate key in the FD: Username → can\_name, DOB, Address,ph\_no

R3 → Worked\_for,Username,Years

It is in BCNF as LHS is a candidate key in the FD: Worked\_for,Username → Years

R4 → Can\_id,Email\_id, Worked\_for,Off\_id, Username

with FD’s: Cab\_id→Username, Email\_id→Username, Username→Email\_id,Can\_id

It is not in BCNF as LHS do not contain super keys.

So decompose the table into

**Decomposition:**

R5 → Worked\_for , Username, Off\_id

all the three attributes together forms a candidate key.

Therefore it is in BCNF.

R6→Username, Email\_id, Can\_id with FD’s: Username→ Can\_id, (Email\_id Can\_id)→ Email\_id, (Username,Email\_id) → Username,Can\_id

Candidate keys are Username , Can\_id, Email\_id as

(Username)+ =Username,Can\_id,Email\_id

(Can\_id)+ =Username,Can\_id,Email\_id

(Email\_id)+ =Username,Can\_id,Email\_id

Since every dependency has LHS as a candidate key it is in BCNF.

Final Tables are: R1→ Username, can\_name, DOB, Address,ph\_no R3 → Worked\_for,Username,Years

R5 → Worked\_for,Off\_id,Username

R6→Username,Email\_id,Can\_id

**Interview\_Type**

|  |  |  |
| --- | --- | --- |
| Int\_type\_id | Int\_type\_name | Int\_id |
| IT1 | Technical | I1 |
| IT2 | HR | I1 |
| IT2 | HR | I3 |
| IT1 | Technical | I4 |

**FD: Int\_type\_id → Int\_type\_name**

**Int\_type\_name → Int\_type\_id**

**Normalization :**

1. Candidate keys: {(Int\_id Int\_type\_id ), (Int\_id int\_type\_name) }

2. It’s already in 1 NF, 2 NF, 3 NF

3. For BCNF decompose into two tables: Int\_type\_id ,Int\_id and Int\_type\_id ,Int\_type\_name

(Int\_type\_id, Int\_id) and (int\_type\_id and int\_type\_name)

**Interview**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Int\_id | Comp\_id | Can\_id | Remarks | Date | Time | Venue |
| I1 | C1 | J1 | Good | 28-Jun-2020 | 9: 00 | MB |
| I2 | C1 | J2 | Excellent | 28-Jun-2020 | 8: 00 | SJT |
| I3 | C2 | J3 | NI | 1-Jul-2020 | 9: 00 | SJT |
| I3 | C3 | J4 | Good | 4-Jul-2020 | 9: 30 | SJT |

**FD: Int\_id → Comp\_id , Remarks , Can\_id , Remarks , Time , Venue**

**Comp\_id , Can\_id → Int\_id , Remarks ,Remarks , Time , Venue**

**Remarks ,Time , → Int\_id , Comp\_id ,Can\_id ,Remarks**

**Normalization :**

1. Candidate keys: {Int\_id , (Comp\_id Can\_id) , (RemarksTimeVenue**)**}

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

3. Hence final table is: Interview(Int\_id, Comp\_id, Can\_id, Remarks, Date, Time, Venue )

With minimal functional dependencies:

Int\_Id → Date, Time, Venue

Comp\_id Can\_id → Date, Time, Venue

Date Time Venue → Int\_id, Comp\_id, Can\_id, Remarks

**Result**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Result\_id | Post | Salary | Can\_id | Int\_id | Recruitement\_status |
| RID1 | Developer | 2000000pa | J1 | I1 | Waiting |
| RID2 | Product Manager | 2000000pa | J2 | I2 | Recruited |
| RID3 | Product Designer | - | J3 | I3 | Rejected |
| RID4 | Consultant | 1000000pa | J4 | I4 | Recruited |

**FD: Result\_id → Post , Salary , Can\_id , Int\_id , Recruitement\_status**

**Int\_id → Result\_id, Post , Salary , Recruitement\_status , Can\_id**

**Can\_id → Result\_id Post , Salary**

**Normalization :**

1. Candidate keys: Result\_id ,Can\_id , Int\_id

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

3. Hence final table is: Skill(Result\_id, Post, Salary , Can\_id, Int\_id, Recruitement\_status) where the functional dependency is -

Result\_id → Post, Salary , Can\_id, Int\_id, Recruitement\_status

Int\_id → Result\_id, Post, Salary , Can\_id , Recruitement\_status

Can\_id → Result\_id, Post, Salary

**Posts**

|  |  |
| --- | --- |
| **Comp\_Id** | **Post** |
| C1 | Consultant |
| C1 | Developer |
| C2 | Consultant |
| C2 | Manager |

No Functional Dependency hence the only candidate key is (Comp\_Id, Post).

Thus it is 1NF, 2NF, 3NF and BCNF.

**Skill**

|  |  |  |  |
| --- | --- | --- | --- |
| Can\_id | Qualification | Co\_Curricular | Technical |
| J1 | M S | Design | C1 |
| J2 | M Tech | Creative Writing | C2 |
| J3 | B Tech | Societies | C3 |
| J4 | Ph. D | Sports | C4 |

**FD: Can\_id → Qualification ,Co\_Curricular ,Technical**

**Normalization :**

4. Candidate keys: Can\_id

5. It’s already in 1 NF, 2 NF, 3 NF and BCNF

6. Hence final table is: Skill(Can\_id, Qualification, Co\_Curricular, Technical ) where the functional dependency is -

Can\_id → Qualification, Co\_Curricular, Technical

**Requisition**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req\_id | Package | Criteria | Comp\_id | Can\_id |
| R1 | 600000 | 2 | C1 | J1 |
| R2 | 700000 | 2 | C2 | J2 |
| R3 | 900000 | 1 | C3 | J3 |
| R4 | 500000 | 1 | C4 | J4 |

**FD:** Req\_id → Package **,** Comp\_id **,** Criteria **,** Can\_id

Comp\_id **,** Can\_id → Req\_id **,** Package **,** Criteria

**Normalization :**

1. Candidate keys: Req\_id , Comp\_id, Can\_id

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

3. Hence final table is: Requisition(Req\_id, Package, Criteria, Comp\_id, Can\_id) where the functional dependency is -

Req\_id → Package, Criteria, Comp\_id, Can\_id

Comp\_id, Can\_Id → Req\_id, Package, Criteria

**User**

|  |  |  |  |
| --- | --- | --- | --- |
| Username | Password | Login\_time | Logout\_time |
| Flynn | Hfsjcavakl | 22:00 UTC | 23:00 UTC |
| Jake | Vahjdjac | 20:00 IST | 21:00 IST |
| Rodger | Hwgyufe | 21:00 UTC | 23:00 UTC |
| Linda | Ahlbuciac | 19:00 IST | 20:00 IST |

**FD: Username → Password ,LogIn\_time, LogOUT\_time**

**Normalization :**

1. Candidate keys: **Username**

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

3. Hence final table is: User(Password, Username, LogIn\_time, LogOUT\_time)

where the functional dependency is –

Username → Password LogIn\_time, LogOUT\_time

**Offer Letter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Off\_id | Comp\_id | Can\_Id | Date | Incharge |
| O1 | C1 | J1 | 2 – 3 – 2001 | J . Murugan |
| O2 | C1 | J2 | 12 – 4 – 2001 | J . Murugan |
| O3 | C2 | J3 | 17 – 5 – 2001 | Dev Mehta |
| O4 | C3 | J2 | 2 – 3 – 2001 | Riya S |

**FD: Off\_id → Comp\_id, Can\_Id , Date , Incharge**

**Comp\_id, Can\_Id → Off\_id , Date , Incharge**

**Normalization :**

1. Candidate keys: Off\_id , Comp\_id, Can\_Id

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

3. Hence final table is: Offer Letter(Off\_id, Comp\_id, Can\_Id, Date, Incharge) where the functional dependency is -

Off\_id → Comp\_id, Can\_Id, Date, Incharge

Comp\_id, Can\_Id → Off\_id, Date, Incharge

**Company**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comp\_id | Comp\_name | Email\_id | Comp\_Type | Vacancy | Username |
| C1 | Campp | campp@ymail.com | Travel | 3 | Campp012 |
| C2 | Trello | trello@ymail.com | Software | 2 | Trello56 |
| C3 | Embibe | embibe@ymail.com | Education | 2 | Embibie09 |
| C4 | TheTribe | thetribe@ymail.com | Software | 1 | TheTribe34 |

**FD : Comp\_id → Comp\_name , Email\_id , Comp\_Type , Vacancy, Username**

**Email\_id → Comp\_id ,Comp\_name ,Comp\_Type ,Vacancy, Username**

**Username → Comp\_name , Email\_id , Comp\_Type , Vacancy,Comp\_id**

**Comp\_name Comp\_Type → Comp\_id ,Email\_id ,Vacancy, Username**

**Normalization :**

1. Candidate keys: {Comp\_id, Email\_id , Username, ( Comp\_name Comp\_Type) }

2. It’s already in 1 NF, 2 NF, 3 NF and BCNF

Hence final table is: Company (Comp\_id , Comp\_name, Email\_id, Comp\_Type, Vacany, Username)

With functional dependencies-

Comp\_Id → Username

Username→Email\_Id

Comp\_name Comp\_Type→ Username

Email\_id → Comp\_id , Comp\_name, Comp\_Type, Vacancy

**Total number of Tables in the final schema: 14**

## **Final schemas:**

1. Company (Comp\_id , Comp\_name, Email\_id,Comp\_Type,Vacany, Username)

2. Offer Letter (Off\_Id, Comp\_Id, Can\_Id, Date, Incharge)

3. User(Username, Password, Login\_Time, Logout\_Time)

4. R1JobSeeker(Can\_id, DOB, Address, ph\_no )

5. R3JobSeeker(Worked\_for, Years, Off\_id)

6. R5JobSeeker(Username, Worked\_for, Off\_id)

7. R6JobSeeker(Username, Email\_id, Can\_id)

8. Requistion (Package , Criteria, Req\_Id, Comp\_Id,Can\_Id)

9. Skill (Can\_Id, Qualification, Co\_Curricular, Technical)

10. Interview (int\_Id, Comp\_Id, Remarks, Date, Time, Venue, Can\_Id)

11. R1Interview\_type(Int\_type\_id, int\_id)

12.R2Interview\_type(int\_type\_id, int\_type\_name)

13. Result(R\_Id,Recruitment\_status, Post, Salary, Can\_Id, int\_Id)

14. Posts(Comp\_Id, Post)

# **PHASE 3 DOCUMENTATION:**

## **Hardware/software requirements:**

Hardware requirements:

1. 2 GB RAM
2. 1.6 GHz CPU
3. Space: 45 MB

Software requirements:

1. Python 3.6+

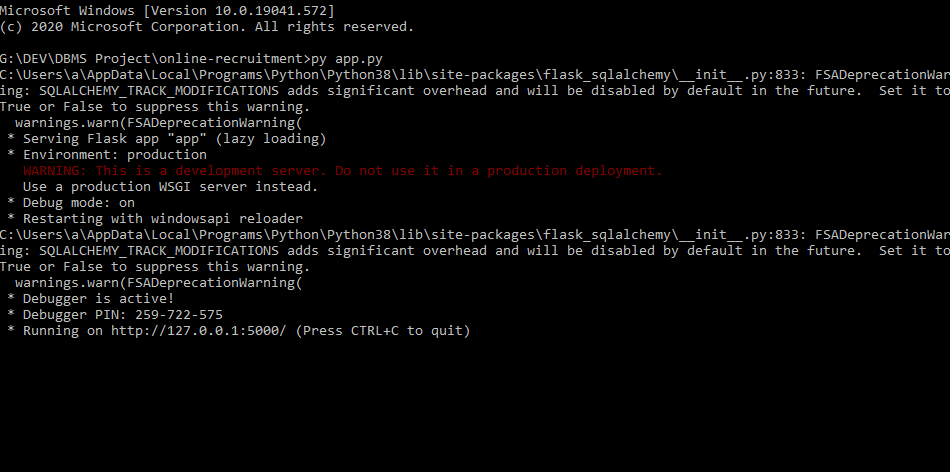
* Flask (Back-end)
* SQLAlchemy (Database connection)

1. HTML/CSS
2. Modern Web Browser
3. Database used: SQLite

## **Help File:**

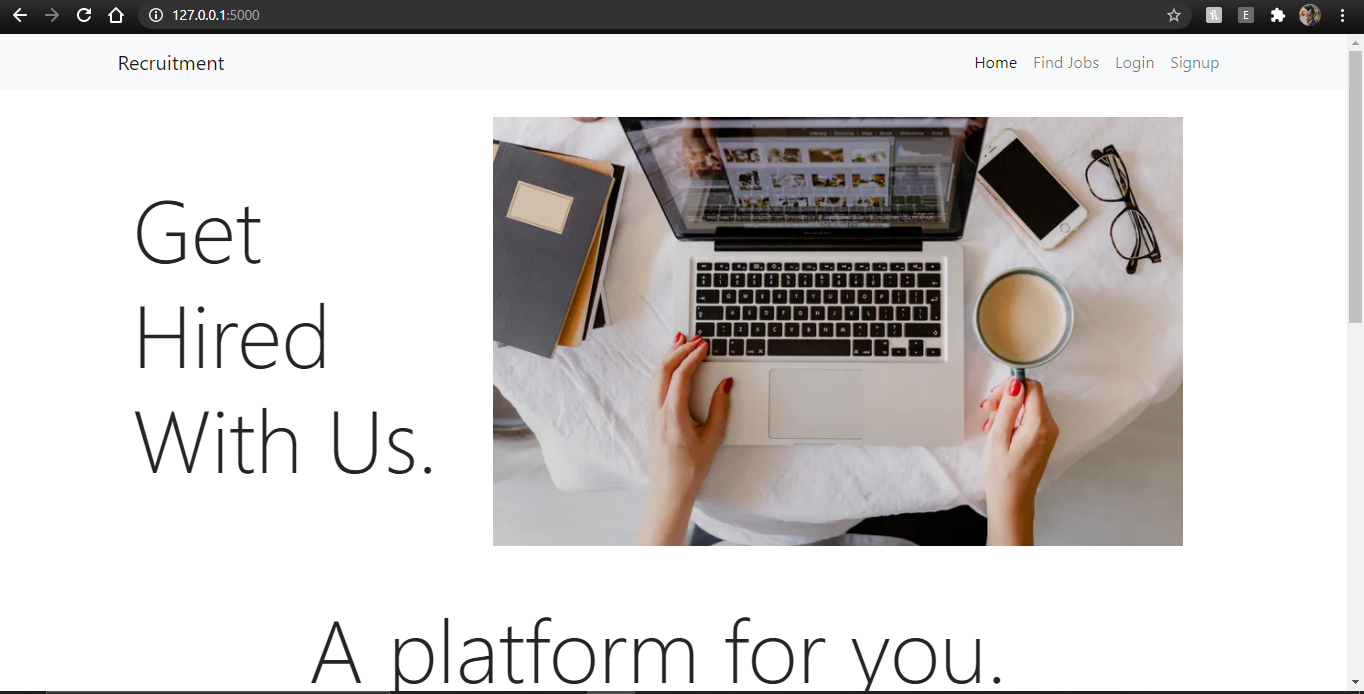
1. Unzip the Application File.
2. Install Python(3.6+).
   * <https://www.python.org/downloads/>
   * 
3. Install pip to install python the packages.
   * python -m pip install -U pip
4. Create a virtual environment for the Folder using Command Line and install the required packages.

* pip install virtualenv
* virtualenv env
* env\Scripts\activate
* pip install -r requirements.txt

1. Run the app using the following command
   * python3 app.py
2. Open your browser window, and enter fire up the local development server at <http://localhost:5000> (alternatively <http://127.0.0.1:5000>)
   * 
3. The application starts at port 5000, now you can use the app in your browser.

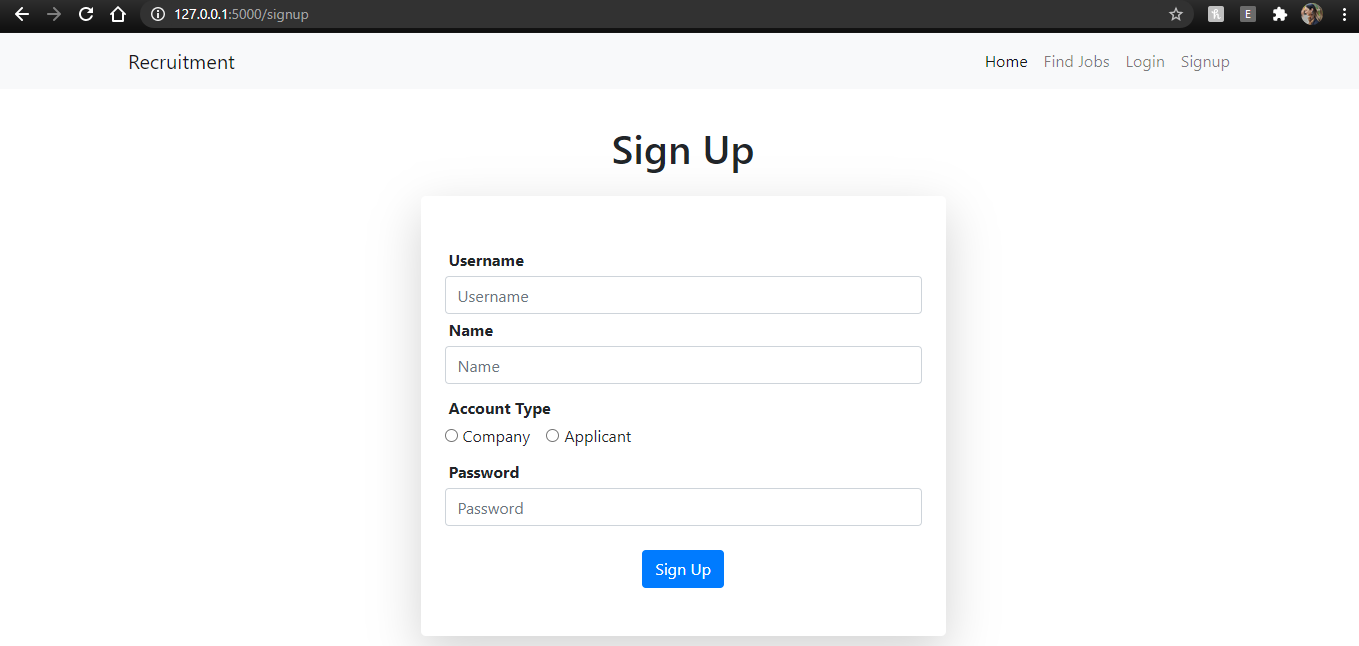
## **Front-end Implementation:**

## For Front-End Implementation we have made a Home Page which has been designed using **HTML,CSS and Bootstrap.** It has three functions: Find Jobs, Login or Sign up. Using Find Jobs, we can browse through job posts by companies.

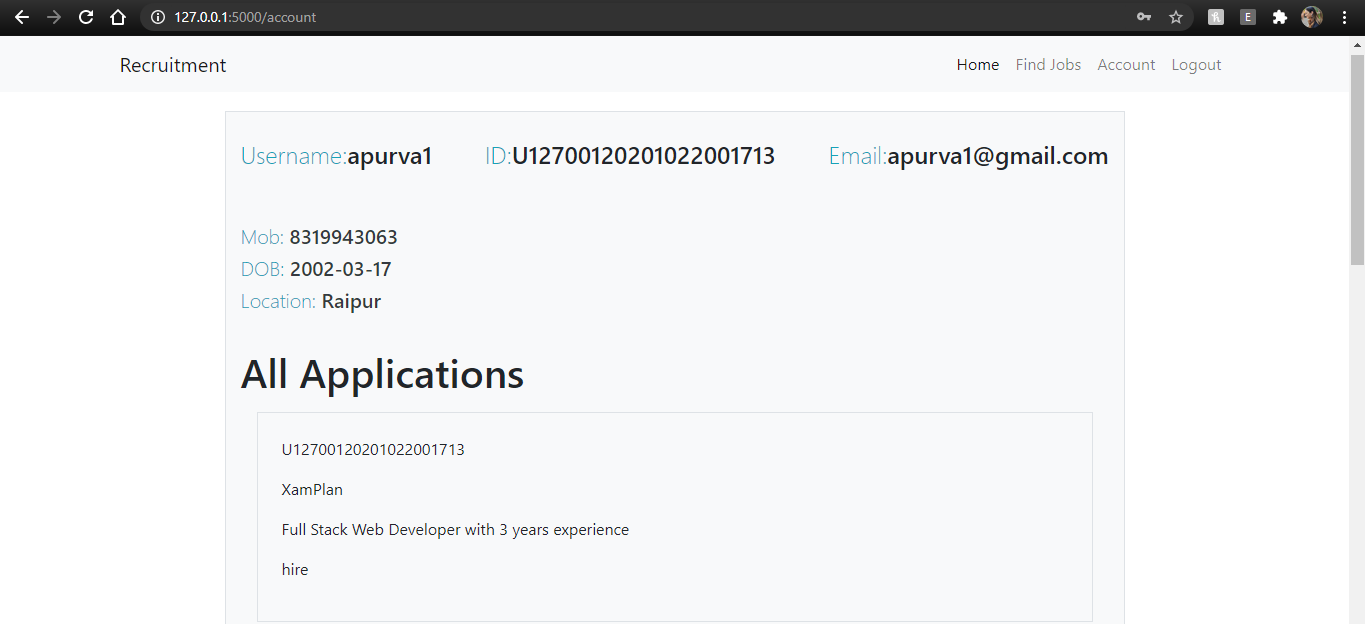


## 

In the Sign up Page you can sign up as an applicant or a company.



## The account page displays details of the user.

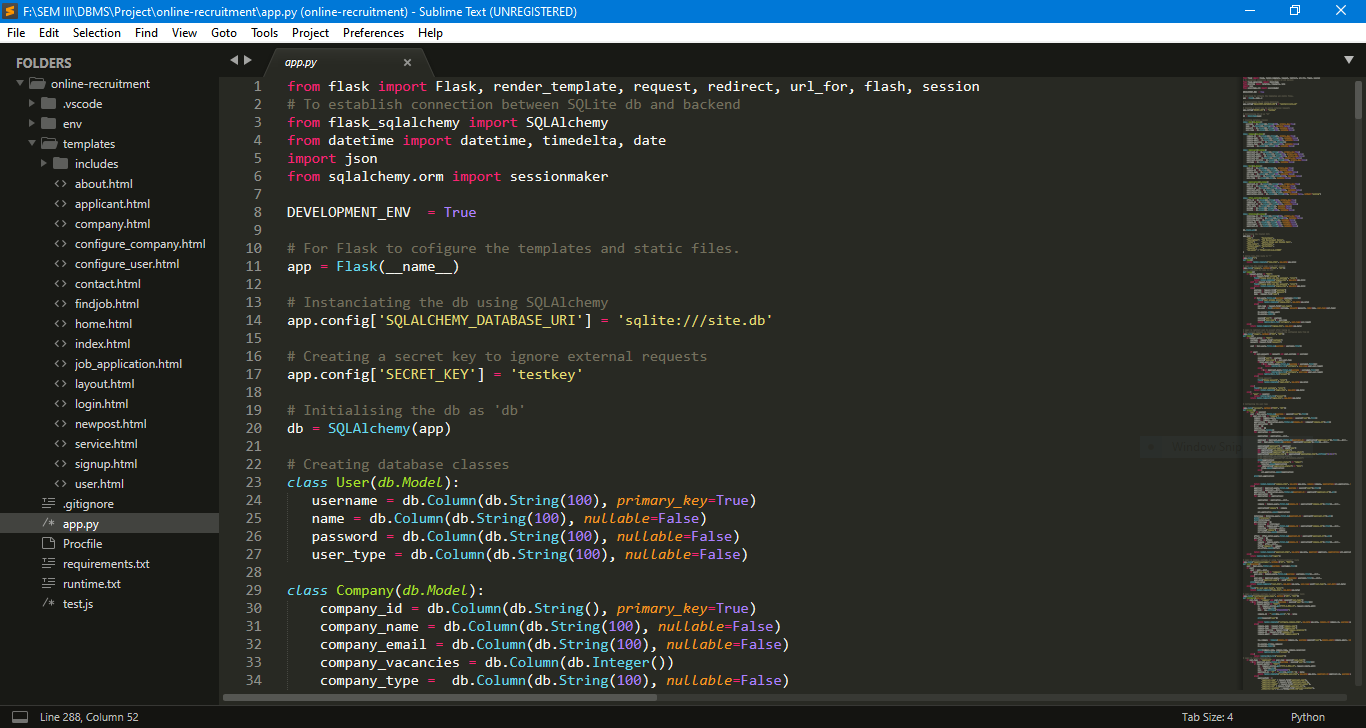


## **Back-end Implementation:**

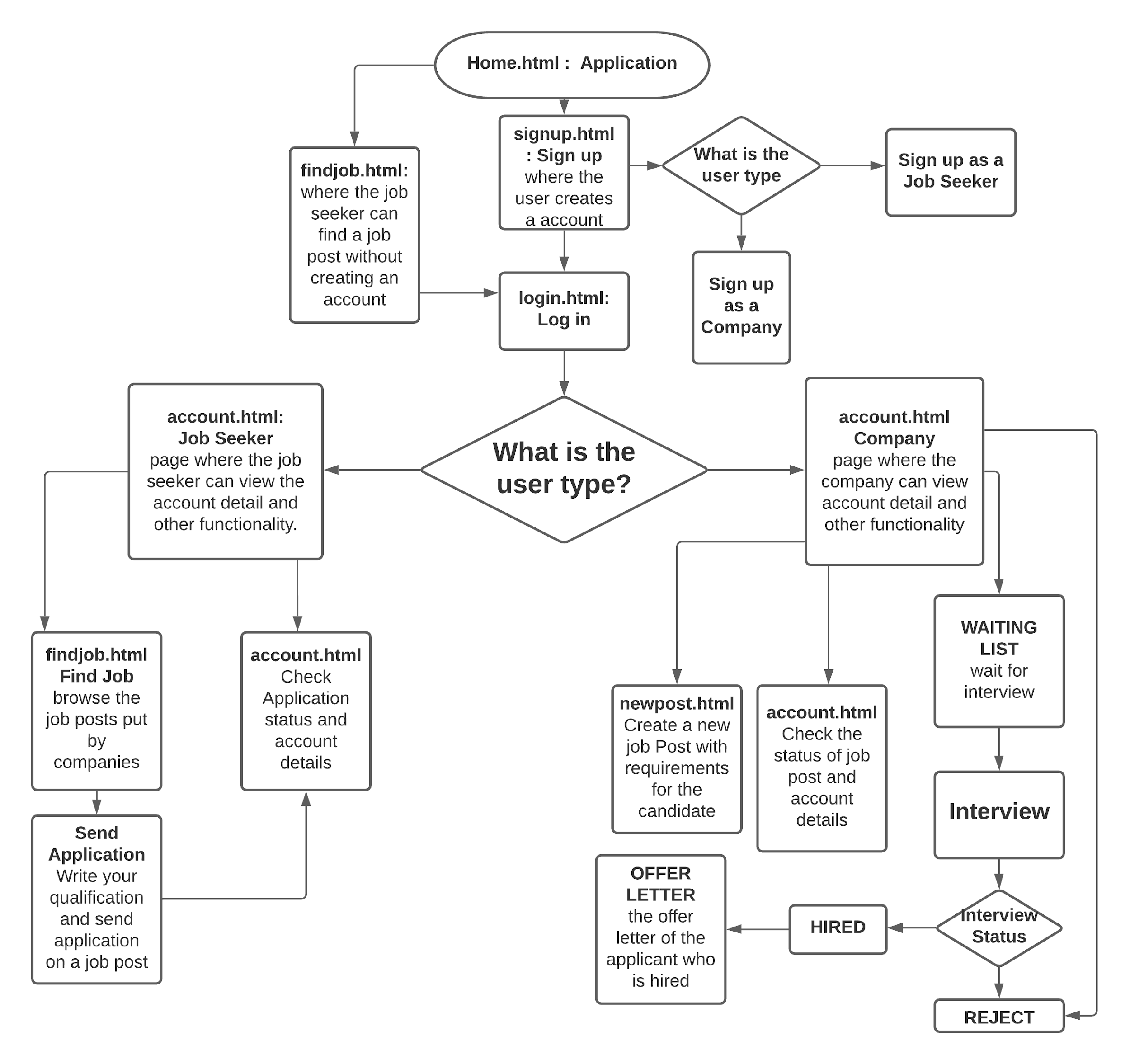
Backend implemented using **Flask** microframework for routing, backend logic, business logic, etc.

Database is implemented using **SQLAlchemy ORM** (Object Relational Mapping model) for DDL, DML, and querying data.

HTML pages use **Jinja2** Templating engine to display content dynamically which is sent through the database via the backend.

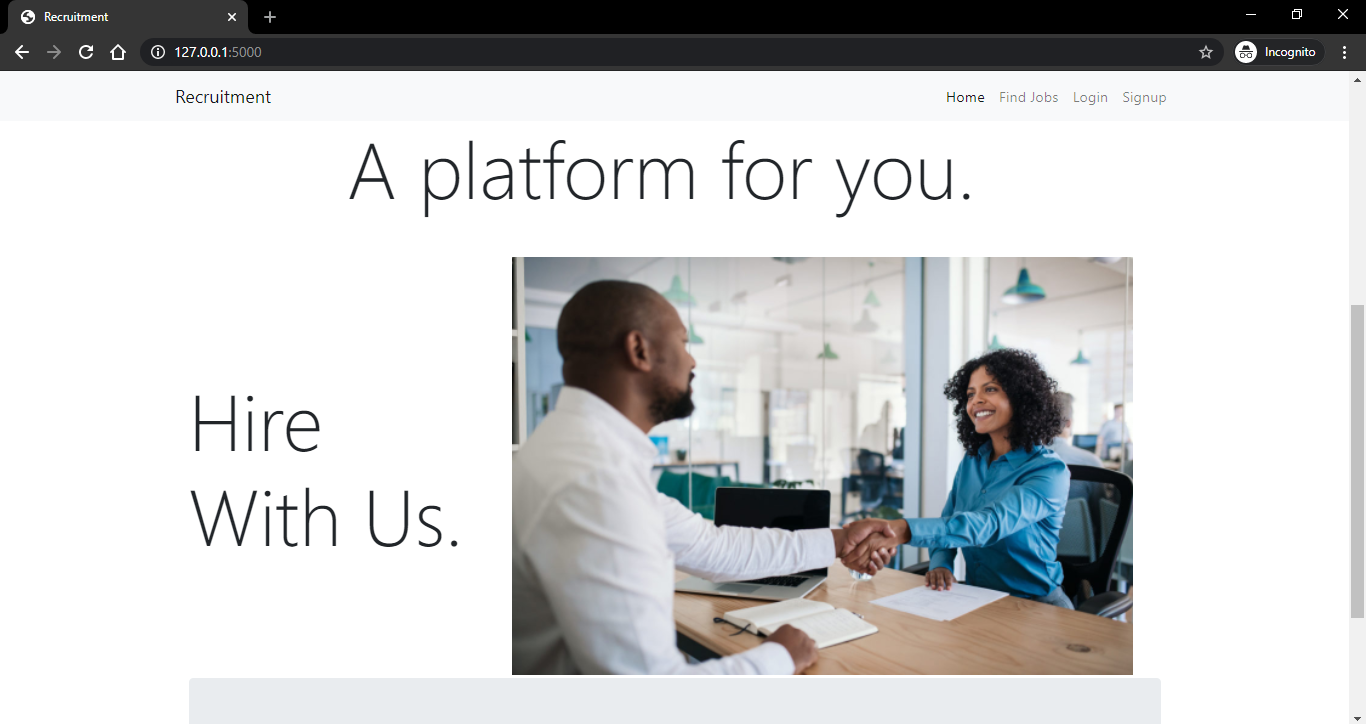


## **Flow of Control with explanation of the purpose of each interface page/form in detail:**

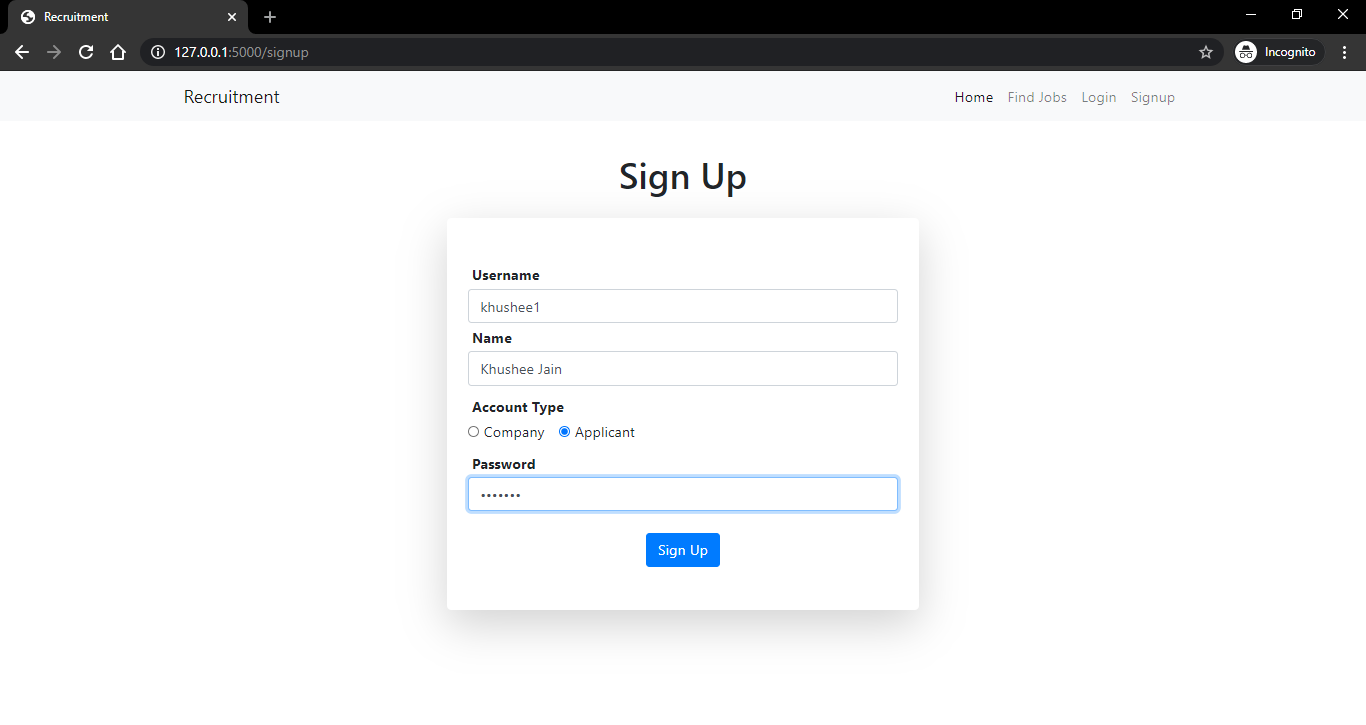


## **Screenshots of working project:**

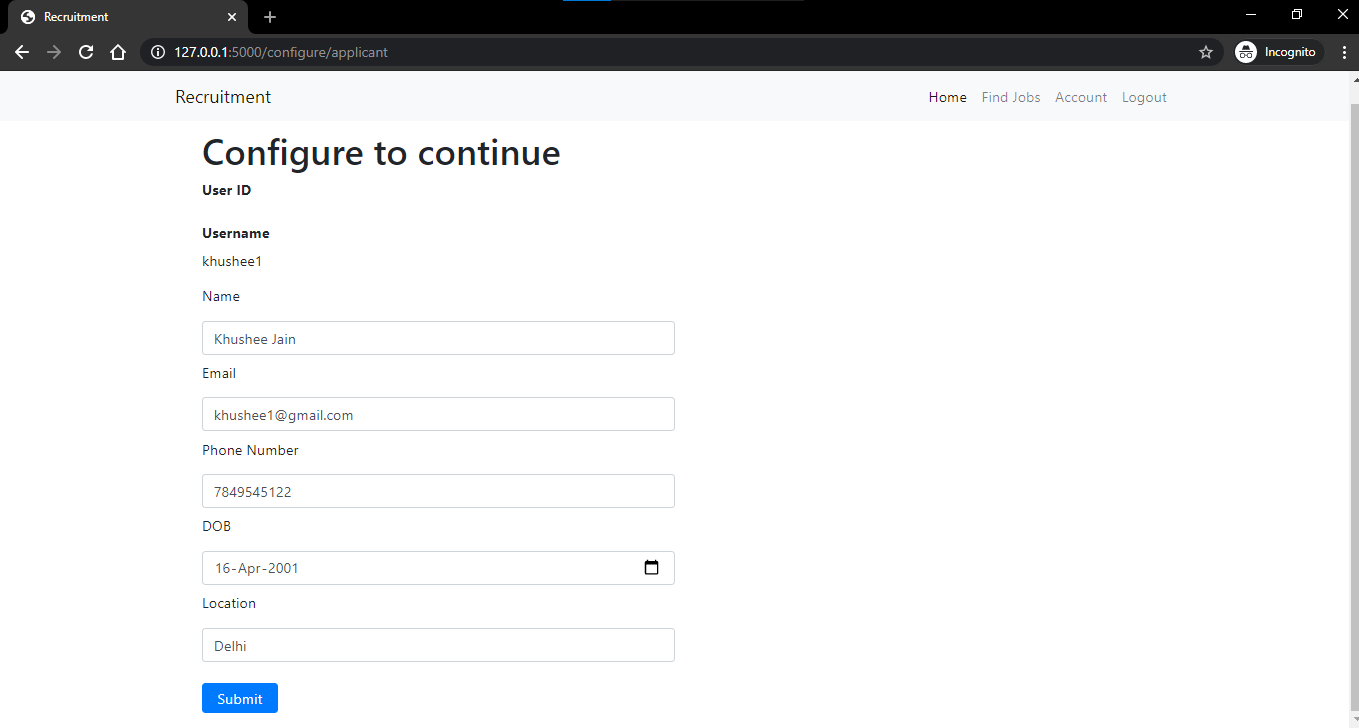
1. Home Page.



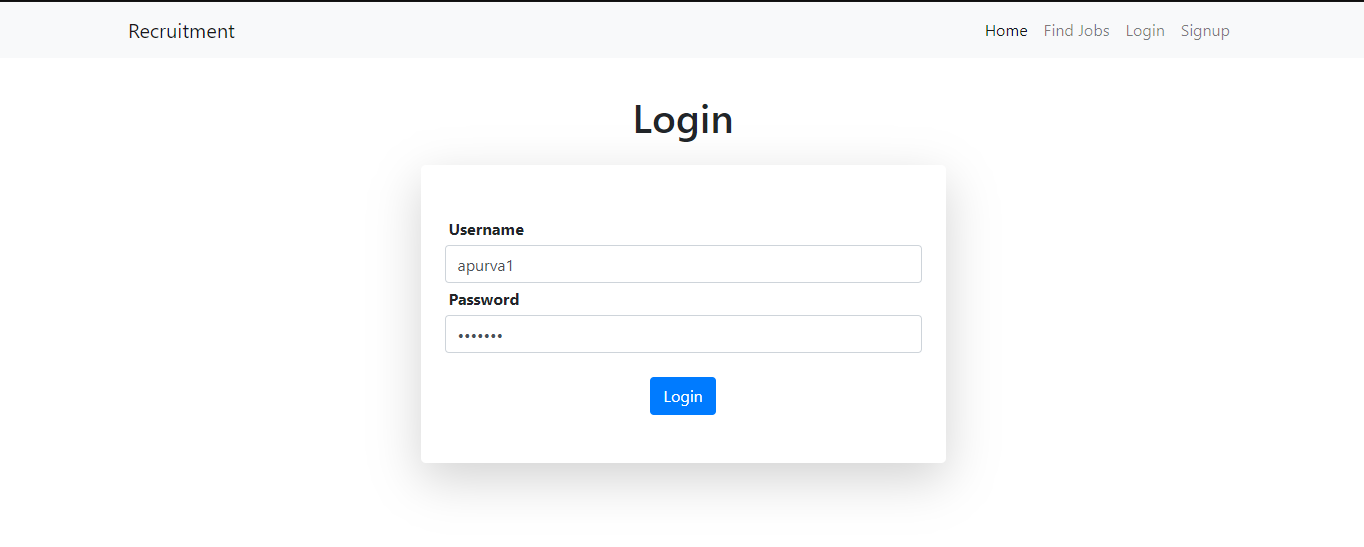
1. **Sign up Page** with Applicant as user type.



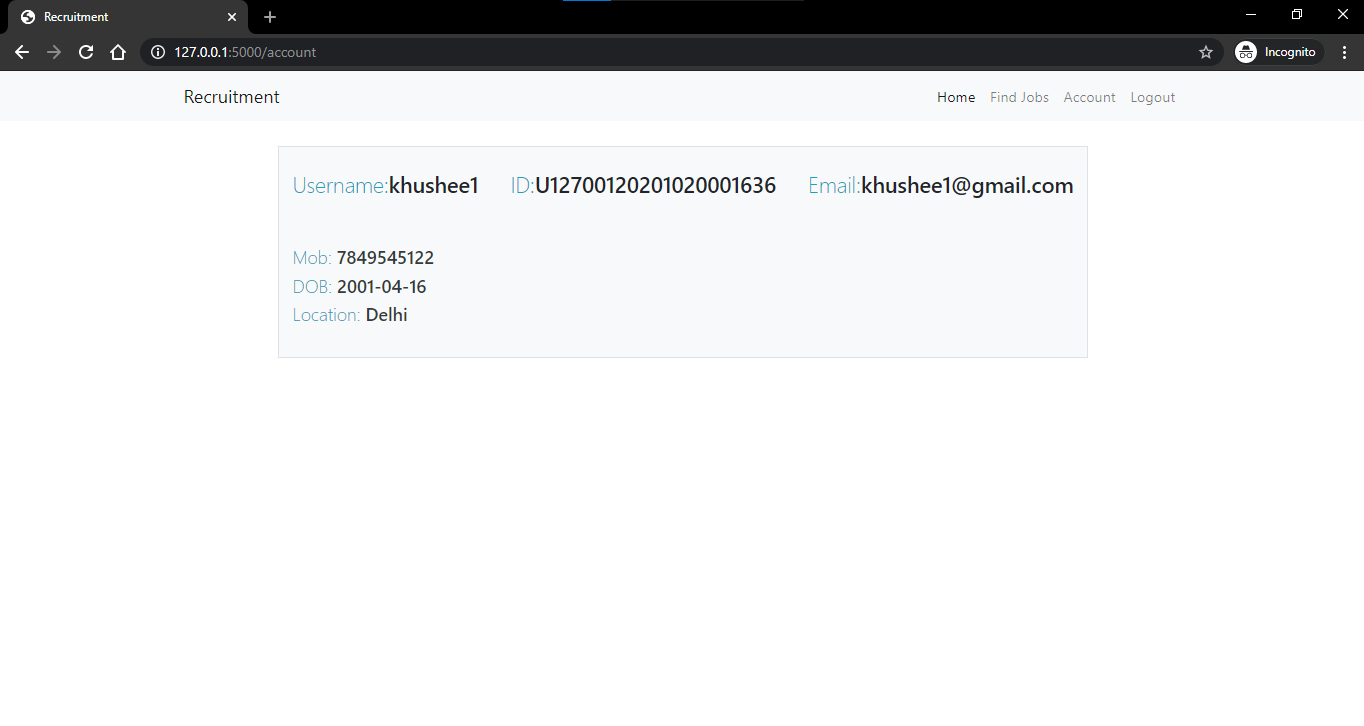
1. Configure the applicant details.



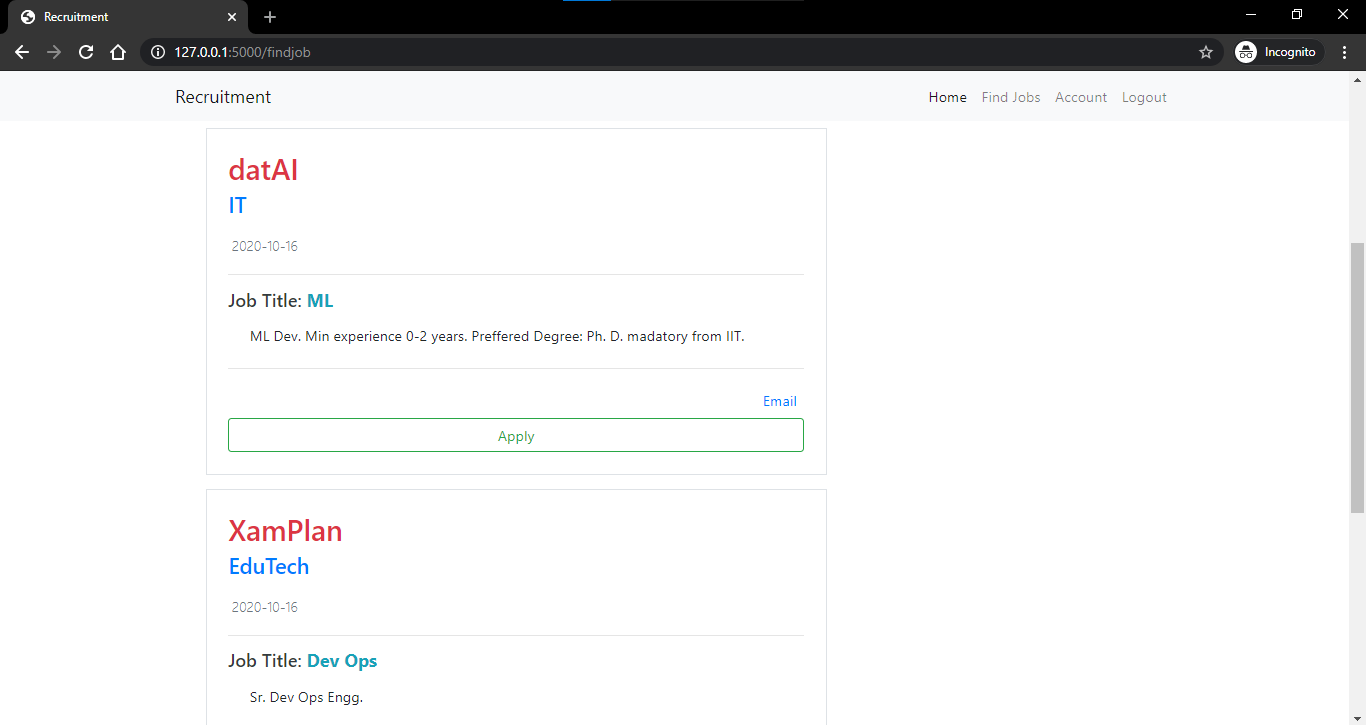
1. Login in as either an applicant or a company.



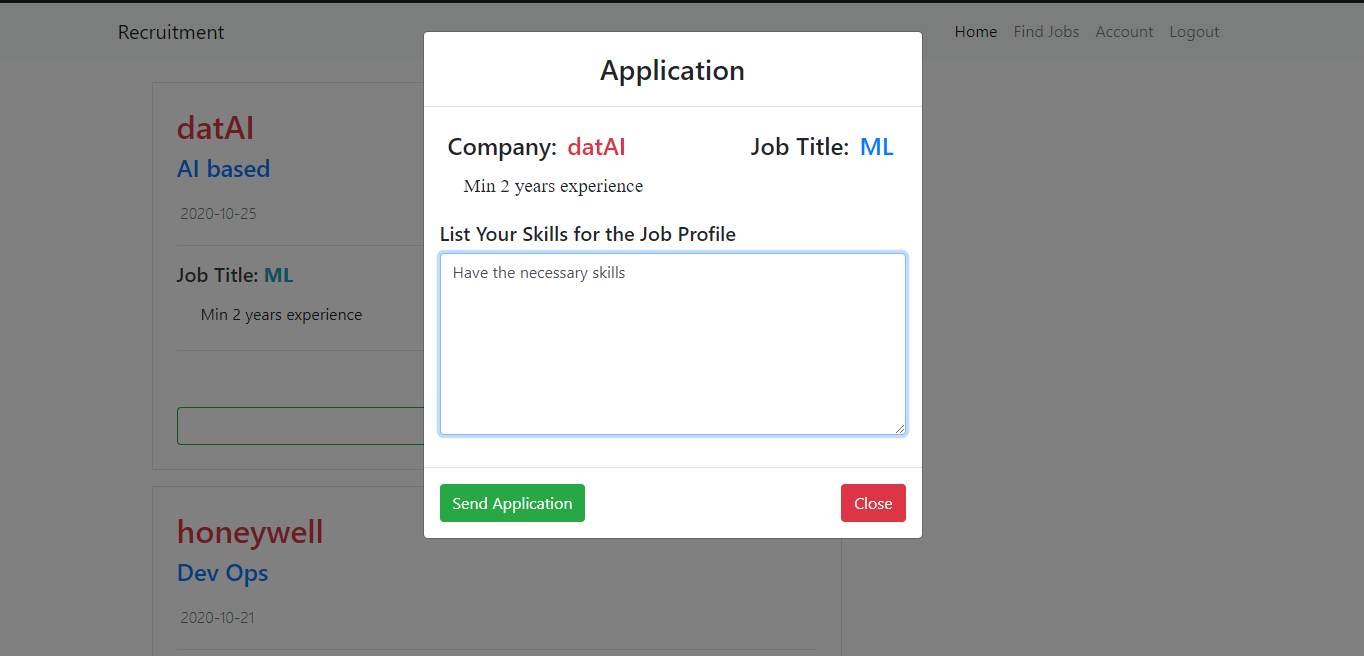
1. Account details of the Applicant.



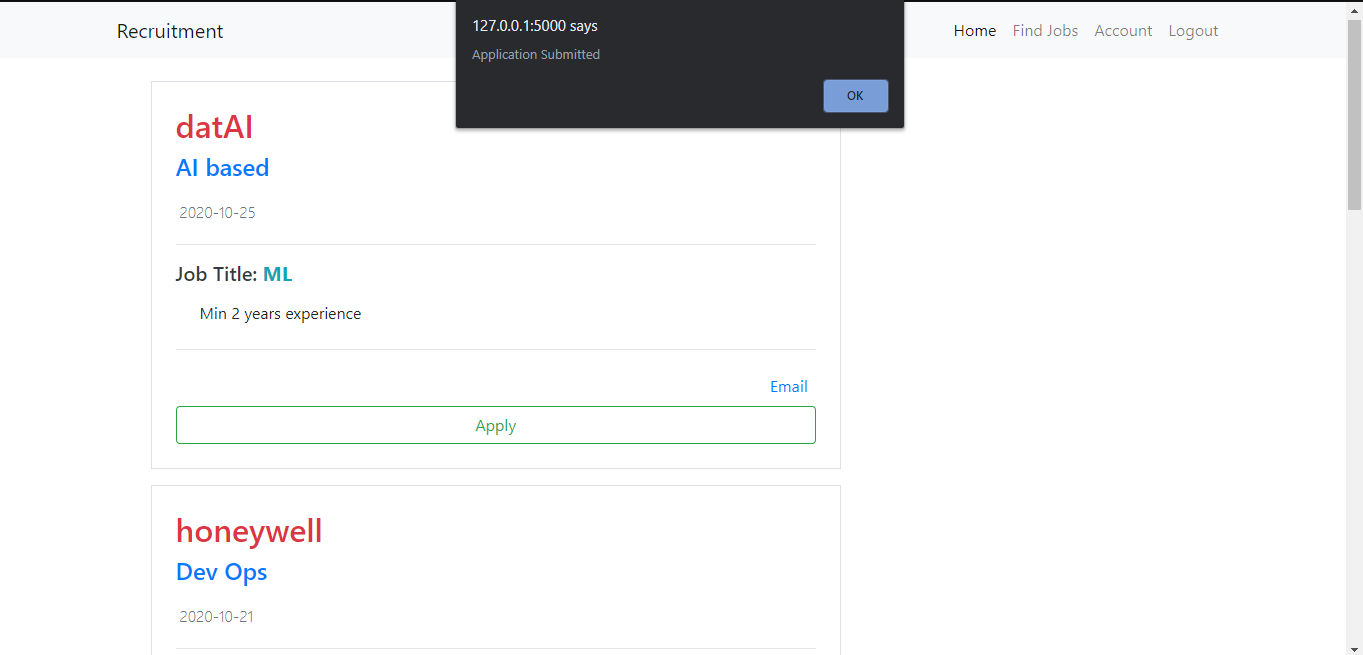
1. Find Jobs and Send Applications to the recruiting company.



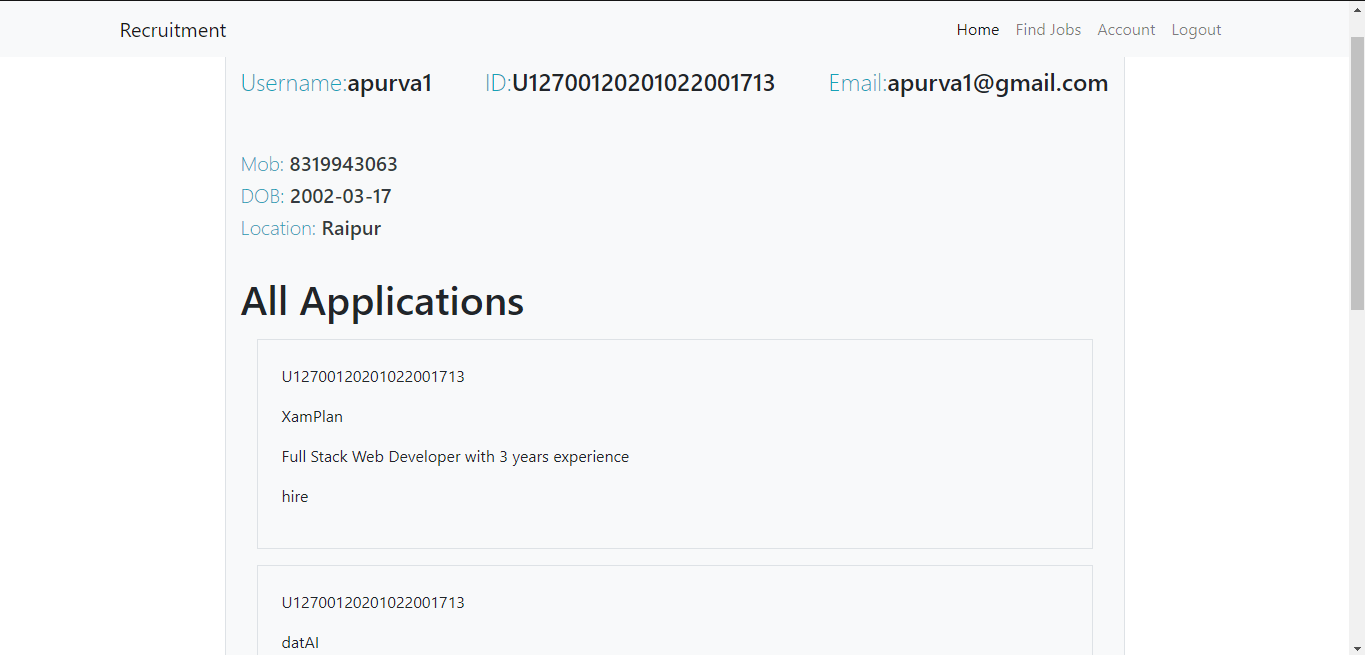
1. List your skills and create an application.



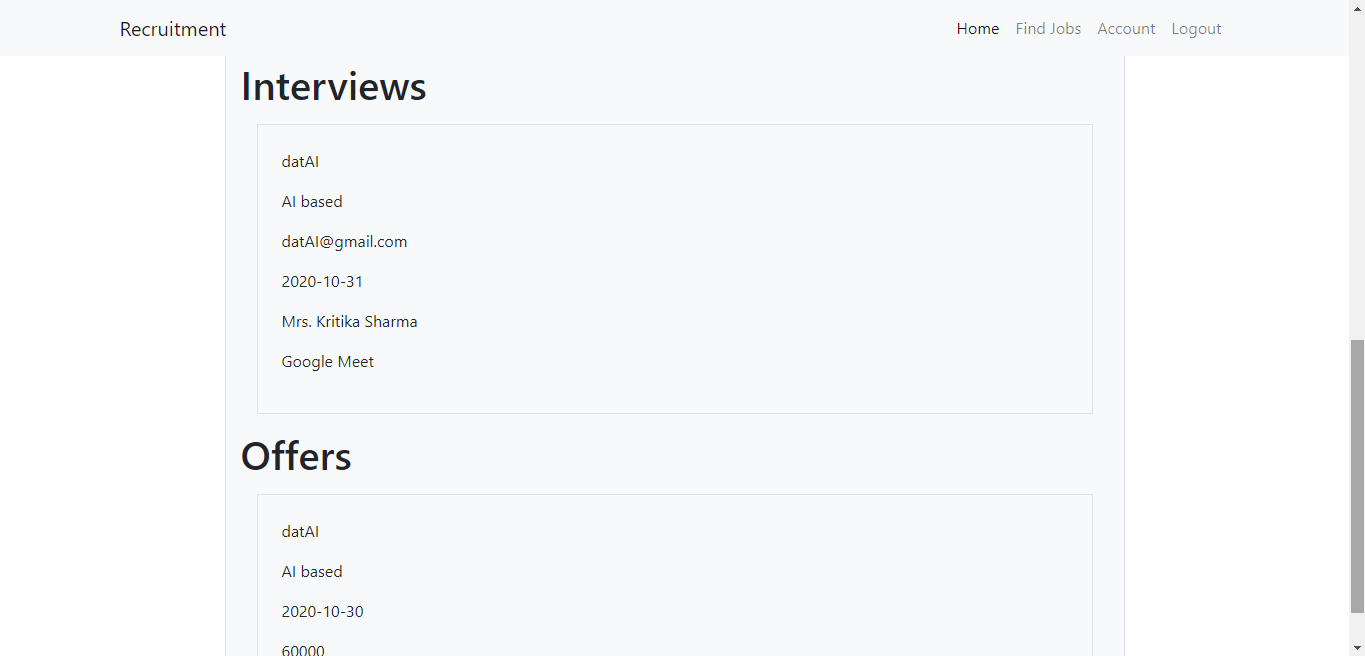
1. Prompt message after sending the application.



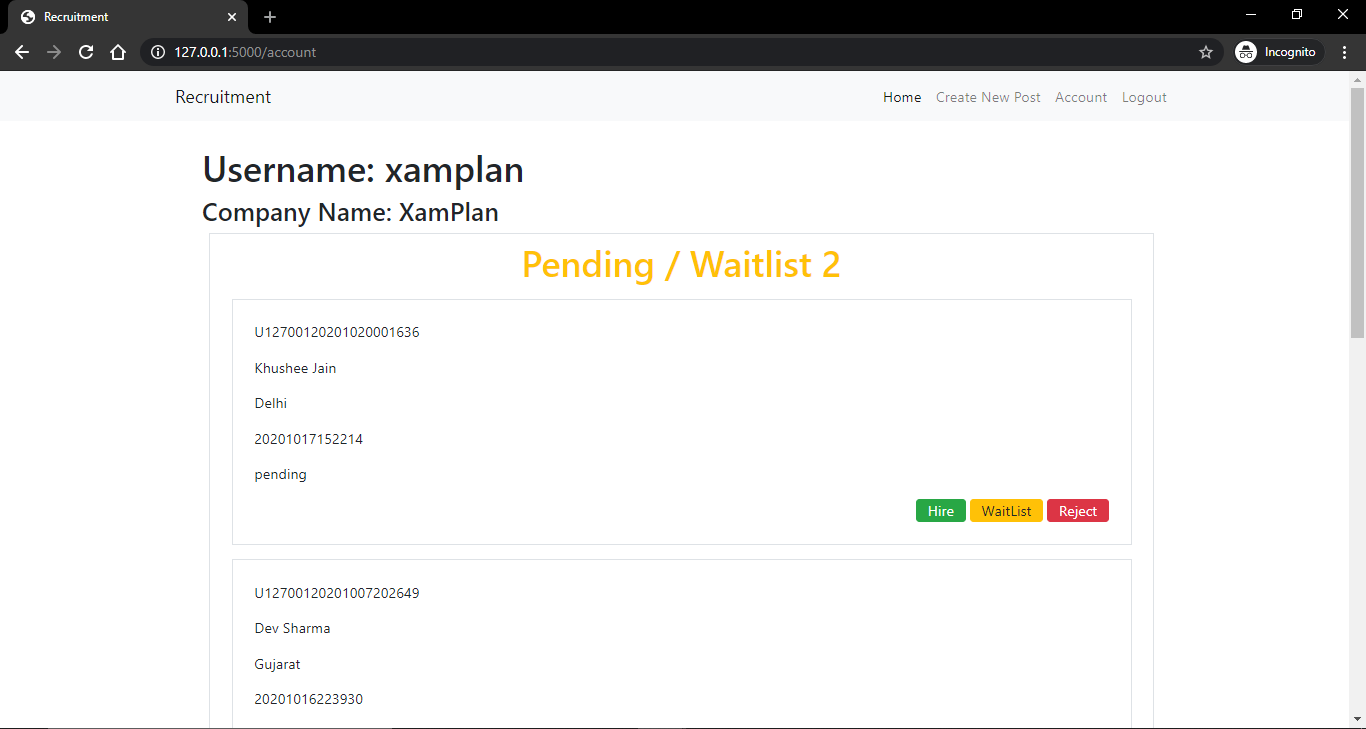
1. Check the status of all application (here: hired)



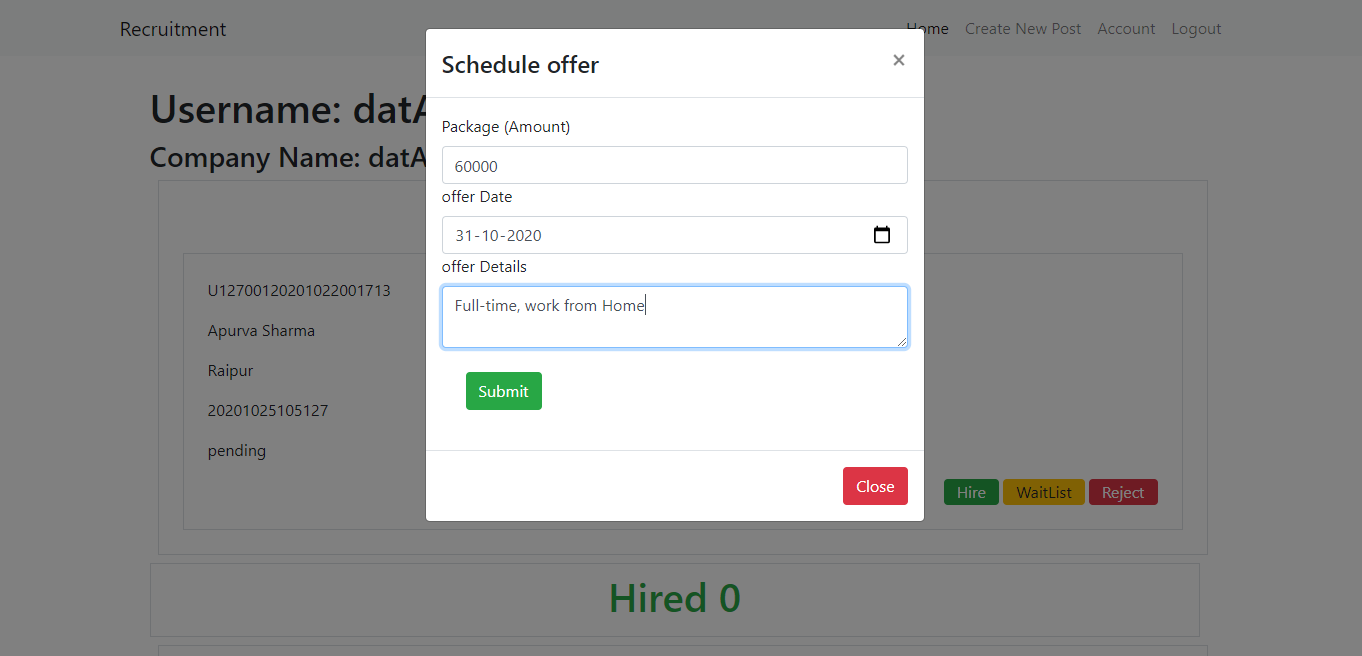
1. Check all your scheduled interviews and offers from the company



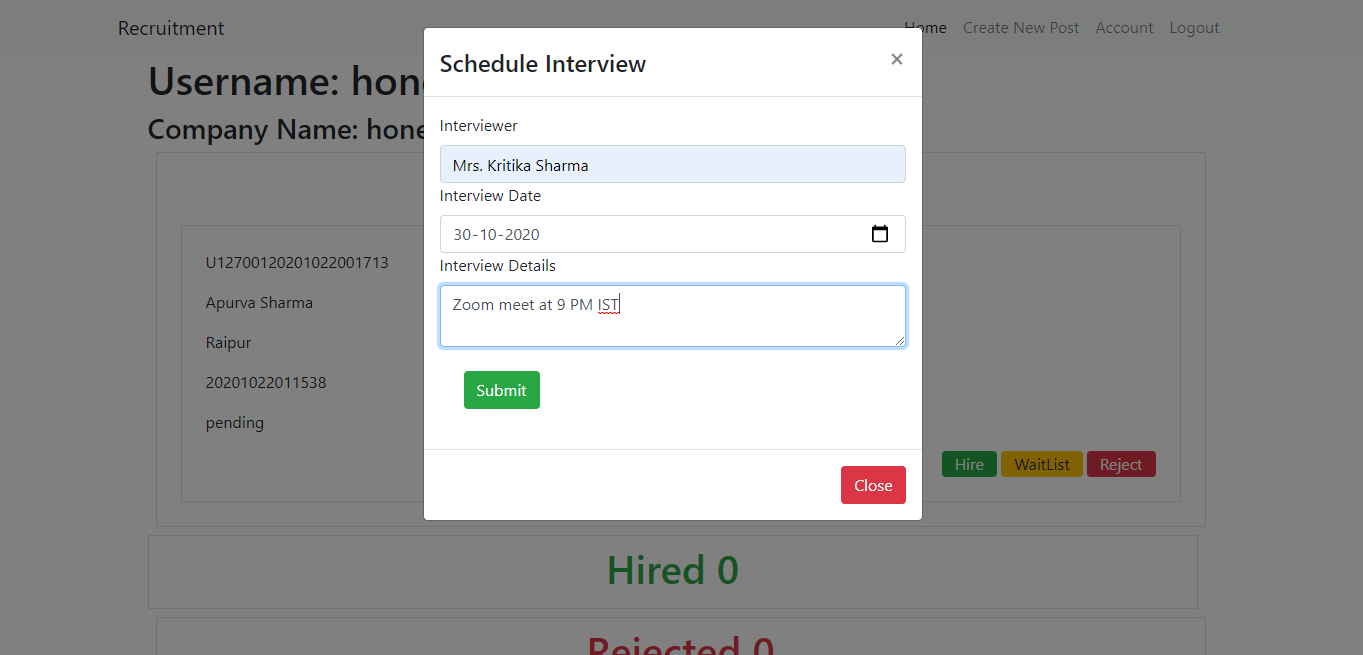
1. Add the applicant to the waiting list/ reject or hire him/her. This interface is for **Company.**



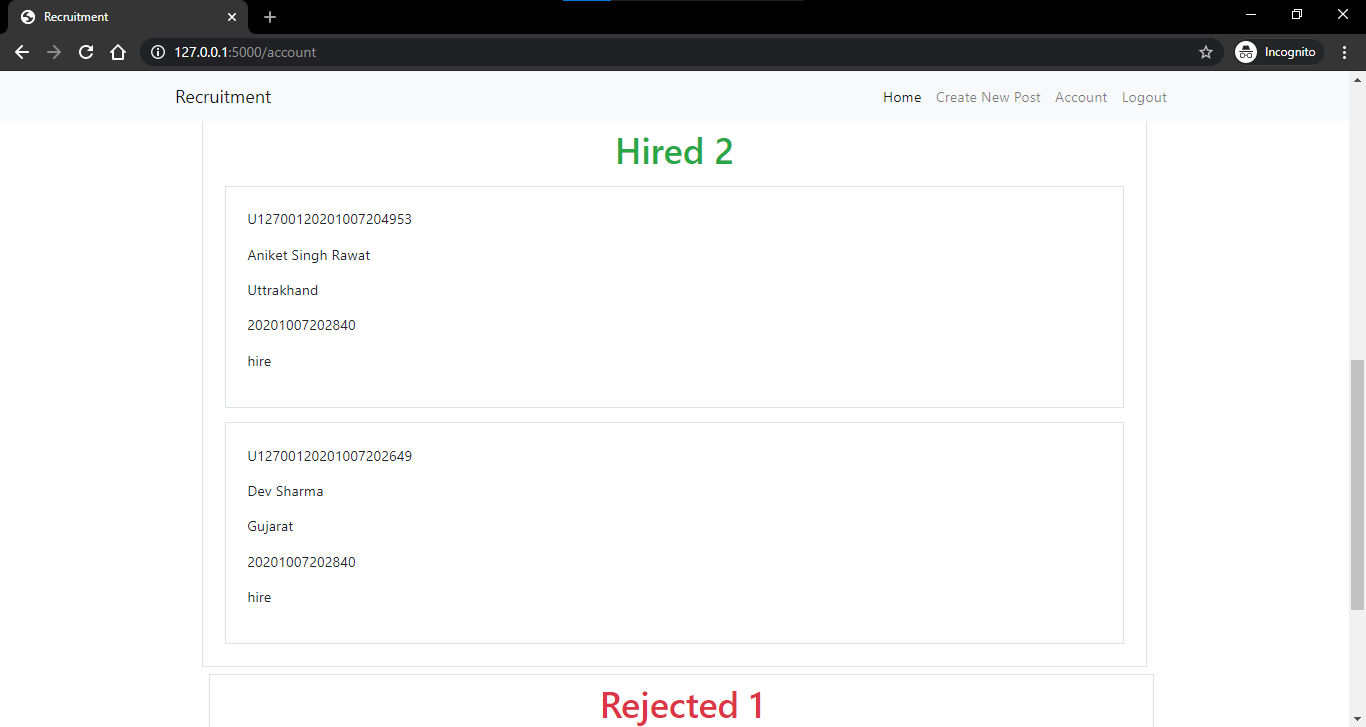
1. Add the offer letter when you hire an applicant



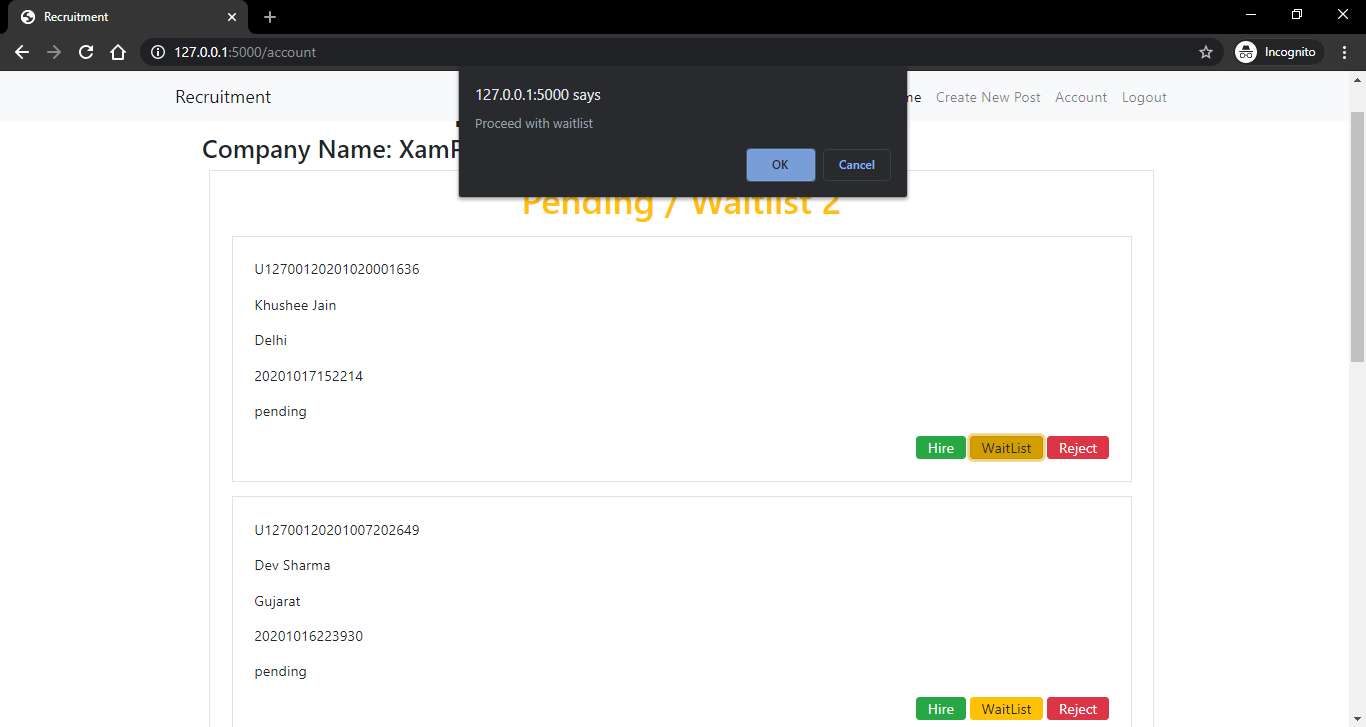
1. Add the interview if you want to add someone to add someone to the waiting list.



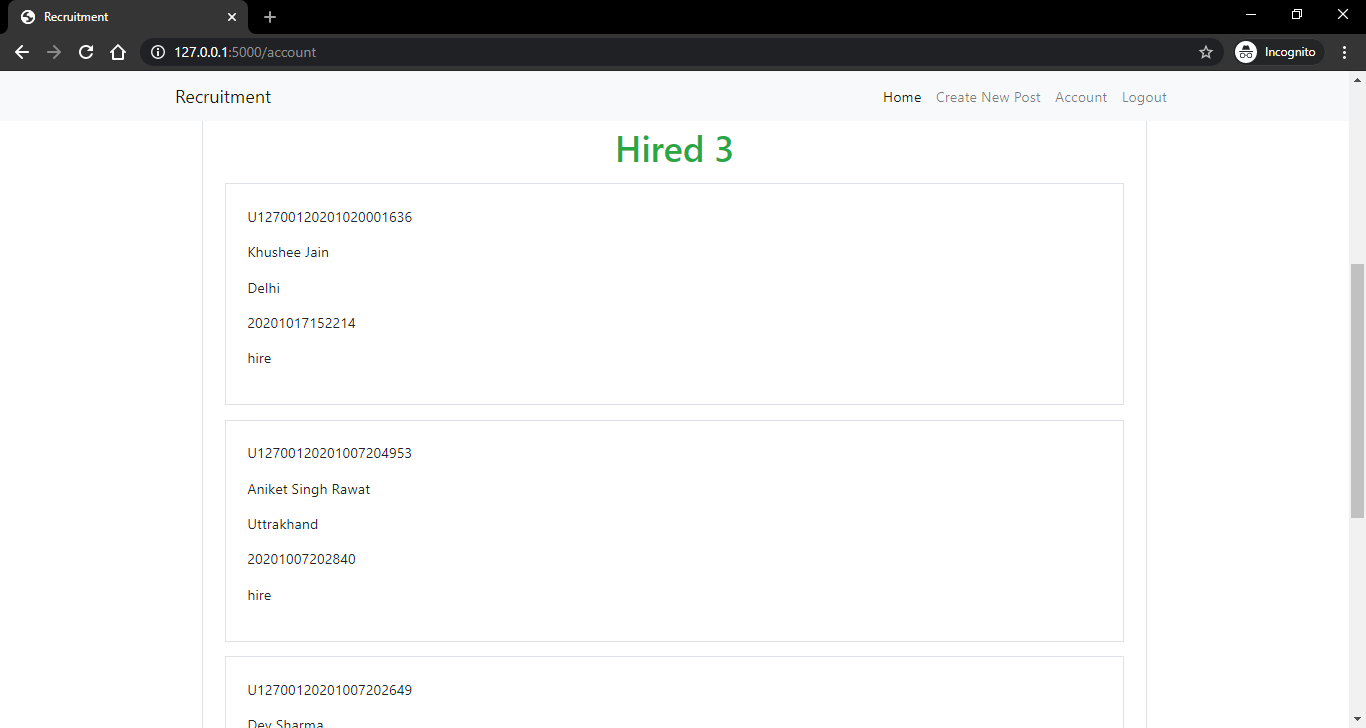
1. Check which applicant is hired/rejected or have been added to the waiting list. This interface is for **Company.**



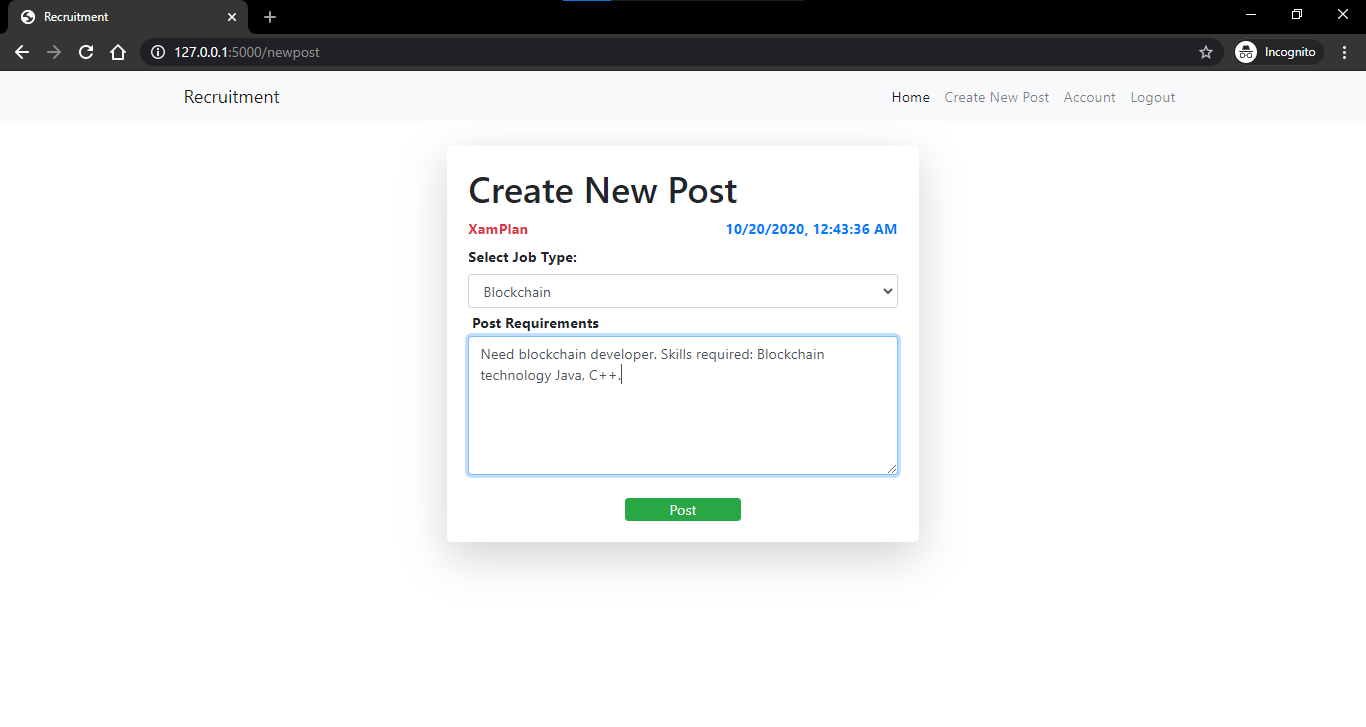
1. Hire Khushee Jain, prompt message for confirmation.



1. Now there are 3 hired people for the particular job post.



1. Create a new job post. This interface is for **Company.**

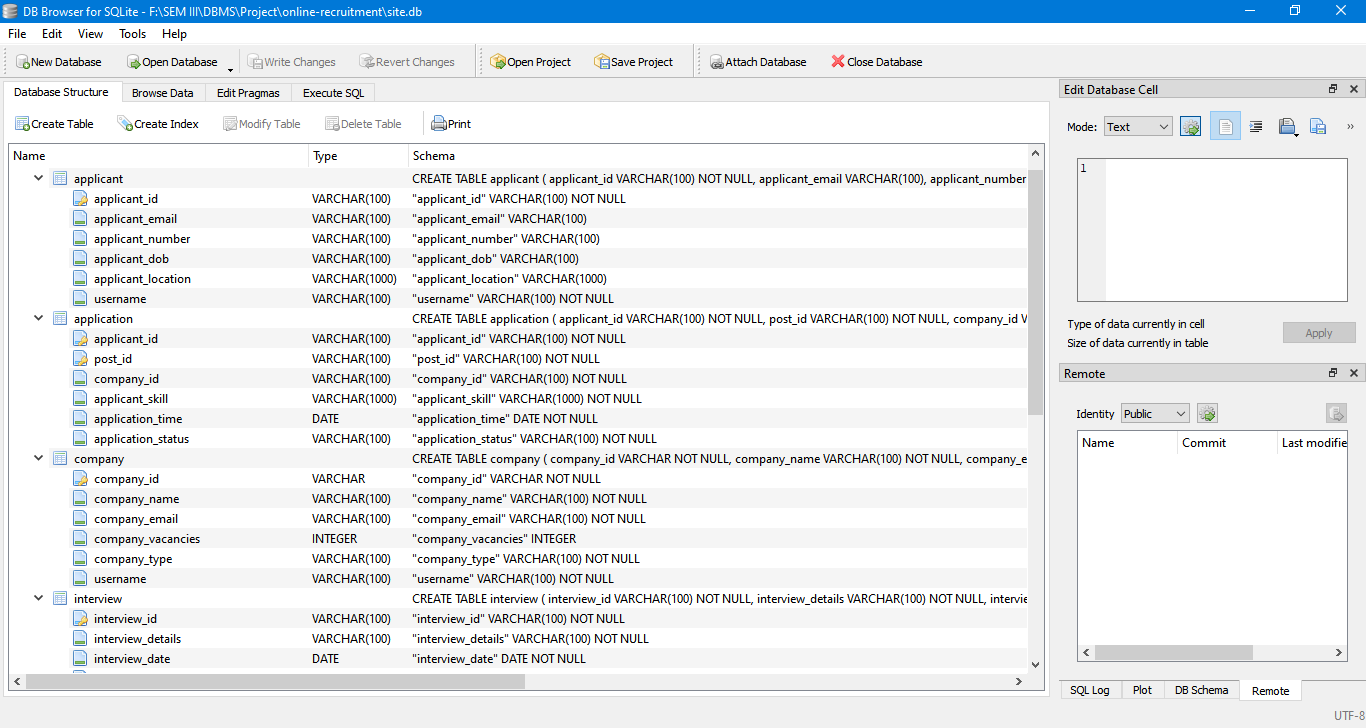


1. Find Jobs without creating an account. And Login for sending an application.



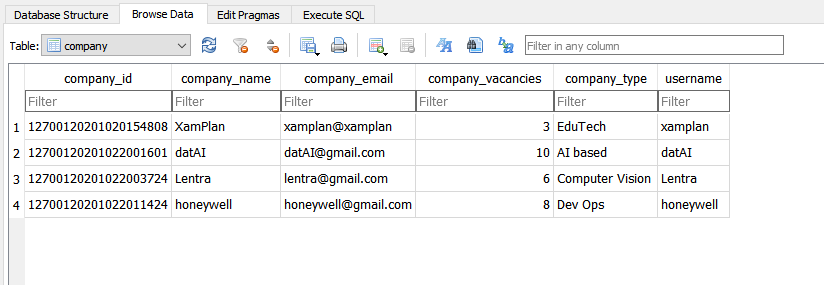
1. Glimpse of the database (SQLite) :

All the tables:



**SOME Database Tables :- Data Dictionary s**

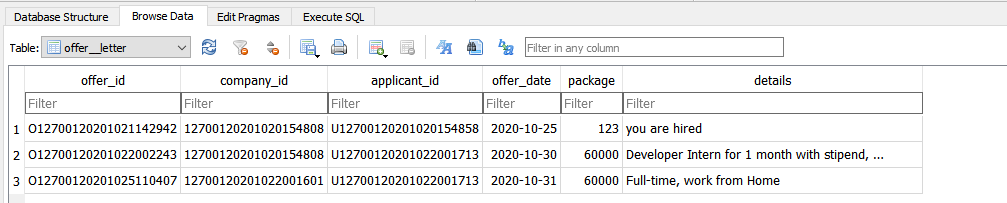
1. The company database:



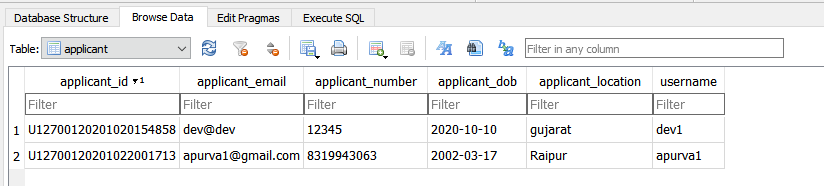
1. The application database:



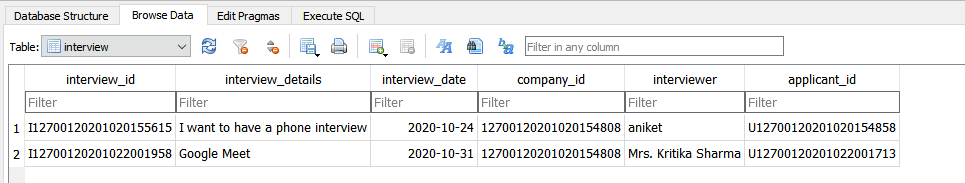
1. Offer\_letter database



1. The applicant database

****

1. The interview database



1. The post database 