JOB FINDING PLATFORM

MINI PROJECT REPORT

Submitted by

ARVIND M

22CSR023

MATHAV Ra

22CSR117

AADHI PRANESH S S

23CSR001

GOWTHAM C D

23CSR068

in partial fulfillment of the requirements
for the award of the degree
of

BACHELOR OF ENGINERING

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

Estd : 1984

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDUTAI ERODE – 638 060

MAY 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

MAY - 2025

BONAFIDE CERTIFICATE

This is to certify that the Project report entitled "**JOB FINDING PLATFORM**" is the bonafide record of project work done by **ARAVIND M (22CSR023), MATHAV Ra (22CSR117), AADHI PRANESH S S (23CSR001), GOWTHAM C D (23CSR068)** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Science and Engineering at Anna University, Chennai during the year 2025-2026.

SUPERVISOR **HEAD OF THE DEPARTMENT**

Date: **(Signature with seal)**

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

NOVEMBER-2024

DECLARATION

We affirm that the Project Report titled **JOB FINDING PLATFORM** being submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering is the original work carried out by us. It has not formed part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Date:

ARVIND M

(22CSR023)

MATHAV Ra

(22CSR117)

AADHI PRANESH S S

(23CSR001)

GOWTHAM C D

(23CSR068)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor with seal

ABSTRACT

This project introduces a **Django-based Job Finding Platform** that serves as a bridge between job seekers and employers by offering a centralized, user-friendly environment for job discovery and recruitment. The platform supports two main types of users: **Job Seekers**, who are looking for suitable employment opportunities, and **Companies**, who are seeking to hire qualified candidates. It allows companies to register, log in, and manage job postings with relevant information like job title, description, location, and eligibility criteria.

Job Seekers can create an account, browse available job listings, and use filters such as job role, company name, and location to find relevant opportunities quickly. The portal is built with secure authentication mechanisms and a responsive interface to ensure a smooth user experience across devices. Django's built-in admin and ORM features are leveraged for efficient data handling and back-end operations.

The platform lays a strong foundation for a scalable and feature-rich job ecosystem. It can be further enhanced with functionalities like resume uploads, application history tracking, and notification systems to keep users updated about new postings or application statuses. This project demonstrates how full-stack web development using Django can be used to solve real-world employment challenges by connecting talent with opportunity.

ACKNOWLEDGEMENT

We express our sincere thanks and gratitude to **Thiru. A. K. ILANGO B.Com., M.B.A., LLB.,** our beloved Correspondent, and all other philanthropic trust members of Kongu Vellalar Institute of Technology Trust who have always encouraged us in academic and co-curricular activities.

We are extremely thankful with no words of a formal nature to the dynamic Principal **Dr. V. BALUSAMY, M.Tech., Ph.D**., for providing the necessary facilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department **Dr. S. MALLIGA, M.E., Ph.D.**, for providing the necessary facilities.

We profoundly thank our Project Coordinator, **Ms.KANNUKINNIYAL, M.E.,** Assistant Professor**, and Ms.GOWTHAMI, M.E.,** Assistant Professor Computer Science and Engineering Department for coordinating our project work with all the necessary facilities to do our project successfully.

We wish to express our sincere thanks to all the Teaching and Non-Teaching staff members of the Computer Science and Engineering Department and Student Friends for their help and cooperation.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In today’s fast-moving world, finding a job or hiring the right talent shouldn't be a frustrating experience—but for many, it still is. Job seekers often have to jump between platforms, filter through irrelevant postings, or struggle with complicated interfaces. On the other side, companies—especially startups or small businesses—don’t always have the resources to post on expensive, feature-heavy job portals. This gap inspired the development of our **Job Finding Platform**, a straightforward and user-friendly solution built with Django.

The idea behind this project is simple: make it easy for job seekers and employers to find each other. Whether you're a graduate looking for your first role or a company trying to build a team, the platform provides a clean and accessible space to connect. It focuses on the core features users actually need—like job browsing, posting, filtering, and secure logins—without overwhelming them with unnecessary complexity.

By using Django as the backend framework, we’re able to handle user authentication, database interactions, and dynamic content efficiently. The result is a responsive, secure, and easy-to-navigate platform that focuses on functionality and user experience. This project not only solves a real-world problem but also showcases how web development can be used to build impactful, meaningful tools.

1.1 EXISTING SYSTEM

Today, most hiring happens through popular job portals like LinkedIn, Indeed, and Naukri. While these platforms offer wide reach, they can be overwhelming due to their complex interfaces, paid features, and the sheer volume of job listings. Small businesses often struggle to stand out, and job seekers may waste time filtering through irrelevant or outdated postings.

On the other hand, many companies still use traditional hiring methods like email

or spreadsheets to track applicants, which is inefficient and prone to errors. There's a clear need for a simple, streamlined platform that removes unnecessary steps and focuses on the essentials helping job seekers find relevant opportunities and allowing companies to post and manage openings with easy.

1.2 OBJECTIVE

The primary objective of this project is to develop a **Job Finding Platform** that streamlines the job search and recruitment process for both job seekers and employers. For job seekers, the platform offers an intuitive and efficient way to explore available job opportunities, apply filters to refine searches, and easily submit applications. The platform aims to simplify the process, making it less time-consuming and more focused on connecting candidates with relevant positions that match their skills and interests. For companies, the goal is to provide an easy-to-use interface where they can create, update, and manage job postings, reaching a broad pool of talent without the complexity of traditional recruitment tools.

In the long run, the platform aims to improve the overall hiring experience by offering features like user authentication, role-based access, and a dynamic search engine. By leveraging Django's capabilities, the system ensures that data management is efficient and secure. The ultimate goal is to make job hunting less overwhelming and to provide companies with a clear, organized way to attract, manage, and evaluate applicants. Future enhancements will focus on adding capabilities such as resume uploads, job application tracking, and real-time notifications, further improving the user experience and providing even more value to both job seekers and recruiters.

1.3 SCOPE

This Job Finding Platform focuses on building a simple, yet powerful system that covers the core needs of both job seekers and companies. Job seekers will be able to register, log in, view job listings, and filter them based on role, company, or location. Companies can sign up, post jobs, and manage their listings with ease. The platform ensures each user type sees only the features relevant to them, keeping the experience clean and focused.

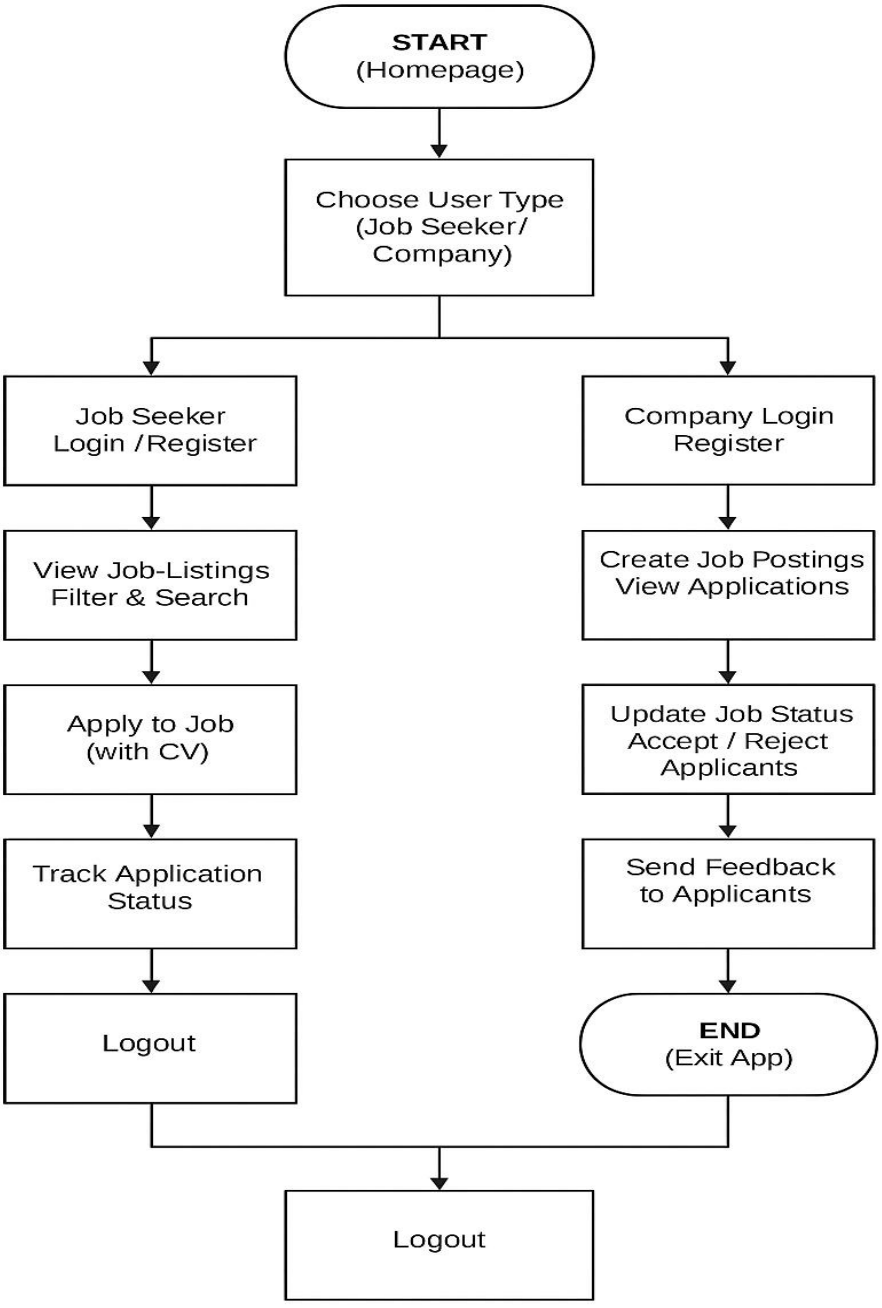
The initial version of the platform includes secure login functionality, role-based dashboards, job posting and editing, and a responsive user interface built with Django. As the platform evolves, there's room to expand with features like resume uploads, application tracking, real-time notifications, saved jobs, and even admin-level controls for monitoring platform usage. The goal is to create a foundation that's simple to start with, but flexible enough to grow into a complete job ecosystem over time.

CHAPTER 2

PROPOSED WORK

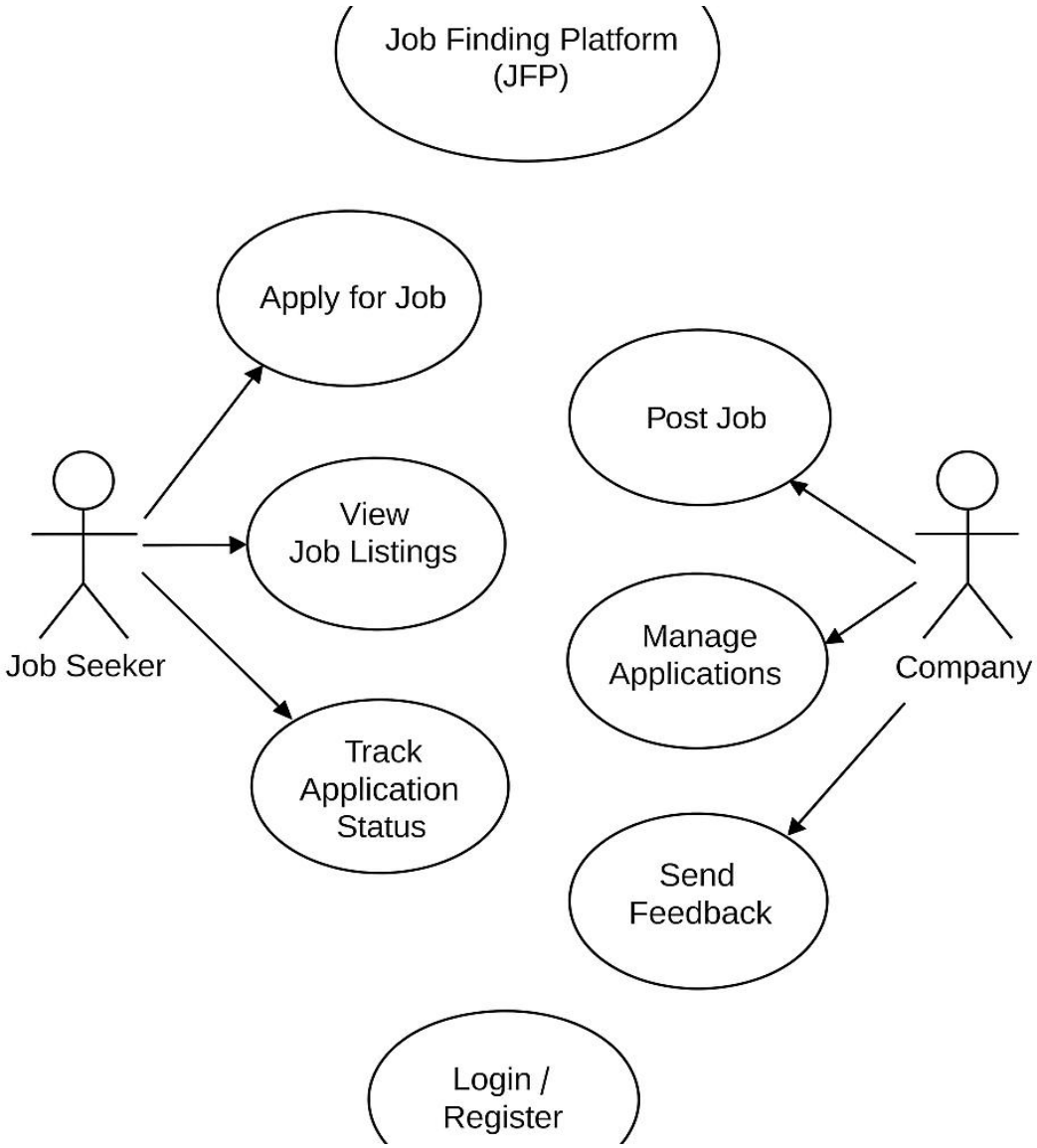
2.1 FLOW CHART

Illustrates the logical flow and interactions within the User Module, showing steps from user authentication shown in FIGURE 2.1 FLOWCHART.



2.2 USECASE DIAGRAM

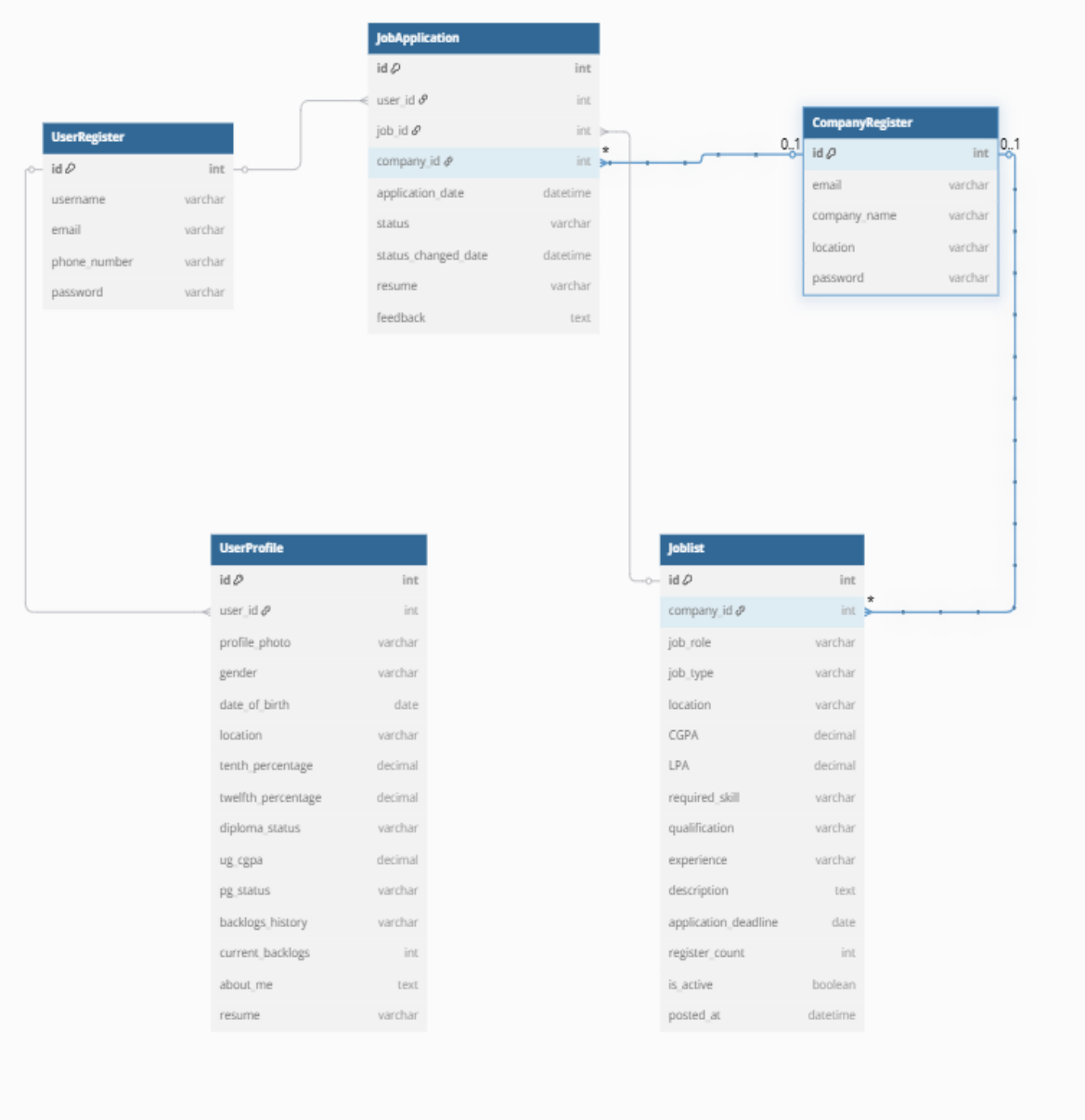
Illustrates the interactions within the User Module, showcasing user actions shown in FIGURE 2.2 USECASE DIAGRAM.



2.3 DATABASE DESIGN

The database design for the Job Finding Platform is structured to support a seamless interaction between job seekers and companies. It consists of well-defined models that reflect the core functionalities of the platform. The **UserRegister** and **CompanyRegister** models store essential authentication and identity details for job seekers and companies

respectively. Each company can post multiple job listings through the **Joblist** model, which includes detailed information such as role, type, skills required, qualifications, salary, and deadlines. Job seekers can apply to these listings via the **JobApplication** model, which tracks the status, resume, feedback, and timestamps of each application. Additionally, the **UserProfile** model complements the user registration with in-depth academic, personal, and professional background, allowing for rich user profiles that enhance employer evaluations. The relationships between these models are designed using foreign keys and constraints to maintain data integrity and prevent duplication, ensuring a robust, scalable, and efficient database structure that meets the dynamic needs of a job recruitment ecosystem shown in FIGURE 2.3



2.4 INTERFACE DESIGN

The Job Finding Platform offers a user-friendly interface designed to make job search and recruitment effortless for both job seekers and employers. The platform's Home page provides an overview of available job opportunities, trending companies, and featured roles, making it easy for job seekers to get started. It also offers a section for employers to view and manage their job posts, track applications, and search for potential candidates.

Users can search and filter job listings based on various criteria such as job title, location, industry, and experience level, helping them find the most relevant opportunities quickly. For job seekers, there's a profile section where they can add their personal details, work experience, and skills, ensuring they present a complete profile to potential employers.

The platform also provides a detailed job application process, allowing users to easily apply for positions and track the status of their applications. Employers can post new jobs, manage applicants, and gain insights into their job postings' performance.

LANDING PAGE

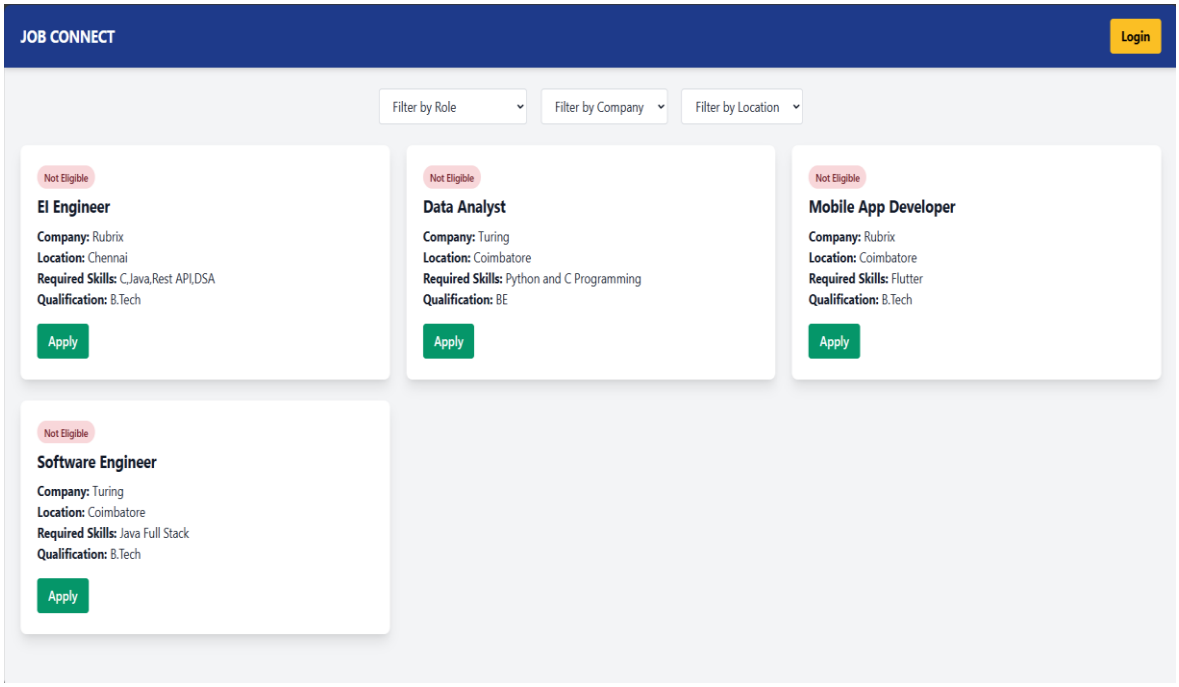


FIGURE 2.4.1

LOGIN PAGE

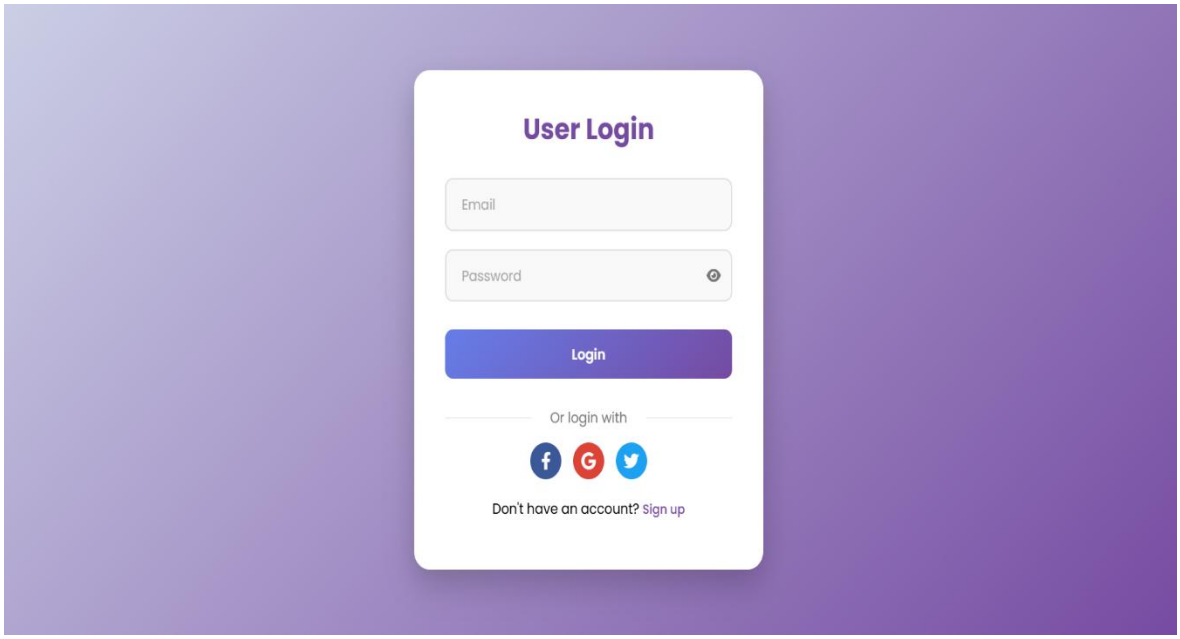


FIGURE 2.4.2

USER DASHBOARD

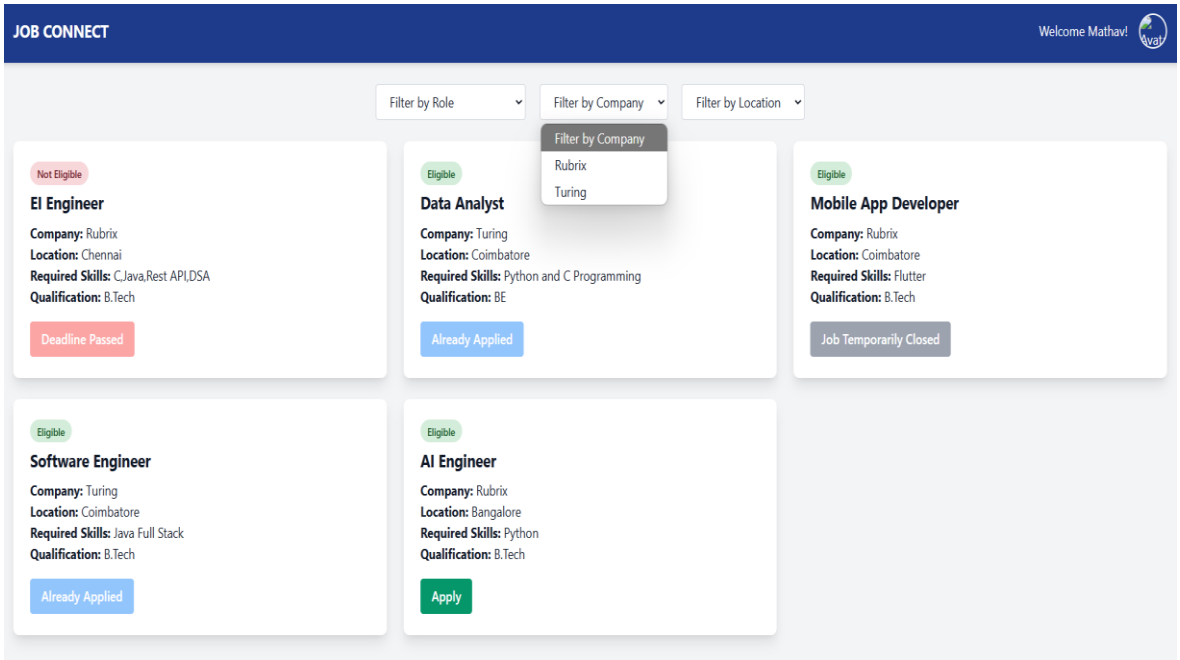


FIGURE 2.4.3

COMPANY DASHBOARD

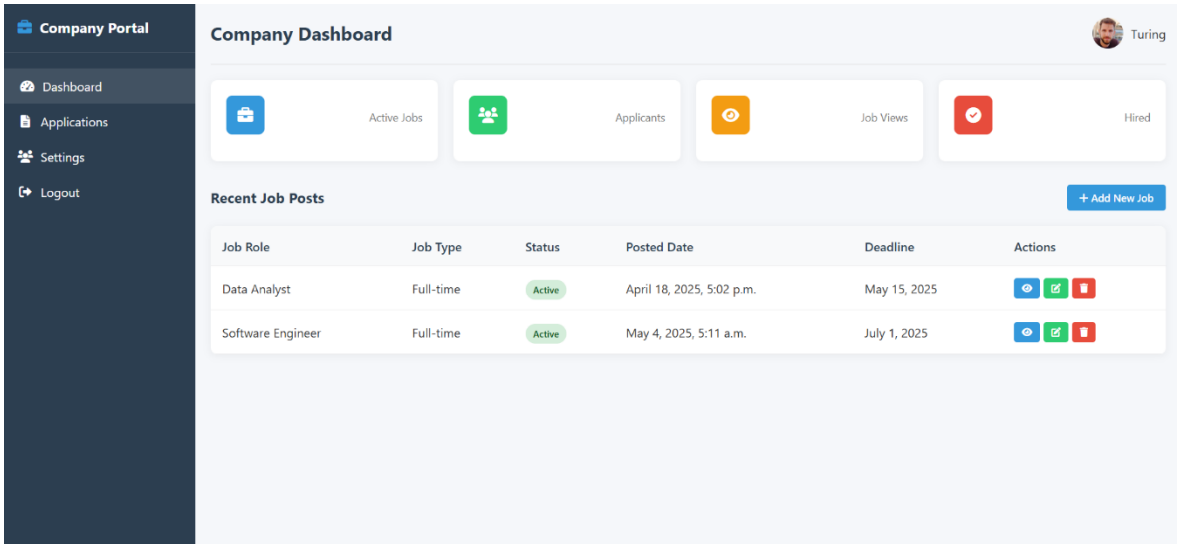


FIGURE 2.4.4

CHAPTER 3

RESULT AND DISCUSSION

RESULT

The implementation of the Job Finding Platform (JFP) successfully fulfilled the core goals of the project. Users were able to register, create profiles, and apply to job listings uploaded by registered companies. Companies, on the other hand, could post jobs, track applications, and manage candidate feedback through a secure and structured system. The platform maintained data consistency and prevented redundant applications by enforcing relational constraints through Django's ORM.

The resume upload and job search functionalities worked seamlessly, with filters like job role, location, CGPA, and experience improving the overall precision of job discovery. Additionally, the company-side dashboard offered clear visibility into candidate applications and statuses, reducing communication gaps.

DISCUSSION

During development and testing, the system proved to be stable and user-friendly. One key strength was the modular design of the database, which allowed for scalable additions like user profiles, academic records, and job application statuses. However, some challenges were encountered in ensuring proper file validation for resume uploads and in handling edge cases such as duplicate user registrations or expired job posts.

Performance-wise, the platform handled concurrent user operations effectively in a controlled environment. User feedback indicated that the interface was easy to navigate, and the real-time updates for application statuses added a professional touch. Overall, the platform serves as a reliable foundation for a full-scale job portal with potential for further expansion and automation.

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

CONCLUSION

The *Job Finding Platform (JFP)* was developed to provide a simplified, efficient, and accessible digital solution for job seekers and recruiters. It brings together users and companies in a unified ecosystem, allowing for job listings, applications, resume submissions, and application status tracking. The use of Django and a structured database ensures a robust backend, while the frontend provides users with an intuitive experience. The platform meets its primary objective—connecting the right candidates with the right opportunities—through a seamless, user-friendly workflow. It also ensures security, prevents duplicate applications, and simplifies administrative processes for companies.

FUTURE SCOPE

To enhance the platform's functionality and user experience, several future upgrades can be considered:

- **AI-Powered Job Recommendations:** Suggest jobs based on skills, education, and user behavior.
- **Real-Time Notifications**: SMS and email alerts for application updates and new job postings.
- **Company-Candidate Chat System**: Enable direct communication for interview scheduling or queries.
- **Dashboard Analytics**: Provide insights like number of views, application performance, and trending roles.
- **Mobile App Integration**: Extend platform usability with a cross-platform mobile app (Flutter/React Native).
- **Resume Builder Tool**: Allow users to create or format resumes directly on the platform.
- **Multi-Language Support**: Expand accessibility to users in diverse regions

APPENDIX 1

CODING

```
from django.contrib import admin
from django.urls import path , include
from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('accounts.urls')),  # Your 'accounts' app URLs
    path('company/', include('companyside.urls')),
    path('user/',include('userSide.urls')),
    # path()
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)


from django.shortcuts import render , redirect
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .models import *
from companyside.models import *
from django.contrib import messages
from django.contrib.auth import authenticate, login
from django.contrib.auth.hashers import check_password
```

```
from django.contrib.auth.hashers import make_password


# Create your views here.
def Home(request):
    if request.method == 'GET':
        joblist = Joblist.objects.all()
        print(joblist)
        return render(request , 'home.html' ,{'job_list':joblist})
    return render(request, 'home.html')


def LandPage(request):
    return render(request,'landingpage.html')


@api_view(["GET","POST"])
def comanyRegister(request):
    # print("iam mathav")
    if request.method == 'POST':
        email = request.POST.get('Email')
        company_name = request.POST.get('Company_name')
        location = request.POST.get('Company_location')
        password = request.POST.get('Password')
        hashed_password = make_password(password)
        #create and save the employee object
        CompanyRegister.objects.create(
                email = email,
                company_name = company_name,
```

```
            location = location,
            password = hashed_password,

        )
        print(f"New Company {company_name} add successfully")
        return render(request,'com_auth.html')
    return render(request,'com_auth.html')



@api_view(["GET", "POST"])
def companyLogin(request):
    if request.method == 'POST':
        email = request.POST.get('Email')
        password = request.POST.get('password')
        try:
            user = CompanyRegister.objects.get(email=email)
            # Check if the password matches the stored hashed password
            if check_password(password, user.password):
                request.session['company_id'] = user.id  # Store company ID in session
                # company_name = user.company_name
               # context = {'user_logged_in': True, 'user': user, 'company_name': company_name}
                return redirect('company_dashboard')
            else:
                return render(request, 'com_auth.html', {'error': 'Invalid credentials'})
        except CompanyRegister.DoesNotExist:
            return render(request, 'com_auth.html', {'error': 'Invalid email or password'})
```

```
    return render(request, 'com_auth.html')


@api_view(['GET','POST'])
def userRegister(request):
    print("user register")
    if request.method == 'POST':
        username = request.POST.get('user_name')
        email = request.POST.get('Email')
        ph_number = request.POST.get('ph_number')
        password = request.POST.get('Password')
        hashed_password = make_password(password)
        UserRegister.objects.create(
            username = username,
            email = email,
            phone_number = ph_number,
            password = hashed_password,
        )
        print(f"New User {username} add successfully")
        return render(request,'user_auth.html')
    return render(request,'user_auth.html')


@api_view(['GET','POST'])
def userLogin(request):
    print("user login")
    if request.method == 'POST':
```

```
    email = request.POST.get('Email')
    password = request.POST.get('password')
    try:
        user = UserRegister.objects.get(email=email)
        if check_password(password, user.password):
            request.session['user_id'] = user.id
            return redirect('user_dashboard')
        else:
            return render(request, 'user_auth.html', {'error': 'Invalid credentials'})
    except UserRegister.DoesNotExist:
        return render(request, 'user_auth.html', {'error': 'Invalid email or password'})
return render(request, 'user_auth.html')
```

APPENDIX 2

SCREENSHOTS

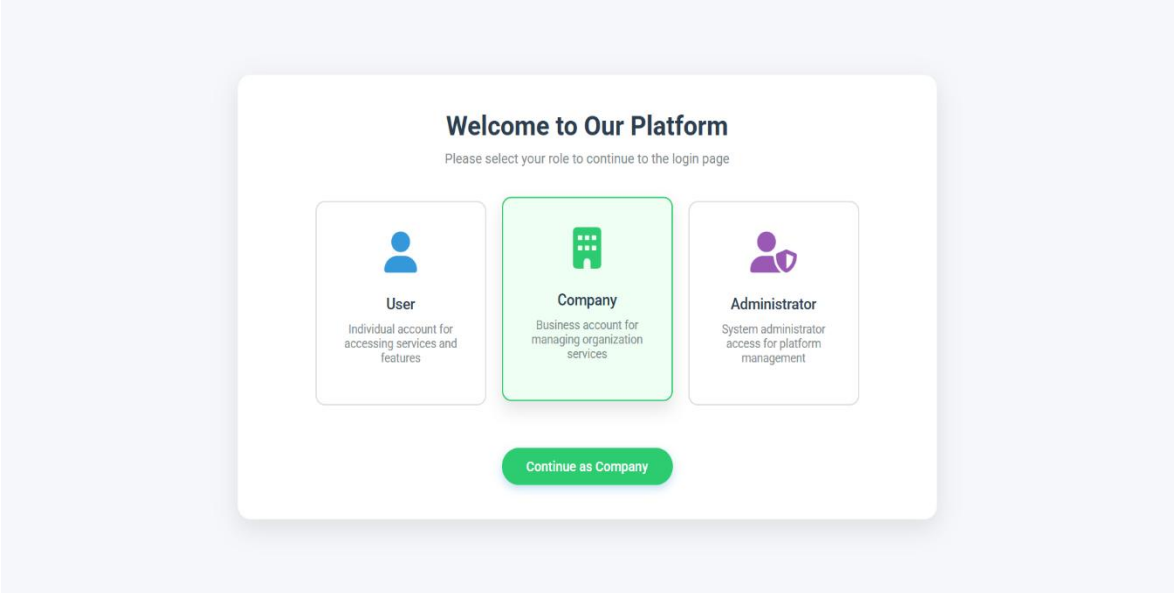


FIGURE A.2.1 USERS-CHOICE

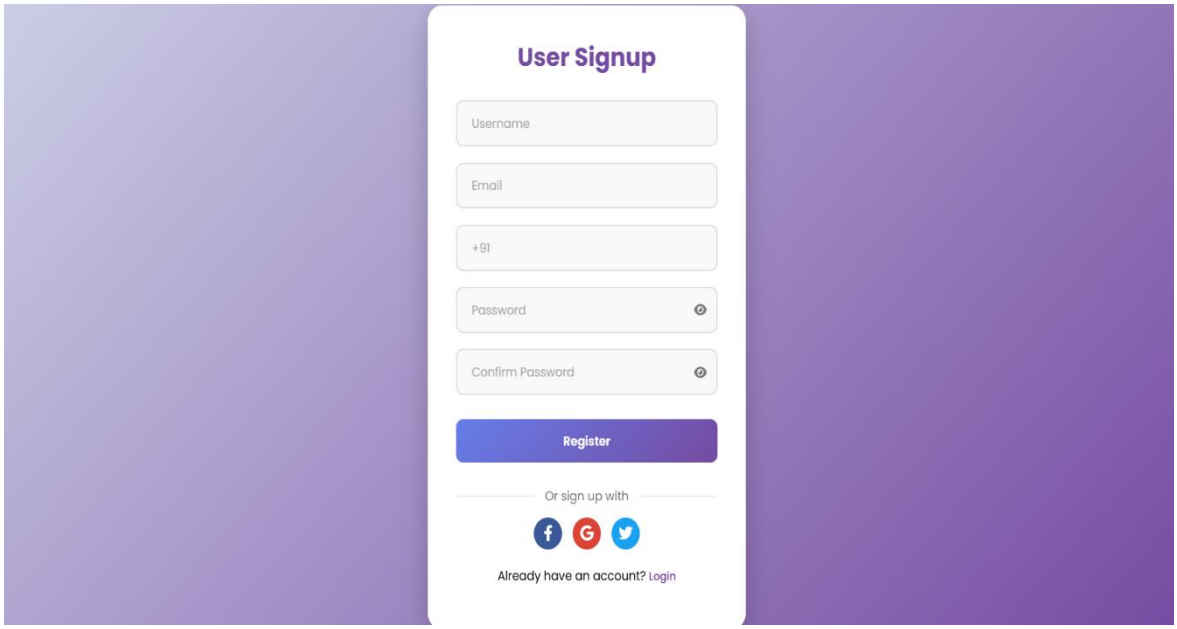


FIGURE A.2.2 SIGNUP PAGE

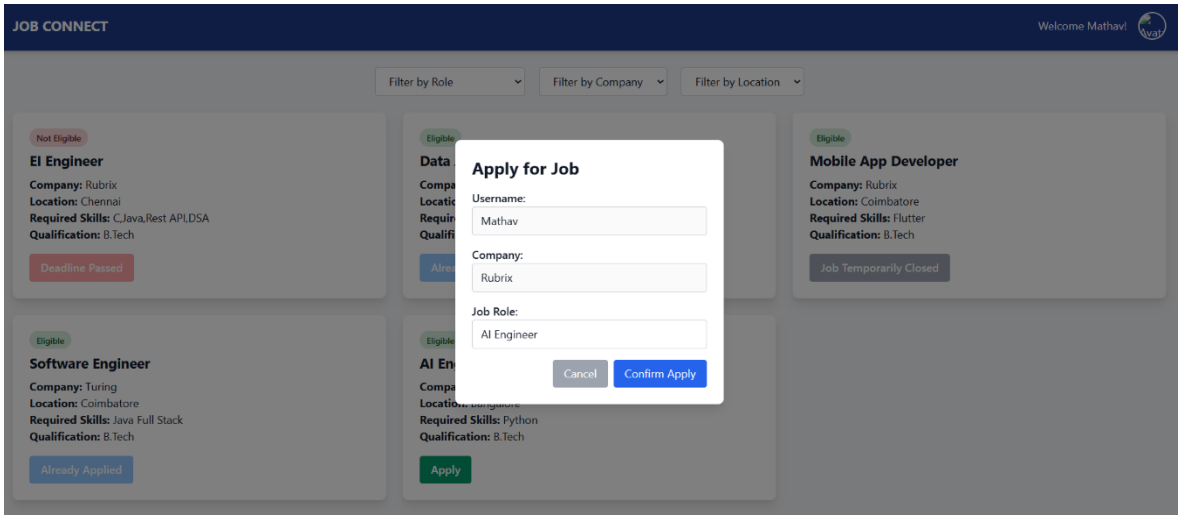


FIGURE A.2.3 USER JOB APPLYING

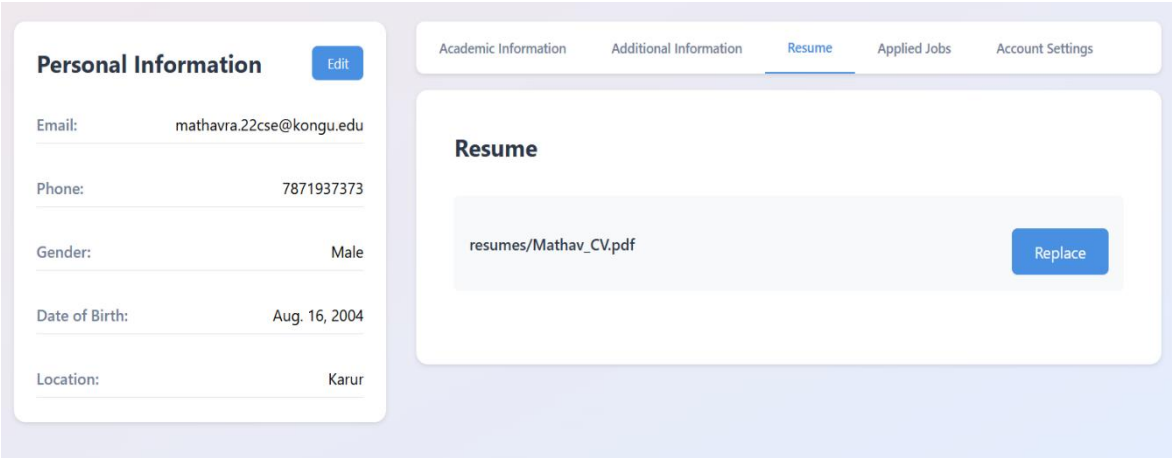


FIGURE A.2.4 USER RESUME PAGE

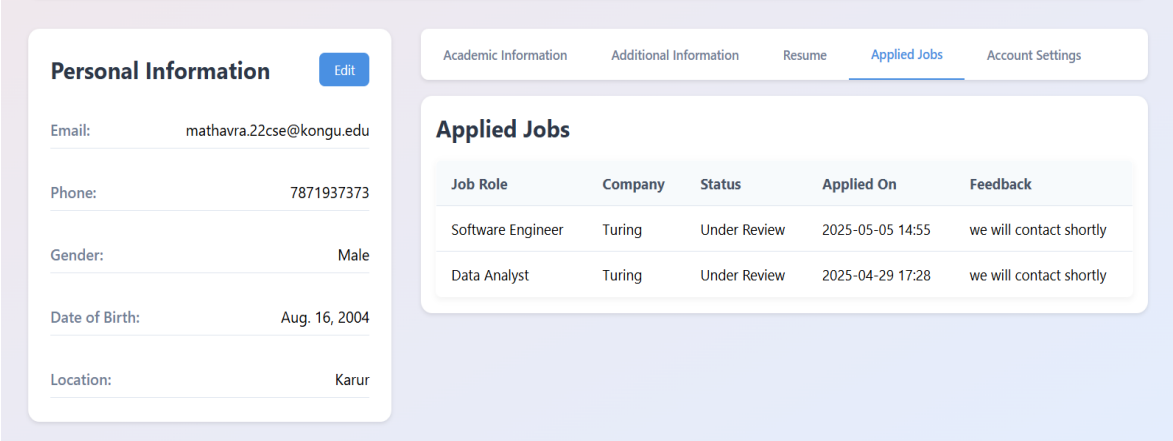


FIGURE A.2.5 USER APPLIED JOBS

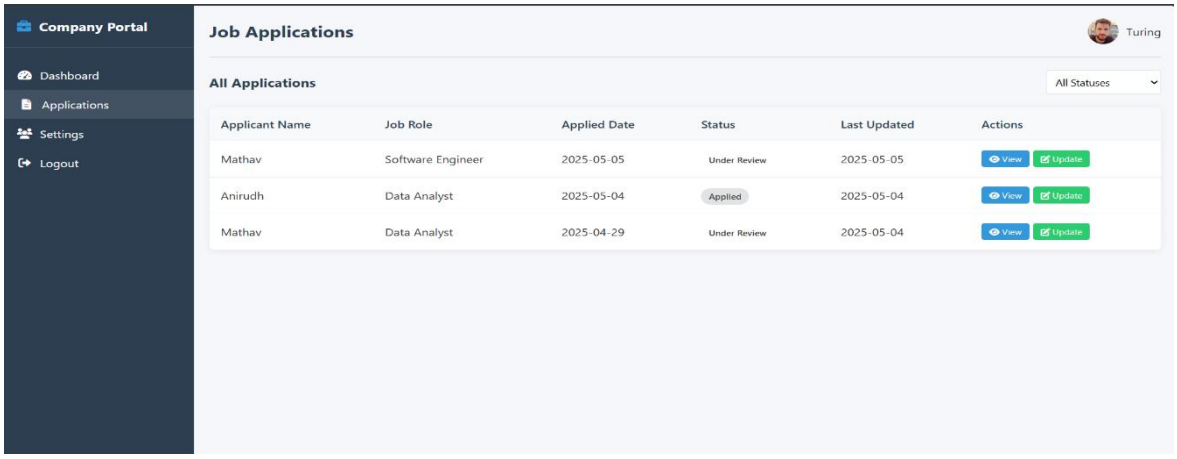


FIGURE A.2.6 JOB APPLICATION FOR COMPANY

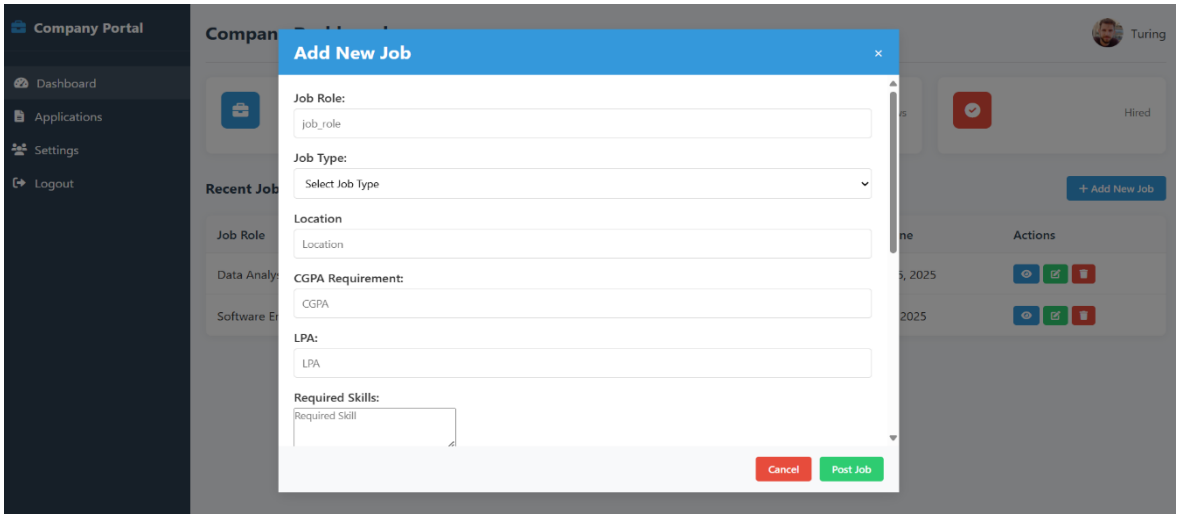


FIGURE A.2.7 COMPANY ADD NEW

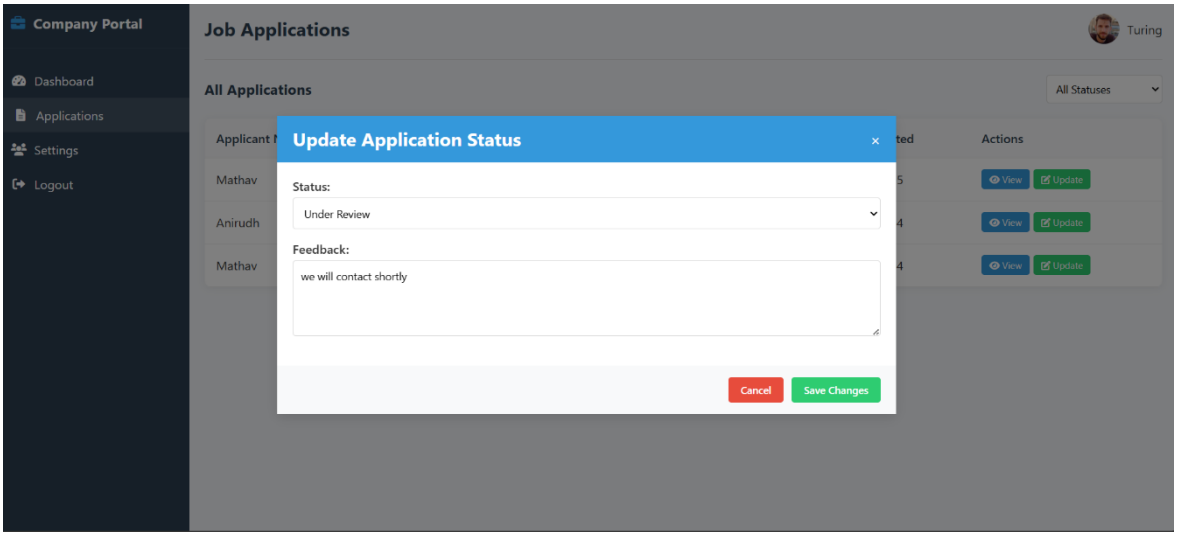


FIGURE A.2.8 JOB APPLICATION UPDATE

REFERENCE

1. Holzner, S. (2016). *Beginning Django: Web Application Development and Deployment with Python*. Apress.

2. Vincent, R. (2019). *Django for Professionals: Production Websites with Python & Django*. WelcomeToCode.

3. Sweigart, A. (2020). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*. No Starch Press.

4. Scheinholtz, C., & Adams, J. (2018). *Mastering Django: Core*. Packt Publishing.

5. Ranjan, A., & Tiwari, R. (2020). "A Study on Efficient Job Matching Algorithms in Recruitment Portals." *International Journal of Computer Applications*, 176(14), 22–26.

6. Gupta, M., & Verma, S. (2021). "Web-based Application for Recruitment Using Django Framework." *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(2), 35–42.

7. W3Schools. (n.d.). *HTML, CSS, JavaScript, and Django Tutorials*. Retrieved from: https://www.w3schools.com/

8. Mozilla Developer Network (MDN). (n.d.). *Web Development Documentation and Guides*. Retrieved from: https://developer.mozilla.org/

9. Stack Overflow. (n.d.). *Programming Questions and Community Solutions*. Retrieved from: https://stackoverflow.com/

10. Django Software Foundation. (n.d.). *Django Official Documentation*. Retrieved from: https://docs.djangoproject.com/