InterJobs

An Online Recruitment website

Supervised By:

Dr. Laila Abd El-Hamid

Team Members

Pola Malak **Abanoub Talaat**

Kirollos Fawzy **Muhammed Bakry**

Big Thanks

First of all in behalf of all team members we would like to thank **Dr. Laila Abd El-Hamid** for her outstanding support through the year and **Eng. Amr Essam** for him being always on our side and guide us through the whole project and we would like to thank the whole faculty community for being our second home in those four years full of experiences, knowledge and new relationships and friends **Thank you.**

Abstract:

These are the documentation of "InterJobs" a graduation project for the faculty of computer and information science Helwan university ,it provides all the needed information for the undertaken project as it contains all the description needed to fully understand the project with all its details ,it contains the scope and context of the project , all the design needed for completing the project , all the planning and analysis , also provides the details of the implementation phase . The main idea of the project is to create a system that aims to facilitate the process of finding the suitable job for anyone by using fast and reliable algorithms to create matches between the right person based on his skills that he provides and his most suitable job so it helps both ordinary people and companies by matching them together and save time and effort for them it also helps in decreasing unemployment.

Table of Contents:

Chapter 1 : Introduction

In this chapter we will go through an overview of the project ,it's purpose, scope, technologies that will be used and know more about the project as all.

1.1 Description:

"InterJobs" is a web app that focus on evolving all the current recruitment system that are available now by using more smart and effective approach to deal with this problem and that it is done by implementing a system which is fast, reliable, effective yet simple for all users,This system does all the work for them and this is done like the following , imagine these scenario as someone want to find a job basically he will go to a recruitment website make an account the search among hundreds of jobs posted to find the best for him and then manually apply for each one of them ,well what "InterJobs" will do is that the app will find the best job for him according to his provided qualifications, apply for him by sending his CV automatically to the HR of the job posted and then notify him if there is any updates like interviews, more information needed, an appointment with the HR,

all useful information he may need. by doing so the job applicant will find a job as fast as possible and save a lot of time also the company will have people fitted for the position they need in no time.

1.2 Problem:

This site tries to solve the problem of work in the community where people can submit their CV to the site and thus try to find them the appropriate company to them according to the requirements of companies that offer the available jobs.

1.3 Objectives:

1.The objectives of the site are to find the right job for the people .
2. Help people to find the right company for him.
3. Help companies find the right employee according to the requirements.
4. Solve the problem of unemployment for people.

1.4 Purpose:

The main purpose of this application is to try and help people looking for job. It provides them with a good environment to seek scientific work and deal with companies and entrepreneurs and connect them to those looking for work.

1.5 Scope:

The approximate work involved to finish the project is divided into these five phases:-

1. Planning:-

• Collecting data about the project and the plenty of unemployment that made us in a need to the Web Application.

• Making surveys in different ways (questionnaires, forms … etc).

• Determining the functional and non-functional requirements.

• Setting a Gantt chart for the project.

• Determining the resources of the team.

2. Designing:-

• Determining the diagrams to be carried out within the project:-

- ERD

- Activity Diagram
- Class Diagram
- Use-case Diagram
- Sequence Diagram
- Context Diagram
- State Diagram

3. Coding:-

• The main supposed functions to be coded in this Web Application are divided into four Sections:-

1- Admin:-
- Sign up
- Login
- Add Applicant
- Delete Applicant
- Add HR
- Add Post
- Delete Post
- Edit post
- Delete HR
- Log out

2- Applicant:-
- Sign up
- Login
- Upload C.V.
- Online Interview
- Send a message
- Log out

3- HR:-

- Sign up
- Login
- Add job post
- View Jobs
- Online Interview
- Send a message
- Log out

4- System:-

- Matching: This function will match between C.V. qualifications and job requirements by finding a suitable job according to his C.V. qualifications. For example, by 80% matching, on the basis of this ratio, the HR will send him a message means initial acceptance of the job.

4. Testing:-

- Functional Testing:-

- Unit testing
- Regression testing
- Integration testing

- Non-functional Testing:-

- Performance testing
- Street testing
- Security testing

5. Documentation:-

The documentation should mainly include these main chapters:-

- Introduction: includes an overview of the project and limitations

- Project Planning: includes the tools and technologies as well as tasks and timeline plan

- Project Requirements: includes the functional and non-functional requirements

- Project Design: includes the diagrams

- Project Implementation: includes user application and administrator system

- Project Testing: includes testing types

1.6 Constraints:

• The Internet connection is a constraint as it is a must to be able to perform all the tasks.

•The Web application is constrained by a number of similar C.V.s which may slow down data analysis.

•The Web application is constrained by doing a lot of online interviews at the same time.

1.7 Approach:

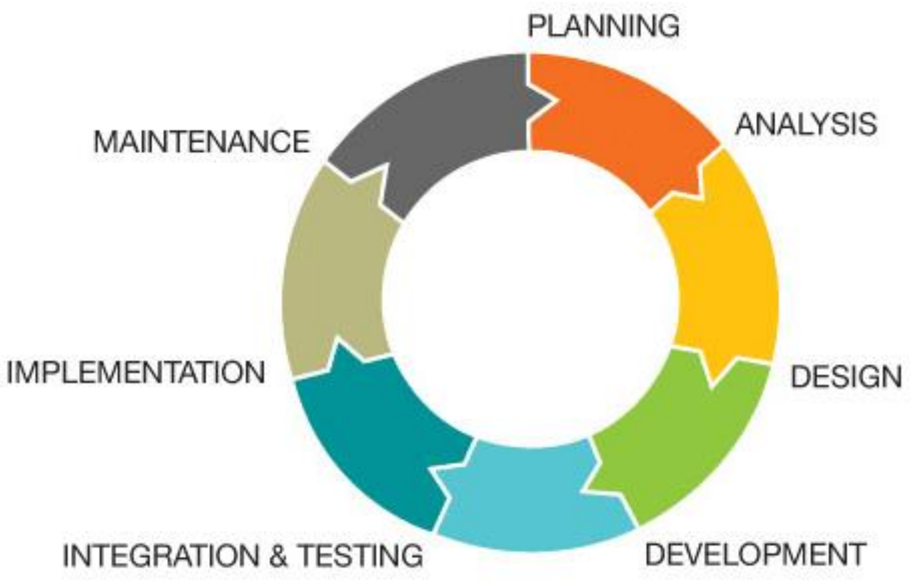
We followed the system development life cycle approach in developing this web app with its phases

1-planing 2-analysis

3-design 4-development

5-integration & testing 6-implementation

7-maintenance



1.8 Project Benefits:

1- User Benefits :

- The user can upload his CV and just wait and company will send his quiz.
- No more exhausting applying for Jobs
- The user will take small quiz to ensure that he his qualified for interview

2- Company Benefits:

- Easy to find employees to work
- Automated quizzes send to people that company want to hire
- Get more qualified people to interview that save money
- More efficient way to hire people.
- Less time needed to find a good worker

1.9 Software tools:

1- AI, sorting and mathcing algorthms That help us to match between the cv and company needs

2- Bootstrap, Html, CSS, and Javascript for Frontend

3- NodeJs And Express as Backend

5- FireBase to manage database.

Chapter 2 : Planning and Analysis

In this chapter, we will go deeper on how we planned and made the required analysis for our project it will have contents like feasibility studies, requirements, Risks, and all needed analysis

.

2.1 Project Planning

2.1.1 Feasibility Study :

A Feasibility study is used to determine if a business or a specific project is achievable, so for determining the achievability of our project we'll go deeper in the following points:-

1- Market Analysis:-

- The application talks about how to develop a working environment so that it is easy for users to find a good job opportunity as soon as possible.

- **Our app has more than a competitive edge like:**

1.The application makes it easy for users to find work easily.

2.The application makes it easier for users to find the nearest work next to the residence.

3.The application supports all regions in Egypt and the world.

4.The app contains the part of social media to make users communicate with each other and exchange experience.

5.The app has the part of ranking the result of the search according to your priorities.

2-Organizational Analysis:-

• The coverage we need to support and improve application:

We will try to develop from the application on a continuous basis so that we will make application developers and observers and will pay them monthly salaries and given these salaries, there will be a profit for the application through investment companies advertisements.

- We upgrade the application by the developers.

- We try to improve the application by developers by collecting feedback from users and their comments on the application and our consultations

3-Operational Analysis:-

We try to offer some services to clients and employees in order to make it easier for customers to find job opportunities in the near future and help them to provide a good environment and communication between users and some of them to exchange experiences among themselves and also to help each other.

- **In our project, we provide some rules and services to our employees such as:**

- An environment suitable for work
- Commitment to the deadline for delivering tasks
- Good salaries.

- **We provide our services through social communication between users and some of them and with the employers.**

- **we used Some equipment like:**

- Visual Studio 2017.
- Visual Studio code.
- Brackets.
- SQL SERVER.

2.1.2 Estimated COST:

There is a bit of adjustment that will be made to accomplish this application of meetings and programs used will provide an easy working environment for Tim. We need to meet once or twice a week and also phone calls all these costs for the application and when the implementation of the application will need a lot of money to announce the application and the emergence of people in a way Good.

Here are some examples:

1. -Purchasing programs like:

- Visual Studio2017
- Visual Studio code
- Brackets
- SQL Server 2018

2. We spent a lot of time and effort to get the best services and functions to be included in our app.

3. 5. Salaries paid to employees

2.2 Analysis and limitations of existing systems:

App logo	App Name	Easy Sign Up	International Jobs	Small time consumed to find a job
	LinkedIn	F	T	F
	Cantalop	T	T	F
	Bayt	T	T	F
	Wuzzuf	F	F	F
	InterJobs	T	T	T

(1)

LinkedIn

Description:

- **Advantage:**
    1. You Can Make important business connections
    2. Join professional groups that include like-minded people
    3. Reach out to the people who are viewing your profile
    4. LinkedIn has extensive job listings.
    5. You can write articles and publish it
    6. You can find international jobs

7. it generates CV from your profile

- **Disadvantage**:
    1. You Can't just upload your cv and wait for job
    2. You must pay to Take full advantages of this website
    3. Time investment, you must invest a lot of time to get a job using LinkedIn
    4. Privacy Concerns, its great risk to share all your information and everyone can see it

(2)

Cantalop

Description:

- **Advantage:**
    1. Easy to sign Up
    2. You can import data from your LinkedIn profile
    3. Easy to fill your own data
    4. Got many job ads in many countries

- **Disadvantage:**
    1. So slow in browsing
    2. User session expire so fast, so he had to login a lot
    3. There is no chatting available with other accounts

(3)

Bayt

Description:

- **Advantage:**
  1. It provides professional CV making
  2. You can save your favorite job ads, so you can see it again easily
  3. You can get messages of other users
  4. Provide good categorization for jobs
- **Disadvantage:**
  1. Got few job ads
  2. Got bad design for logo and website
  3. You must pay to gain full access
  4. Jobs ads not well organized

(4)

Wuzzuf

WUZZUF
Egypt's #1 Online Recruitment Job Site

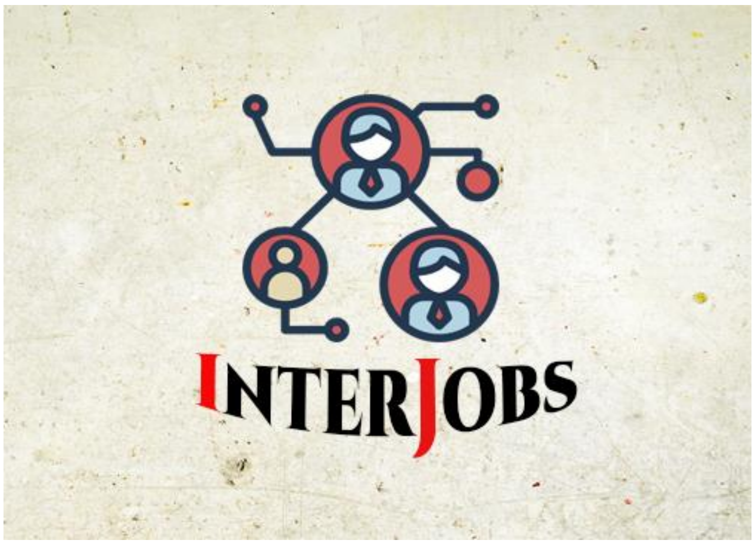
Description:

- **Advantage:**
    1. Got many jobs in Egypt, KSA and UAE
    2. Send Emails with suggested jobs
    3. Got many connections so he can know the new jobs
    4. So famous in Egypt
    5. Got good design and user friendly
- **Disadvantage**:
    1. Got jobs only for three countries
    2. Limited job applying
    3. Sending too many emails

Our project:

(5)

InterJobs



Description:

- This web app enables users to upload there cv to find a job or post job and the system will help you to find the best workers there are two types of users

1.The applicant that upload his cv and wait for the job

2.The company that post job and system will suggest some workers and give them a quiz to make sure that they are good enough for this job

- **Features:**
  1. Our web app contains jobs all over the world
  2. It easy to sign up to our web app
  3. You will have to just put job title that you want and upload your cv and we will do the rest
  4. The Company will upload job requirements and we will match and suggest some workers if the company want to interview them, we will send quizzes to make sure the workers are qualified for this job

2.3 Need for the new system:

This web app solves serious issues that are time invested in searching for a job and time consumed by the company in filtering cv and get the best applicants that same money and time for both sides. We thought of this web app because of challenges that face the company and applicant like.

1. Time consumed by company to filter and interview many applicants by sending quizzes for best cv for this job
2. Easy to sign up
3. Provide jobs in everywhere in world
4. Well categorized jobs ads
5. Free full access you don't have to pay to use this web app.

2.4 Analysis for the new System:

2.4.1 User Requirments :

Features for the ordinary user (Job Seeker):

*User can create a personal account fully detailed about himself so that it can show a complete image for the applicant to companies any account will include :

-The User Full Name

-Residence place (Must be specified as job Searches and matching will be affected by the place and region)

-education level and degrees

-telephone/Mobile

-Email

-Personal image(Optional)

*User can view and edit information in his account.

*User Must add a CV that follows certain guidelines so that the cv would be used in the matching process done by the site

*User can Edit CV or upload new versions of it anytime

*User can search for a job with applied filters this may include:

-specified category

-specified location/region

-Job post Date

*User can apply to any job he wants

*User can message and chat with HRs and other users

*user can deactivate his account

Features for Companies (HR/Recruitment):

*HR can create an account for the company as a workplace and can also edit information created any time

*HR can post a Job Post any time specifying it's availability over time and specifying its details which include :

-job name

-job category

-preferred job requirements

-preferred years of experience

-expected salary or salary range

*HR can edit or cancel any job he already posted

*HR can view the recommended applicant (from the site) and those ordinary applicants

*HR can choose the suitable Applicant for the job

*HR can determine and schedule an online interview with any applicant

2.4.2 System Requirments:

- **Hardware requirements:**

-A modern PC is needed that comes with a keyboard and mouse to be able to use the website and interact with it easily.

-A SmartPhone can be used with any operation system Android, iOS, windows...

- **Software requirements:**

-Any browser that supports HTML5 can be used to be able to access and surf the site example:

-Mozilla Firefox

-Google Chrome

-Opera

- The user must have a reliable internet connection as all the processes is done online and any unreliable connection could lead to a loss in data between transferring it from user to the server and vice versa.

2.4.3 Domain Requirements:

1) Multiple users must be able to use the application simultaneously without corrupting the database (whatever form it may be).

2) The programs to run this application on the web are Google Chrome, Firefox, Internet Explorer and some other engines.

3) This application takes data from the CV of each client and saves it in the database and then compares them to the requirements of the companies.

4) Each customer and each company has its own email and a data warehouse of its own, with every new addition added by the client or companies.

5) The comparison is carried out in the application by means of similar words in the CV of the client and their conformity with the functions offered and available in the companies.

6) A server must be set up to host the database, and the server must be accessible by all the systems running the inventory tracking software.

7) The database should be backed up every once in a while in case the original does become corrupt.

8) The application must verify all values before making the change in the database.

9) The application must have update capabilities for future models and accessories.

2.4.4 Functional Requirments:

1-User Register:

Description: first The user should register a new account to be able to use the website

Input: User enters his information with all its details such as first name, last name email, password, phone, address, picture, type (job seeker, HR) and so on.

Output: an account has been created and saved in the database.

Actors: clients, HRs.

2-Sign In:

Description: a sign in form so users can access their accounts

Input: user email and password.

Output: user account opens.

Actors: Clients,HRs,Admins.

3-Edit Account:

Description: a function that gives the users the ability to change and edit information in their accounts.

Input: new data (text, numbers)

Output: data saved and replaced old information.

Actors: Clients, HRs

4-Add CV:

Description: Client adds his CV to complete the profile

Input: a CV with (Pdf,word)extension.

Output: CV has been uploaded and saved in the database.

Actors: Clients.

5-Edit CV:

Description: Client can change his old saved CV.

Input: new CV.

Output: new CV is saved and replaced the old one in the database.

Actors: Clients

6-Search Jobs:

Description: Clients can search for jobs in different categories.

Input: Job name and category and region.

Output: a list of all available jobs.

Actors: Client

7-Apply for a job:

Description: Client can apply for any job position he searched for.

Input: an apply confirmation from the user.

Output: his application is saved and send to the HR who posted the job.

Actors: Client.

8-Chat/Messages:

Description: all users can send messages and chat with other users.

Input: a message written by any user(collection of text and numbers).

Output: the message is sent successfully to the right user.

Actors:Admins,Clients,HRs

9-Add job:

Description: HR can make a job post with his preferred description.

Input: a Job form is made with all requirements and information filled**.**

Output: the job form is saved and posted in the right category.

Actors: HRs.

10-Edit/Cancel Job:

Description: HR can edit information for a job he posted before or cancel it completely**.**

Input: new information is entered for a certain job or cancelation confirmed**.**

Output: The post is updated and saved or canceled and deleted from the database.

Actors: HRs

11-View job applications:

Description: HR can view all applications sent to the job he posted.

Input: an order made to view all applicants.

Output: a list of all application that are saved for these specific post.

Actors: HRs.

12-Choose suitable applicant:

Description: HR choose a suitable applicant for the job.

Input: an application **is selected.**

Output: the applicant is chosen as the suitable person for this Job and will be notified**.**

Actors: HRs.

13-Schedule an interview or quiz:

Description: any HR can make an appointment with a certain date with any applicant to the job he posted.

Input: a certain date is declared for a certain candidate as an interview or quiz.

Output: a notification is sent to the applicant with the date stated.

Actors: HRs.

14-Create an online quiz:

Description: HR can make an online quiz for one or more applicants.

Input: an MCQ quiz form is created with a declared deadline.

Output: a notification is sent and then the quiz to the selected applicants**.**

Actors: HRs.

15-Deactivate Account:

Description: any user can deactivate his account at any time.

Input: Deactivation account Confirmation is sent to the system**.**

Output: the specific account is deactivated and deleted from the database**.**

Actors: clients, HRs.

16-View full account details:

Description: admin can view any account with all its details.

Input: any account admin choose.

Output: a page with all account information is viewed.

Actors: Admins.

17-Edit Accounts:

Description: Admin can edit any information he wants in any certain account.

Input: new data (text, numbers)

Output: data saved and replaced old information.

Actors: Admins.

18-Delete Account:

Description: Admin can delete any account from the system.

Input: any account admin choose.

Output: the Selected account is deleted from the database.

Actors: Admins**.**

19-Edit or delete Job Post:

Description: Admin can edit any information in any job post.

Input: new information is entered for a certain job or cancelation confirmed.

Output: The post is updated and saved or canceled and deleted from the database.

Actors: Admins.

20-Notification:

Description: a notification with a certain message is sent to users to inform them with information.

Input: message content (Chosen for job, Interview, Quiz) with dates if needed.

Output: notification is sent successfully for the right user.

Actor: System.

21-Matching:

Description: the system matches between a job posted with a suitable CV(s) saved in the database.

Input: job requirements and CV(s) saved

Output: a match is made by the system matching algorithm and the CV is sent to the HR.

Actor: System.

2.4.5 Non-Functional Requirments:

Accessibility:

The system is accessible anywhere if there is an internet connection and also it’s simple and not complex design doesn’t require a high internet speed it is also accessible on a variety of platforms and screens sizes because of its responsive design.

Availability:

All system pages are available in normal state and it doesn’t take a long time to be executed any page that will take a relatively high time to be processed will notify the user on when the system is going to be up again.

Efficiency:

The system should handle any situations of high data capacity and also it's throughput is efficient and reliable and could complete a large number of processes per unit time, also the response time should be very minimal and the time difference between submission of a request and the response is very small.

Reliability:

The failure rate of any process should be very low as the system is consistently perform the specified functions smoothly and without failure.

Safety:

The system does not cause any harm to any person or the environment as it helps to eliminate the unemployment problem.

Usability:

The system is designed with a simple UI without any complexity so that anyone even without any technical knowledge could use and operate the system easily also the interaction between the system and the user would be done by a very simple language so no misunderstanding or problems will be made.

Confidently:

The system will be very confidential about all user sensitive data and would be stored safely and only authorized users are able to access also no data would be given to anyone or shared to the public without the user approval.

Access Security:

The system is safeguarded against deliberate and intrusive from faults internal and external sources also the database would be very safe against all injection and unauthorized access.

2.5 Advantages of the new system:-

1. InterJobs app uses more efficient ways to hire people.

2. InterJobs app helps the applicant to find a suitable job for him easily in many fields.

3. InterJobs app helps companies to find a suitable employee easily.

4. InterJobs helps companies in less time needed to find a good employee.

5. InterJob helps companies to get more qualified people to interview that save money.

6. InterJobs app uses artificial intelligence (AI) to matching between C.V. qualifications and job requirements and this point makes finding a job easier because the applicant doesn’t exhaust to find a suitable job.

7. InterJobs app helps companies HR to hold a small quiz for the applicant after the matching process to ensure that he his qualified for interview.

2.6 Risk and Risk Management:-

Risk management process:-

1- Risk identification:-

Risk Type	Possible Risks
Technology	-The database used in the app cannot process as many transactions per second as expected. -InterJobs app components that should be reused contain defects that limit their functionality.
People	-The applicant cannot find the department he wants to work for. -The company cannot find the department it wants.
Tools	-The code generated by CASE tools is inefficient. CASE tools cannot be integrated.
Requirements	-Changes to requirements that require major design rework are proposed. Users fail to understand the impact of requirements changes
Estimation	-The time required to develop the app is underestimated. -The rate of defect repair is underestimated. -The size of the software is underestimated.

2 Risk analysis:-

Risk	Probability	Effects
-The database used in the app cannot process as many transactions per second as expected.	Moderate	Serious
-InterJobs app components that should be reused contain defects that limit their functionality.	Moderate	Serious
-The applicant cannot find the department he wants to work for.	Moderate	Catastrophic
-The company cannot find the department it wants.	Moderate	Catastrophic
-CASE tools cannot be integrated.	High	Tolerable
-Changes to requirements that require major design rework are proposed.	Moderate	Serious
-The time required to develop the app is underestimated.	High	Serious
-The rate of defect repair is underestimated.	Moderate	Tolerable
-The size of the software is underestimated.	High	Tolerable

3 Risk Planning:

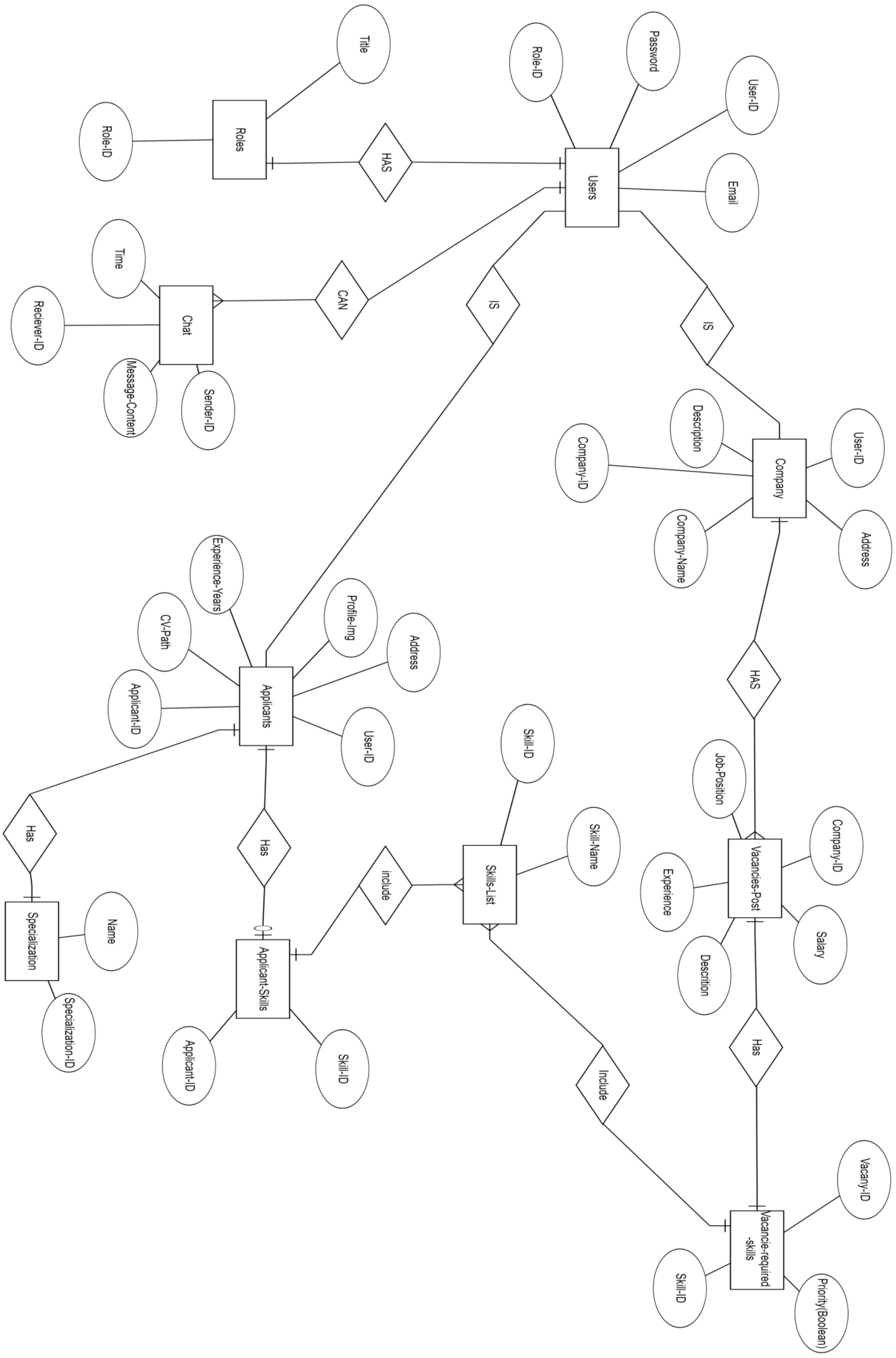
Risk	Strategy
Database performance	Investigate the possibility of buying a higher performance database
Defective components	Replace potentially defective components with bought components of known reliability.
The applicant cannot find the department	The Determination of the best ways to get a suitable job for him.
The company cannot find the department	The Determination of the best ways to get a suitable employee for it.
Requirements change	Derive traceability information to assess requirements change impact, maximize information hiding in the design.

Chapter 3 : Design

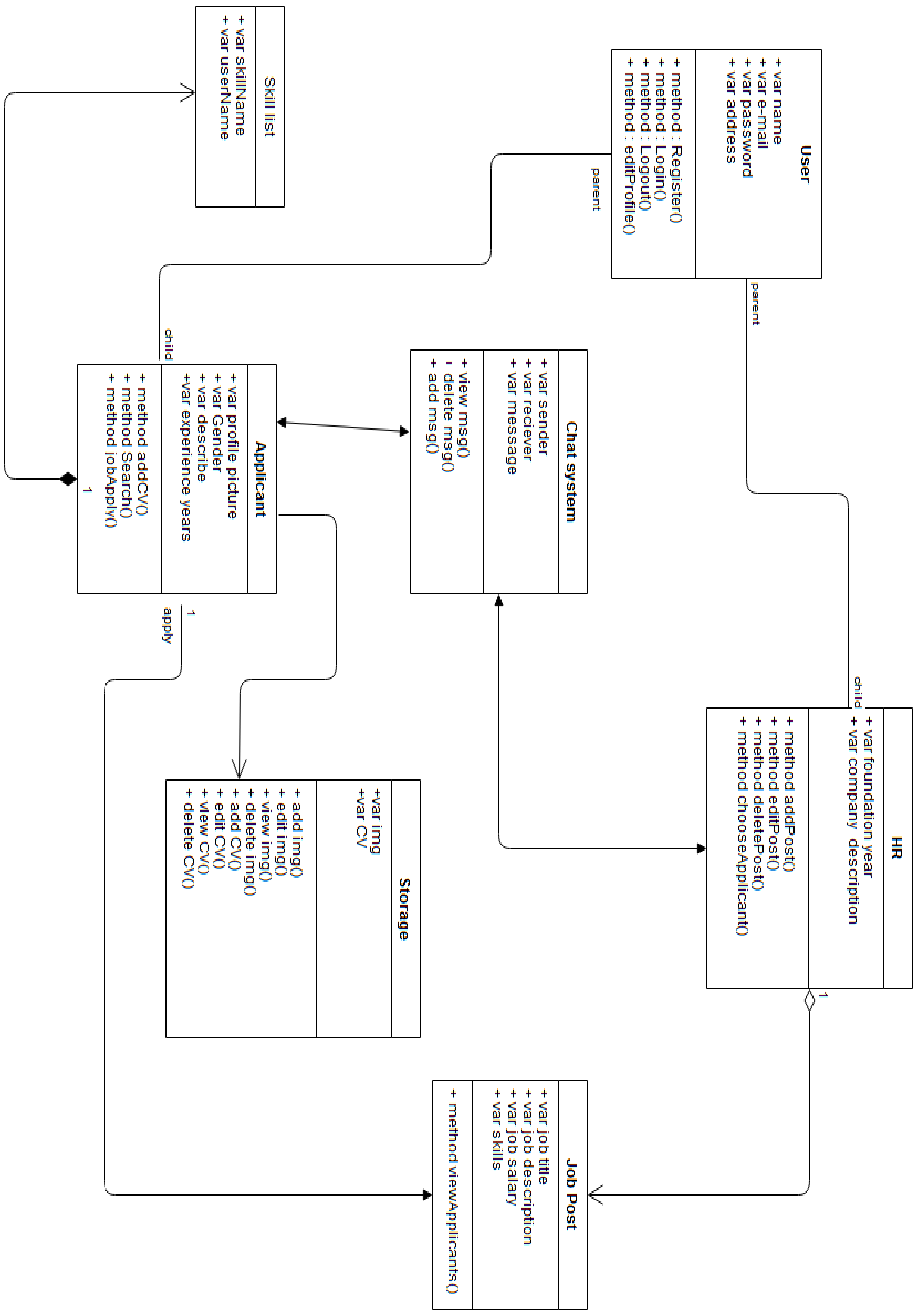
In this chapter, we will include all the Desigs and Diagrams needed to implement the system afterwards

.

3.1 DataBase Schema (ERD):

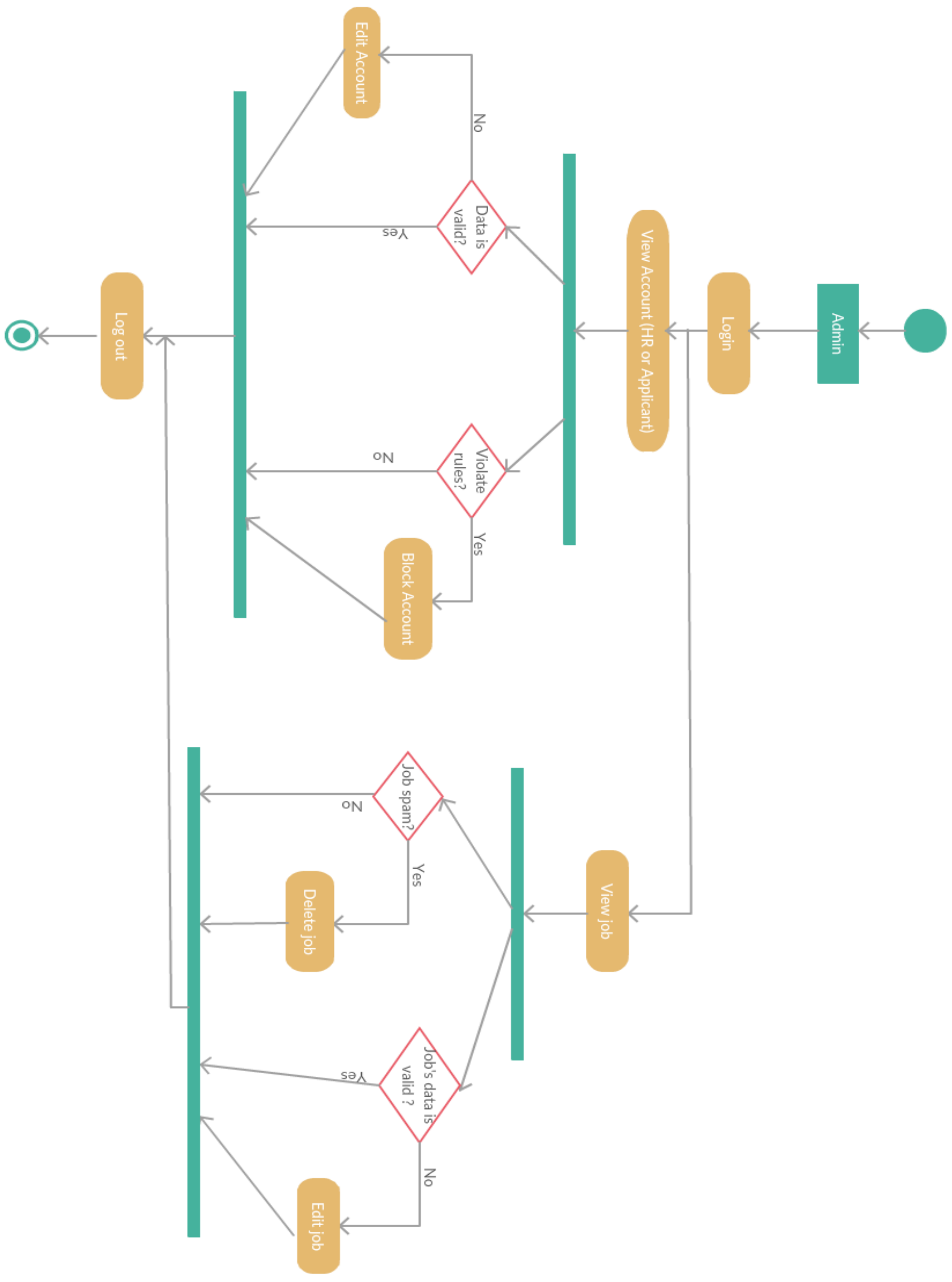

Title
Role-ID
Roles
HAS
Role-ID
Password
User-ID
Users
Email
Time
Reciever-ID
Chat
Message-Content
Sender-ID
CAN
IS
IS
Description
Company-ID
Company
User-ID
Company-Name
Address
Experience-Years
CV-Path
Profile-Img
Address
Applicant-ID
Applicants
User-ID
HAS
Skill-ID
Job-Position
Skill-Name
Company-ID
Experience
Vacancies-Post
Salary
Descrition
Has
Has
Specialization
Name
Specialization-ID
include
Skills-List
Applicant-Skills
Applicant-ID
Skill-ID
Include
Has
Vacany-ID
Vacancie-required -skills
Skill-ID
Priority(Boolean)

3.2 Class Diagram:

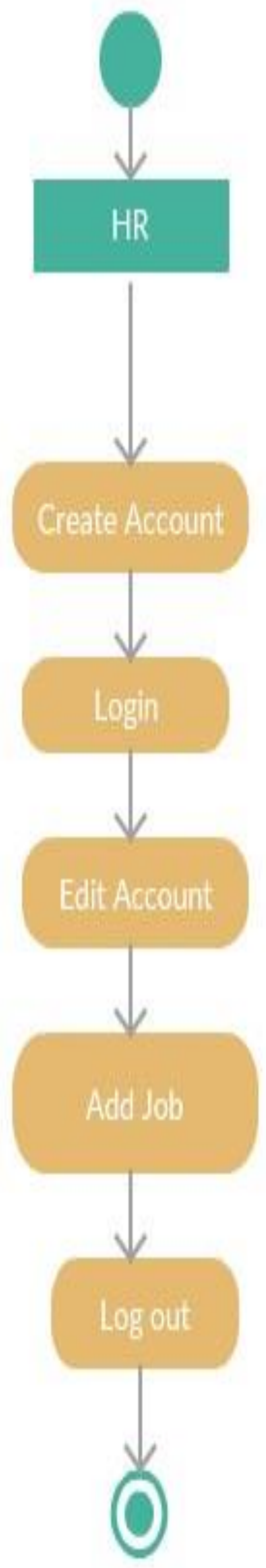


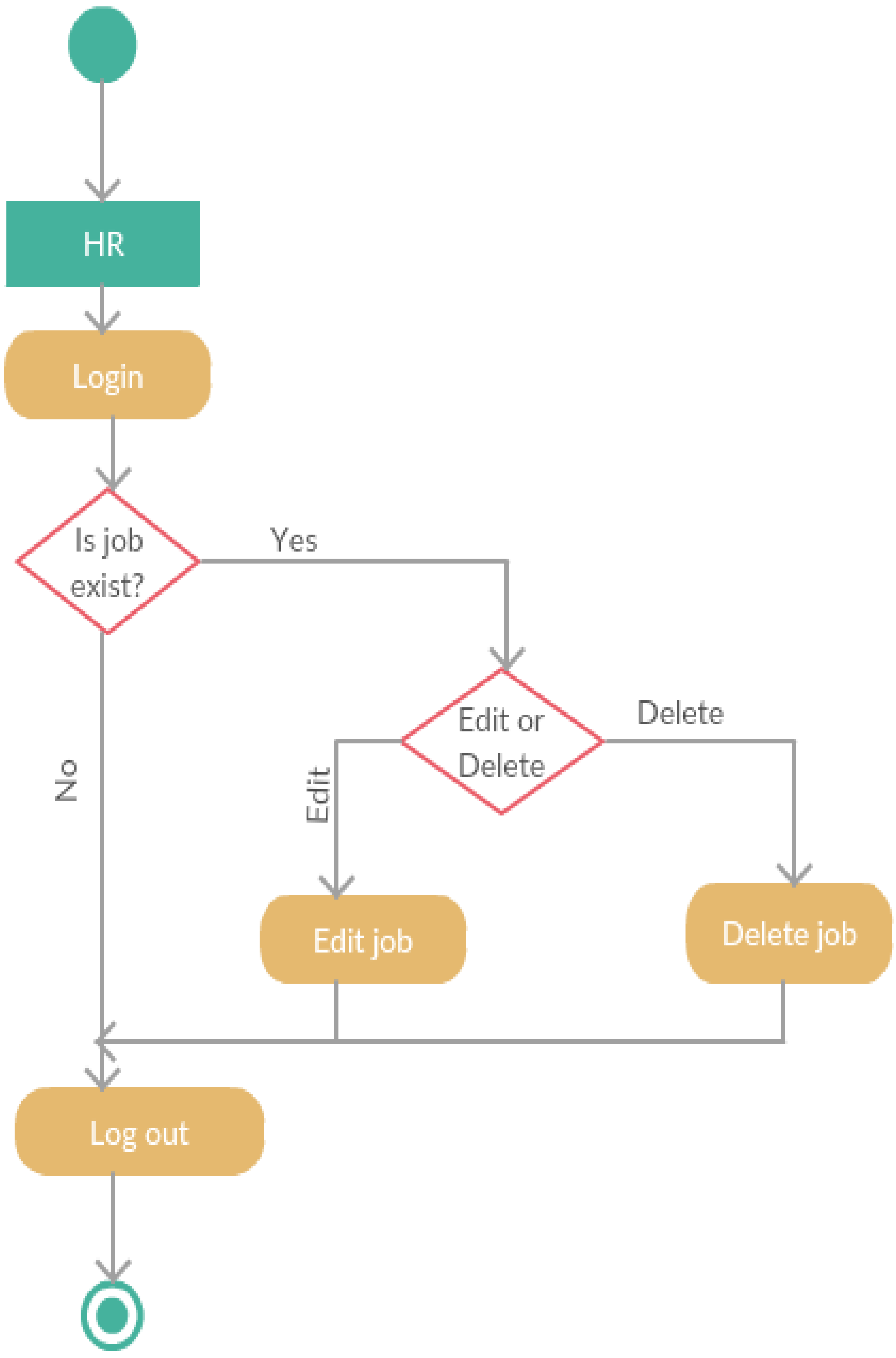
3.2 Activity Diagram:

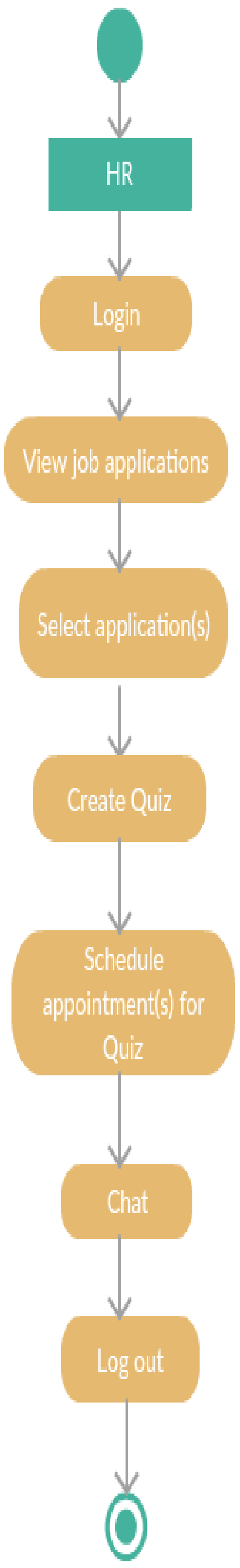
1 Admin:-

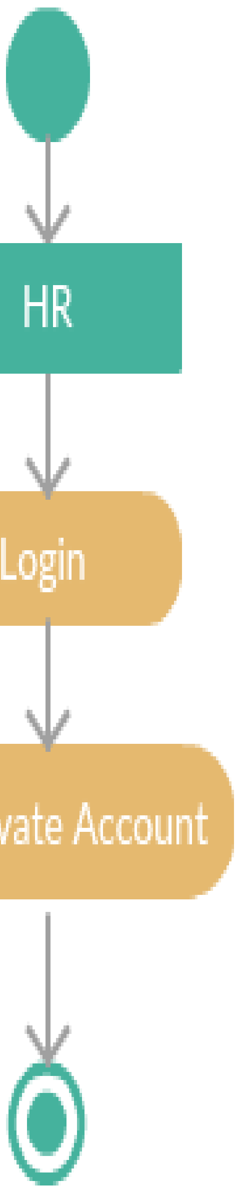


2 HR(Company):-

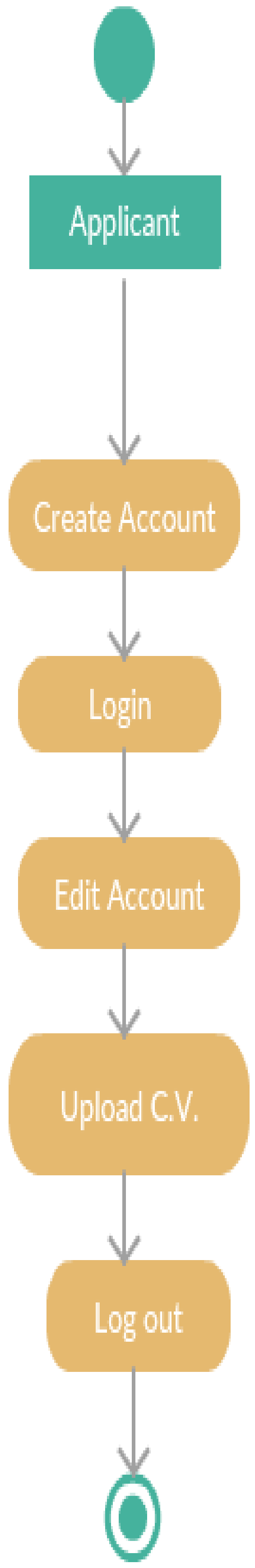


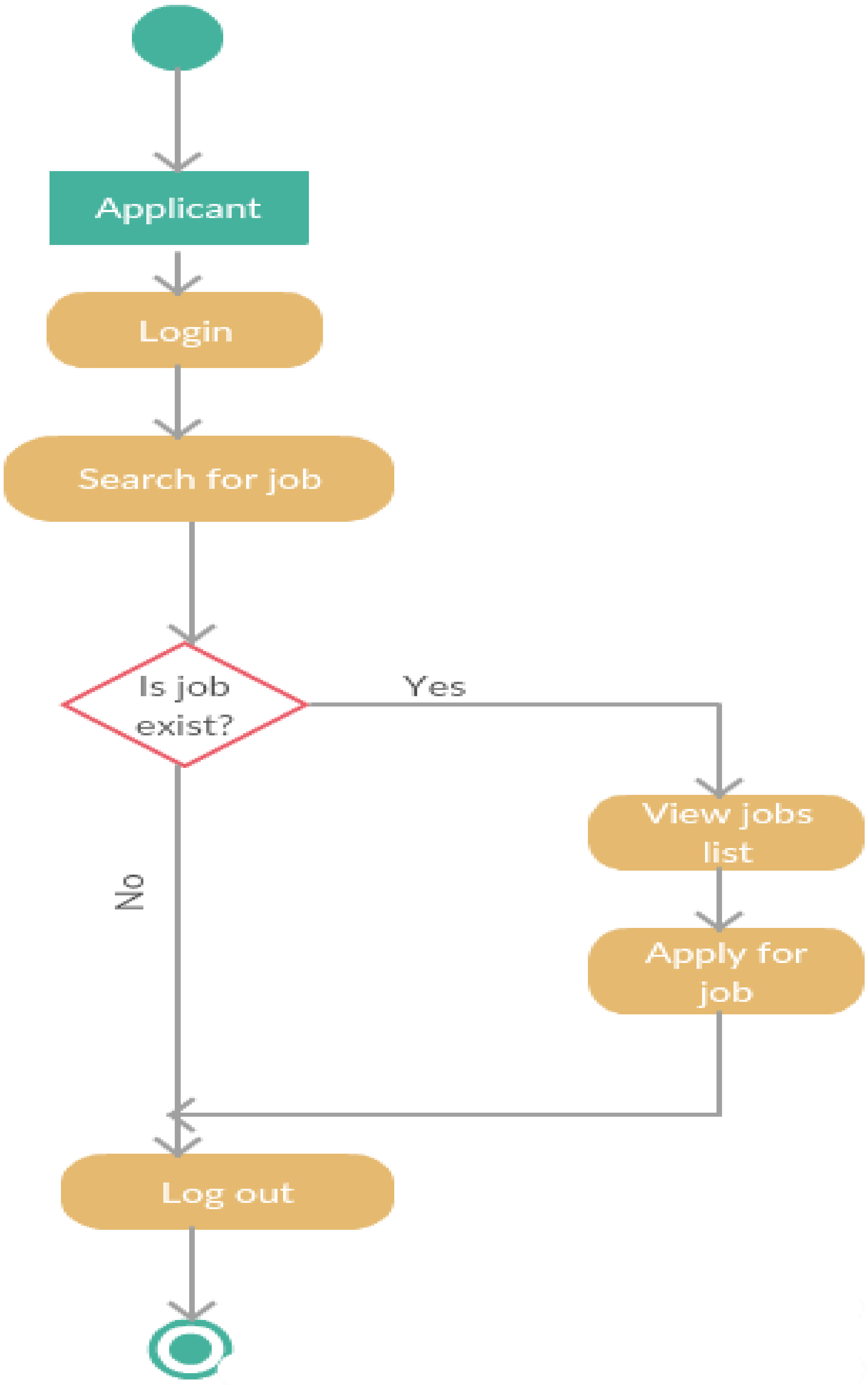

HR
Login
Is job exist?
Yes
No
Edit or Delete
Delete
Edit
Edit job
Delete job
Log out

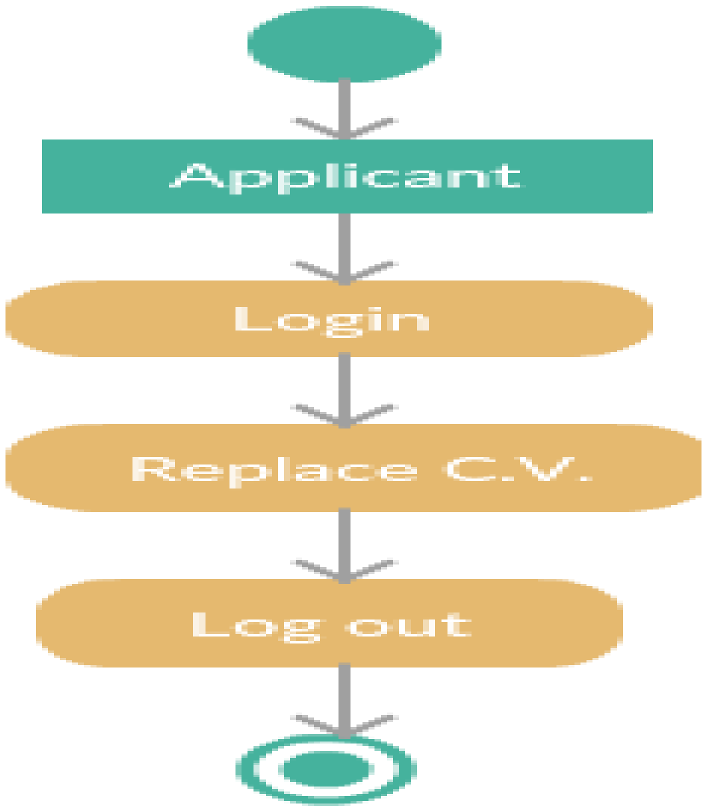

HR
Login
View job applications
Select application(s)
Create Quiz
Schedule appointment(s) for Quiz
Chat
Log out

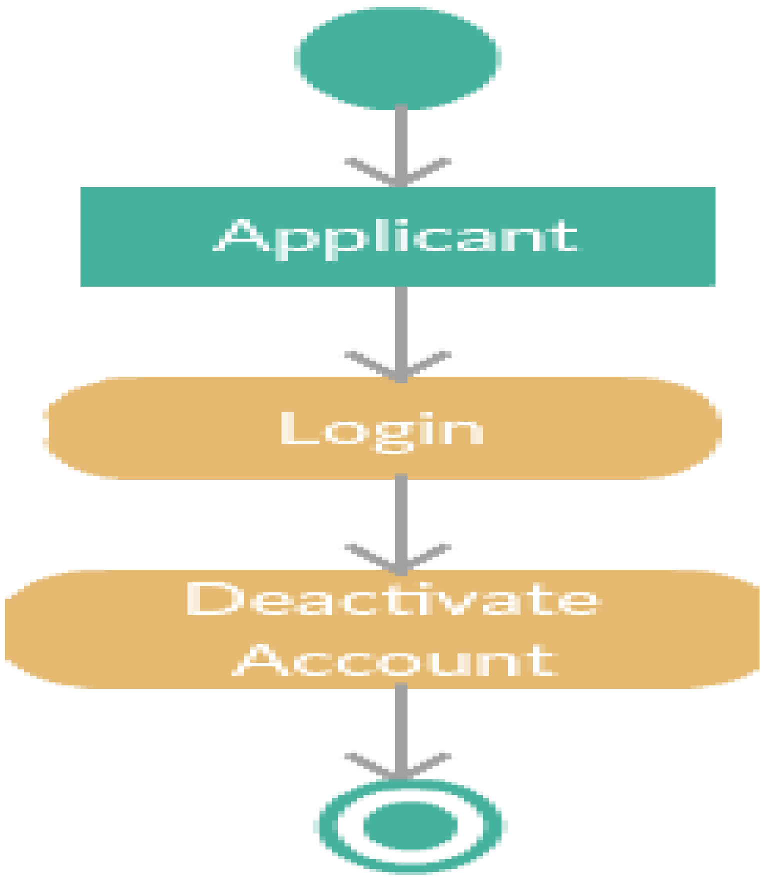

HR
Login
Deactivate Account

3 Applicant:-

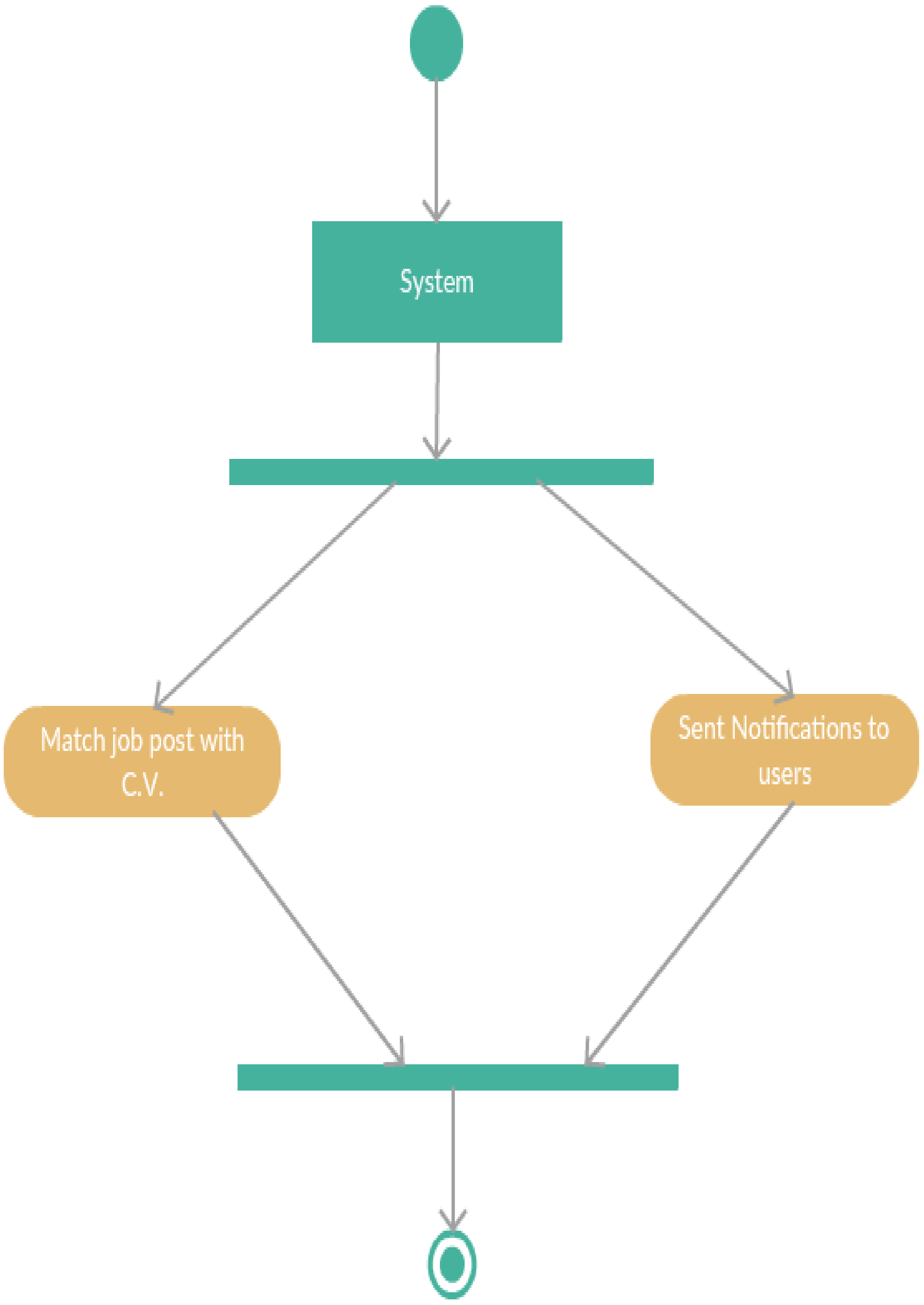

Applicant
Create Account
Login
Edit Account
Upload C.V.
Log out


Applicant
Login
Search for job
Is job exist?
Yes
No
View jobs list
Apply for job
Log out

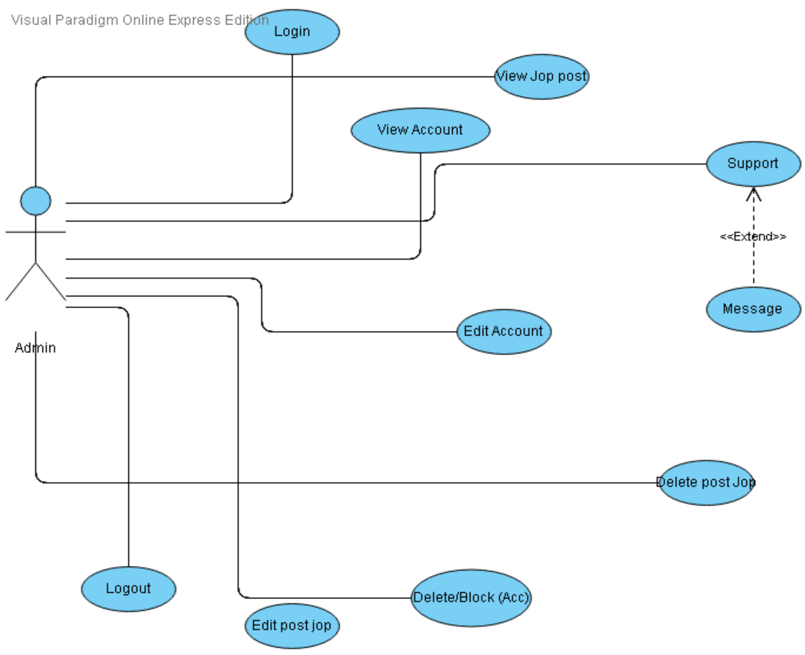

Applicant
Login
Replace C.V.
Log out

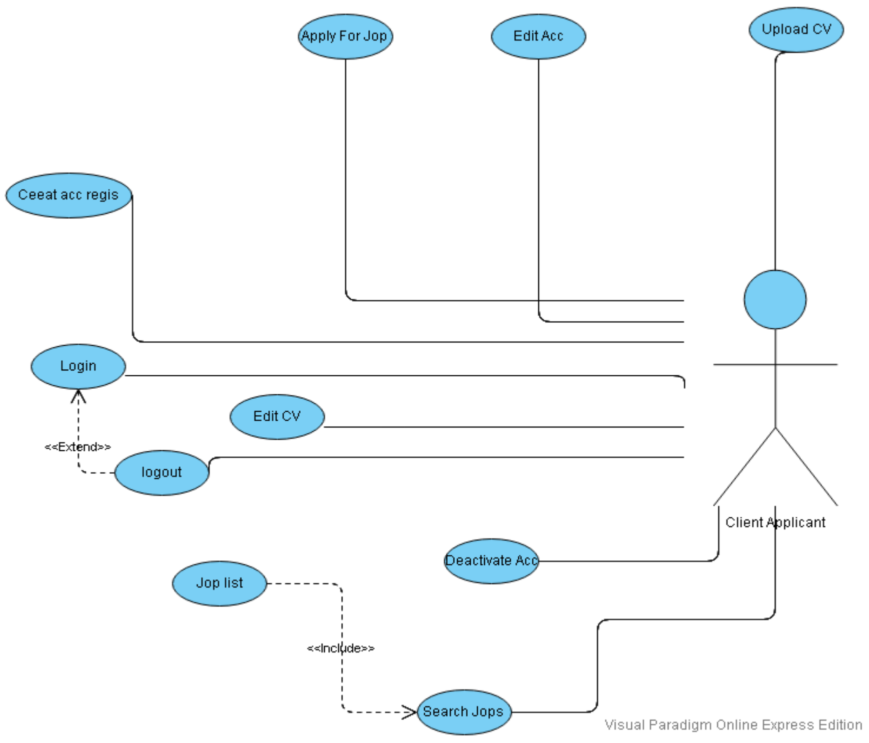

Applicant
Login
Deactivate Account

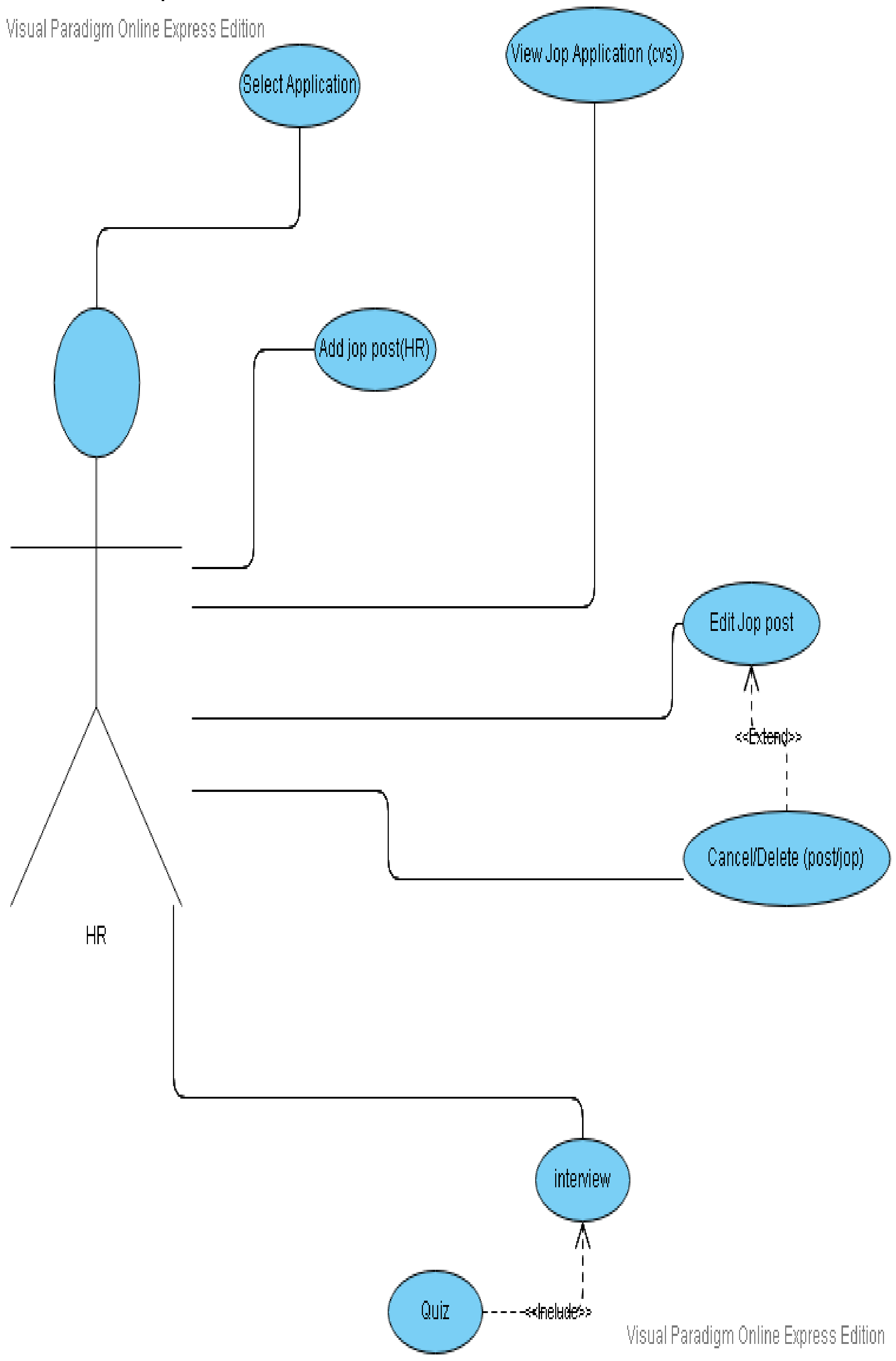
4 Main System (Logic):



3.3 Use Case:


Visual Paradigm Online Express Edition
Login
View Jop post
View Account
Support
<<Extend>>
Message
Edit Account
Admin
Delete post Jop
Logout
Delete/Block (Acc)
Edit post jop


Apply For Jop
Edit Acc
Upload CV
Ceeat acc regis
Login
Edit CV
<<Extend>>
logout
Client Applicant
Deactivate Acc
Jop list
<<Include>>
Search Jops
Visual Paradigm Online Express Edition


Visual Paradigm Online Express Edition
Select Application
View Jop Application (cvs)
Add jop post(HR)
Edit Jop post
<<Extend>>
Cancel/Delete (post/jop)
HR
interview
Quiz
<<Include>>
Visual Paradigm Online Express Edition

Chapter 4 : Implementation

In this chapter, we will discuss how we implemented the system ,the software architecture, design patterns and technologies used.

.

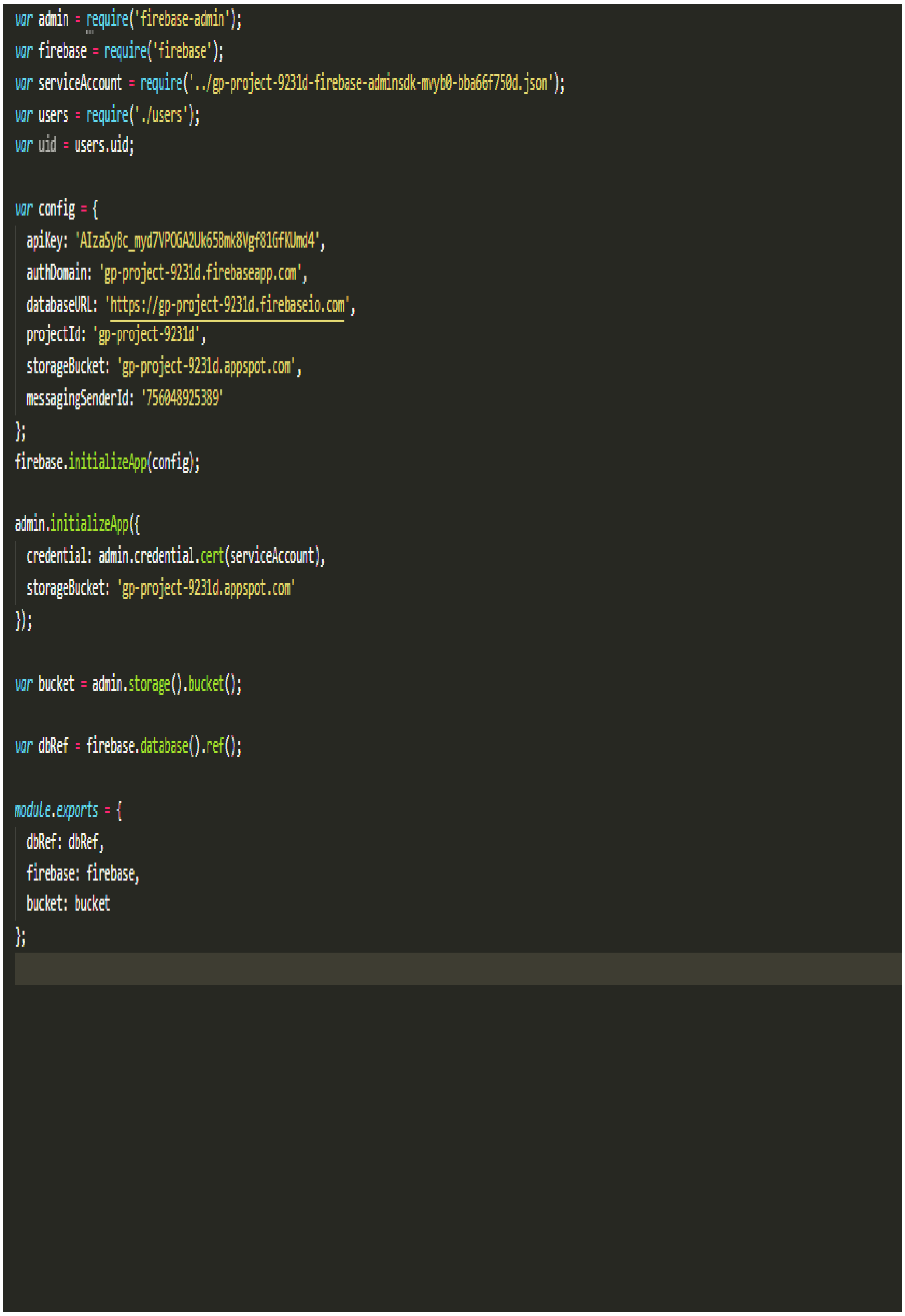
4.1 Design Pattern:

We designed the system using a Model-View-Controller (MVC) architectural pattern where the model is the responsible for maintaining the data or the "DataBase" , View is the user interface seen by the client or user and the controller is what connects both of them.

The Model is designed using Firebase NoSql DataBase, View is designed using EJS template engine and the Controller is designed using Node.js and Express.

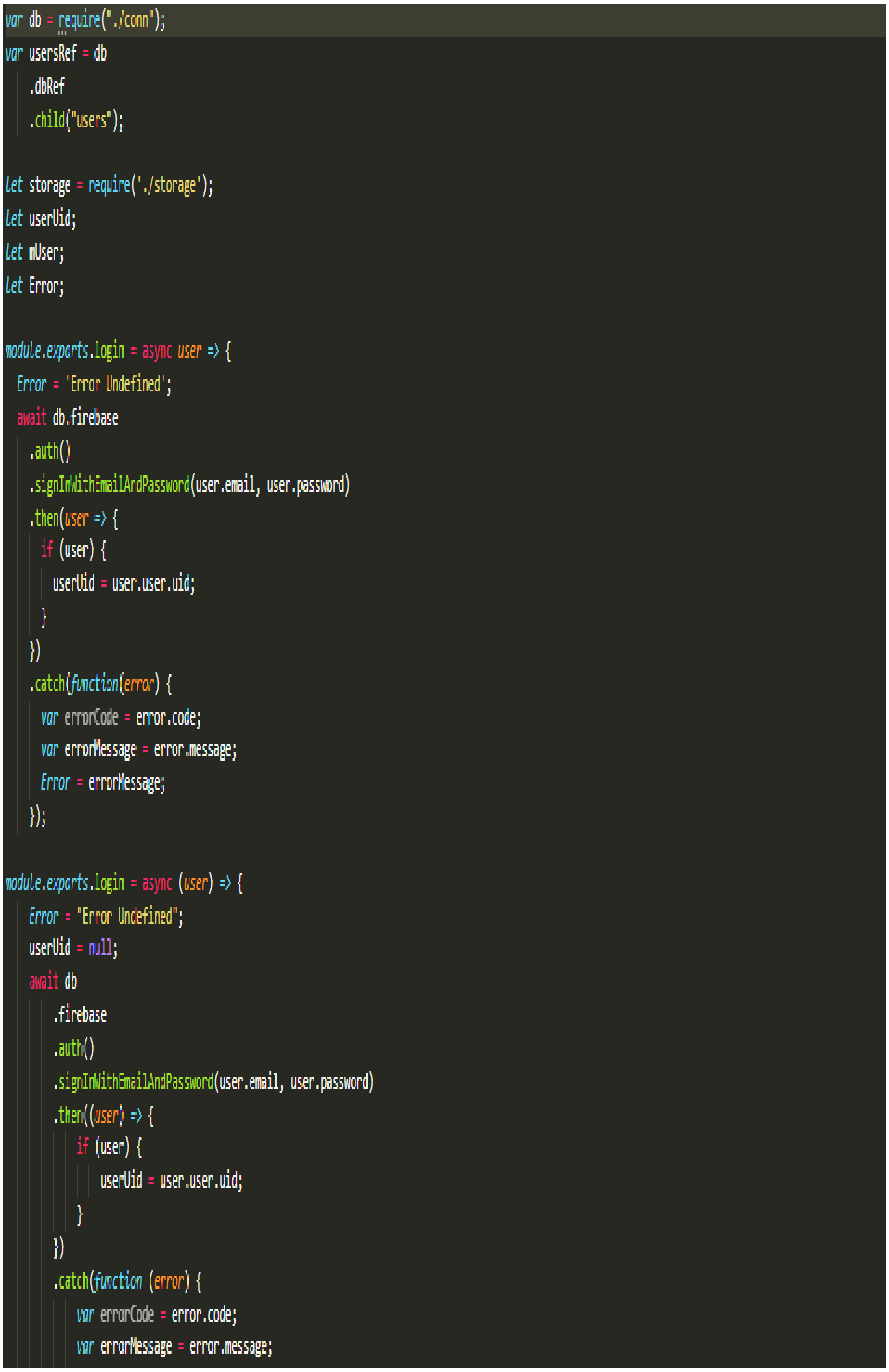
4.2 Software Architecture:

4.2.1 DB Connection:

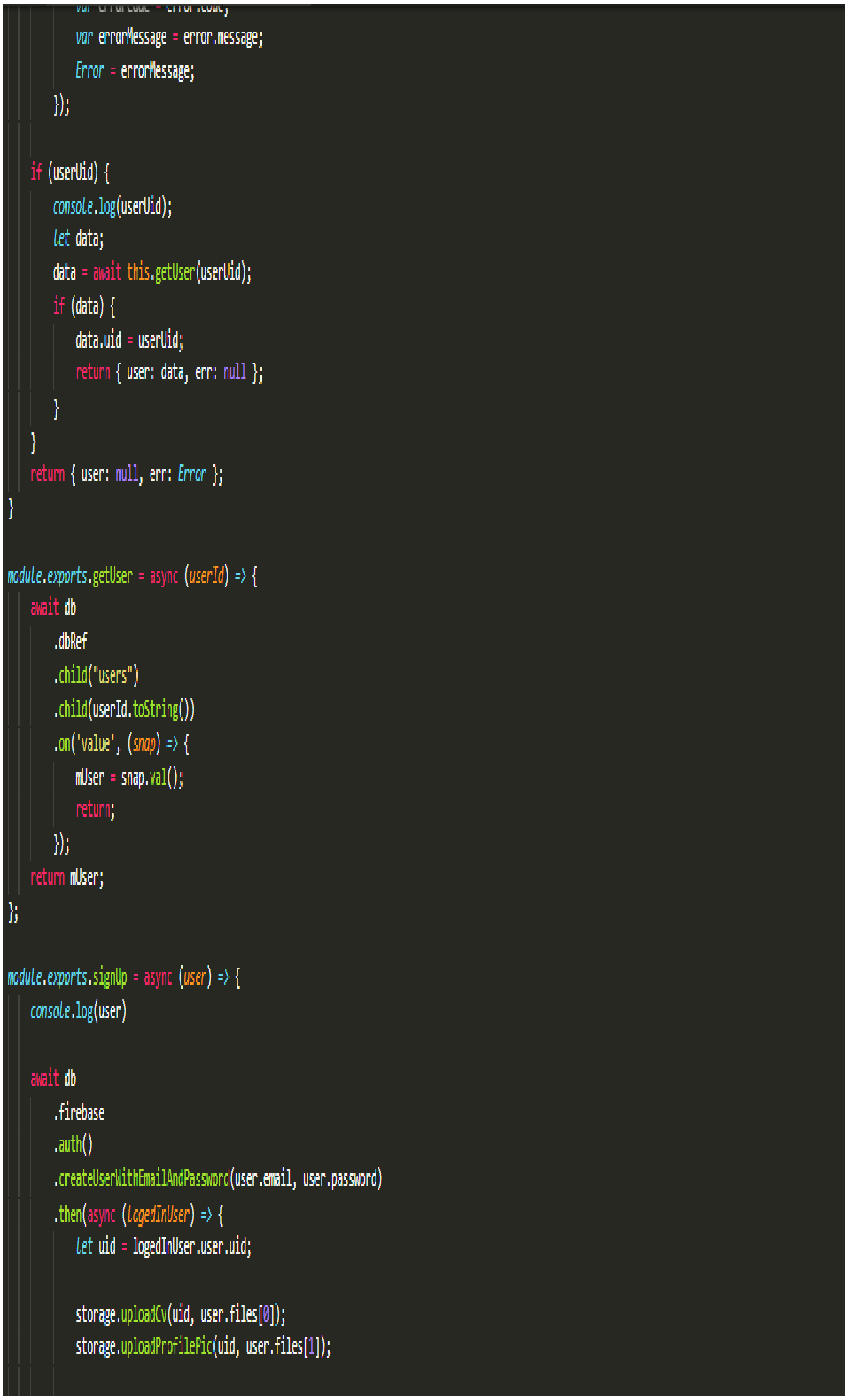


```
var admin = require('firebase-admin');
var firebase = require('firebase');
var serviceAccount = require('../gp-project-9231d-firebase-adminsdk-mvyb0-bba66f750d.json');
var users = require('./users');
var uid = users.uid;

var config = {
  apiKey: 'AIzaSyBc_myd7VPOGA2Uk65Bmk8Vgf81GfKUmd4',
  authDomain: 'gp-project-9231d.firebaseapp.com',
  databaseURL: 'https://gp-project-9231d.firebaseio.com',
  projectId: 'gp-project-9231d',
  storageBucket: 'gp-project-9231d.appspot.com',
  messagingSenderId: '756048925389'
};
firebase.initializeApp(config);

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  storageBucket: 'gp-project-9231d.appspot.com'
});

var bucket = admin.storage().bucket();

var dbRef = firebase.database().ref();

module.exports = {
  dbRef: dbRef,
  firebase: firebase,
  bucket: bucket
};
```

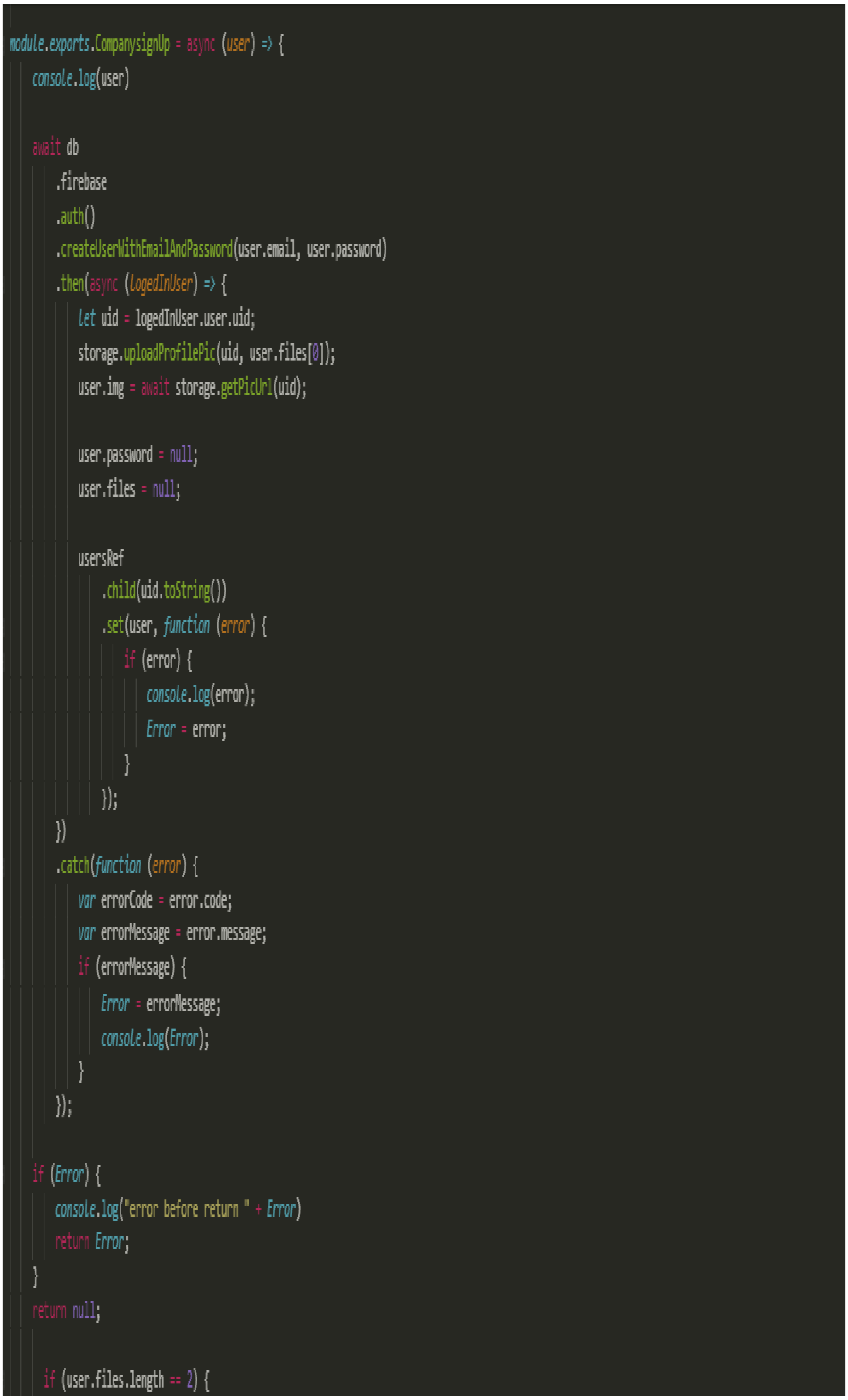
DB users data:



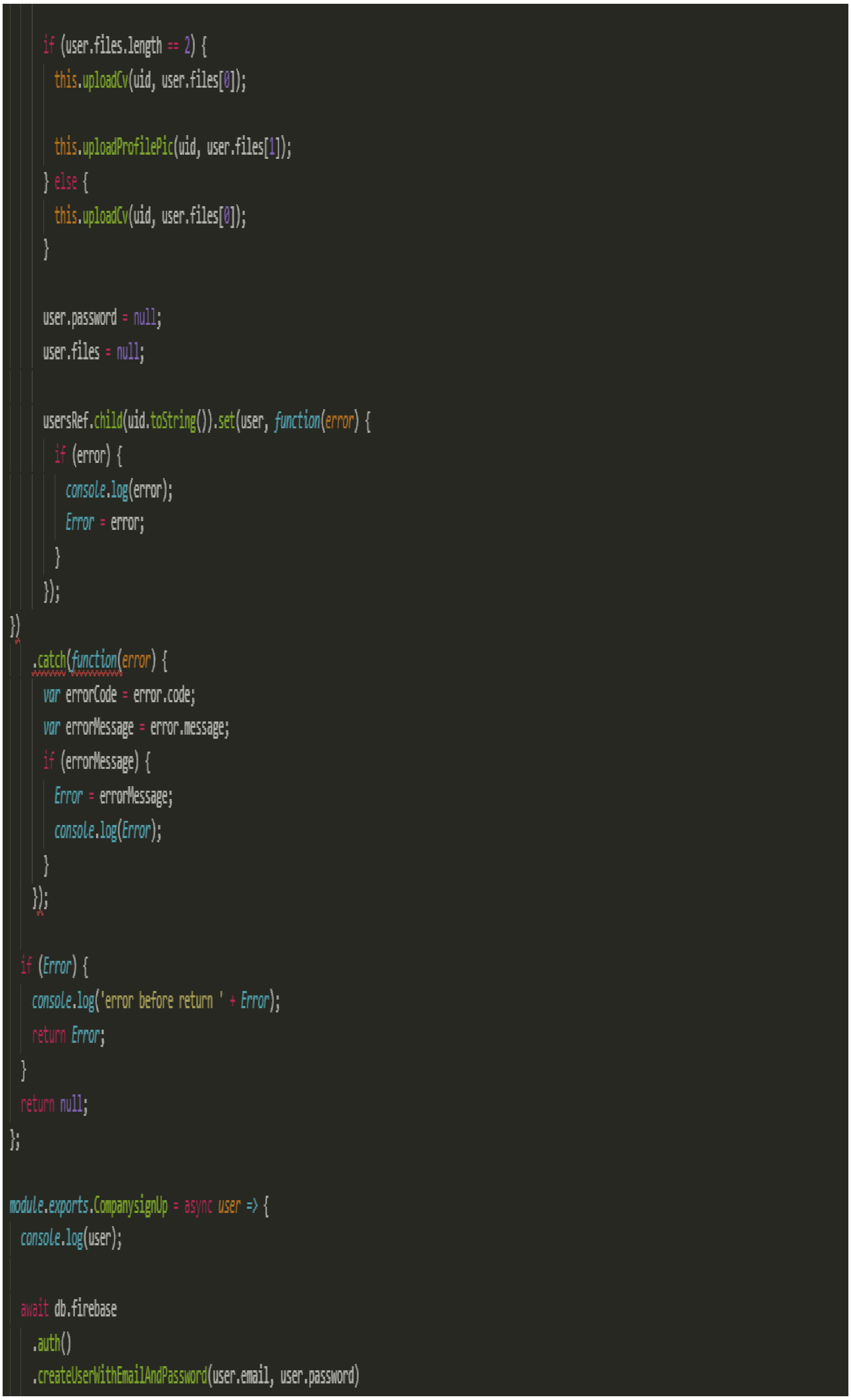
```
var db = require("./conn");
var usersRef = db
    .dbRef
    .child("users");

let storage = require('./storage');
let userUid;
let mUser;
let Error;

module.exports.login = async user => {
  Error = 'Error Undefined';
  await db.firebase
    .auth()
    .signInWithEmailAndPassword(user.email, user.password)
    .then(user => {
      if (user) {
        userUid = user.user.uid;
      }
    })
    .catch(function(error) {
      var errorCode = error.code;
      var errorMessage = error.message;
      Error = errorMessage;
    });

module.exports.login = async (user) => {
    Error = "Error Undefined";
    userUid = null;
    await db
        .firebase
        .auth()
        .signInWithEmailAndPassword(user.email, user.password)
        .then((user) => {
            if (user) {
                userUid = user.user.uid;
            }
        })
        .catch(function (error) {
            var errorCode = error.code;
            var errorMessage = error.message;
```



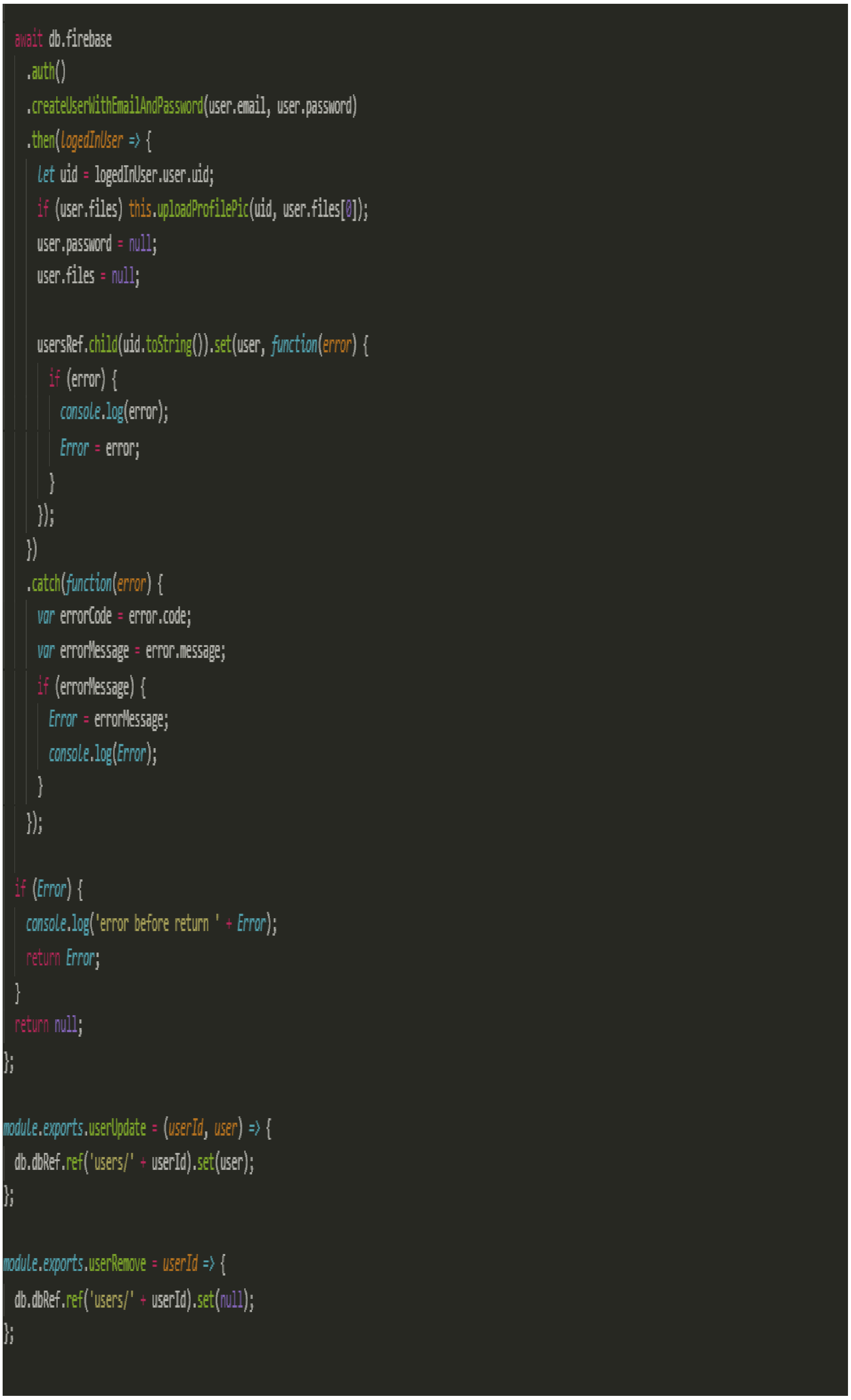
```
            var errorMessage = error.message;
            Error = errorMessage;
        });

    if (userUid) {
        console.log(userUid);
        let data;
        data = await this.getUser(userUid);
        if (data) {
            data.uid = userUid;
            return { user: data, err: null };
        }
    }
    return { user: null, err: Error };
}

module.exports.getUser = async (userId) => {
    await db
        .dbRef
        .child("users")
        .child(userId.toString())
        .on('value', (snap) => {
            mUser = snap.val();
            return;
        });
    return mUser;
};

module.exports.signUp = async (user) => {
    console.log(user)

    await db
        .firebase
        .auth()
        .createUserWithEmailAndPassword(user.email, user.password)
        .then(async (logedInUser) => {
            let uid = logedInUser.user.uid;

            storage.uploadCv(uid, user.files[0]);
            storage.uploadProfilePic(uid, user.files[1]);
```



```
module.exports.CompanysignUp = async (user) => {
    console.log(user)

    await db
        .firebase
        .auth()
        .createUserWithEmailAndPassword(user.email, user.password)
        .then(async (logedInUser) => {
            let uid = logedInUser.user.uid;
            storage.uploadProfilePic(uid, user.files[0]);
            user.img = await storage.getPicUrl(uid);

            user.password = null;
            user.files = null;

            usersRef
                .child(uid.toString())
                .set(user, function (error) {
                    if (error) {
                        console.log(error);
                        Error = error;
                    }
                });
        })
        .catch(function (error) {
            var errorCode = error.code;
            var errorMessage = error.message;
            if (errorMessage) {
                Error = errorMessage;
                console.log(Error);
            }
        });

    if (Error) {
        console.log("error before return " + Error)
        return Error;
    }
    return null;

     if (user.files.length == 2) {
```

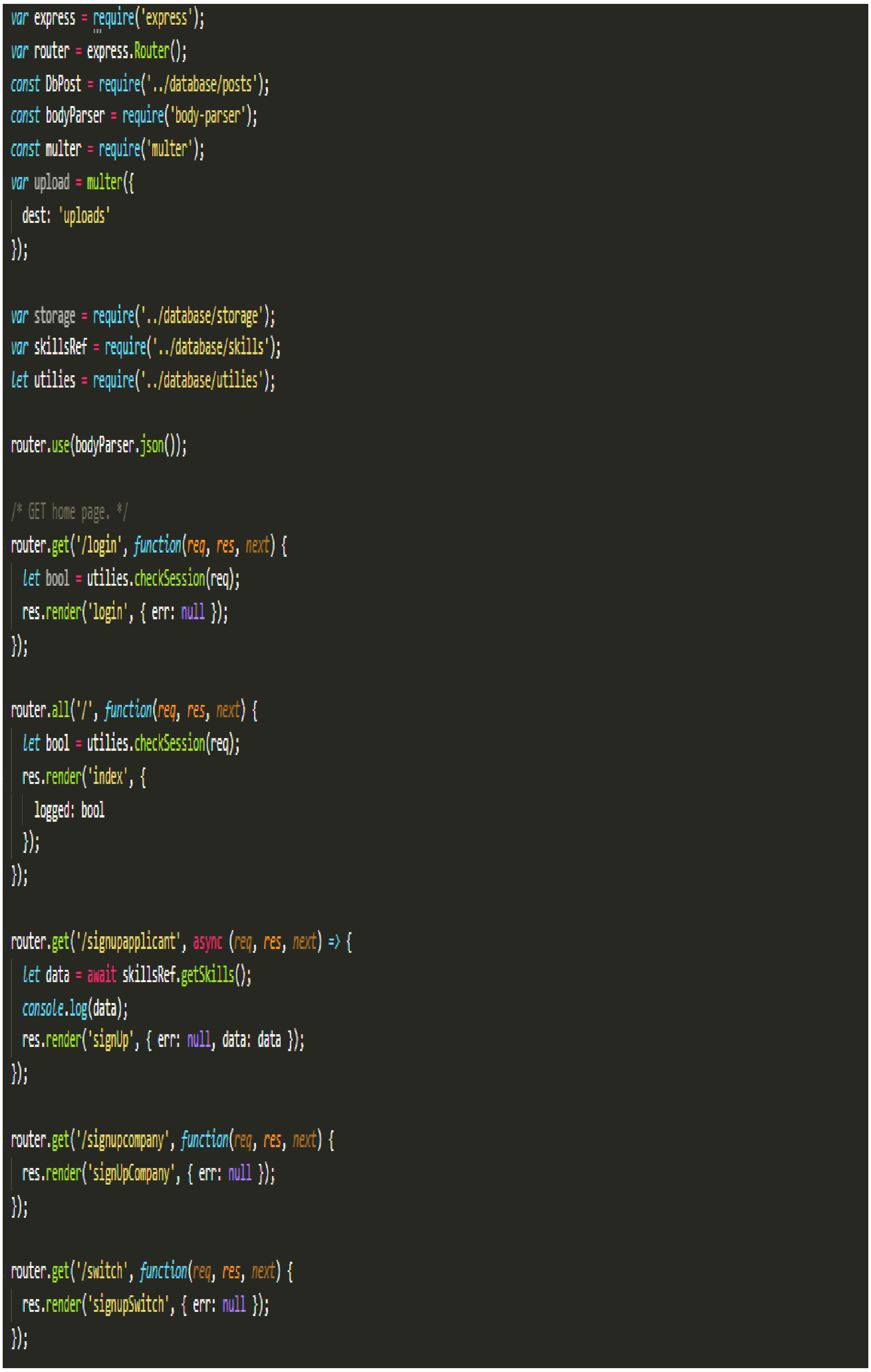


```
      if (user.files.length == 2) {
        this.uploadCv(uid, user.files[0]);

        this.uploadProfilePic(uid, user.files[1]);
      } else {
        this.uploadCv(uid, user.files[0]);
      }

      user.password = null;
      user.files = null;

      usersRef.child(uid.toString()).set(user, function(error) {
        if (error) {
          console.log(error);
          Error = error;
        }
      });
})
    .catch(function(error) {
      var errorCode = error.code;
      var errorMessage = error.message;
      if (errorMessage) {
        Error = errorMessage;
        console.log(Error);
      }
    });

  if (Error) {
    console.log('error before return ' + Error);
    return Error;
  }
  return null;
};

module.exports.CompanysignUp = async user => {
  console.log(user);

  await db.firebase
    .auth()
    .createUserWithEmailAndPassword(user.email, user.password)
```

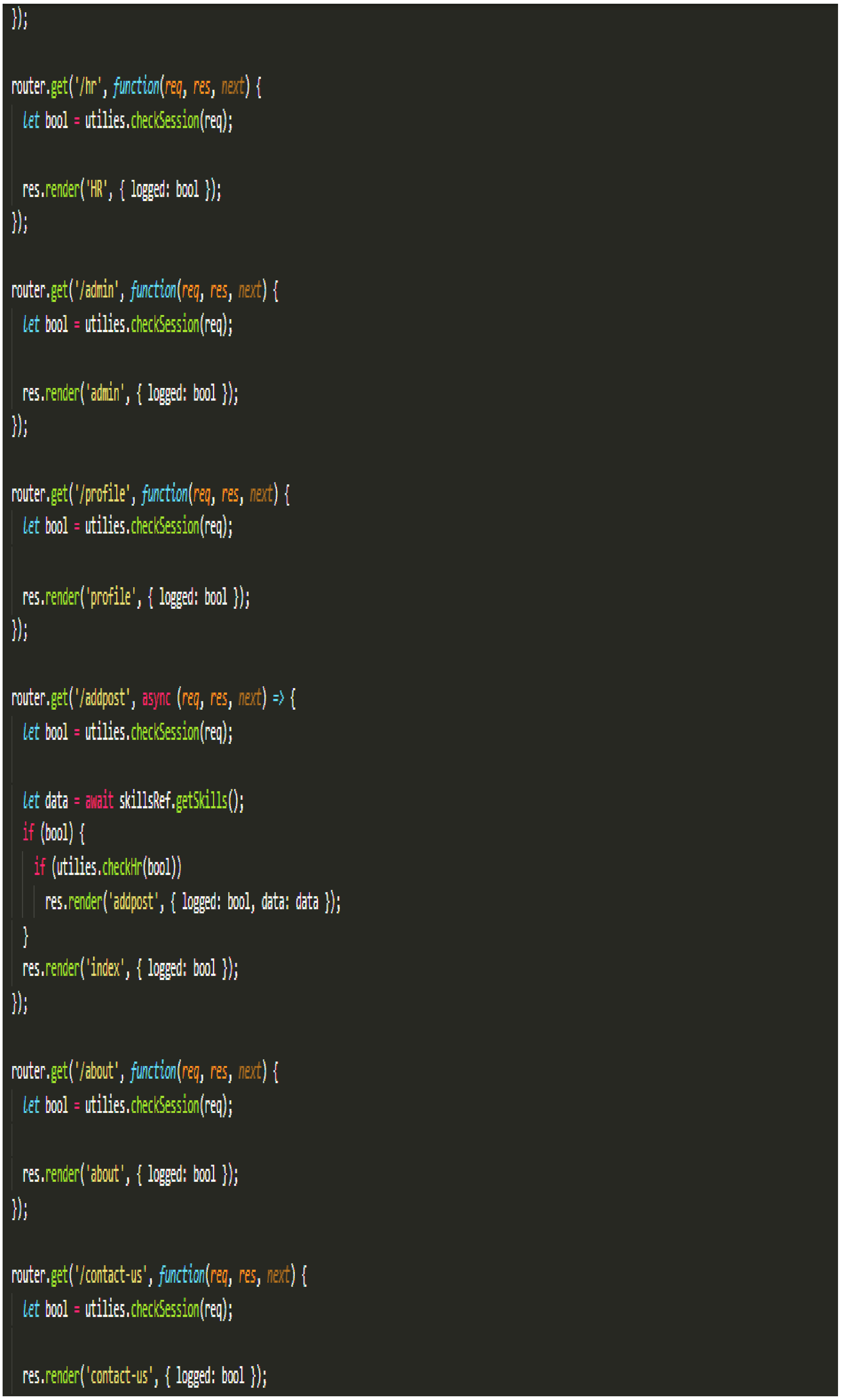


```
  await db.firebase
    .auth()
    .createUserWithEmailAndPassword(user.email, user.password)
    .then(logedInUser => {
      let uid = logedInUser.user.uid;
      if (user.files) this.uploadProfilePic(uid, user.files[0]);
      user.password = null;
      user.files = null;

      usersRef.child(uid.toString()).set(user, function(error) {
        if (error) {
          console.log(error);
          Error = error;
        }
      });
    })
    .catch(function(error) {
      var errorCode = error.code;
      var errorMessage = error.message;
      if (errorMessage) {
        Error = errorMessage;
        console.log(Error);
      }
    });

  if (Error) {
    console.log('error before return ' + Error);
    return Error;
  }
  return null;
};

module.exports.userUpdate = (userId, user) => {
  db.dbRef.ref('users/' + userId).set(user);
};

module.exports.userRemove = userId => {
  db.dbRef.ref('users/' + userId).set(null);
};
```

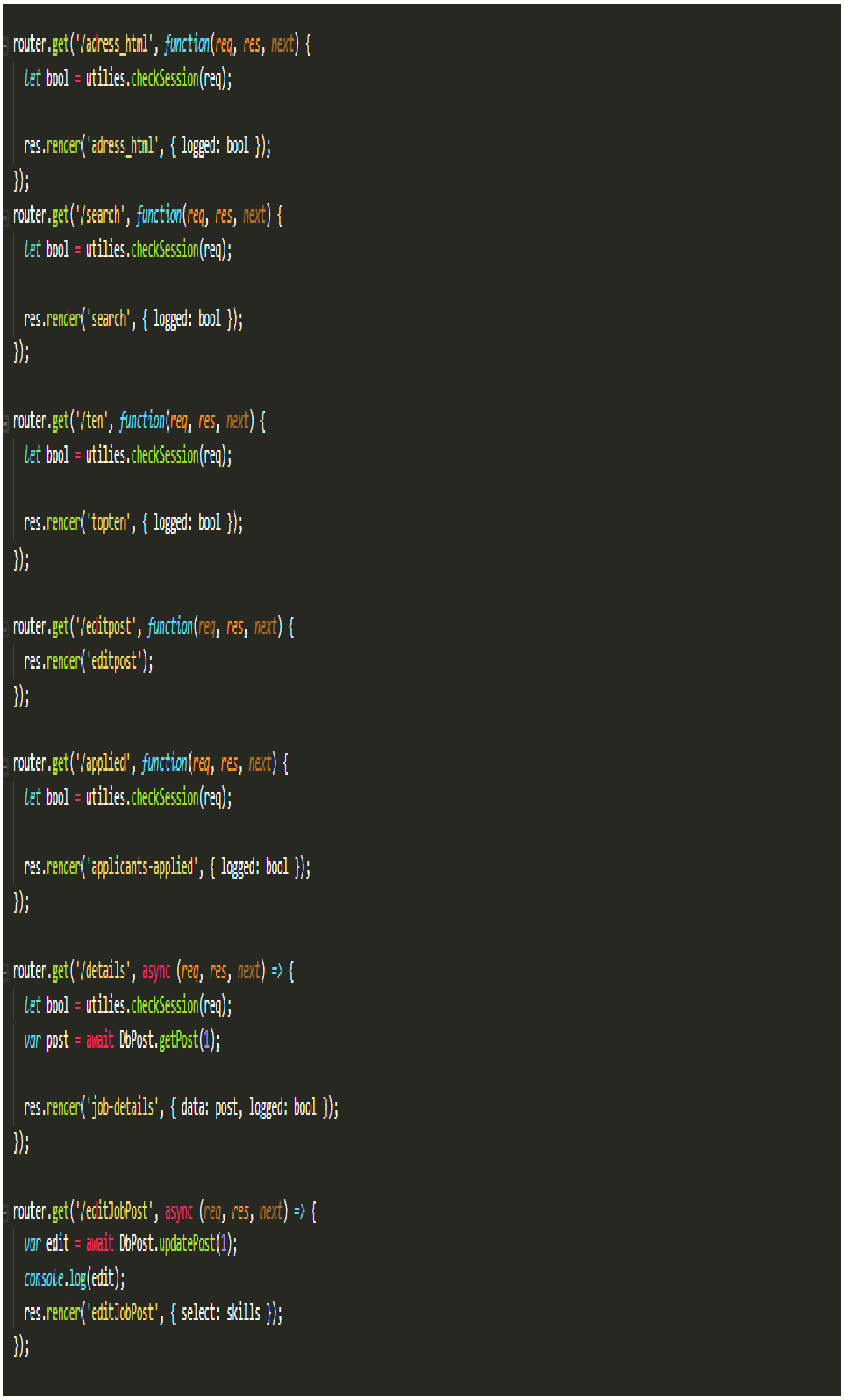
4.2.2 Controller index:



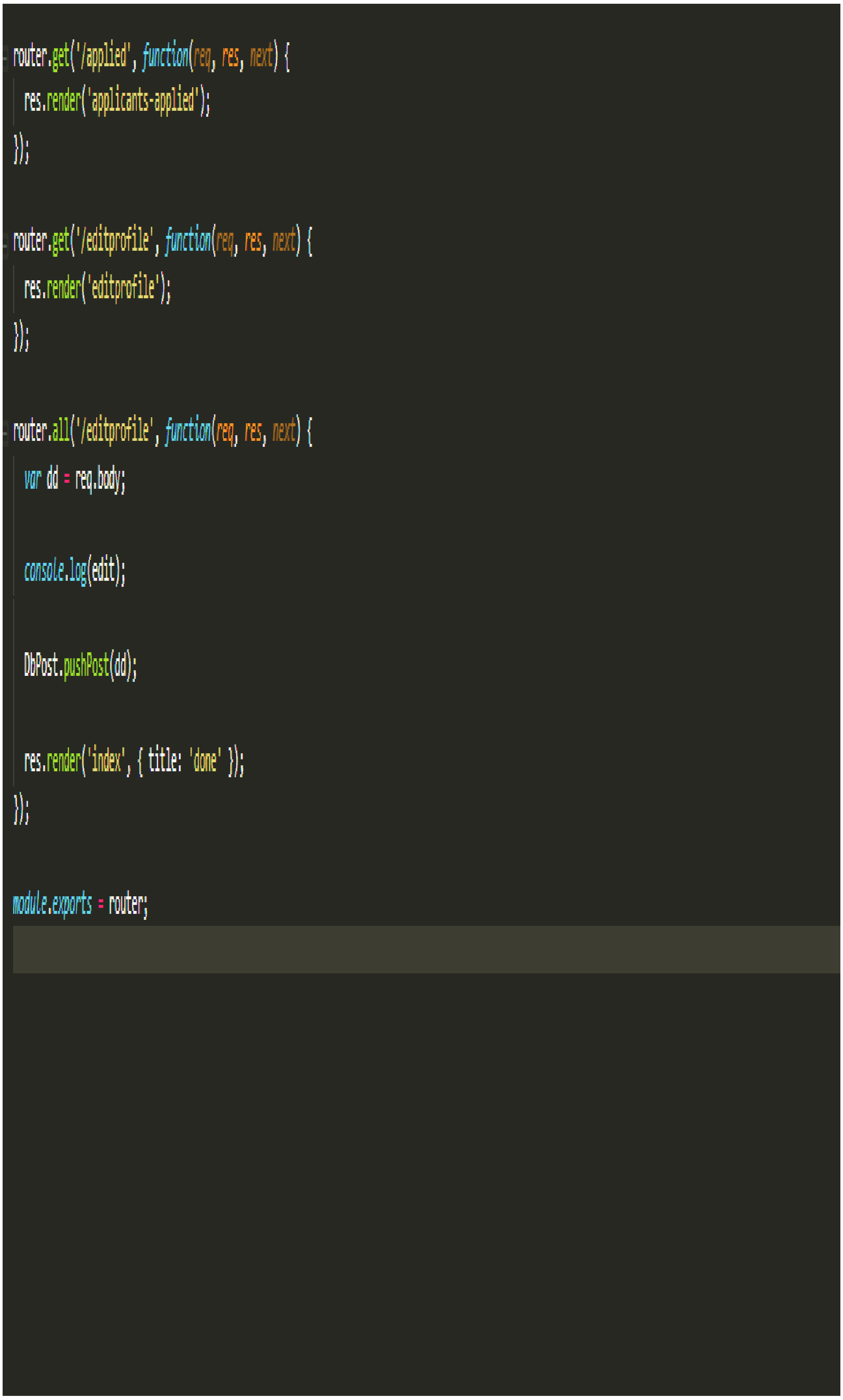
```
var express = require('express');
var router = express.Router();
const DbPost = require('../database/posts');
const bodyParser = require('body-parser');
const multer = require('multer');
var upload = multer({
  dest: 'uploads'
});

var storage = require('../database/storage');
var skillsRef = require('../database/skills');
let utilies = require('../database/utilies');

router.use(bodyParser.json());

/* GET home page. */
router.get('/login', function(req, res, next) {
  let bool = utilies.checkSession(req);
  res.render('login', { err: null });
});

router.all('/', function(req, res, next) {
  let bool = utilies.checkSession(req);
  res.render('index', {
    logged: bool
  });
});

router.get('/signupapplicant', async (req, res, next) => {
  let data = await skillsRef.getSkills();
  console.log(data);
  res.render('signUp', { err: null, data: data });
});

router.get('/signupcompany', function(req, res, next) {
  res.render('signUpCompany', { err: null });
});

router.get('/switch', function(req, res, next) {
  res.render('signupSwitch', { err: null });
});
```



```
});

router.get('/hr', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('HR', { logged: bool });
});

router.get('/admin', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('admin', { logged: bool });
});

router.get('/profile', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('profile', { logged: bool });
});

router.get('/addpost', async (req, res, next) => {
  let bool = utilies.checkSession(req);

  let data = await skillsRef.getSkills();
  if (bool) {
    if (utilies.checkHr(bool))
      res.render('addpost', { logged: bool, data: data });
  }
  res.render('index', { logged: bool });
});

router.get('/about', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('about', { logged: bool });
});

router.get('/contact-us', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('contact-us', { logged: bool });
```

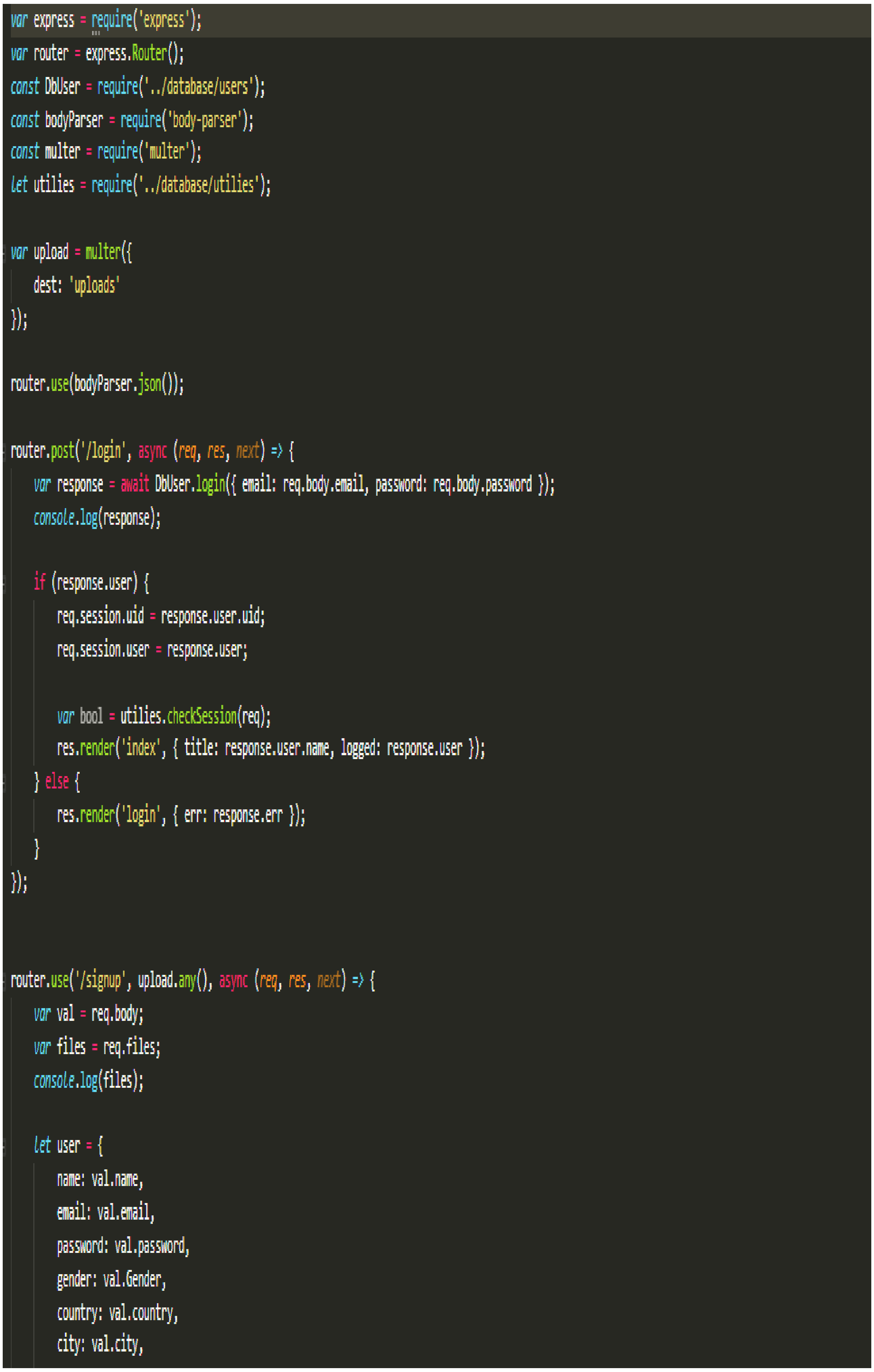


```
router.get('/adress_html', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('adress_html', { logged: bool });
});
router.get('/search', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('search', { logged: bool });
});

router.get('/ten', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('topten', { logged: bool });
});

router.get('/editpost', function(req, res, next) {
  res.render('editpost');
});

router.get('/applied', function(req, res, next) {
  let bool = utilies.checkSession(req);

  res.render('applicants-applied', { logged: bool });
});

router.get('/details', async (req, res, next) => {
  let bool = utilies.checkSession(req);
  var post = await DbPost.getPost(1);

  res.render('job-details', { data: post, logged: bool });
});

router.get('/editJobPost', async (req, res, next) => {
  var edit = await DbPost.updatePost(1);
  console.log(edit);
  res.render('editJobPost', { select: skills });
});
```

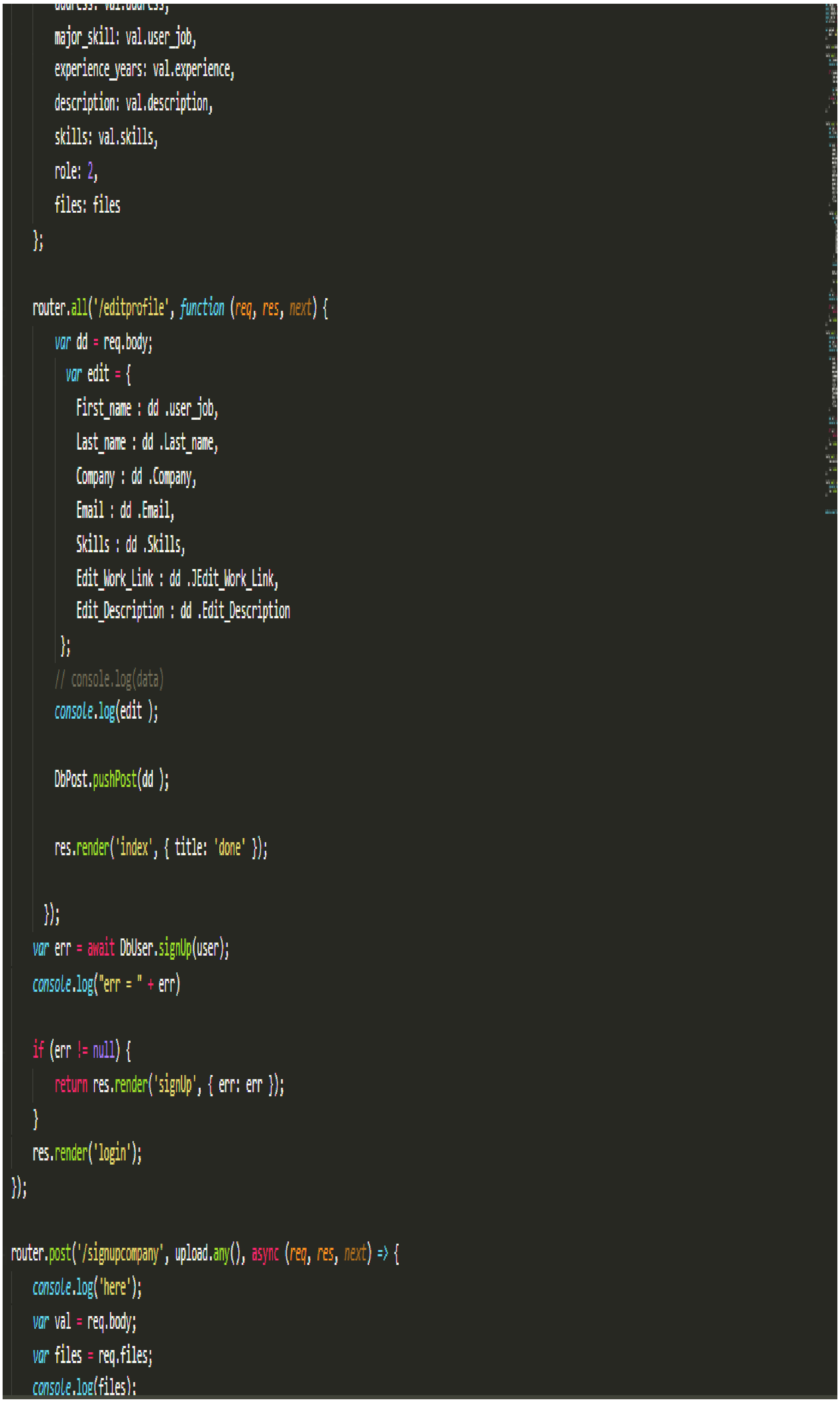


```
router.get('/applied', function(req, res, next) {
  res.render('applicants-applied');
});

router.get('/editprofile', function(req, res, next) {
  res.render('editprofile');
});

router.all('/editprofile', function(req, res, next) {
  var dd = req.body;

  console.log(edit);

  DbPost.pushPost(dd);

  res.render('index', { title: 'done' });
});

module.exports = router;
```

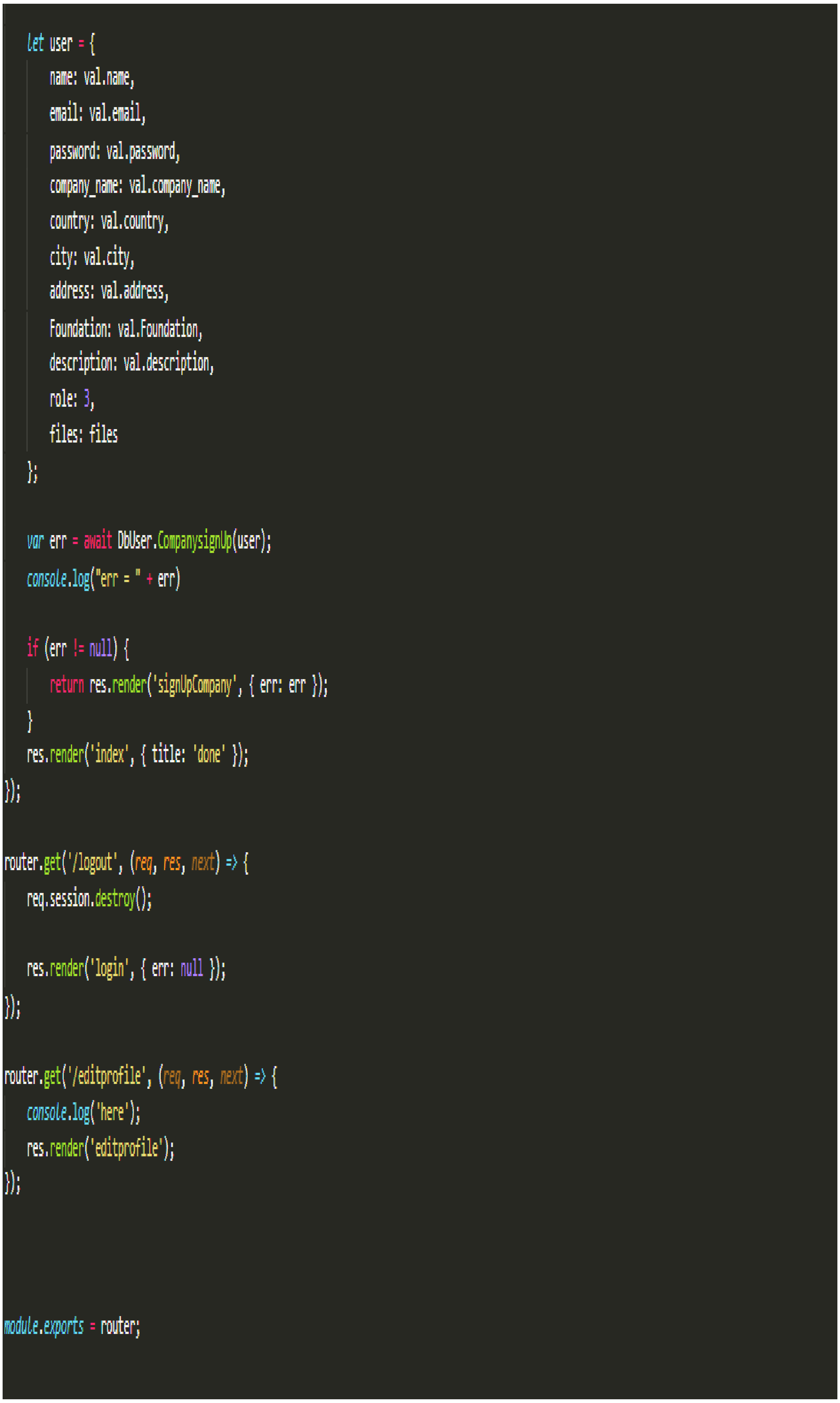
Controller User routes:



```
var express = require('express');
var router = express.Router();
const DbUser = require('../database/users');
const bodyParser = require('body-parser');
const multer = require('multer');
let utilies = require('../database/utilies');

var upload = multer({
    dest: 'uploads'
});

router.use(bodyParser.json());

router.post('/login', async (req, res, next) => {
    var response = await DbUser.login({ email: req.body.email, password: req.body.password });
    console.log(response);

    if (response.user) {
        req.session.uid = response.user.uid;
        req.session.user = response.user;

        var bool = utilies.checkSession(req);
        res.render('index', { title: response.user.name, logged: response.user });
    } else {
        res.render('login', { err: response.err });
    }
});


router.use('/signup', upload.any(), async (req, res, next) => {
    var val = req.body;
    var files = req.files;
    console.log(files);

    let user = {
        name: val.name,
        email: val.email,
        password: val.password,
        gender: val.Gender,
        country: val.country,
        city: val.city,
```



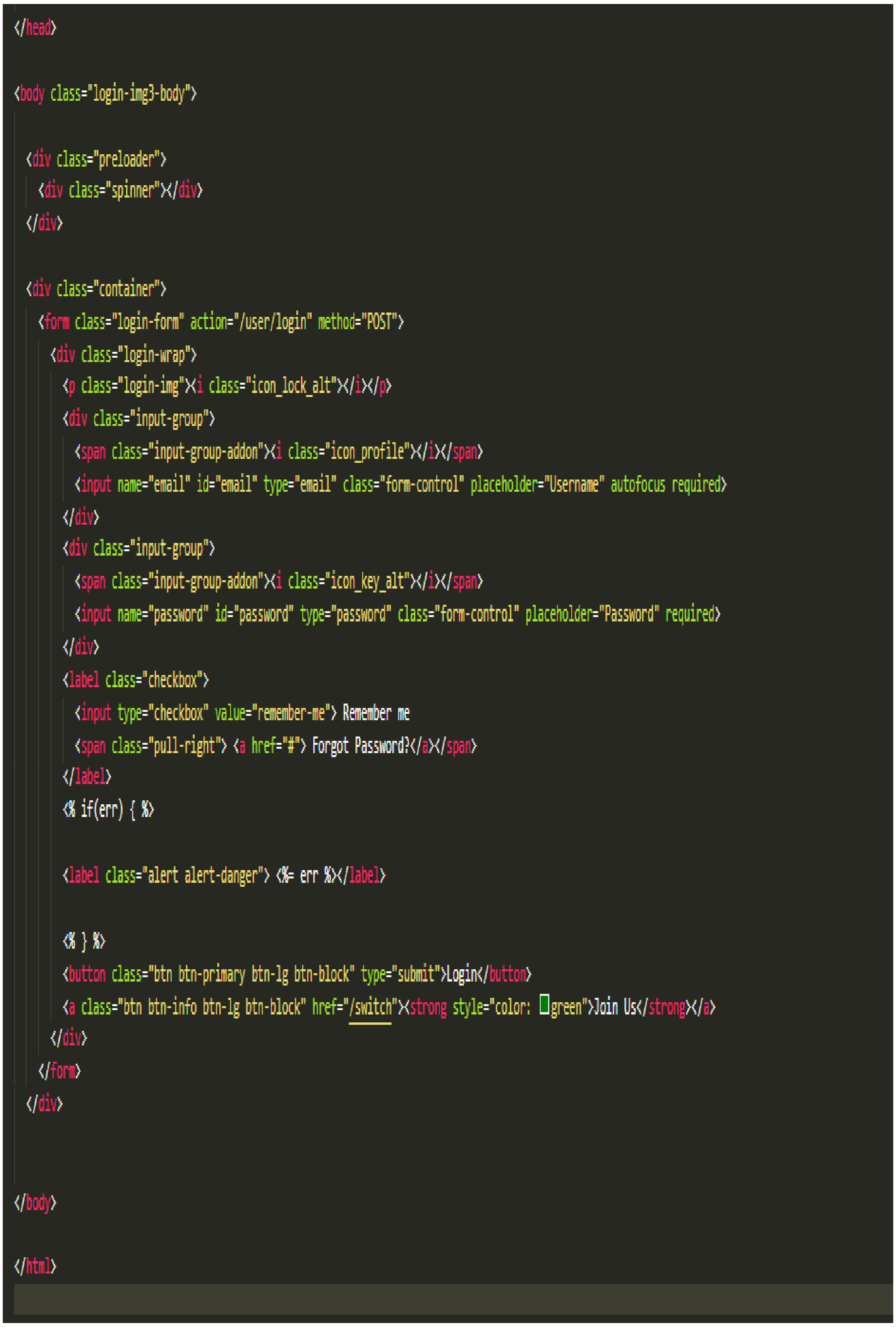
```
        major_skill: val.user_job,
        experience_years: val.experience,
        description: val.description,
        skills: val.skills,
        role: 2,
        files: files
    };

    router.all('/editprofile', function (req, res, next) {
        var dd = req.body;
          var edit = {
            First_name : dd .user_job,
            Last_name : dd .Last_name,
            Company : dd .Company,
            Email : dd .Email,
            Skills : dd .Skills,
            Edit_Work_Link : dd .JEdit_Work_Link,
            Edit_Description : dd .Edit_Description
          };
        // console.log(data)
        console.log(edit );

        DbPost.pushPost(dd );

        res.render('index', { title: 'done' });

      });
    var err = await DbUser.signUp(user);
    console.log("err = " + err)

    if (err != null) {
        return res.render('signUp', { err: err });
    }
    res.render('login');
});

router.post('/signupcompany', upload.any(), async (req, res, next) => {
    console.log('here');
    var val = req.body;
    var files = req.files;
    console.log(files);
```



```
    let user = {
        name: val.name,
        email: val.email,
        password: val.password,
        company_name: val.company_name,
        country: val.country,
        city: val.city,
        address: val.address,
        Foundation: val.Foundation,
        description: val.description,
        role: 3,
        files: files
    };

    var err = await DbUser.CompanysignUp(user);
    console.log("err = " + err)

    if (err != null) {
        return res.render('signUpCompany', { err: err });
    }
    res.render('index', { title: 'done' });
});

router.get('/logout', (req, res, next) => {
    req.session.destroy();

    res.render('login', { err: null });
});

router.get('/editprofile', (req, res, next) => {
    console.log('here');
    res.render('editprofile');
});



module.exports = router;
```

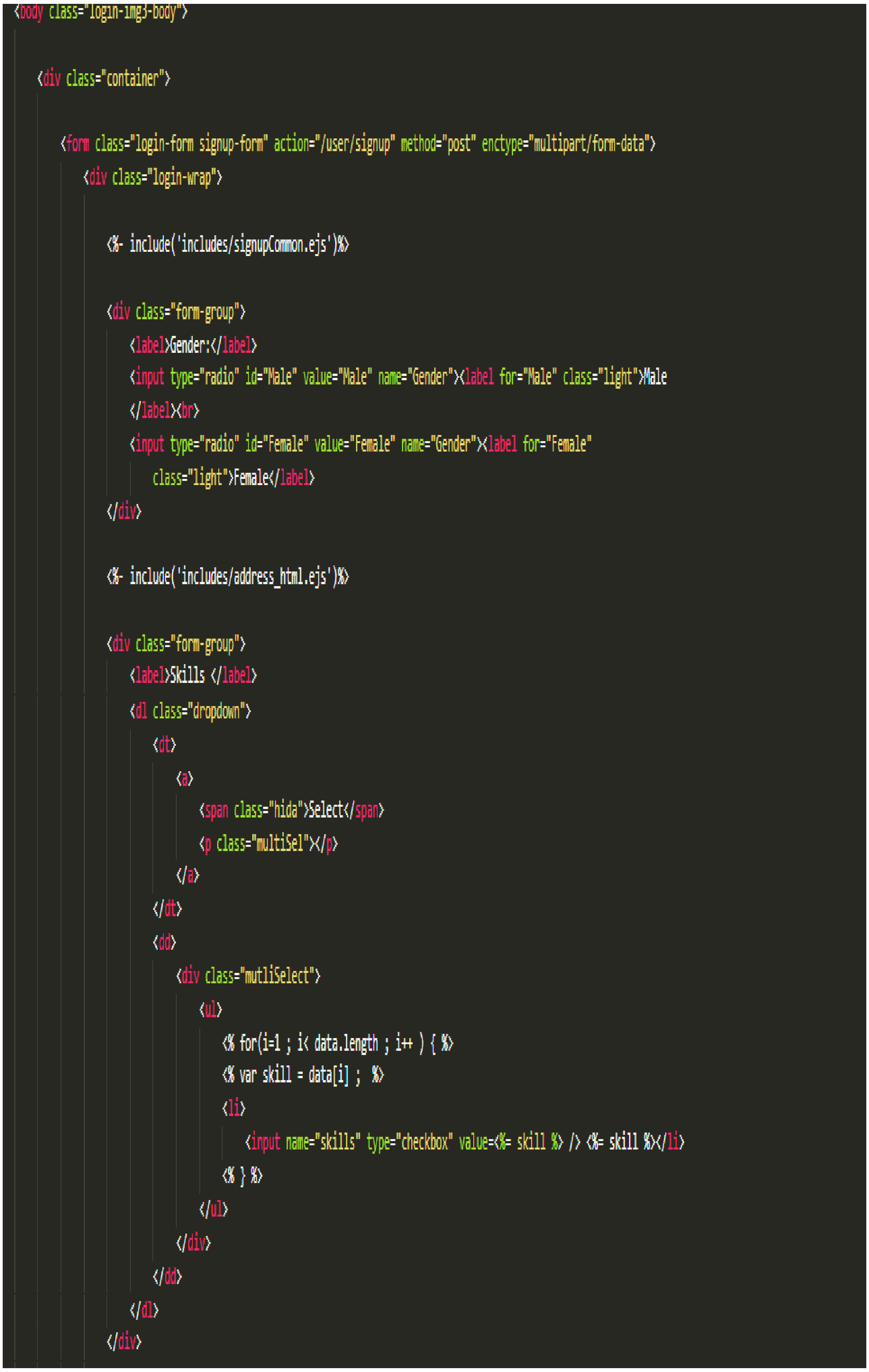
4.2.3 Views:

Login:

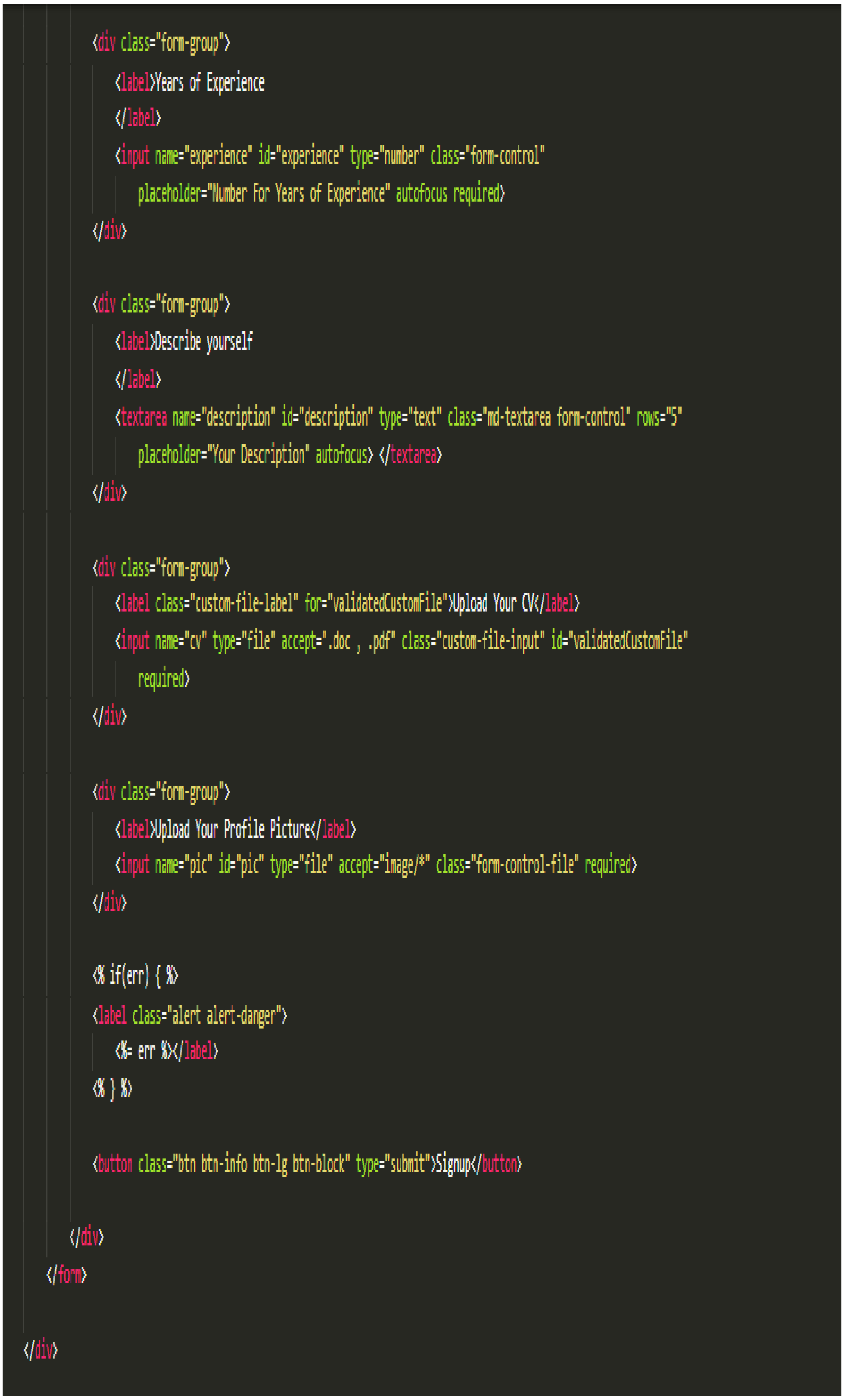


```
</head>

<body class="login-img3-body">

  <div class="preloader">
    <div class="spinner"></div>
  </div>

  <div class="container">
    <form class="login-form" action="/user/login" method="POST">
      <div class="login-wrap">
        <p class="login-img"><i class="icon_lock_alt"></i></p>
        <div class="input-group">
          <span class="input-group-addon"><i class="icon_profile"></i></span>
          <input name="email" id="email" type="email" class="form-control" placeholder="Username" autofocus required>
        </div>
        <div class="input-group">
          <span class="input-group-addon"><i class="icon_key_alt"></i></span>
          <input name="password" id="password" type="password" class="form-control" placeholder="Password" required>
        </div>
        <label class="checkbox">
          <input type="checkbox" value="remember-me"> Remember me
          <span class="pull-right"> <a href="#"> Forgot Password?</a></span>
        </label>
        <% if(err) { %>

        <label class="alert alert-danger"> <%= err %></label>

        <% } %>
        <button class="btn btn-primary btn-lg btn-block" type="submit">Login</button>
        <a class="btn btn-info btn-lg btn-block" href="/switch"><strong style="color: green">Join Us</strong></a>
      </div>
    </form>
  </div>



</body>

</html>
```

Register:

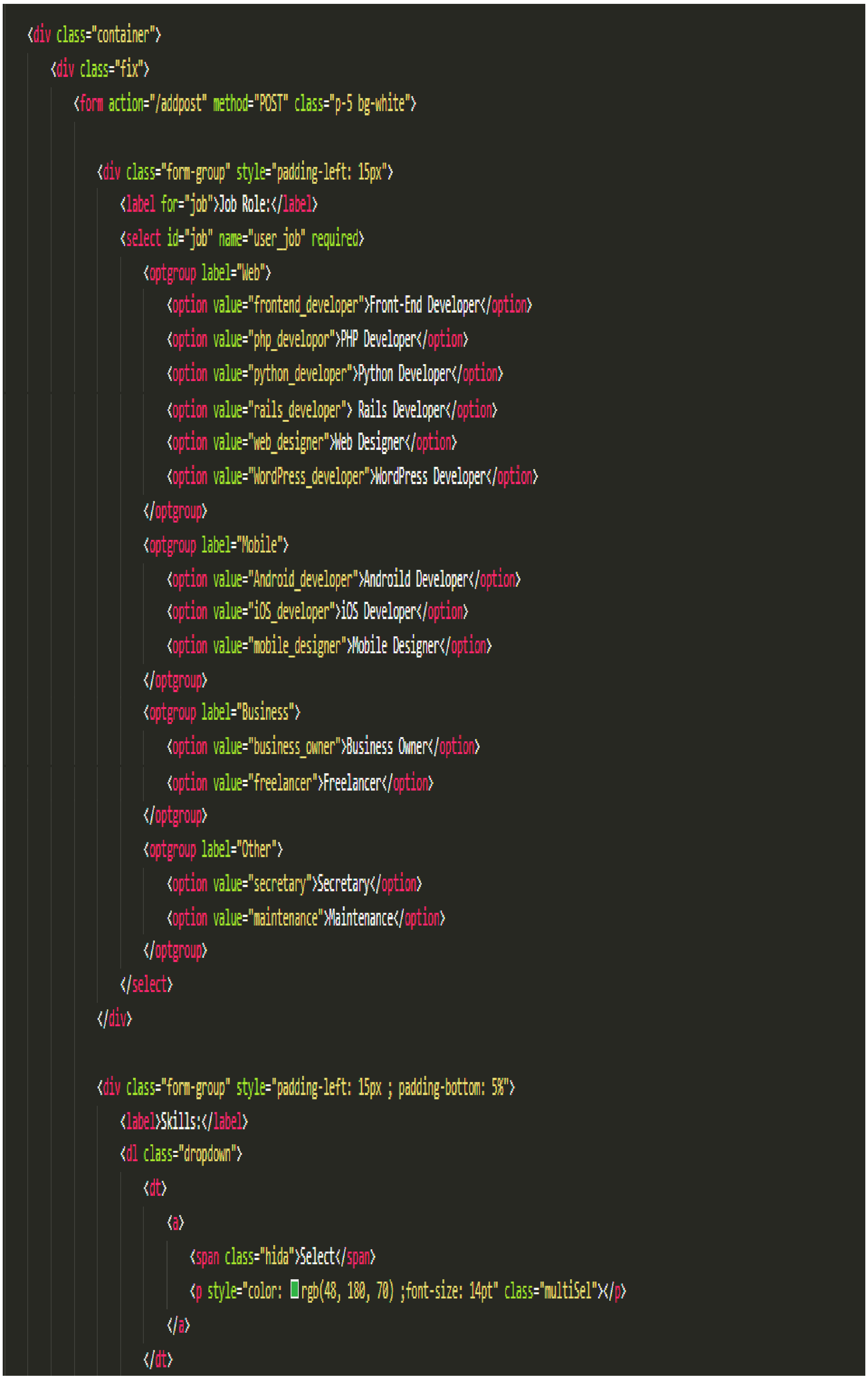


```
<body class="login-img3-body">


    <div class="container">

        <form class="login-form signup-form" action="/user/signup" method="post" enctype="multipart/form-data">
            <div class="login-wrap">

                <%- include('includes/signupCommon.ejs')%>

                <div class="form-group">
                    <label>Gender:</label>
                    <input type="radio" id="Male" value="Male" name="Gender"><label for="Male" class="light">Male
                    </label><br>
                    <input type="radio" id="Female" value="Female" name="Gender"><label for="Female"
                        class="light">Female</label>
                </div>

                <%- include('includes/address_html.ejs')%>

                <div class="form-group">
                    <label>Skills </label>
                    <dl class="dropdown">
                        <dt>
                            <a>
                                <span class="hida">Select</span>
                                <p class="multiSel"></p>
                            </a>
                        </dt>
                        <dd>
                            <div class="mutliSelect">
                                <ul>
                                    <% for(i=1 ; i< data.length ; i++ ) { %>
                                    <% var skill = data[i] ;  %>
                                    <li>
                                        <input name="skills" type="checkbox" value=<%= skill %> /> <%= skill %></li>
                                    <% } %>
                                </ul>
                            </div>
                        </dd>
                    </dl>
                </div>
```

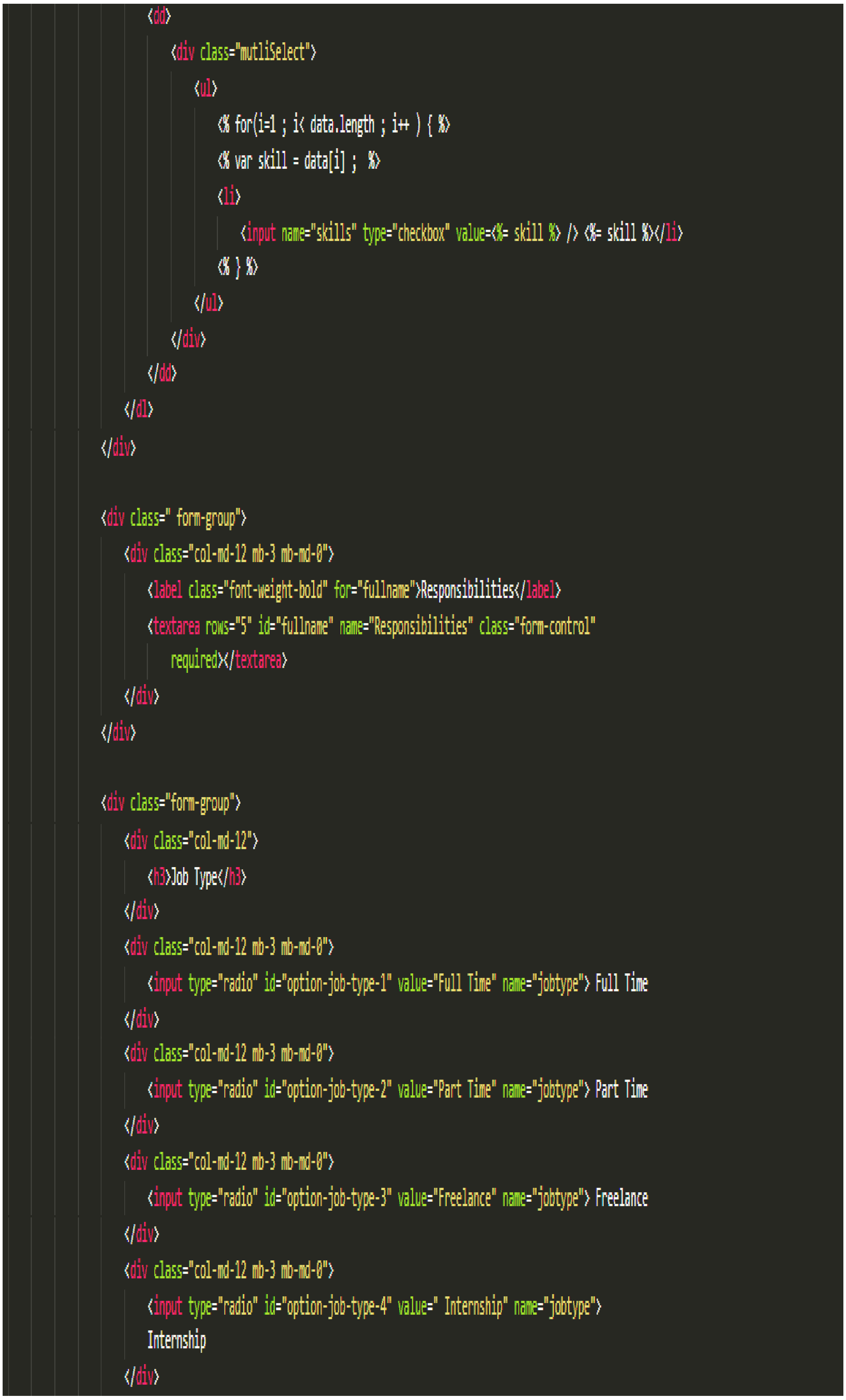


```
            <div class="form-group">
                <label>Years of Experience
                </label>
                <input name="experience" id="experience" type="number" class="form-control"
                    placeholder="Number For Years of Experience" autofocus required>
            </div>

            <div class="form-group">
                <label>Describe yourself
                </label>
                <textarea name="description" id="description" type="text" class="md-textarea form-control" rows="5"
                    placeholder="Your Description" autofocus> </textarea>
            </div>

            <div class="form-group">
                <label class="custom-file-label" for="validatedCustomFile">Upload Your CV</label>
                <input name="cv" type="file" accept=".doc , .pdf" class="custom-file-input" id="validatedCustomFile"
                    required>
            </div>

            <div class="form-group">
                <label>Upload Your Profile Picture</label>
                <input name="pic" id="pic" type="file" accept="image/*" class="form-control-file" required>
            </div>

            <% if(err) { %>
            <label class="alert alert-danger">
                <%= err %></label>
            <% } %>

            <button class="btn btn-info btn-lg btn-block" type="submit">Signup</button>

        </div>
    </form>

</div>
```

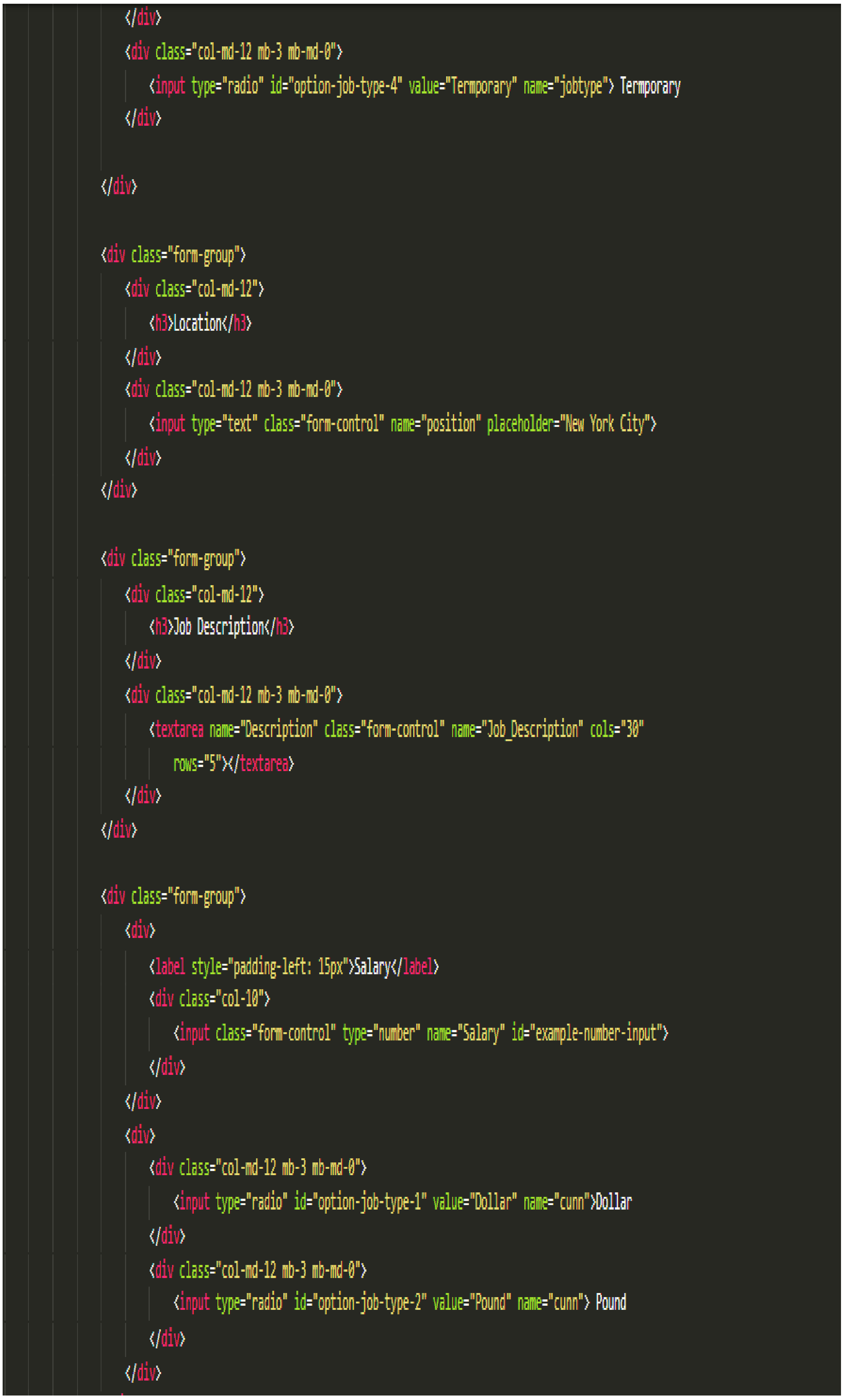
Add Post:



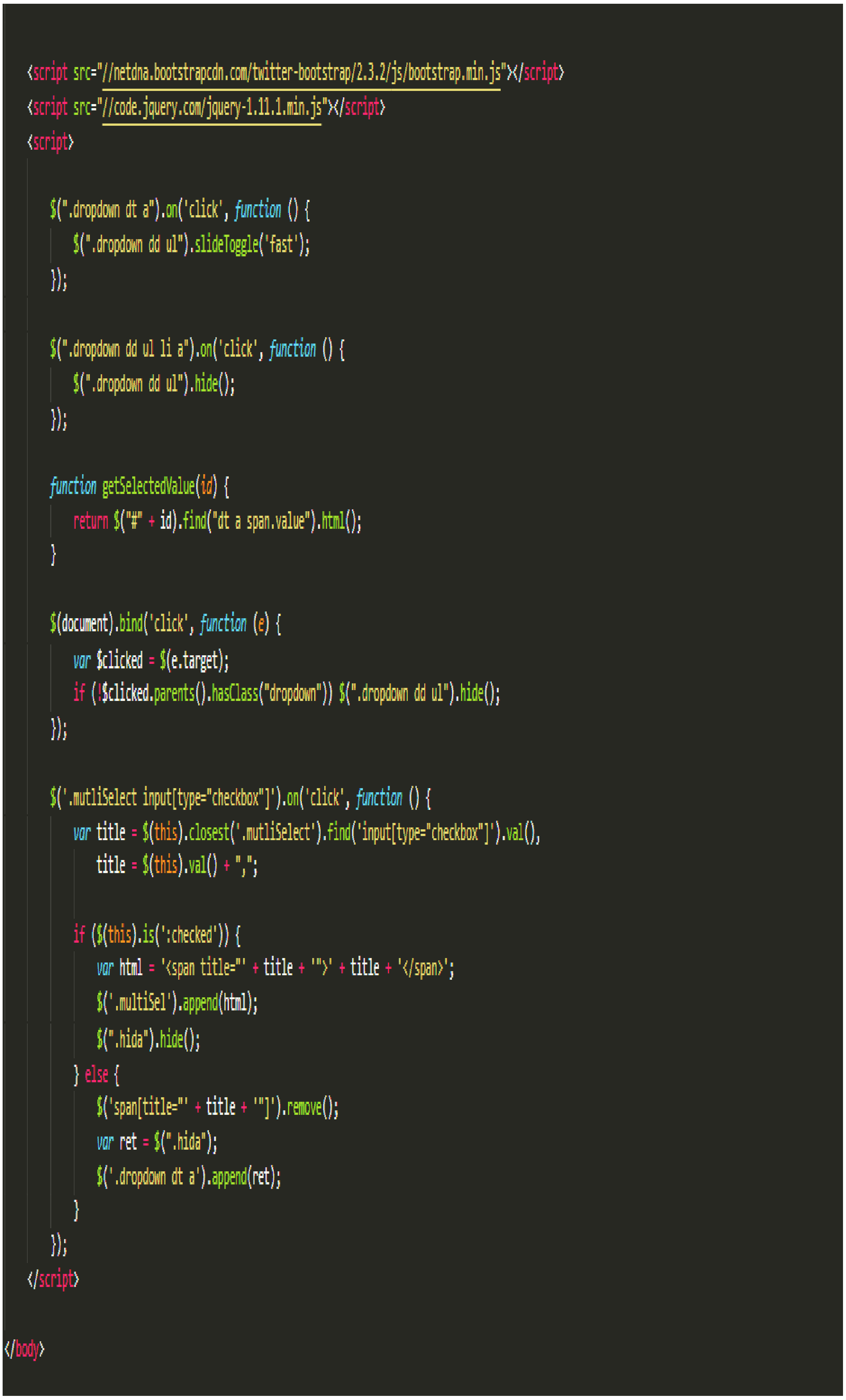
```
<div class="container">
    <div class="fix">
        <form action="/addpost" method="POST" class="p-5 bg-white">

            <div class="form-group" style="padding-left: 15px">
                <label for="job">Job Role:</label>
                <select id="job" name="user_job" required>
                    <optgroup label="Web">
                        <option value="frontend_developer">Front-End Developer</option>
                        <option value="php_developor">PHP Developer</option>
                        <option value="python_developer">Python Developer</option>
                        <option value="rails_developer"> Rails Developer</option>
                        <option value="web_designer">Web Designer</option>
                        <option value="WordPress_developer">WordPress Developer</option>
                    </optgroup>
                    <optgroup label="Mobile">
                        <option value="Android_developer">Androild Developer</option>
                        <option value="iOS_developer">iOS Developer</option>
                        <option value="mobile_designer">Mobile Designer</option>
                    </optgroup>
                    <optgroup label="Business">
                        <option value="business_owner">Business Owner</option>
                        <option value="freelancer">Freelancer</option>
                    </optgroup>
                    <optgroup label="Other">
                        <option value="secretary">Secretary</option>
                        <option value="maintenance">Maintenance</option>
                    </optgroup>
                </select>
            </div>

            <div class="form-group" style="padding-left: 15px ; padding-bottom: 5%">
                <label>Skills:</label>
                <dl class="dropdown">
                    <dt>
                        <a>
                            <span class="hida">Select</span>
                            <p style="color: rgb(48, 180, 70) ;font-size: 14pt" class="multiSel"></p>
                        </a>
                    </dt>
```



```
        <dd>
            <div class="mutliSelect">
                <ul>
                    <% for(i=1 ; i< data.length ; i++ ) { %>
                    <% var skill = data[i] ;  %>
                    <li>
                        <input name="skills" type="checkbox" value=<%= skill %> /> <%= skill %></li>
                    <% } %>
                </ul>
            </div>
        </dd>
    </dl>
</div>

<div class=" form-group">
    <div class="col-md-12 mb-3 mb-md-0">
        <label class="font-weight-bold" for="fullname">Responsibilities</label>
        <textarea rows="5" id="fullname" name="Responsibilities" class="form-control"
            required></textarea>
    </div>
</div>

<div class="form-group">
    <div class="col-md-12">
        <h3>Job Type</h3>
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="radio" id="option-job-type-1" value="Full Time" name="jobtype"> Full Time
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="radio" id="option-job-type-2" value="Part Time" name="jobtype"> Part Time
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="radio" id="option-job-type-3" value="Freelance" name="jobtype"> Freelance
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="radio" id="option-job-type-4" value=" Internship" name="jobtype">
        Internship
    </div>
```



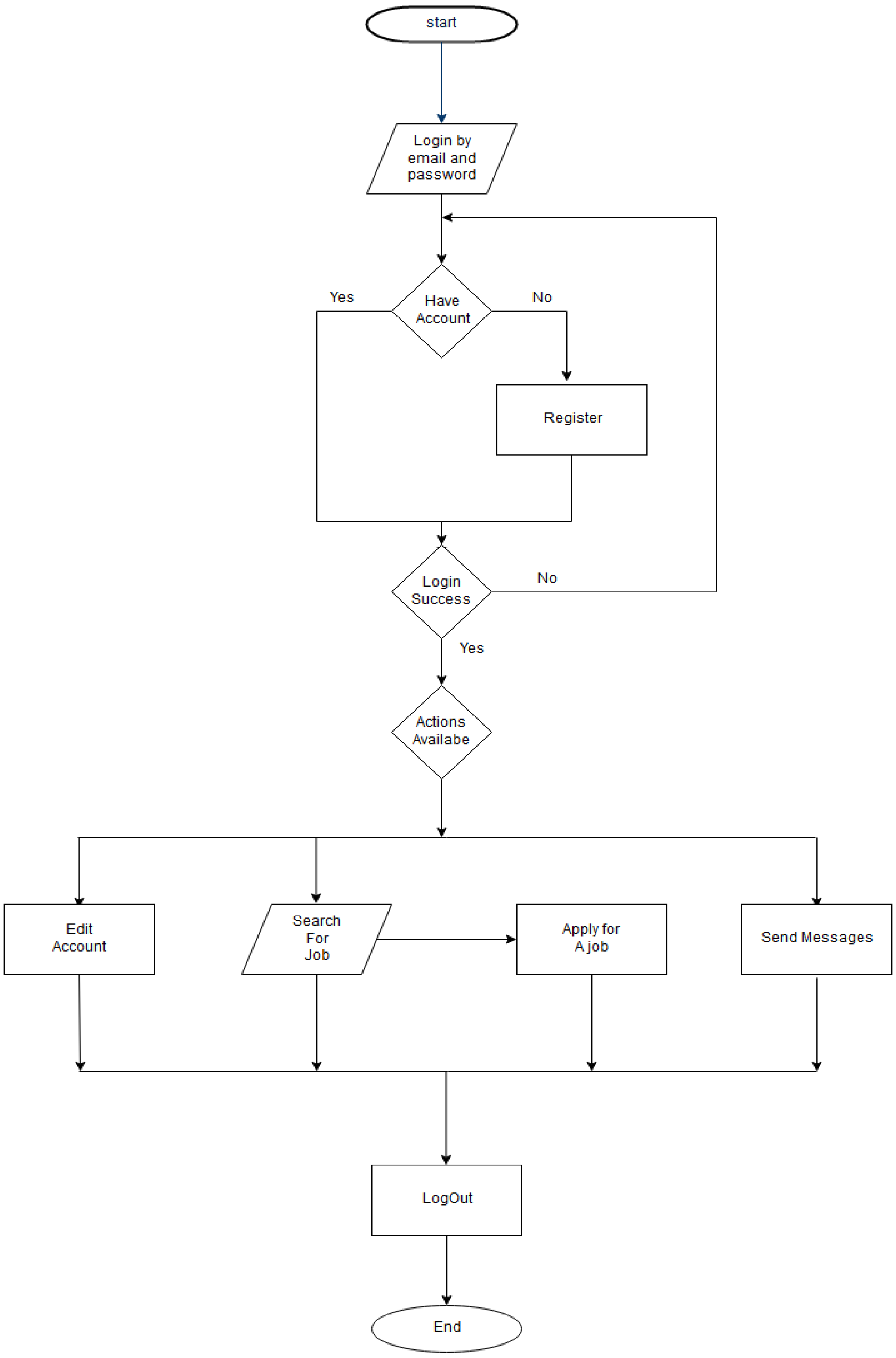
```
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="radio" id="option-job-type-4" value="Termporary" name="jobtype"> Termporary
    </div>

</div>

<div class="form-group">
    <div class="col-md-12">
        <h3>Location</h3>
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <input type="text" class="form-control" name="position" placeholder="New York City">
    </div>
</div>

<div class="form-group">
    <div class="col-md-12">
        <h3>Job Description</h3>
    </div>
    <div class="col-md-12 mb-3 mb-md-0">
        <textarea name="Description" class="form-control" name="Job_Description" cols="30"
            rows="5"></textarea>
    </div>
</div>

<div class="form-group">
    <div>
        <label style="padding-left: 15px">Salary</label>
        <div class="col-10">
            <input class="form-control" type="number" name="Salary" id="example-number-input">
        </div>
    </div>
    <div>
        <div class="col-md-12 mb-3 mb-md-0">
            <input type="radio" id="option-job-type-1" value="Dollar" name="cunn">Dollar
        </div>
        <div class="col-md-12 mb-3 mb-md-0">
            <input type="radio" id="option-job-type-2" value="Pound" name="cunn"> Pound
        </div>
    </div>
```



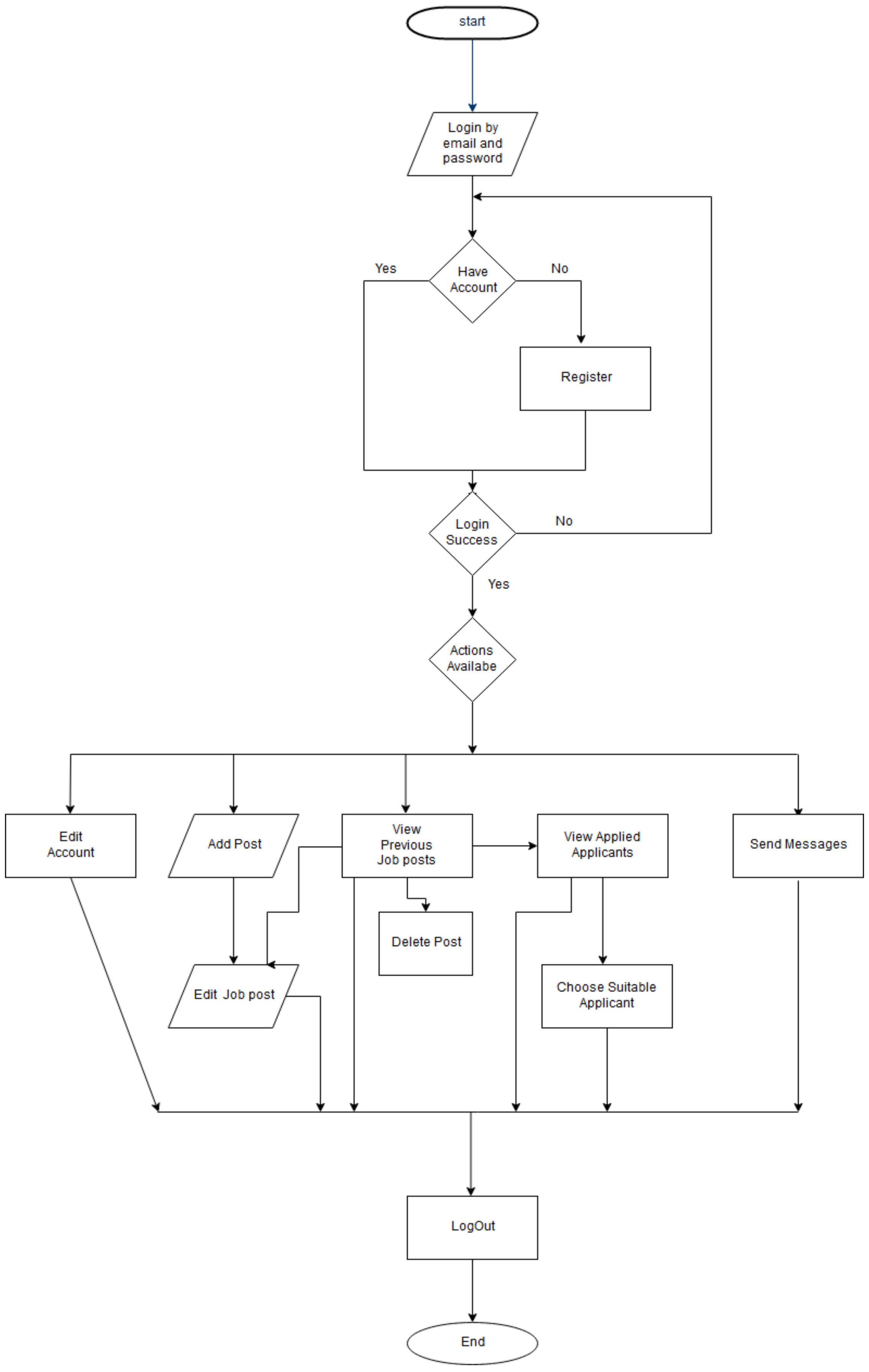
```
    <script src="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/js/bootstrap.min.js"></script>
    <script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
    <script>

        $(".dropdown dt a").on('click', function () {
            $(".dropdown dd ul").slideToggle('fast');
        });

        $(".dropdown dd ul li a").on('click', function () {
            $(".dropdown dd ul").hide();
        });

        function getSelectedValue(id) {
            return $("#" + id).find("dt a span.value").html();
        }

        $(document).bind('click', function (e) {
            var $clicked = $(e.target);
            if (!$clicked.parents().hasClass("dropdown")) $(".dropdown dd ul").hide();
        });

        $('.mutliSelect input[type="checkbox"]').on('click', function () {
            var title = $(this).closest('.mutliSelect').find('input[type="checkbox"]').val(),
                title = $(this).val() + ",";

            if ($(this).is(':checked')) {
                var html = '<span title="' + title + '">' + title + '</span>';
                $('.multiSel').append(html);
                $(".hida").hide();
            } else {
                $('span[title="' + title + '"]').remove();
                var ret = $(".hida");
                $('.dropdown dt a').append(ret);
            }
        });
    </script>

</body>
```

4.3 FlowCharts:

User:



HR:


start
Login by email and password
Have Account
Yes
No
Register
Login Success
No
Yes
Actions Availabe
Edit Account
Add Post
View Previous Job posts
View Applied Applicants
Send Messages
Delete Post
Edit Job post
Choose Suitable Applicant
LogOut
End

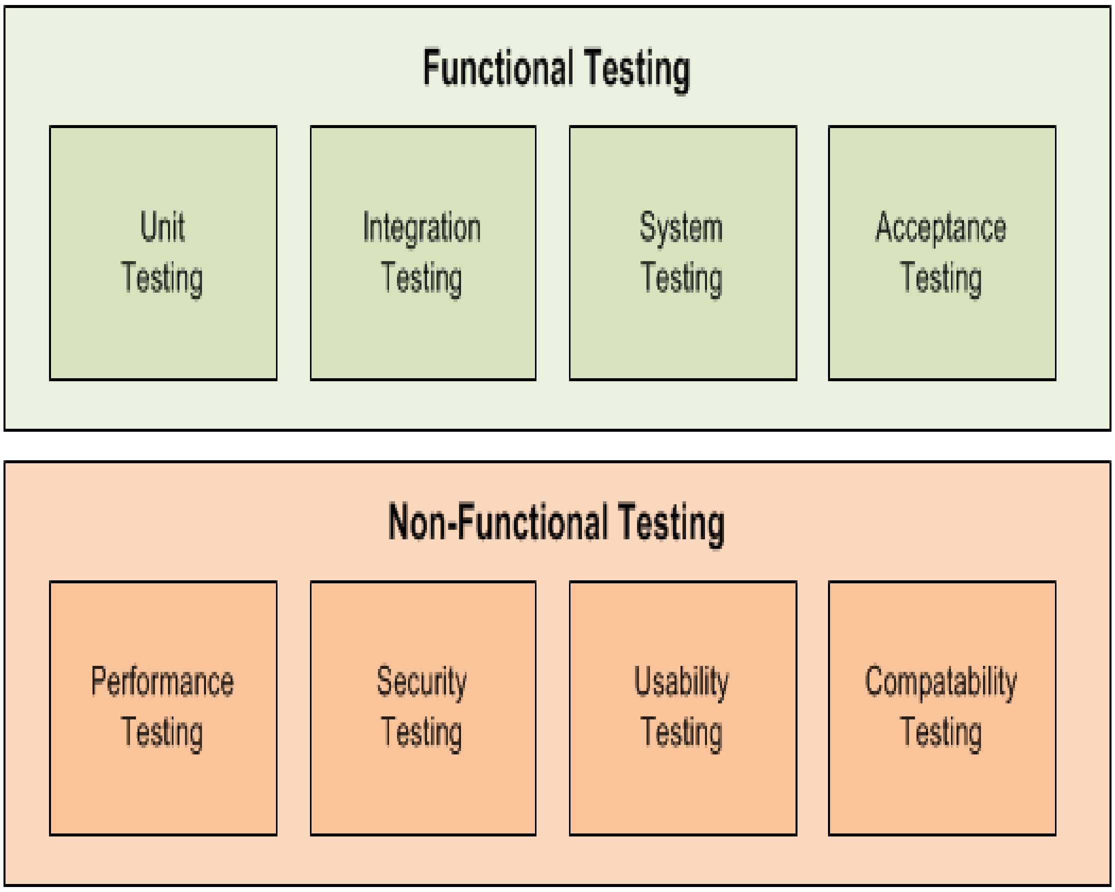
Chapter 5 : Testing

In this chapter, we will discuss the testing method and test cases used to make sure the the system is reliable and error-free.

.

5.1 Types of testing:

There are a lot of testing methods and ways to measure a lot of aspects of a project such as performance, scalability, reliability, etc.



5.2 Functional Testing:

Unit Testing:

Testing of individual items (e.g. modules, programs, objects, classes, etc.)Usually as part of the coding phase, in isolation from other development item sand the system as a whole.

Integration testing:

Testing the interfaces between major (e.g. systems-level application modules) and minor (e.g. individual programs or components)items within an application which must interact with each other.

System testing:

Testing a system behavior as a whole when development is finished and the system can be tested as a complete entity.

Acceptance testing:

Testing to ensure that development is ready to be deployed into the business, operational or production environment.

5.3 Non-Functional Testing:

Performance Testing:

Accomplished a designated function regarding processing time and throughput rate.

Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources.

Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system.

Compatibility:

Test if the system is compatible with other platforms and different hardware and systems.

5.4 Test Cases:

-Test Scenario Objective:

Make Sign up (as Applicant) successfully

-Assumptions/Dependencies:

Valid full name: Abanoub Talaat
Valid email: abanoub@gmail.com
Valid password: 123456
Gender: Male
Country: Egypt
City/Town: Cairo
Valid address line: 145 st Shubra
Skills: HTML, CSS, JS, Node.JS
Job Role: Web Developer
Years of experience: 1
Describe your self: Creative
Upload your C.V.: abanoub.docx
Upload your profile picture: img.png

Step#	**Description**	**Expected Result**	**Actual Result**	**Error Type**
1.	Navigate to sign up(as applicant) screen.	User navigate to sign up page.		
2.	Enter full name : Abanoub Talaat	User can enter full name		
3.	Press tab	Move to text field called email		

4.	Enter email : abanoub@gmail.com	User can enter email		
5.	Press tab	Move to text field called password		
6.	Enter password : 123456	User can enter password		
7.	Press tab	Move to Checkbox field called gender		
8.	Select Gender : Male	User can select gender		
9.	Press tab	Move to selection field called country		
10.	Select country : Egypt	User can select country		
11.	Press tab	Move to text field called City/Town		
12.	Enter City/Town: Cairo	User can enter city/town		
13.	Press tab	Move to text field called address line		
14.	Enter address line: 145 st Shubra	User can enter address line		
15.	Press tab	Move to selection field called skills		
16.	Select skills : HTML, CSS, JS, Node.JS	User can select skills		
17.	Press tab	Move to selection field called job role		

18.	Select job role : Web developer	User can select job role		
19.	Press tab	Move to text field called years of experience		
20.	Enter years of experience: 1	User can enter years of experience		
21.	Press tab	Move to text field called describe your self		
22.	Enter describe your self : Creative	User can enter describe your self		
23.	Press tab	Move to choose file called upload your C.V.		
24.	Upload your C.V.: abanoub.docx	User can upload his C.V.		
25.	Press tab	Move to choose file called upload your profile picture		
26.	Upload your profile picture: img.png	User can upload his profile picture		

-Test Scenario Objective:

Make Sign up (as Company) successfully

-Assumptions/Dependencies:

Valid full name: Facebook
Valid email: facebook@gmail.com
Valid password: 123
Company name : Facebook
Country: Egypt
City/Town: Cairo
Valid address line: 9 elmaadi
Foundation year : 1990
Company Describe: Social Media
Upload your profile picture: img.png

Step#	**Description**	**Expected Result**	**Actual Result**	**Error Type**
1.	Navigate to sign up(as campany) screen.	User navigate to sign up page.		
2.	Enter full name : Facebook	User can enter full name		
3.	Press tab	Move to text field called email		
4.	Enter email : facebook@gmail.com	User can enter email		
5.	Press tab	Move to text field called password		
6.	Enter password : 123	User can enter password		
7.	Press tab	Move to text field called company name		

8.	Enter company name : Facebook	User can enter company name		
9.	Press tab	Move to selection field called country		
10.	Select country : Egypt	User can select country		
11.	Press tab	Move to text field called City/Town		
12.	Enter City/Town: Cairo	User can enter city/town		
13.	Press tab	Move to text field called address line		
14.	Enter address line: 9 elmaadi	User can enter address line		
15.	Press tab	Move to text field called foundation year		
16.	Enter Foundation year: 1990	User can enter foundation		
17.	Press tab	Move to selection field called company description		
18.	Enter company description: Social Media	User can enter company description		
19.	Press tab	Move to choose file called upload your profile picture		
20.	Upload your profile picture: img.png	User can upload it profile picture		

-Test Scenario Objective:

Verify login successfully with correct email and correct password

-Assumptions/Dependencies:

Valid email: abanoub@gmail.com

Valid password: 123

Step#	Description	Expected Result	Actual Result	Error Type
1.	Navigate to login screen.	User navigate to login page.		
2.	Enter email : abanoub@gmail.com	User can enter email		
3.	Press tab	Move to text field called password		
4.	Enter password:123	User can enter password		
5.	Click on login button	Login will make successfully, home page will open	passed	

-Test Scenario Objective:

Add post successfully

-Assumptions/Dependencies:

Valid job title: Full Stack .NET developer
Valid job description : We are a software house working in Web Development, Mobile Applications and Data Analytics

Valid responsibilities: Participatein requirements analysis

Valid skills needed : HTML5, CSS3, JS, Web API
Salary : 5000$
Valid company address : 9 elmaadi
Valid HR Email: ahmed@gmail.com

Step#	**Description**	**Expected Result**	**Actual Result**	**Error Type**
1.	Navigate to add post screen.	User navigate to add post page.		
2.	Enter job title : Full Stack .NET developer	User can enter job title		
3.	Press tab	Move to text field called job desc.		
4.	Enter job desc. : : We are a software house working in Web Development, Mobile Applications and Data Analytics	User can enter job desc		
5.	Press tab	Move to text field called responsibilities		
6.	Enter responsibilities : Participatein requirements analysis	User can enter responsibilities		
7.	Press tab	Move to Selection field called skills		

		neeeded		
8.	Select skills needed : HTML5, CSS3, JS, Web API	User can select skills needed		
9.	Press tab	Move to text field called salary		
10.	Enter Salary : 5000$	User can enter salary		
11.	Press tab	Move to text field called Company address		
12.	Enter Company address: 9 elmaadi	User can enter company address		
13.	Press tab	Move to text field called HR Email		
14.	Enter HR Email:ahmed@gmail.com	User can enter HR Email		
15.	Click on add post button	Add post will make successfully,and message will appear to tell you that you added post	passed	

Chapter 6 : Results and Discussion

In this chapter, we will discuss the results achieved and what was the targets from the beginning and their progress till now.

.

6.1 Expected Results:

From the beginning we wanted to create a system that would deliver a better experience as a recruitment website for all users, some of the targets are:

1-both applicants and HRs can register and login in a simple way.

2-HRs can add job posts detailing all they want in the candidates in an easy way.

3-applicant can search and find the perfect job for him.

4-applicant can apply for any job he finds suitable.

5-HR can reach the applicant afterward and communicate with him

4-The system would make things easy for both applicants and HR by matching them together automatically

6.2 Actual Results achieved:

We were able to achieve all the predefined goals except for a good matching system which would be added in future work.

6.3 Discussion:

The system was full of features and we tried to implement them in an effective and reliable way a lot of work was done but there is more to come.

Chapter 7 : Conclusion and Future work

FInally by the end of the project we were very happy by the results we achieved it was a great experience full of knowledge and learning we tried to make a new system that could help many of people like us who are trying to find a job or work but found it difficult or didn't know where to start we made the system by making it simple and ease of use as a priority we are considering future work such as adding the matching system and implementing geolocation API and many more.