A

Project Report On

“Job Portal”

By

Darshik Marvaniya (CE-067) (20CEUOS111)

Prashant Nakum(CE-074)(20CEUBG040)

B. Tech CE Semester – VI

Subject: System Design Practice

Guided by :
Prof. Brijesh S. Bhatt

Dharmsinh Desai University, Nadiad
Faculty of Technology
Department of Computer Engineering

Dharmsinh Desai University, Nadiad
Faculty of Technology
Department of Computer Engineering

CERTIFICATE

This is to certify that **System Design Practice** Project entitled "**Job Portal**" is the bonafide report of work carried out by,

Darshik Marvaniya (CE-067) (20CEUOS111)
Prashant Nakum(CE-074)(20CEUBG040)

of B. Tech semester **VI** in the branch of **Computer Engineering** during the academic year **2022-2023.**

Prof. Brijesh S. Bhatt
Professor,
Computer Engg. Department
Faculty of Technology
Dharmsinh Desai University

Dr. C. K. Bhensdadia
Professor and HoD,
Computer Engg. Department
Faculty of Technology
Dharmsinh Desai University

Table of Contents

1. Introduction.....4
    - Brief Overview.....4
    - Plan.....4
    - Approach.....4
2. Analysis.....5
    - Problem Statement.....5
    - Scope.....5
    - Software requirement specification(SRS).....5
    - UseCase Diagram.....9
3. Design.....10
    - Class Diagram.....10
    - Sequence Diagram.....11
    - Activity Diagram.....12
    - Module Of System.....13
    - Database Design.....14
4. Implementation Details.....16
    - Tools and Technology.....16
    - Function prototype.....18
5. Testing.....22
6. Screen-Shots.....23
7. Conclusion.....29
8. Limitation and Future Extension.....30
9. Bibliography.....31

Introduction

- Job Portal is about the online recruitment process. The system manages the recruitment process. This platform provides us the opportunity to choose the best amongst the various available jobs in a particular field.
- The person will be having the account after registration and will be referred to as the applied user.
- Nowadays the job market is so extensive that a variety of industries and companies are searching for the right candidates and the prospective candidates are searching for the right companies for growth opportunities. This purpose is served by most of the Job Portal.
- The Job Portal with an open environment for applicant and recruiters to meet on the same dais and know about each other so that the right candidate is placed in the right company.
- In the Job Portal, Employers can advertise the vacancies by taking the membership, logging in, and posting the job information with the eligibility criteria for the jobs. Any job seeker can search for available jobs at any moment with updated information. When he finds a job, he can post his application to the job online.
- Job Portal provides easy job application and gets a secure job of user interest.

Plan

- The first step would be to gather requirements. Based on requirements, we would design the system architecture, database schema, and user interface. The technology would be used is MERN stack (MongoDB, ExpressJs, ReactJs, NodeJs). Testing would be conducted to ensure code quality and functionality. Finally, I would maintain the application by monitoring it, fixing bugs, and enhancing features based on requirement of project.

Approach

- The approach to developing an online job portal project should involve following best practices in software development, such as modular programming, naming conventions, and code commenting. It's important to use version control to ensure code quality and catch errors early in the development process. Overall, our approach would be to prioritize code quality, functionality while also ensuring that the project is maintainable.

Analysis

Problem Statement

The problem that an online job portal project aims to solve is the lack of acentralized platform for applicant and Companies to connect with each other. In traditional job search methods, applicant have to manually search for job vacancies in newspapers, company websites, or through personal contacts. Similarly, companies have to go through the cumbersome process of advertising job vacancies in newspapers or on company websites, screening resumes, and conducting interviews.

Scope

The scope of an online job portal project includes the development of a web-based platform that allows Applicant to search for job vacancies based on their preferences and qualifications, and submit job applications online. Companies can post job vacancies, review resumes of potential candidates, and communicate with them through the platform. The project should also include features such as user authentication, and analytics to track user activity and job performance.

Software Requirement Specification

User of Application:

- Applicants
- Company
- Admin

Functional Requirements :

R.1 User authentication

R.1.1 Register

Description: If a user, if Applicants want to apply for a job or If Company want to add a job then first they have to register on the portal, after that they are able to apply for a job or add the job.

Input: User details (Name, Email, password etc.)

Output: Register successfully and able to login.

R.1.2 Login

Description: if users are already registered and want to apply for a job or want to add the job then they need to login first.
Input : User detail (name, email and password etc.)
Output : User will logged in and navigate to home page of application.

R.1.3 Log Out

Description: user can logout from the system by the click on logout.
Input: Click on logout.
Output: Logout from the portal.

R.2 Manage Applicants

R.2.1 Search job

Description: applicant can search for a job as per their perspective job.
Input: Enter the job title.
Output : Related to entered job title, jobs are filtered and displayed.

R.2.2 Apply For job

Description: If Applicants want to apply for any job then they can apply for a job.
Input: Click on the apply button. Enter the required information and click on the submit button.
Output : Display message your application is successfully submitted.

R.2.3 Update Application

Description: user can update their application details.
Input: click on update button and modify details.
Output : Display message your application updated successfully.

R.2.4 Delete Application

Description: If a user wants to delete their applied job application then the user can delete it.
Input: click on delete application button.
Output : Display message your application deleted successfully.

R.2.5 View/Update Profile

Description: applicant can view/update their profile.
Input: Click on view/update profile.
Output: Profile of user is displayed.

R.2.5 Give Suggestation/feedback to admin

Description: applicant can give any suggestation/feedback to admin.
Input: Click on contact us and fill the detail, write message.
Output: Display message message sended successfully.

R.3 Manage Company

R.3.1 Add job

Description : Company can add the job on a portal.
Input : Enter the job details (Company name, salary, technology etc.)
Output : Display message Job is added successfully.

R.3.2 Update job

Description : Company can update the details of the job which is added on the portal.
Input : Edit all the information and click on the update button.
Output : Display message data is successfully updated.

R.3.3 Delete job

Description : Company can delete added job from the portal.
Input : Click on the delete button for that job.
Output : Display message job is deleted successfully.

R.3.4 Accept/Reject Applicants Application

Description: After a user applies for the job, company can approve or delete him/her application for the job.
Input : Click on the approve/reject button for accept or reject the application of the user.
Output : User application is approved and user is informed.

R.3.5 View/Update Profile

Description: company can view/update their profile.
Input: Click on view/update profile.
Output: Profile of user is displayed.

R.3.6 Give Suggestation/feedback to admin

Description: applicant can give any suggestation/feedback to admin.
Input: Click on contact us and fill the detail, write message.
Output: Display message message sended successfully.

R.4 Manage Admin

R.4.1 Manage company/applicant /job/application details

Description: Admin can view/delete company/applicant/job/applicayion
Input: click on view/delete.
Output: all companies/applicant/job/application is displayed/deleted.

R.4.2 View/Delete user message

Description: admin can view the suggestation/feedback from the users.
Input : Click on view/delete message.
Output : All message is displayed/delete.

Non-functional requirement

N.1 Platform : The system should work on all kinds of OS and any web browser.

N.2 Database : A database management system that is available free of cost in the public domain should be used.

N.3 Availability : Our system will display the only Jobs that are available.

N.4 Recoverability : If our system fails then our system will be recovered within 20 minutes. As per the bond maintenance will be given to use.

N.5 Reliability: The system should not update the data in database for any processes. The system is able to update and delete information which is provided by system user very easily.

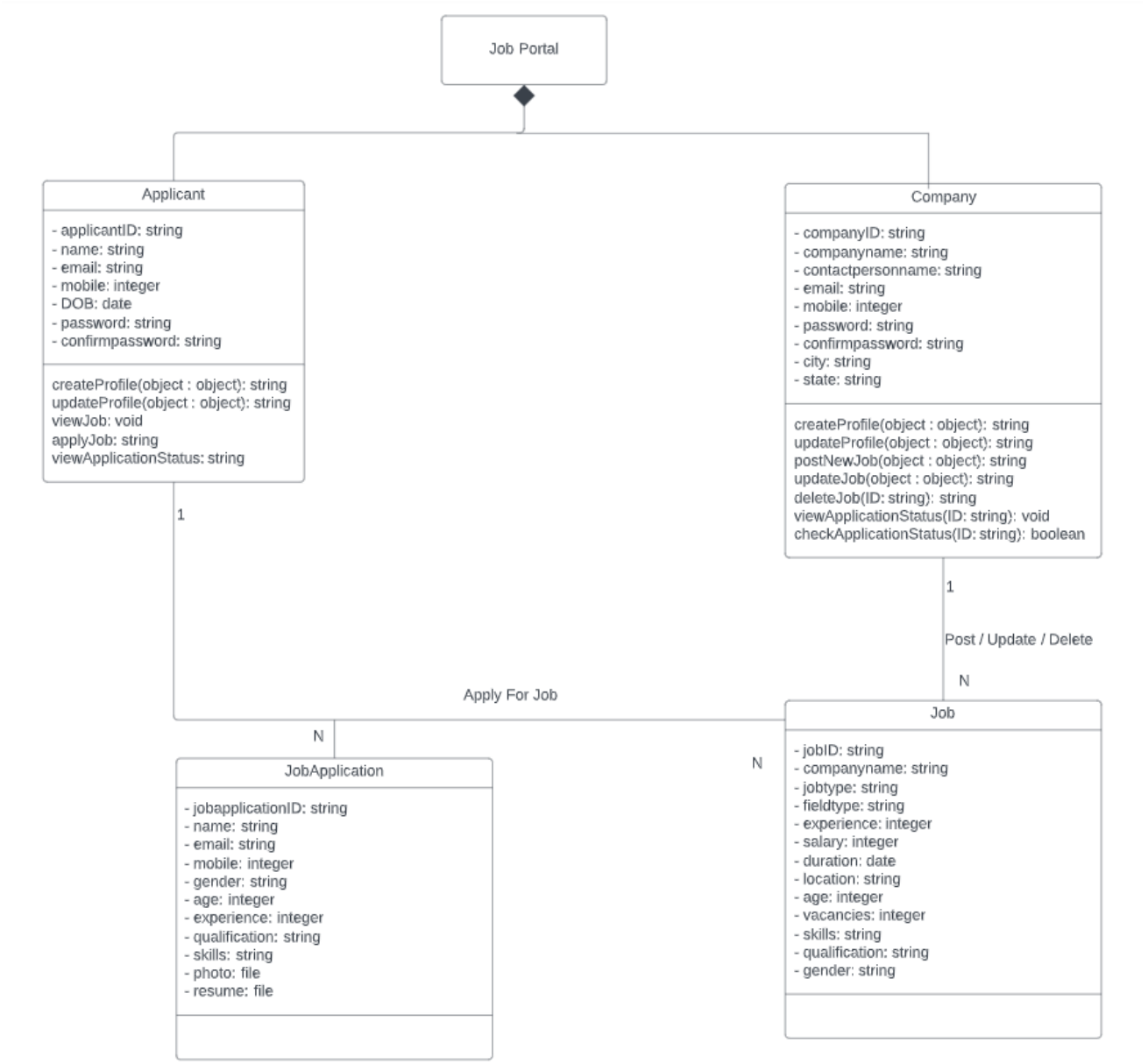
UseCase Diagram

Actors : Applicant, Company and Admin.
UseCases : Registration, Login, Apply For Job, Add job, Search Job, accept Reject Application etc.
The use case diagram should provide a clear and concise overview of how different users interact with the system and what tasks they can perform.

Design

Class Diagram

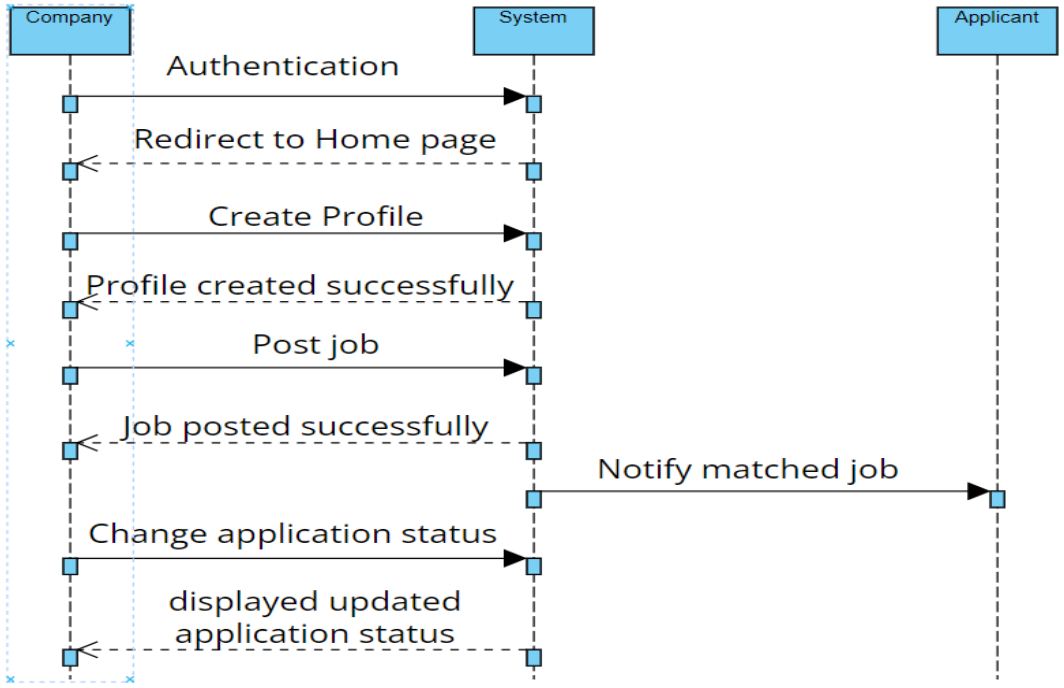
Classes : Applicant, Job, Job application and Company.
Class diagram will contain the relation, attributes and methods of the all classes. A class diagram provides a visual representation of the system's architecture and helps to define the structure of the system.



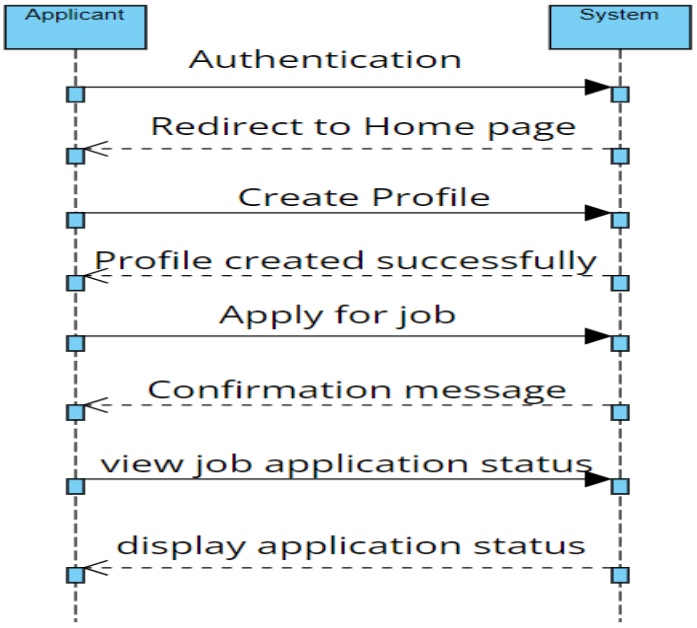
Sequence Diagram

Sequence diagram shows interactions between components of a system over time.How the all component will interect with each other and messages will be displayed. A sequence diagram provides a visual representation of the flow of events in a system, and can be used to analyze and optimize the system's performance.

- **For company**



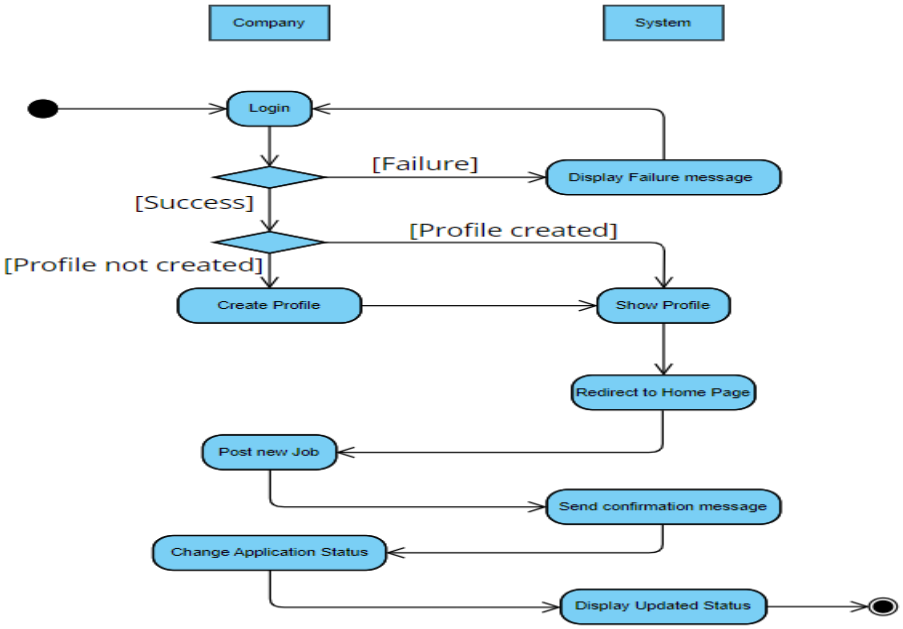
- **For Applicant**



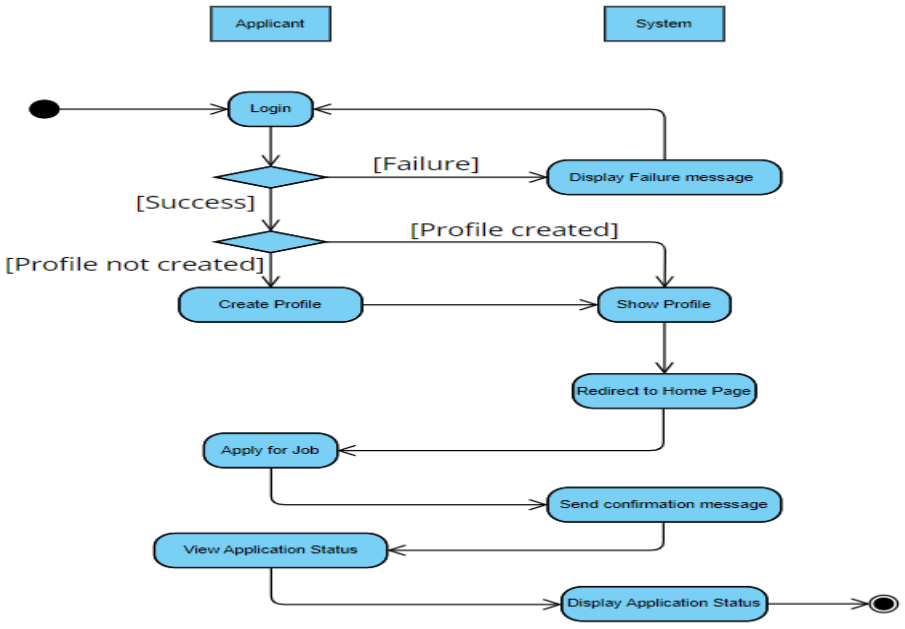
Activity Diagram

An activity diagram provides a visual representation of the flow of activities in a system, and can be used to analyze and optimize the system's performance. Here in dimond shape represents decisions and the Actions will be represented by rectangle with rounded corner.

- **For Company**



- **For Applicant**



Modules Description

The system consists of 2 main modules namely

1. Company Module

2. Applicant Module

Each module consists of several methods to implement the required functionality. Implementation is done using ReactJS, NodeJS, ExpressJS and MongoDB (for data storage).

Company Module :

Profile : In this section,company can view/update their profile.

Jobs : In this section,company can add job or update/delete/view all jobs.

Application : In this section, company can view all the application for that company and accept/reject application.

Applicant Module :

Profile : In this section,applicant can view/update their profile.

Jobs : In this section,applicant view all the jobs.

Search Jobs : In this section, Applicant can search the jobs based on job location or required skills for any job.

View Application status: In this section, Applicant can see the application status and all other detail and also they can update/delete application.

Database Design

Applicant Model		
Field Name	**Datatype**	**Constraints**
Id	String	Auto generated
Name	String	Required, maxLength=32
DOB	Date	Required
Email	String	Required, Unique
Mobile	Number	Required, Length=10
Password	String	Required, minLength=6
Confirm Password	String	Required
Role	Number	Default=1
Photo	String	-

Job Model		
Field Name	**Datatype**	**Constraints**
Id	String	Auto generated
Company Name	String	Required, maxLength=32
Company Email	String	Required
Job Type	String	Required
Field Type	String	Required
Gender	String	Required
Experience	String	Required
Salary	Number	Required
Open Date	Date	Required
Close Date	Date	Required
City	String	Required
State	String	Required
AgeLimit	Number	Required
Vacancies	String	Required
Skills	String	Required
Qualification	String	Required

Company Model		
Field Name	**Datatype**	**Constraints**
Id	String	Auto generated
Company Name	String	Required, maxLength=32
ContactPersonName	String	Required, maxLength=32
Email	String	Required, Unique
Mobile	Number	Required, Length=32
Password	String	Required, minLength=6
Confirm Password	String	Required
State	String	Required
City	String	Required
Role	Number	Default=1
Website	String	-
Linkedin	String	-

Application Model		
Field Name	**Datatype**	**Constraints**
Id	String	Auto generated
Name	String	Required, maxLength=32
Email	String	Required, Unique
Mobile	Number	Required, Length=32
Gender	String	Required
Age	Number	Required
Experience	String	Required
Qualification	String	Required
Skills	String	Required
Resume	File	Required
Hired	Number	Default=0
Company_id	ObjectId	Ref Company
Job_id	ObjectId	Ref Job

Message Model		
Field Name	**Datatype**	**Constraints**
Id	String	Auto generated
Name	String	Required, maxLength=32
Email	String	Required
Message	String	Required
Marked	Number	Default=0

Implementation

The project is divided into two parts: Client side and Server side.

Client side is implemented using ReactJs. It has all the required Components and Routing.

Server side is implemented using NodeJs, ExpressJs and mongoose.

MongoDB is used for data storage.

Technology

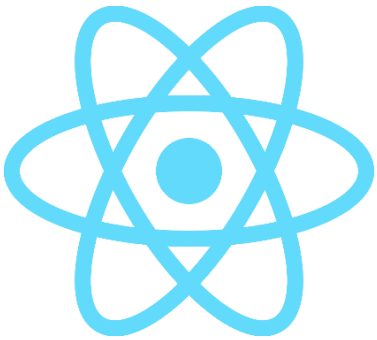
- MongoDB

MongoDB is a non-relational document database that provides support for JSON- like storage. The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support and replication with rich and intuitive APIs.

- ExpressJS

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on top of the Node js that helps manage servers and routes.

- ReactJS



React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.

- NodeJs

Node. js (Node) is an open-source development platform for executing JavaScript code server-side. Node is useful for developing applications that require a persistent connection from the browser to the server and is often used for real-time applications such as chat, news feeds, and web push notifications.

Tools

- Visual Studio Code
- Postman

Function Prototype

APIs:

Add new Job

```
  router.post('/addJob', async (req, res) => {

    const { vacancies, city, state, skills, ageLimit, experience, salary,
qualification, openDate, closeDate, jobType, fieldTypes, gender } = req.body;

    //check if any field is empty
    if(!vacancies || !city || !state || !skills || !ageLimit || !experience ||
!salary || !qualification || !openDate || !closeDate || !jobType || !fieldTypes ||
!gender){
          res.json({message : 'plz fill all the field'});
    }
  try{
    //verify with jwt token and get _id of current user from jwt token
    const token = req.cookies.jwttoken;
    const tokenVerify = jwt.verify(token, process.env.SECRET_KEY);
    const companyExist = await Company.findOne({ _id: tokenVerify._id,
"tokens.token": token});

    const email = companyExist.email;
    //create new job
    const job = new Job({ companyName:companyExist.companyName , email, vacancies,
city, state, skills, ageLimit, experience, salary, qualification, openDate,
closeDate, jobType, fieldTypes, gender });

    //save the job in database
    const addJob = await job.save();
    if(addJob){
        res.status(201).json({message : "Job added successfully"});

    }
    else{
        res.status(500).json({error: "Failed to add job"});
    }
}catch(err){
    console.log(err);
}
});
```

Search Job

```
router.get('/searchJob/:key', async (req, res) => {

   try{
      // find the jobs on the maching fields

      const result = await Job.find({
        "$or" : [
        {companyName : {$regex : req.params.key, $options: 'i'}},
         {fieldTypes : {$regex : req.params.key, $options: 'i'}},
         {email : {$regex : req.params.key, $options: 'i'}},
         {jobType : {$regex : req.params.key, $options: 'i'}},
         {experience : {$regex : req.params.key, $options: 'i'}},
         {salary : {$regex : req.params.key, $options: 'i'}},
         {skills : {$regex : req.params.key, $options: 'i'}},
         {qualification : {$regex : req.params.key, $options: 'i'}},
         {city : {$regex : req.params.key, $options: 'i'}},
         {state : {$regex : req.params.key, $options: 'i'}}
        ]
      });
      res.status(200).send(result);
    }
}
catch(err){
  console.log(err);
}
});
```

Apply for Job

```
router.post("/addApplication", upload.single("resume"), async (req, res) => {

    const { name, email, mobile, gender, age, experience, qualification, skills,
company_email, job_id } = req.body;

      //first check if any field is empty
    if(!name || !email || !mobile || !gender || !age || !experience ||
!qualification || !skills || !job_id){
                    return res.status(400).json({
                        error : "Plz fill all the fields"
                    })}
    try{
      //upload the resume pdf onto the cloudinary
       await cloudinary.uploader.upload(req.file.path, {resource_type : "raw"},
async (err, result) => {
             if(err){
```

```
                return res.status(500).json({error : "Pdf is not uploaded
successfully in cloudinary"});
             }
             else{
          //get the current user from the jwt token
            const token = req.cookies.jwttoken;
            const tokenVerify = jwt.verify(token, process.env.SECRET_KEY);
            const userExist = await Applicant.findOne({ _id: tokenVerify._id,
"tokens.token": token});
            //compare the current user email and entered email in the application
form
            if(userExist.email !== email){
                return res.status(400).json({error: "Email is not match"});
            }
            //create new pplication
                const application = new Application({
                    name, email, mobile, gender, age, experience, qualification,
skills, company_email, resume : result.secure_url, company_id:companydetails._id,
job_id
                });
                //save application in database
                const applicationRegister = await application.save();

                if(applicationRegister){
                    res.status(201).json({message : "Application added
successfully"});
                }
                else{
                    res.status(400).json({error: "Application not added
successfully"});
                }
             }
       })
    }
    catch(err){
        console.log(err);
    }
});
```

Display jobs

```
router.get("/getJobs",authenticate, async (req, res) => {
    try{
         //get current user from the jwt token
        const token = req.cookies.jwttoken;
        const tokenVerify = jwt.verify(token, process.env.SECRET_KEY);
        const company = await Company.findOne({ _id: tokenVerify._id,
"tokens.token": token});
```

```
      //check if current user is company then return only job of that company
       if(company){
           //company
           const currentrole = {'currentrole':company.role};
           const jobs = await Job.find({"email":company.email});
           jobs.push(currentrole);
           return res.status(200).json(jobs);
       }
       else{
         const Applicant = await Applicant.findOne({ _id: tokenVerify._id,
"tokens.token": token});
         const currentrole = {'currentrole':Applicant.role};
           //Applicant
           const jobs = await Job.find();
           jobs.push(currentrole);
           return res.status(200).json(jobs);
       }
   }
   catch(err){
     console.log(err);
   }
 });
```

Check if user logged in

```
const Authenticate = async (req, res, next) => {
    try{
       const token = req.cookies.jwttoken;
       const tokenVerify = jwt.verify(token, process.env.SECRET_KEY);
       //Check if currently Applicant is logged in
       const rootEmployee = await Employee.findOne({ _id: tokenVerify._id,
"tokens.token": token});

       if(!rootEmployee){
          //Check if currently company is logged in
           const rootCompany = await Company.findOne({ _id: tokenVerify._id,
"tokens.token": token});
           if(!rootCompany){
           //if no one is logged in then throw an error
              throw new Error('User not Found');
           }
  //if user present then add their detail in request and call next middleware
           else{
                req.token = token;
                req.rootUser = rootCompany;
                req.userID = rootCompany._id;
           }
```

```
        }
        else{
        req.token = token;
        req.rootUser = rootEmployee;
        req.userID = rootEmployee._id;
        }
        next();
    }
    catch (err){
        res.status(400).send('Unauthorized: No token provided');
        console.log(err);
    }
}
```

Testing

For testing our application, integration testing and manual testing is performed.

• **Integration testing :** Each small part of application is tested first and after that combine them and whole application testing is performed.

• **Manual testing**

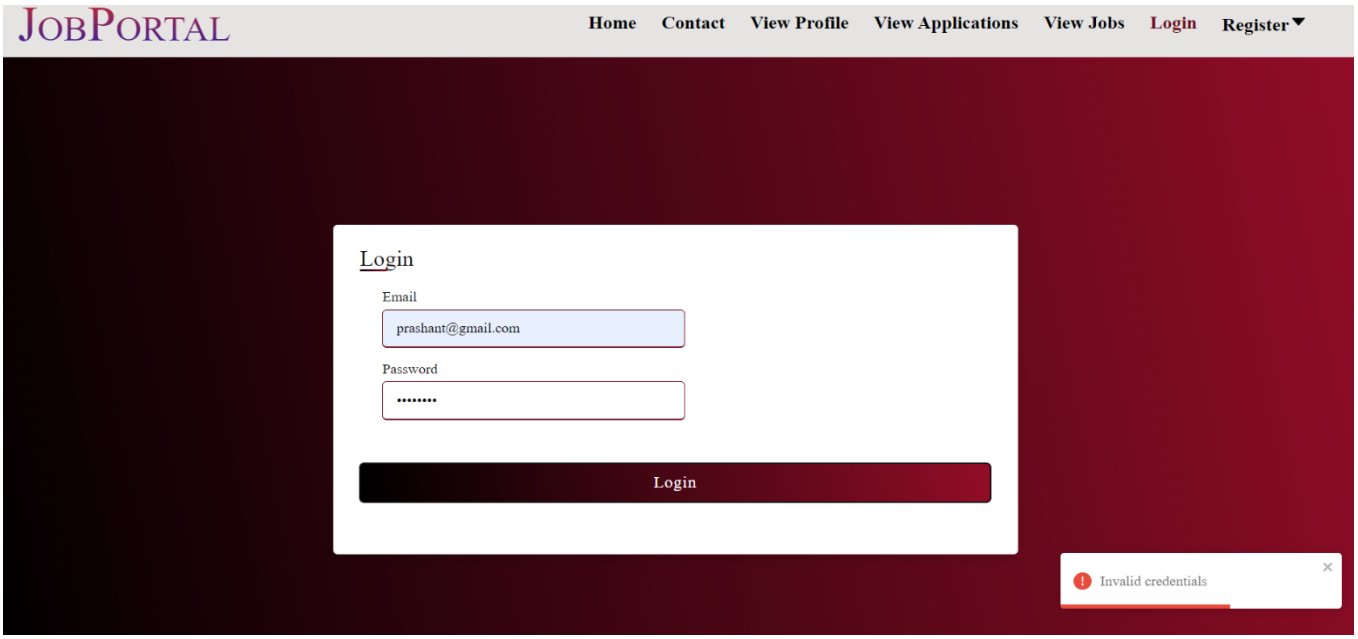
Sr No.	Test Scenario	Expected Result	Actual Result	Status
1	Login with incorrect credentials	User should not be able to login	Error message : “Invalid credentials!”	pass
2	Login with correct credentials	User should be able to login	Success message: “login successfully”	pass
3	Register with different password and confirm password	User should not be able to register	Warning message: “both password must same”	pass
4	Register with valid details	User should be able to register	Success message: “Registration successful”	pass
5	Without login view profile/companies/jobs	User not be able to view	Error message: “login required”	pass
6	Update profile	Profile should be updated	Success message : “updated successfully”	pass

7	Apply for job	Application shows to company	Success message: “applied successfully"	pass
8	Update application	Application should be update	Success message: “updated successfully”	pass
9	Add job	Show posted job in list	Success message: “job added successfully”	pass
10	Update job	Job should be update	Success message: “updated successfully”	pass
11	Change application status	Application accepted/rejected then disable accept/reject button	Accept/reject button is disabled afted accept/reject	pass
12	Send message with incorrect email	Message should noy be sended	Error message: “invalid email”	pass
13	Logout	User should be able to log out and restricted from the system until next login	User is successfully log out and unable to access system without login again	pass

Screen-Shots

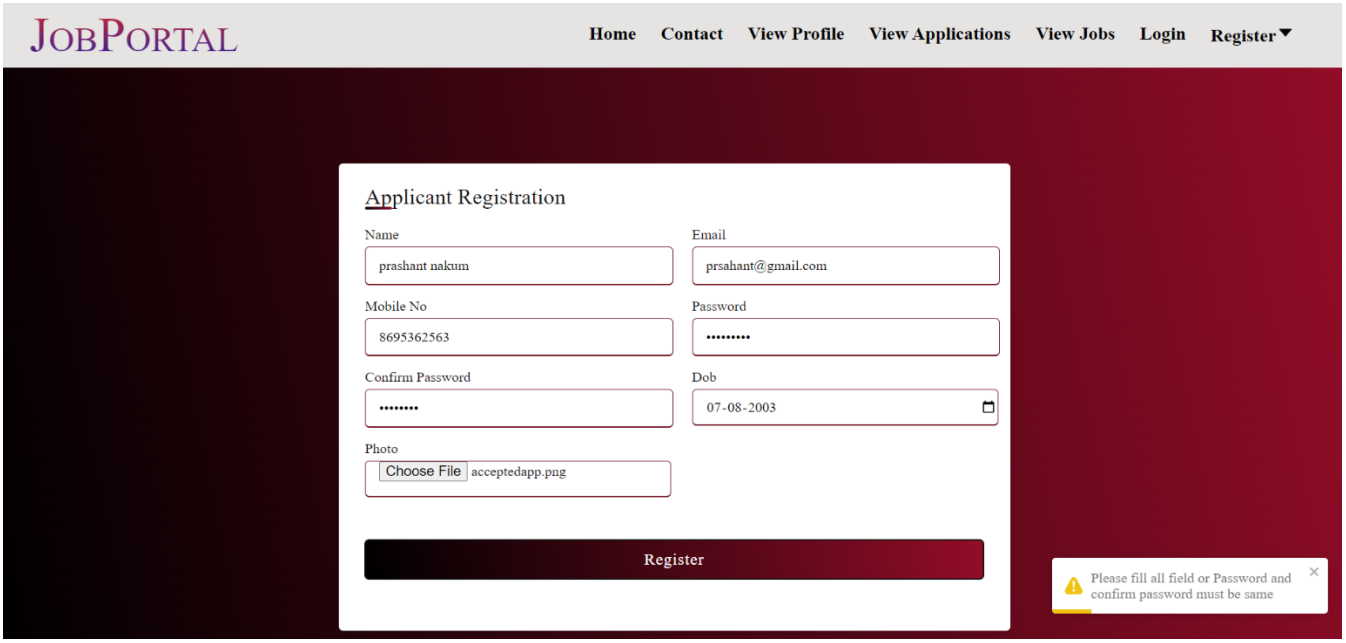
Login Page with invalid credentials

If we provide invalid credential then give error. If we provide valid email and password then successfully login.

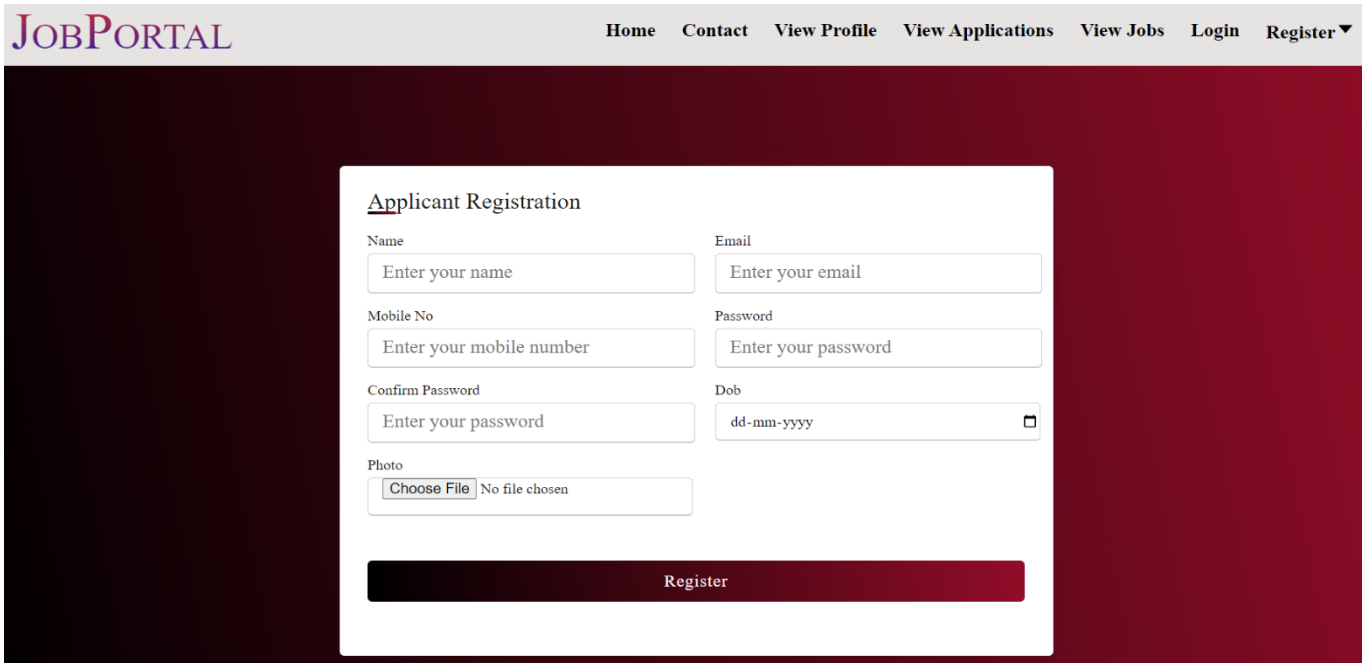


Company Register with different password and confirm password

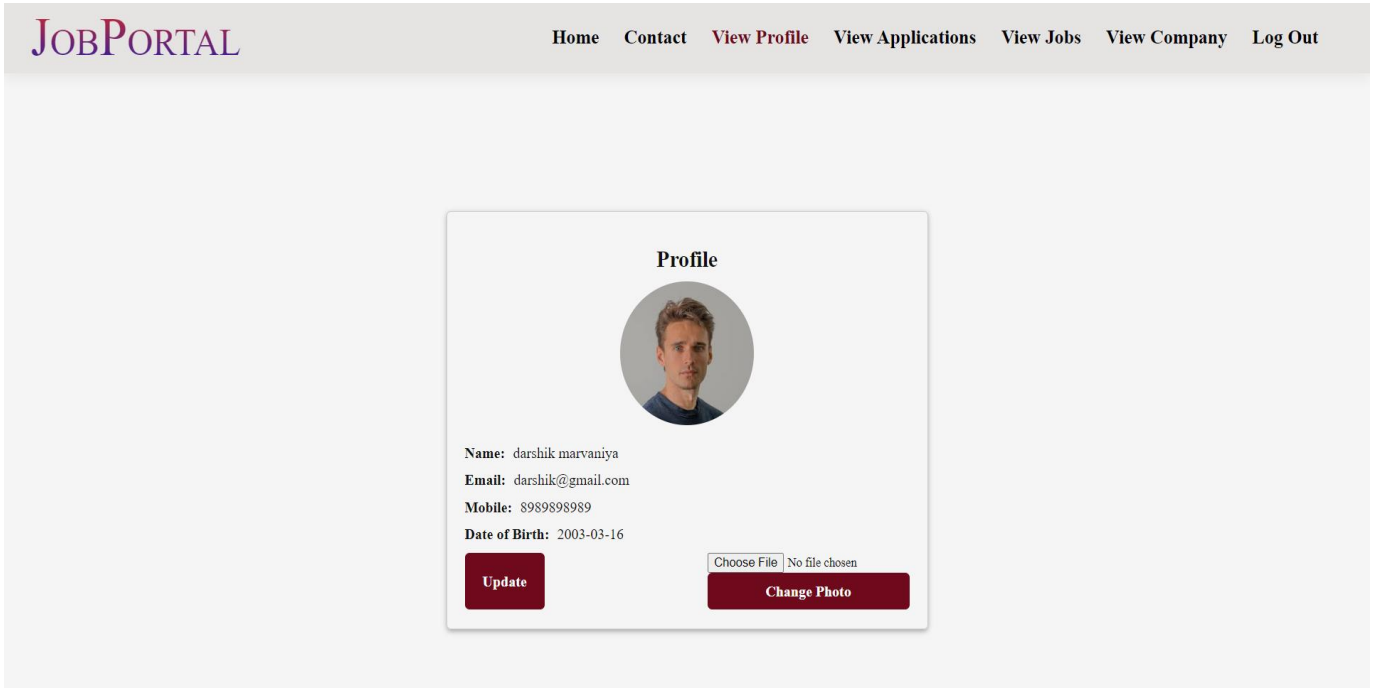
If Any field is empty or if password and confirm password is not match then give warning. If we fill valid data then successfully registered.



Applicant Register

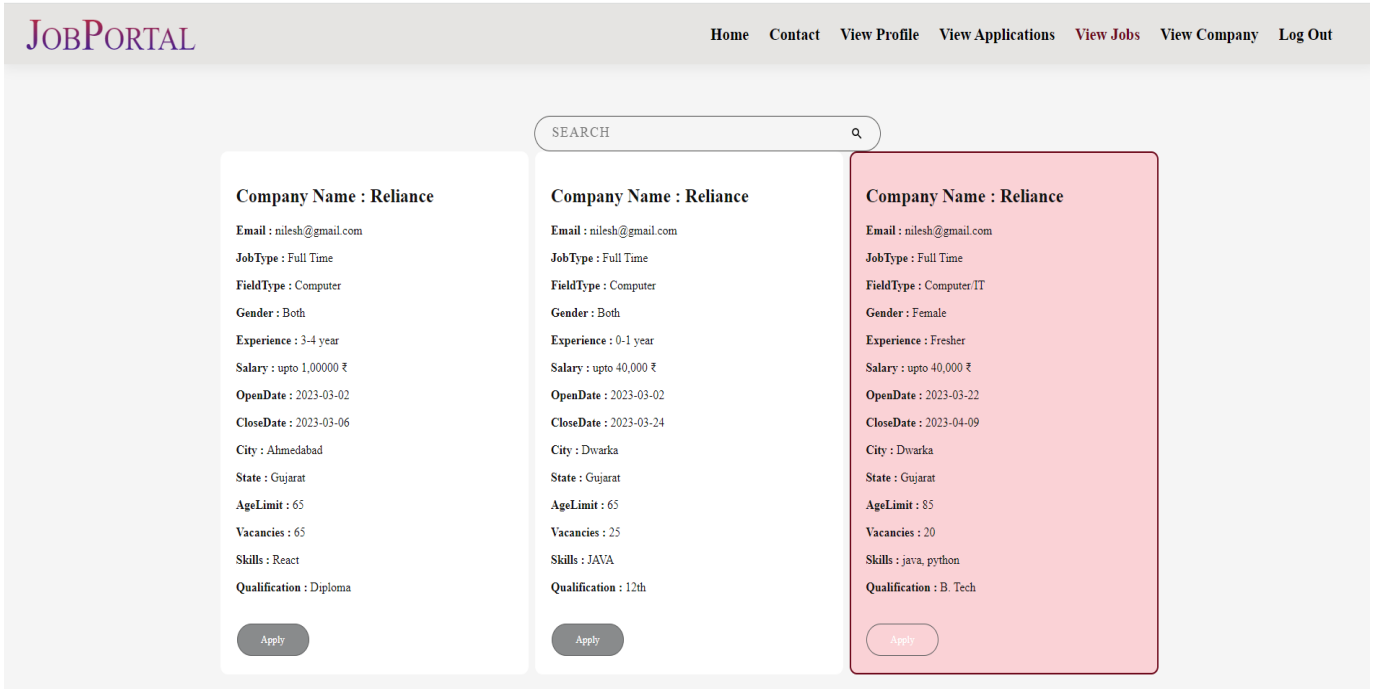


Applicant Profile Page

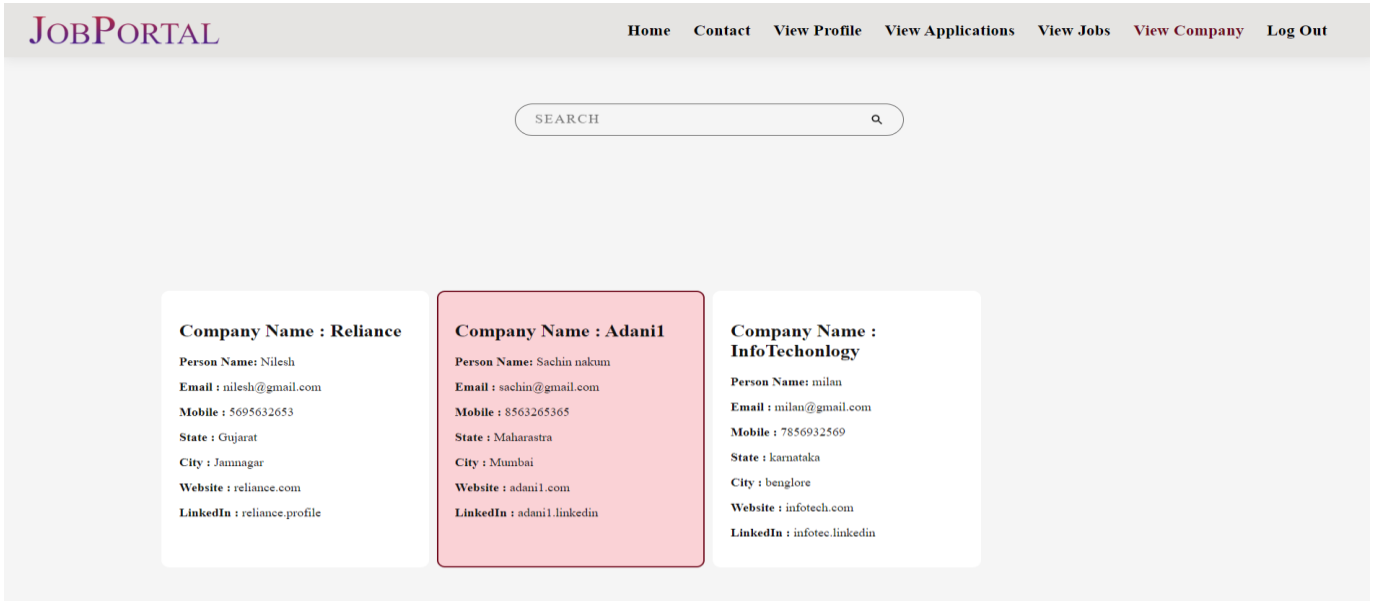


View Job Page

Display all the job posted by the Companies and user can apply for that job by click on apply button.

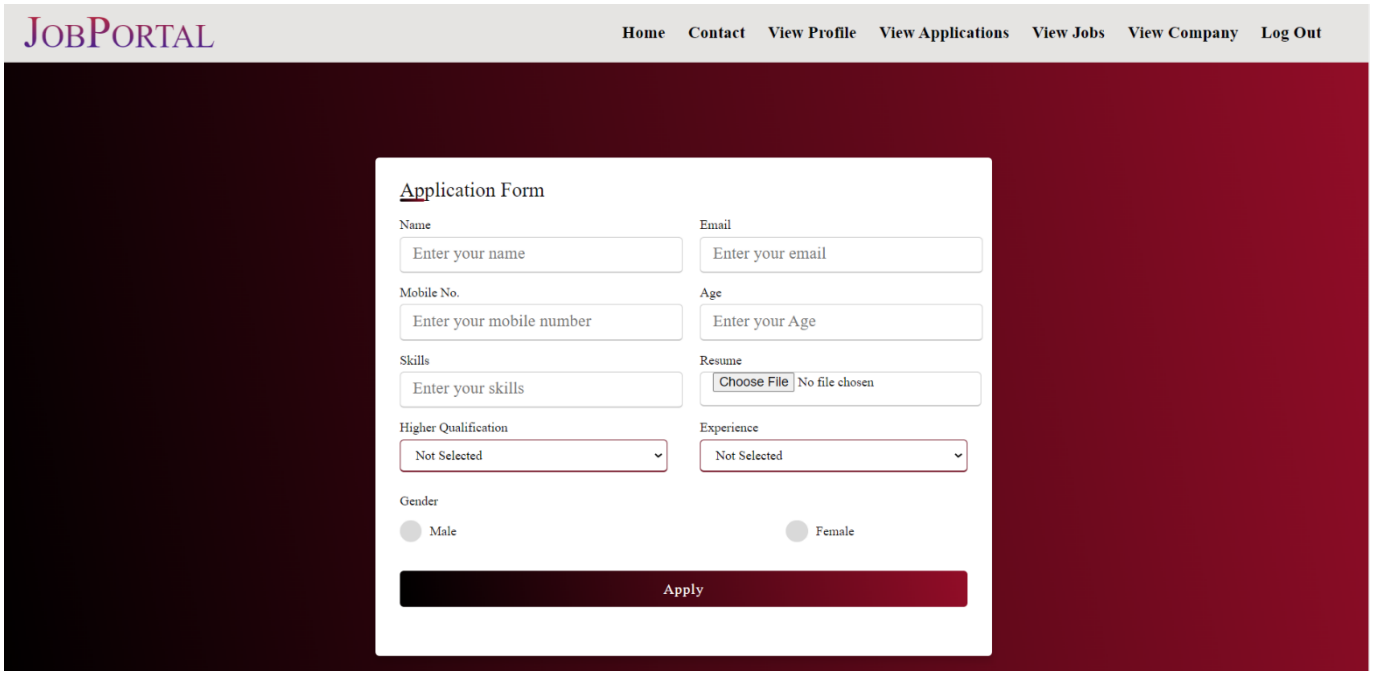


View company Page



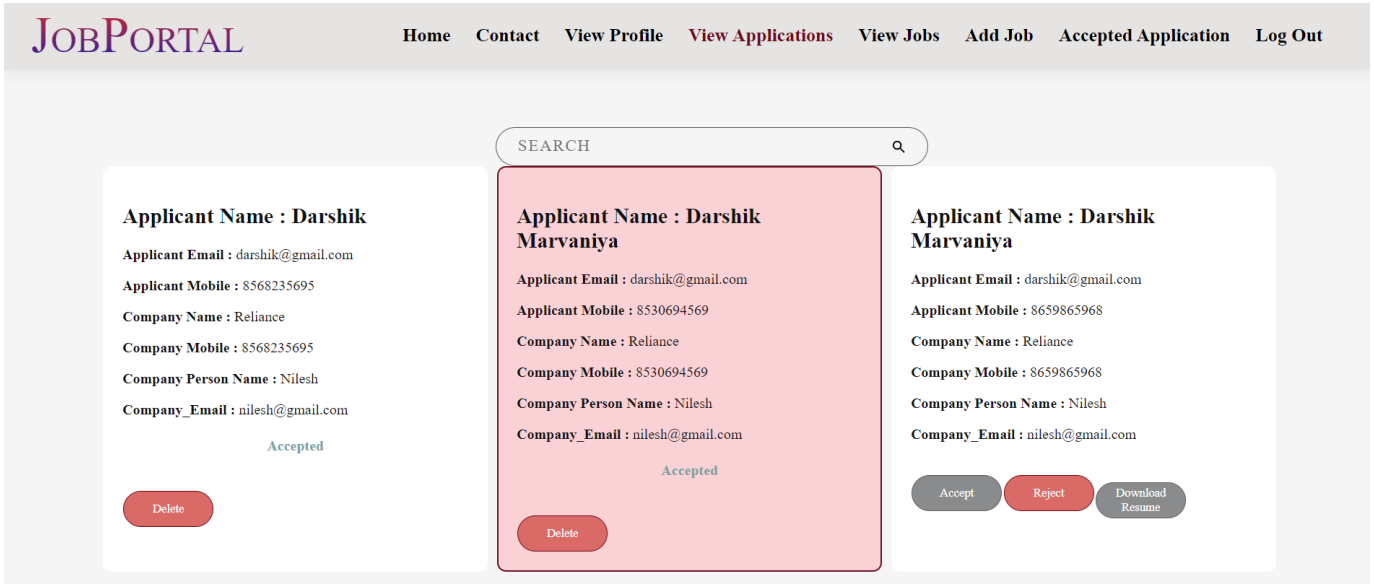
Apply for job Page

Here applicant have to fill all the details to apply for any job. If all the details valid then application will be successfully submitted.



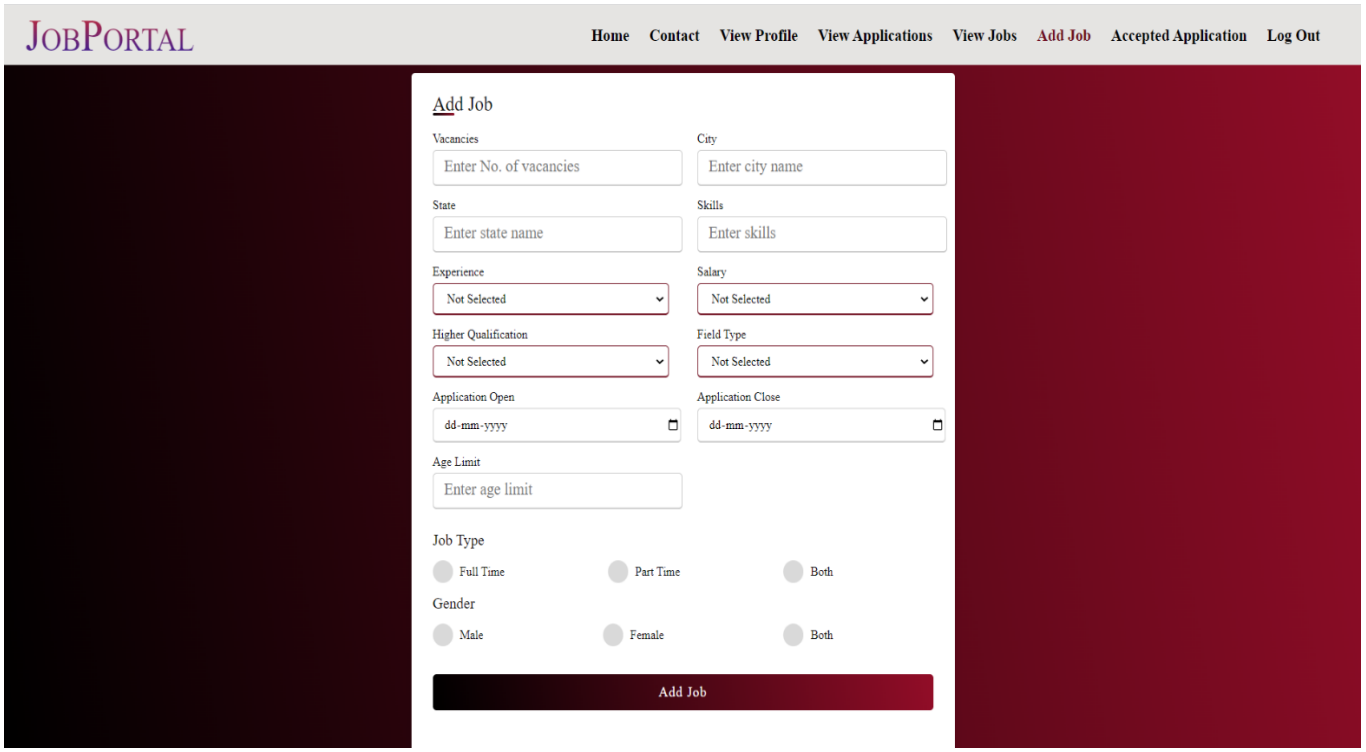
View Application Page

Here all the application of company is displayed and company can view the resume of applicant and accept/reject/delete the application.

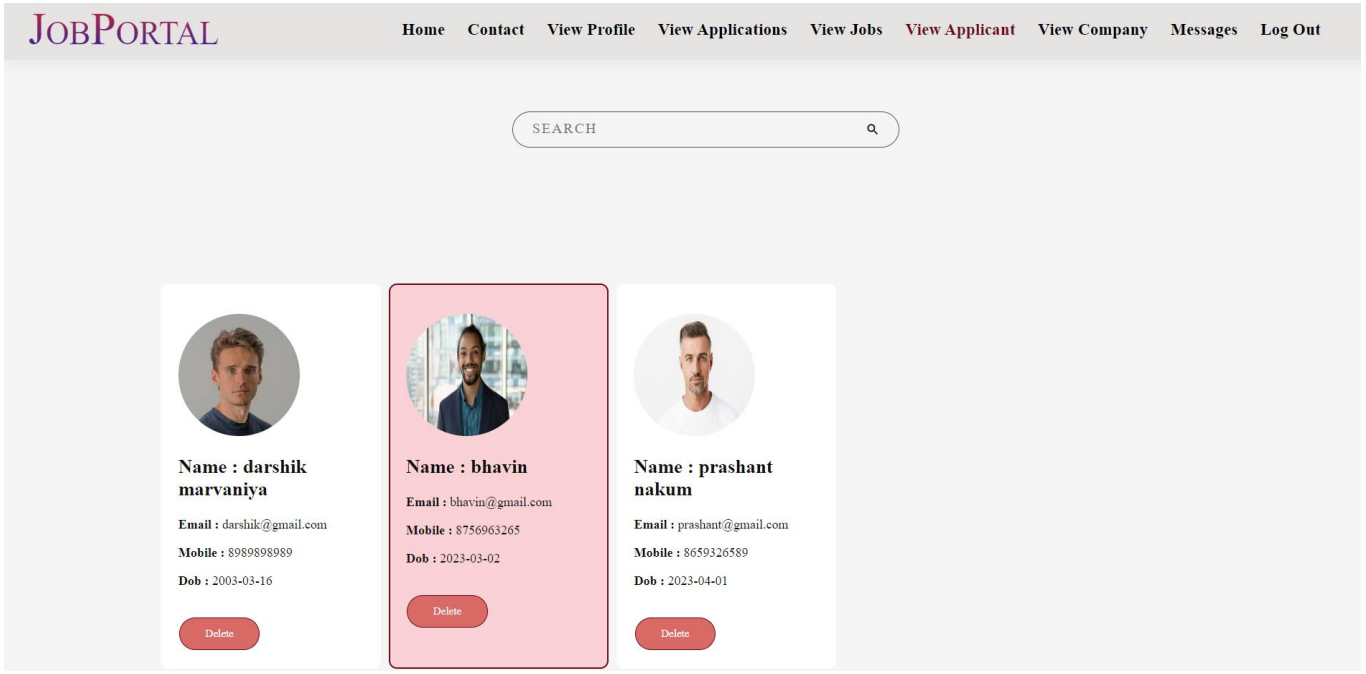


Add Job Page

Company have to fill this all the details correctly and it can post the job.

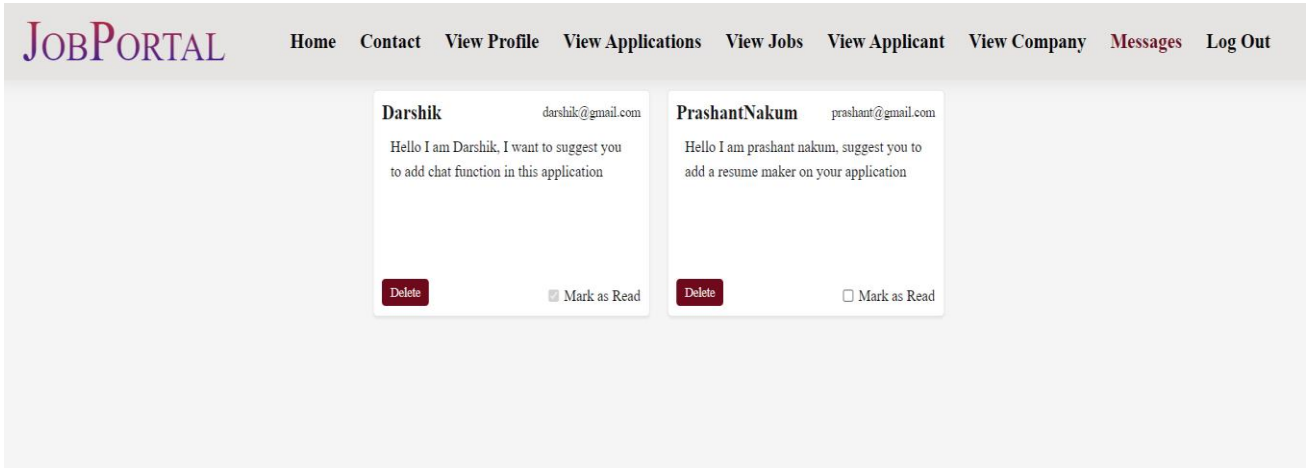


View Applicant Page



View User Feedback Message

Admin can view the messages sended by the companies and applicants.



Conclusion

The functionalities implemented in the system were done after understanding all the system modules according to the requirements.

Functionalities implemented in the system are :

- Authentication based on JWT token and Email verification
- Logout

Applicant functionalities:

- View/update profile
- Search job
- Apply for job
- Send msg to admin
- View application status

Company functionalities:

- View/update profile
- Add new job
- Update/delete job
- Accept/reject application
- Send msg to admin

Admin functionalities:

- View/delete company/applicant
- View feedback msg

After the implementation of the system manual testing was performed on the system to determine possible flow of the system.

Limitations and Future Extensions

Limitations of Project

- In this project there is no job alerting for user when new job posted by the company.
- We have not implemented chatting functionality in our system.
- Skill assessment is the one another limitation of this project.

Future Extensions

- Job alerting feature will be added for applicant for new job alert.
- System can be expanded to implement chatting functionality between applicant and company.
- Skill assessment feature will be added.

Bibliography

Websites:

- Mongoose : https://mongoosejs.com/docs/
- React js : https://reactjs.org/docs/getting-started.html
- Node js : https://nodejs.org/en/docs/
- Mongodb : https://www.mongodb.com/docs/
- For Error Solving : https://stackoverflow.com/

Youtube channels :

- Mern stack tutorial in hindi- you tube : https://youtu.be/fSmp7Cv-c_0