**Online Job Portal Project Report**

**Project Report**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Topic** | **Page no** |
| 1 | Title of Research | 5 |
| 2 | Certificate | 2 |
| 3 | Abstract | 5 |
| 4 | Introduction | 6-9 |
| 5 | Objective | 9- |
| 6 | Programming Language | 9-18 |
| 7 | Data gathering | 19 |
| 8 | System design /UML Diagrams | 20-22 |
| 9 | Proposed system features | 23-25 |
| 10 | Evaluation | 26-27 |
| 11 | Future scope | 28 |
| 12 | Conclusion | 29 |
| 13 | Biography | 30 |
| 14 | References | 31 |

**Acknowledgement**

I take this opportunity to express my profound gratitude and deep regards to my teachers **Prof. Rajashri Thete** for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them time to time shall carry me a long way in the journey of life on which I am about to embark.

Special gratitude I give to my respected head of the division Miss. Swarali Bhalerao, for allowing me to use the facilities available and also help me to coordinate my project

Furthermore, I would also like to acknowledge with much appreciation the crucial role of faculty members on this occasion.

Lastly, I thank almighty, my parents and friends for their constant encouragement without which this project would not be possible.

**Abstract**

The project titled **Online Job Portal** focuses on creating a web-based platform that connects job seekers with potential employers in a seamless and efficient manner. This system replaces the traditional, time-consuming methods of job hunting and recruitment with a centralized, digital solution where users can search for jobs, apply online, and upload their resumes, while administrators can manage job listings, view applications, and maintain the portal's content with ease.

The portal enables job seekers to explore job opportunities based on categories, such as IT, Marketing, HR, and more, and apply without the need for physical documentation or in-person visits. Users can also create personal profiles and upload resumes directly through the portal. For administrators, the system provides tools to post, edit, and remove job listings and review candidate information in an organized format.

The platform is responsive, user-friendly, and compatible with mobile devices. It supports real-time updates, secure login, and streamlined data management. The system improves the overall efficiency of the job application process, saves time for both applicants and employers, and ensures broader accessibility to job opportunities. This project emphasizes automation, accessibility, and digital empowerment in the employment sector.

**1. Introduction**

With the rapid growth of digital infrastructure, the internet has revolutionized how we communicate, work, and seek opportunities. Among the many areas influenced by this transformation, the job market has undergone significant changes. Gone are the days when job seekers relied solely on newspapers or physical visits to companies. Today, web applications have become essential tools in connecting talent with opportunity, providing instant access to job listings, application forms, and employer details.

This project focuses on building an **Online Job Portal** that streamlines the process of job searching and recruitment through a centralized digital platform. It is designed to cater to both job seekers and employers by offering features such as job listings, resume uploads, and administrative control over job postings.

The platform simplifies the overall recruitment process by delivering real-time updates, job categorization, and a user-friendly interface. Candidates can browse jobs based on industry, location, or keywords and apply by uploading their resumes. Employers, on the other hand, can log in securely, post job openings, and review applicant details with ease.

Built using Django for backend development and HTML/CSS/Bootstrap for frontend design, this system ensures scalability, data security, and a smooth user experience across devices. The aim is to provide a robust and accessible solution that reduces the barriers in traditional hiring workflows and enhances engagement for both parties involved.

## **1.1 Objective of the Present Work**

The objectives of this project are as follows:

* To develop a web-based Job Portal system to connect job seekers with potential employers.
* To allow users to create profiles, upload resumes, and apply for jobs efficiently.
* To provide real-time job listings and updates across multiple categories.
* To enable administrators to post, edit, and manage job vacancies with ease.
* To offer a simple, visually appealing, and responsive user interface accessible on all devices.
* To make job opportunities accessible to users at any time and from anywhere via the internet.
* To reduce the time, effort, and cost involved in the traditional hiring and job search processes.

**3. System analysis**

**3.1 PROBLEM DEFINITION**

In the traditional job recruitment process, both job seekers and employers face multiple challenges. Job seekers often struggle to find relevant opportunities and must physically submit resumes or depend on third-party agencies. Employers, on the other hand, find it difficult to manage applicant data, track applications, or reach a broad audience efficiently.

This project addresses these limitations by offering a web-based platform where job listings can be posted and managed digitally, and where applicants can register, browse jobs, and apply with just a few clicks. It simplifies the process of recruitment for employers and enhances accessibility for job seekers, making the entire hiring pipeline faster, more organized, and more effective.

**3.2 Preliminary Investigation**

**Purpose**

The Online Job Portal is designed to digitize the job application process and provide a centralized system for job listings, resume submissions, and application management. It eliminates the need for manual processes and provides a fast, secure, and structured way for both users and administrators to interact with the employment system.

**Benefits**

The portal offers several advantages:

* **Real-Time Job Access**: Users can browse job openings sorted by categories such as IT, Marketing, HR, and more.
* **Resume Management**: Candidates can upload and update resumes directly through the portal.
* **Streamlined Job Posting**: Admins can add, edit, and delete job listings with minimal effort.
* **Application Tracking**: Admins can view applications and filter candidate data quickly.
* **User-Friendly Experience**: The platform provides a clean and responsive interface, accessible across various devices.

**Proposed System**

The proposed Online Job Portal provides a reliable, user-centric, and automated approach to recruitment:

* **Job Seekers** can register, view categorized job listings, upload resumes, and apply for jobs.
* **Administrators** can log in to manage job posts, review applicants, and maintain platform integrity.
* The system supports responsive design, file uploads, and secure access for different user roles.

**3.3 Feasibility Study**

The feasibility study evaluates whether the **Online News Portal with Sentiment Analysis** project is practical, achievable, and beneficial. This assessment ensures the system can be successfully implemented within available resources, technology, and constraints.

**Types of Feasibility Analysis**

* **Technical Feasibility**
  + Developed using Django (Python), a widely used backend framework.
  + Frontend built using HTML, CSS, and Bootstrap ensures responsive UI.
  + Database management using SQLite (or MySQL for production).
  + Secure user authentication and form validation built into the Django framework.
* **Economic Feasibility**
* Utilizes free and open-source tools such as Django, Bootstrap, and SQLite.
* Reduces the cost of recruitment through automation and digital management.
* Eliminates the need for paid job boards or recruitment services.

  **Operational Feasibility**

* Easy to navigate for both users and admins.
* Mobile-responsive design allows access from any device.
* Supports file uploads (resumes) and interactive features like job filtering.

 **Schedule Feasibility**

* Project development is divided into structured phases with achievable deadlines.
* Incremental development ensures early feedback and continuous improvement.

 **Social Feasibility**

* Provides broader access to employment opportunities, especially for users in remote or under-resourced areas.
* Empowers users with less technical knowledge to engage with job listings and submit applications online.

3.4 Project Planning

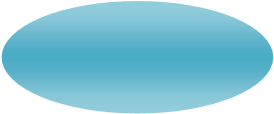
Purpose of Project Planning

Project planning ensures smooth execution, efficient resource use, and timely delivery. It lays out a roadmap from initial investigation to full deployment of the **Online Job Portal**.

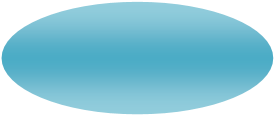
Phases Covered in the Plan

The planning is divided into different phases to ensure smooth execution:

1. **Preliminary Investigation** – Understanding the project scope and objectives.
2. **System Analysis** – Identifying challenges, gathering requirements, and defining solutions.
3. **System Design** – Structuring modules, database design, and UI development.
4. **Coding** – Developing the portal using Python (Django) and integrating sentiment analysis.
5. **Security** – Implementing authentication, data encryption, and user privacy measures.
6. **Testing** – Performing unit testing, integration testing, and user acceptance testing.
7. **Implementation** – Deploying the final system and ensuring smooth operation.



**Stop**



**Start**



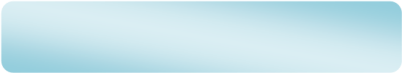
Preliminary



System Analysis



System Design



Coding



Testing



Security



Implementation

3.5 Project Scheduling

The project is structured over multiple weeks as follows:

|  |  |  |
| --- | --- | --- |
| **Phase** | **Duration** | **Deliverables** |
| Requirement Analysis | Week 1 | System goals, user roles, core modules |
| Design | Week 2-3 | UI wireframes, database schema |
| Development | Week 4-7 | Job listing, registration, resume upload |
| Testing | Week 8 | Bug fixing, validation |
| Deployment | Week 9 | Final system launch |

3.6 Software Requirement Specification (SRS)

The **Software Requirement Specification (SRS)** outlines the fundamental requirements of the **Online Job Portal** to ensure efficient functionality, usability, and maintainability.

System Overview

The system consists of the following modules:

1. **User Module** – Allows users to register, upload resumes, and apply for jobs.
2. **Admin Module** – Enables job posting, application viewing, and platform management.

Software & Hardware Requirements

Software Requirements

* **Frontend**: HTML, CSS, Bootstrap
* **Backend**: Python (Django Framework)
* **Database**: SQLite / MySQL
* **Server**: Apache or Nginx (for deployment)

Hardware Requirements

* **Processor**: Intel i5 or higher
* **RAM**: Minimum 8GB
* **Storage**: At least 100GB for database and media files
* **Connectivity**: Internet access for real-time updates

3.7 Functional Requirements

1. User Module

The **User Module** is designed to facilitate job seekers in their journey of exploring and applying for job opportunities with ease. It offers an intuitive and responsive interface for registration, job search, application, and resume submission. Key features include:

* **User Registration and Secure Login**: New users can create accounts using their email addresses and passwords. A secure authentication mechanism ensures that only registered users can access application features.
* **Job Listing Browsing**: After logging in, users can explore job openings categorized by fields such as IT, Marketing, Finance, HR, etc. Each listing includes details like job title, company name, location, qualifications required, and application deadlines.
* **Resume Upload**: Users can upload their resumes in supported formats (PDF or DOCX). This resume is stored in the user’s profile and can be attached automatically when applying to jobs.
* **Job Search and Filtering**: Users can search jobs based on keywords, categories, or location to find relevant opportunities quickly.
* **One-Click Job Application**: With a single click, users can apply for a job directly through the portal. The system automatically links the user's resume and profile details with the job application.
* **Application History Tracking**: Users can view the list of jobs they have applied to, helping them keep track of their application progress.

2. Admin Module

The **Admin Module** is the backbone of the portal's content management system. It enables administrators to manage users, job postings, and overall platform content with efficiency and control. Key features include:

* **Secure Admin Login**: Admins access the dashboard using a secure authentication system to prevent unauthorized access.
* **Job Post Management**: Admins can create new job posts, update existing listings, or delete old/inactive jobs. Each post includes fields such as job title, description, category, qualification, salary, and deadline.
* **Applicant Tracking and Resume Access**: Admins can view a list of users who applied for specific jobs. They can also download or preview resumes uploaded by applicants for screening purposes.
* **User Account Management**: Admins can monitor user activity, deactivate or delete user accounts if necessary, and resolve user-related issues.
* **Content Moderation and Platform Maintenance**: Admins have access to static content pages (e.g., About Us, Contact Us) and can make changes as required. They are also responsible for maintaining data integrity and ensuring that the portal runs smoothly.
* **System Monitoring**: The admin panel includes an overview dashboard that summarizes active jobs, total applications, and user statistics for better operational oversight.

3. Sub-Admin Module

Sub-Admins can:

* Perform **news management** (add, edit, delete) but cannot create new sub-admins.
* Access sentiment insights for the articles they manage.
* Moderate comments within assigned categories.

3.8 Software Engineering Paradigm

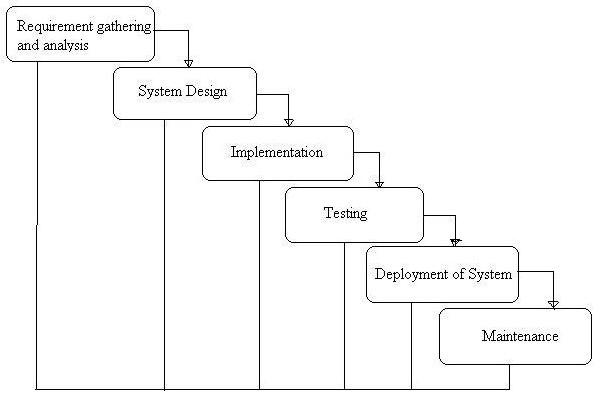
The development of the **Online News Portal with Sentiment Analysis** follows a structured approach to ensure efficiency, reliability, and maintainability. The chosen paradigm helps streamline the project by defining clear phases while allowing iterative improvements.

Development Model: Adapted Waterfall Model

**The project follows an Adapted Waterfall Model**, which ensures a structured, phase-wise development process. While the traditional Waterfall Model is linear, this adapted version introduces flexibility by incorporating feedback loops, especially during the testing phase. This approach helps ensure that issues discovered later can still lead to improvements in earlier stages without disrupting the overall flow.

Key Adaptations in the Waterfall Model:

1. **Sequential Stages**: Each development phase—requirement analysis, design, coding, testing, and deployment—progresses in a logical order. Every stage builds upon the deliverables of the previous one, maintaining a clear development structure.
2. **Feedback Integration**: Unlike the rigid structure of the classic model, this adapted approach allows feedback gathered during testing to inform and refine previous stages, particularly the coding and design phases. This helps improve the final product's quality and usability.
3. **Defined Milestones**: Each phase has clear deliverables and review checkpoints. Progression to the next stage only occurs once the current phase has been reviewed and approved, ensuring completeness and consistency throughout the development lifecycle.



Phases of Development

1. **Requirement Analysis & System Study**
   * Identifying project goals, challenges, and functional specifications.
   * Gathering stakeholder requirements and defining core functionalities.
2. **System Design**
   * Structuring the **database, modules, and architecture**.
   * Designing **user interfaces** for optimal accessibility.
3. **Implementation (Coding)**
   * Backend development using **Python (Django)**.
   * Frontend design using **HTML, CSS, Bootstrap**.
   * Database integration with **MySQL**.
   * Sentiment analysis integration with **TextBlob**.
4. **Testing & Debugging**
   * Unit testing, integration testing, and usability checks.
   * Debugging for performance improvements.
5. **Deployment & Maintenance**
   * Hosting on a scalable environment.
   * Continuous updates for feature enhancements.

3.9 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

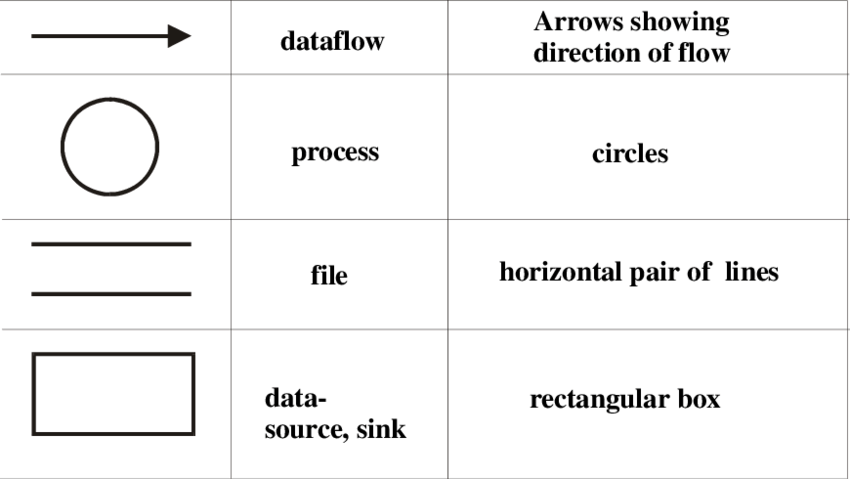
It shows how data enter and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

**The following observations about DFDs are essential:**

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

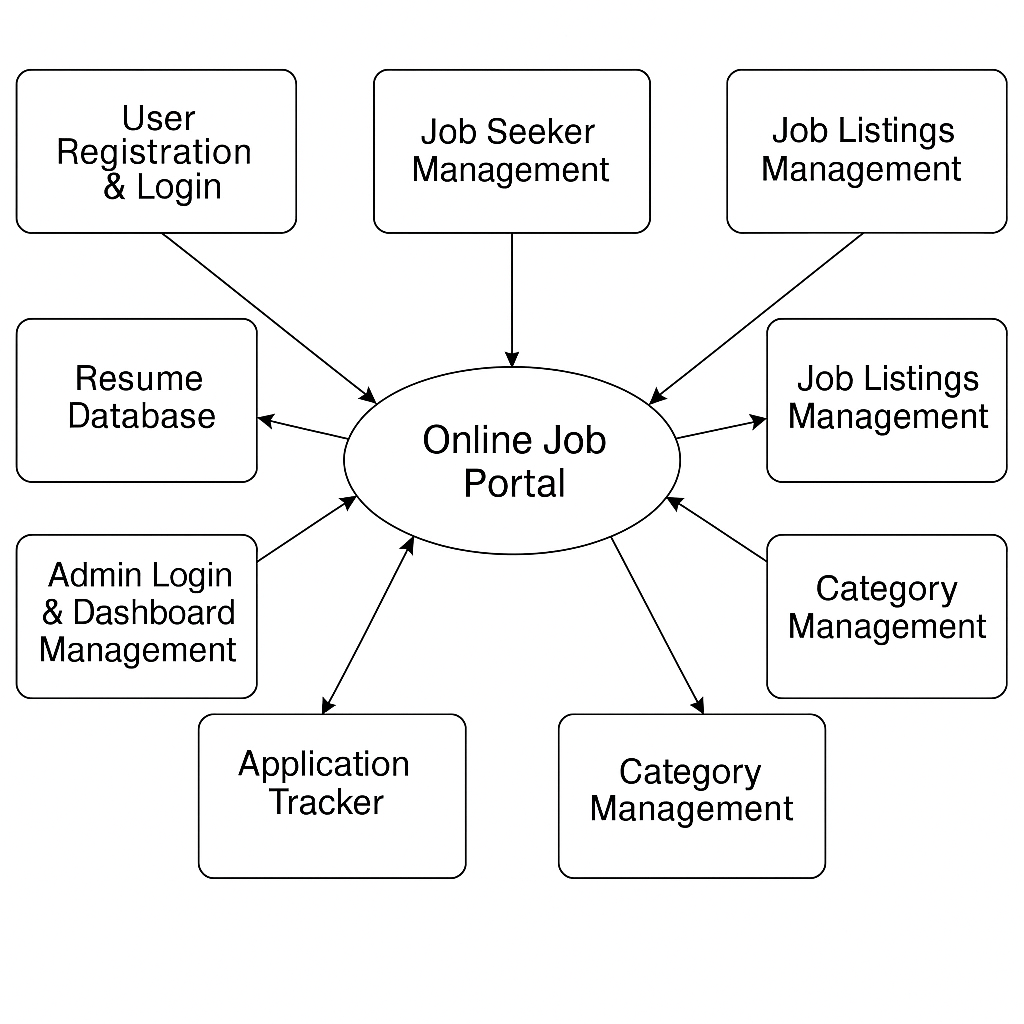
******

**Circle:** A circle (bubble) shows a process that transforms data inputs into data outputs.

**Data Flow:** A curved line shows the flow of data into or out of a process or data store.

**Data Store:** A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

**Source or Sink**: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

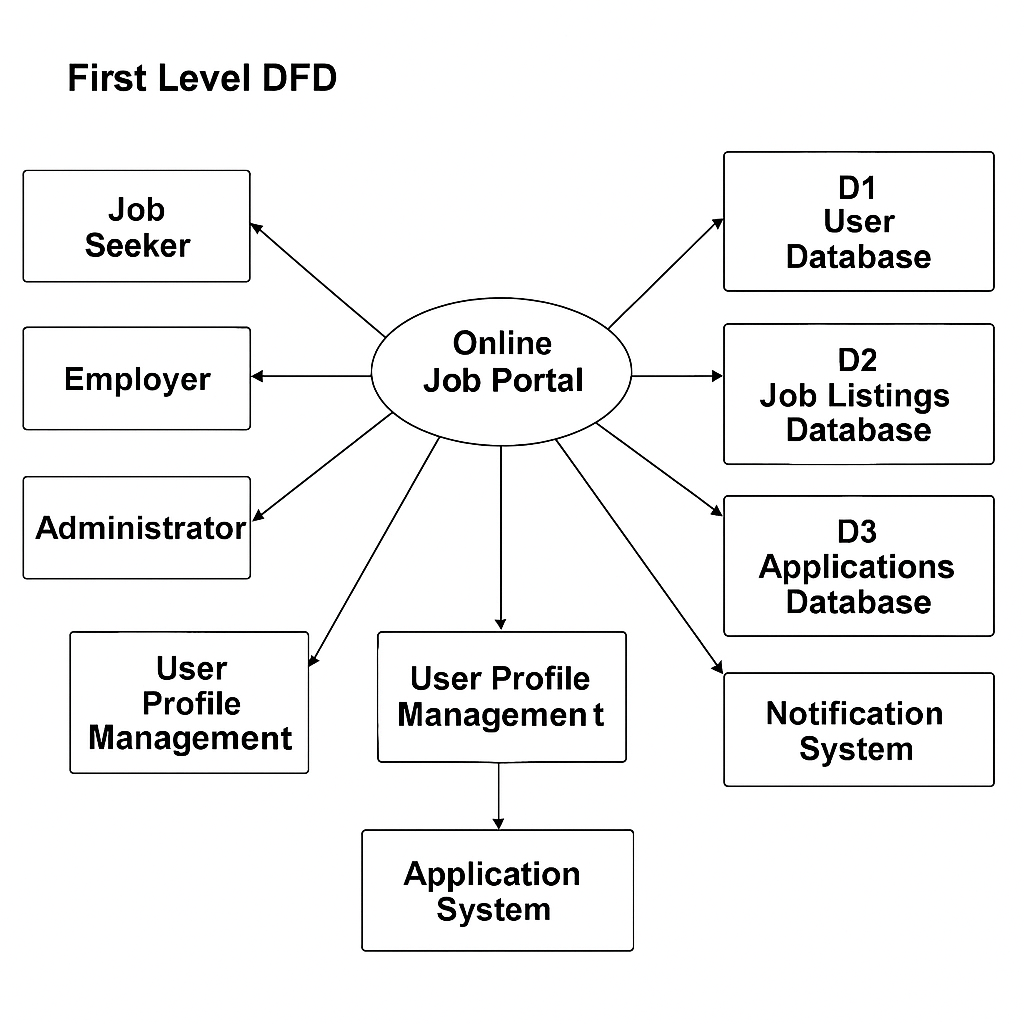


#### **Central Process**:

**OJP** (Online Job Portal) — This is the main system where all core processes are handled.

#### **External Entities and Processes** (connected to the main system):

1. **User Registration & Login** ↔ Handles user signup, login, and authentication.
2. **Job Seeker Management** ↔ User profile creation, resume upload, application tracking.
3. **Job Listings Management** ↔ Admin posts, edits, and deletes job vacancies.
4. **Job Application Management** ↔ Job seekers apply for jobs; resumes are linked to applications.
5. **Resume Database** ↔ Stores uploaded user resumes in various formats.
6. **Application Tracker** ↔ Admin views list of candidates for each job post.
7. **Admin Login & Dashboard Management** ↔ Admin login, dashboard access, and system monitoring.
8. **Category Management** ↔ Organizes jobs into categories like IT, Marketing, HR, etc.



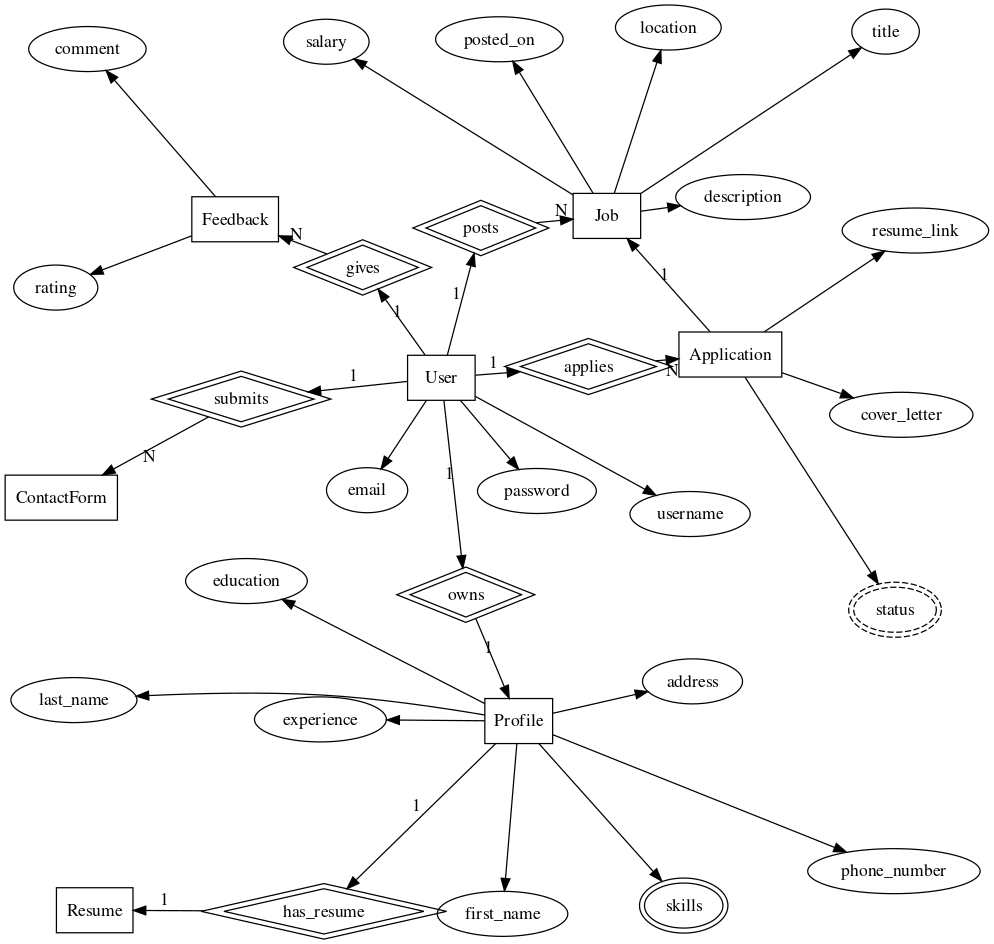
The **First-Level DFD of the Online Job Portal** illustrates how the system manages essential operations such as user authentication, profile management, job postings, and application tracking. It connects these core processes with data-driven modules like resume storage, job listings, and application databases to ensure seamless interaction between job seekers, employers, and administrators. The system also includes a notification mechanism to keep users informed about job-related updates, enhancing overall engagement and efficiency across the platform.

**Sentiment analysis**

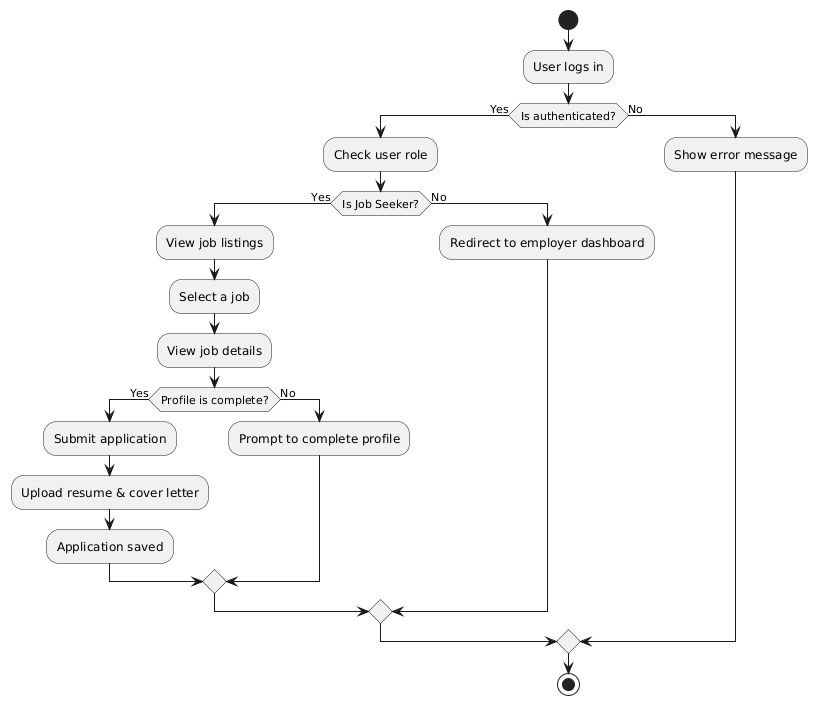
ER diagram

The ER Diagram of the **Online Job Portal System** illustrates the structured relationships among Users (Job Seekers and Employers), Admins, Job Listings, Applications, Categories, and Feedback mechanisms, ensuring efficient data management and facilitating insightful recruitment processes.

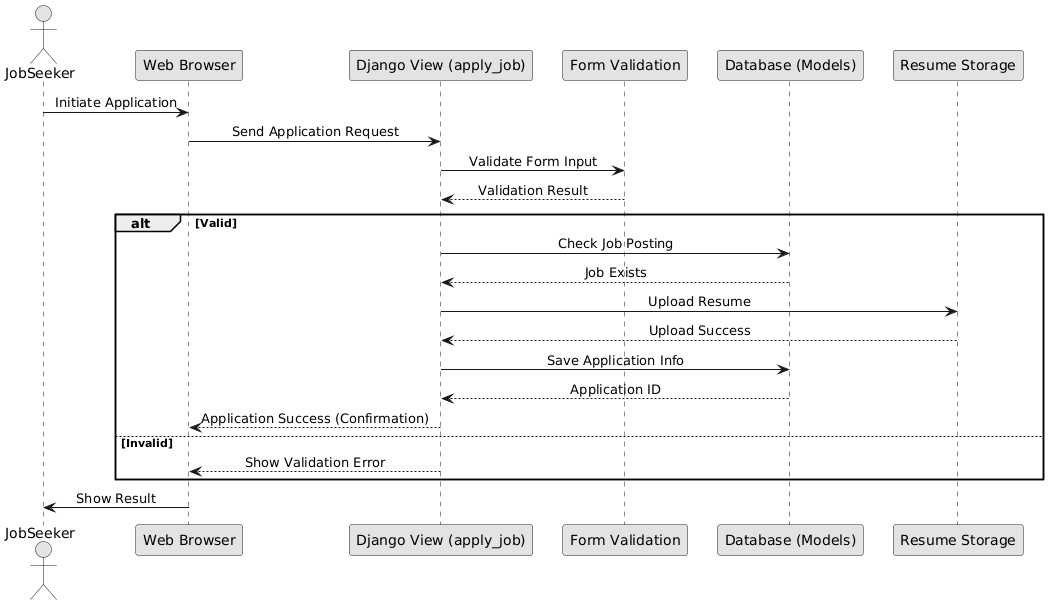
**ER Diagram:**

****

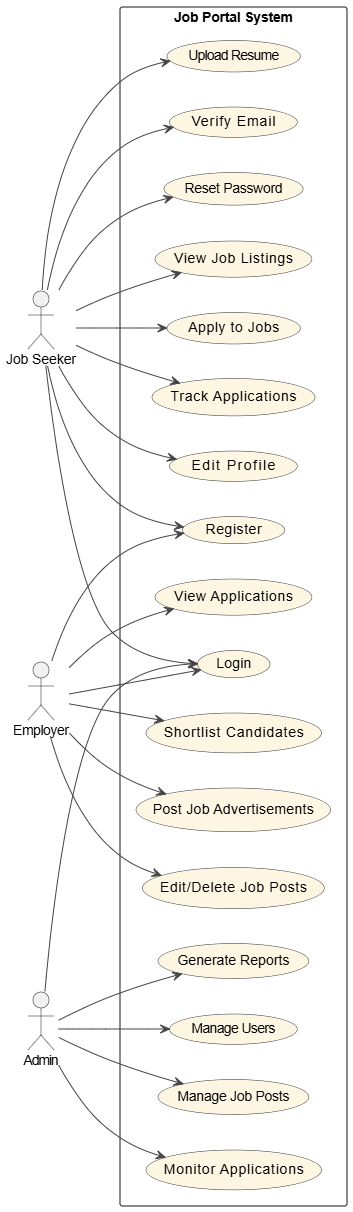
**Activity Diagram:**

****

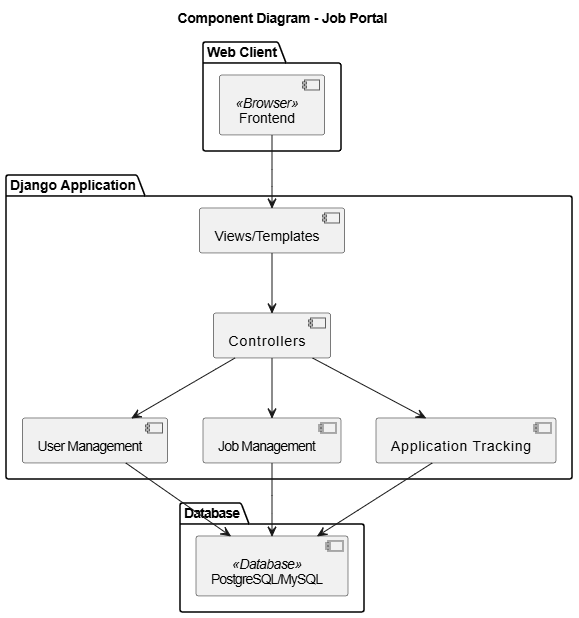
**Sequence diagram:**

****

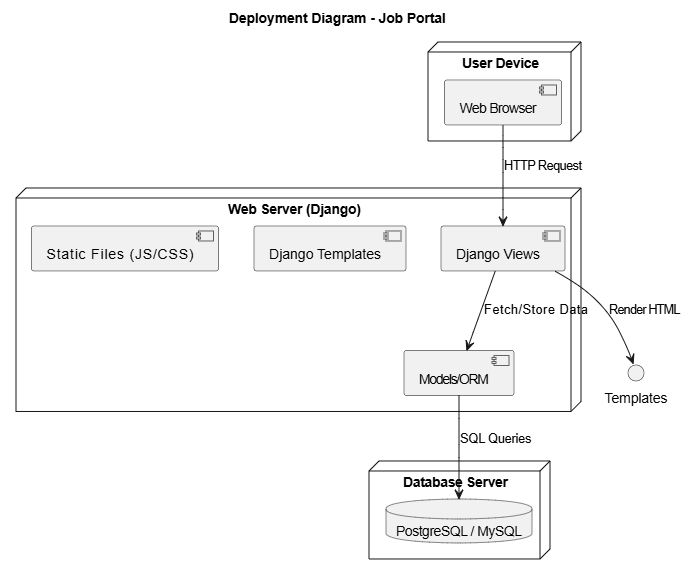
**UseCase Diagram:**

****

**Component Diagram:**

****

**Deployment diagram:**

****

**4. system design**

#### 1. **User Module (Job Seekers)**

Users can:

* Register and log in securely.
* Create and update their profile with resumes, experience, and skills.
* Search and filter job listings based on keywords, location, and categories.
* Apply for jobs directly through the portal.
* Track the status of their job applications.
* Receive email notifications for relevant job postings or status updates.

#### 2. **Employer Module**

Employers can:

* Register and log in securely.
* Post new job openings with detailed descriptions, requirements, and categories.
* Edit or delete previously posted jobs.
* View applications received for their job listings.
* Shortlist or reject candidates.
* Receive notifications when new applications are submitted.

#### 3. **Admin Module**

Admins have full control over the platform and can:

* Secure Login System – Access the admin dashboard through authentication.
* Dashboard Management – Manage job categories, user roles, and platform settings.
* User Moderation – Activate, deactivate, or delete user/employer accounts.
* Job Moderation – Approve, reject, or remove job listings.
* Sub-Admin Management – Create and manage sub-admin roles with specific permissions.
* Content Management – Edit static pages like ‘About Us’, ‘Contact Us’, etc.
* Site Monitoring – View metrics such as total users, job posts, and applications.

#### 4. **Sub-Admin Module**

Sub-Admins can:

* Add, edit, or delete job listings and manage user queries.
* Moderate employer and job seeker activities within assigned categories.
* Generate reports and oversee specific platform segments as assigned by the Admin.

### 4.2 DATA STRUCTURE OF ALL MODULES:

The system is backed by a structured MySQL database customized for job portal operations. It is accessed programmatically through Django's ORM layer and consists of:

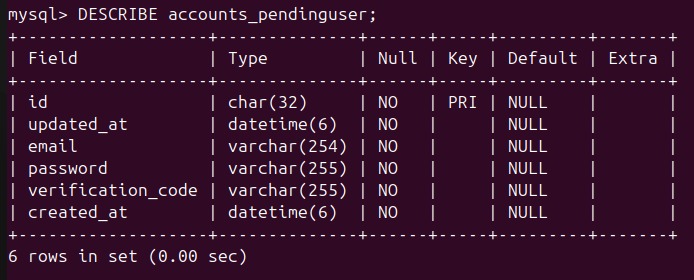
* **Custom Tables** for:
  + Users (Job Seekers and Employers)
  + Job Listings
  + Applications
  + User Profiles
  + Categories/Tags
  + Notifications
* **Default Django Tables** handling:
  + Authentication
  + Sessions
  + Permissions
  + Admin metadata

These tables are interrelated through foreign key relationships and can be easily maintained and queried for analytics, reporting, and moderation tasks.

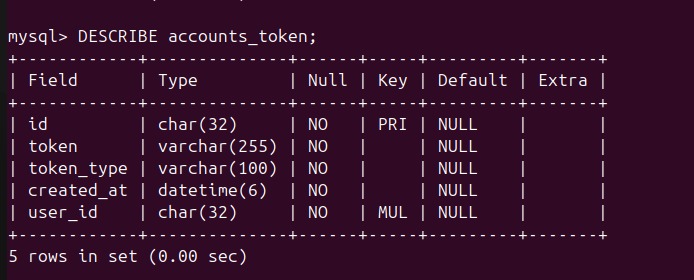
**Customized Tables Details**

**User Management Tables-**

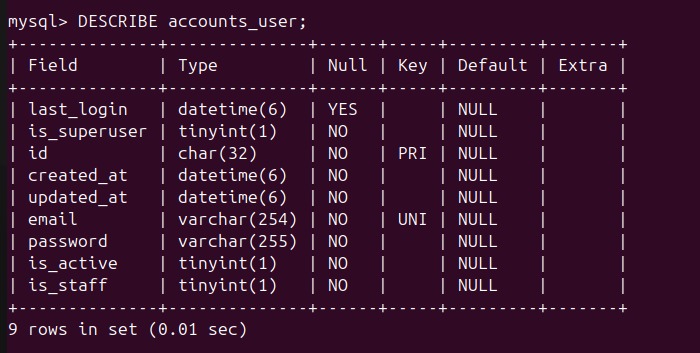
**accounts\_pendinguser**:  
Stores information of users who are in the signup process but not yet verified. Includes fields like email, password, and verification code.



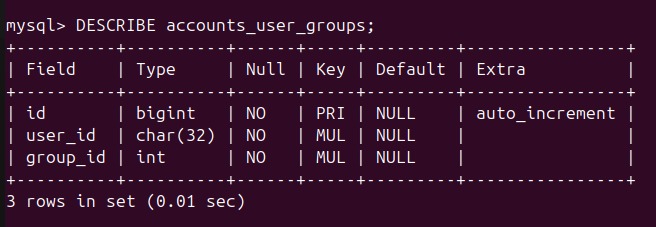
**accounts\_token:**  
Manages authentication tokens linked to users, specifying the token value and type, along with a reference to the user.



**accounts\_user**:  
Stores registered user data including login credentials, status flags (active, staff, superuser), and timestamps.

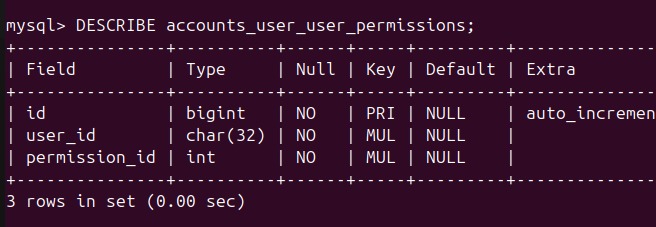


**accounts\_user\_groups**:  
Maps users to groups for role-based access control by linking user IDs to group IDs.



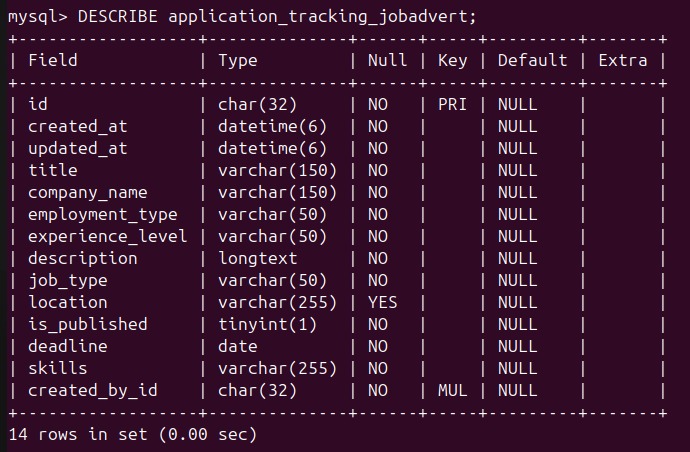
**accounts\_user\_user\_permissions:**

Associates users directly with specific permissions, enabling fine-grained access control beyond group membership.



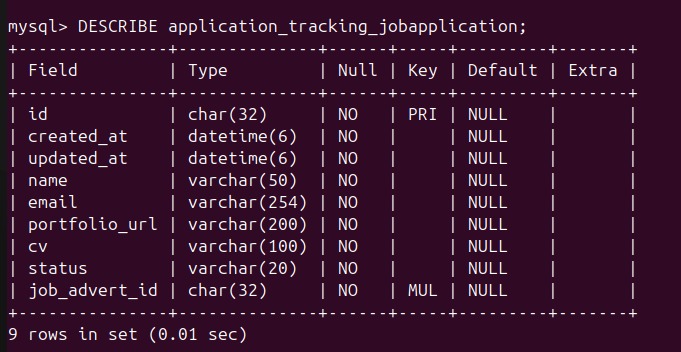
**Job Application System Tables :**

**application\_tracking\_jobadvert:**  
Contains job postings with details like title, company, type, location, skills, and creator information.

****

**application\_tracking\_jobapplication:**  
 Stores applicant submissions for job ads, including personal info, portfolio URL,

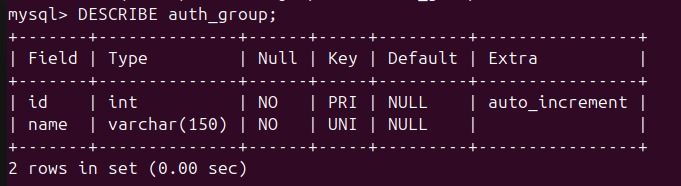
and application status.



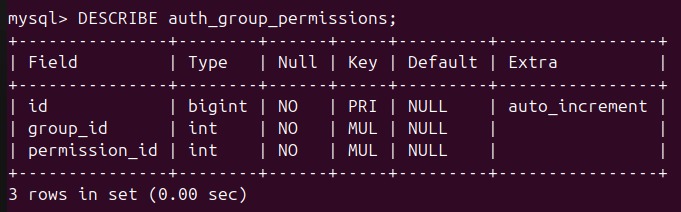
**Default Tables Details:**

**auth\_group:**

Stores user groups for role-based access management, each with a unique name.

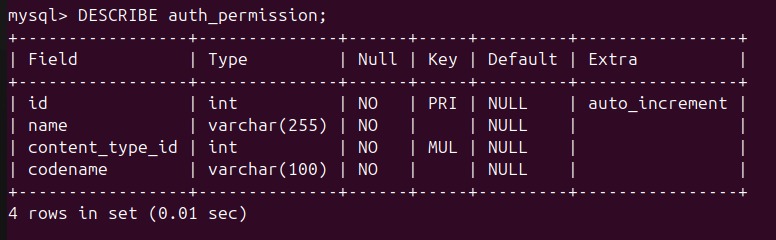


**auth\_group\_permissions:**

Links groups to permissions to define group-level access rights

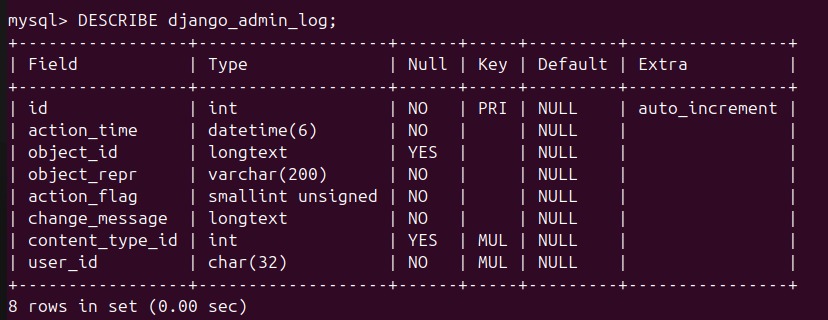
**auth\_permission:**

Defines individual permissions (e.g., add, edit, delete) for Django models, linked to content types



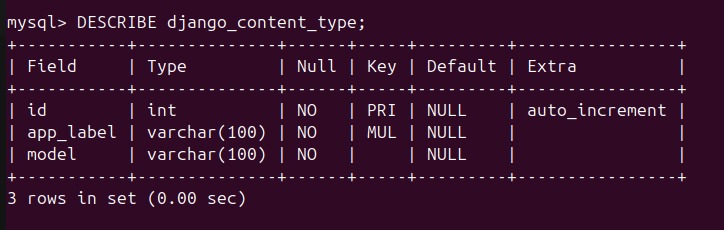
**Django Internal Tables**:

**django\_admin\_log:**  
Tracks admin actions (like add/change/delete) for auditing, including who did what and when.

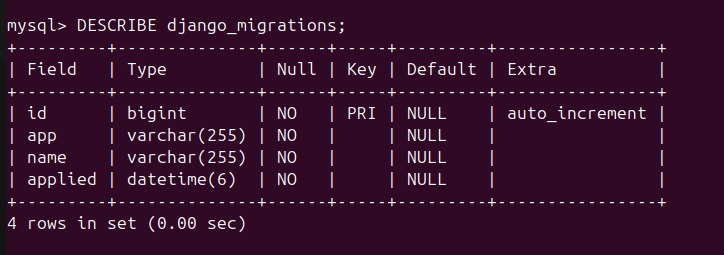


**django\_content\_type:**

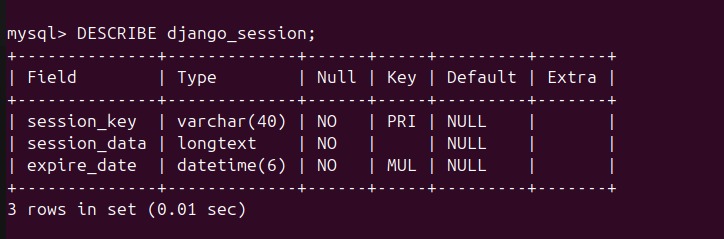
Maps Django models to their content types, used for generic relationships and permission mapping.



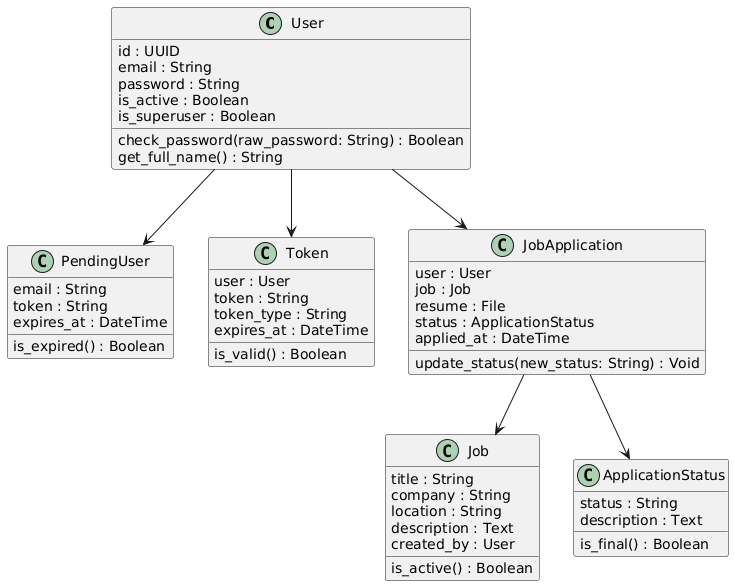
**django\_migrations:**Records applied database migrations for Django apps to manage schema versioning.



**django\_session:**Stores session data for logged-in users, including session key, data, and expiration timestamp.

****

**Relationship between tables (Class Diagram)**

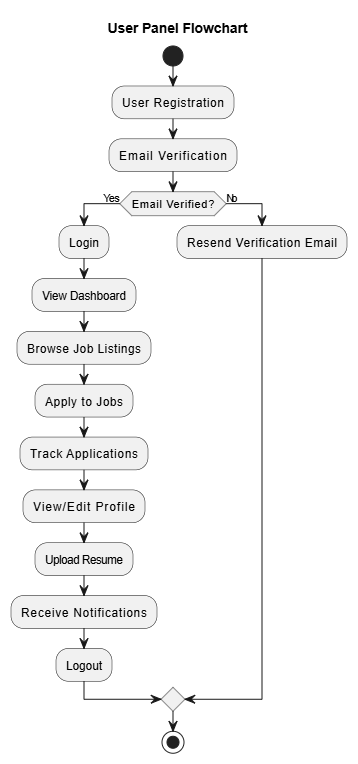


4.3 PROCEDURAL DESIGN:

Process logic (flowchart ) of each module

**4.3.1 User Panel Design**

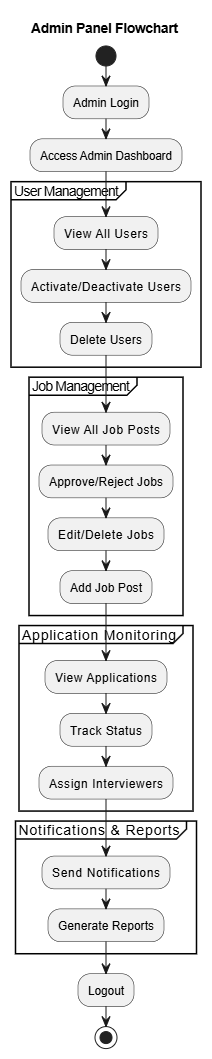
In user panel design we have done our task for user. Here we provide facility about Online News Paper. In index page user can select any options which is needed by him/her. By selecting options he/she can see the desired page. Then he/she can get the all oriented information finally. The design of user panel is shown in following flow chart….

****

**4.3.2 Admin Panel Design**

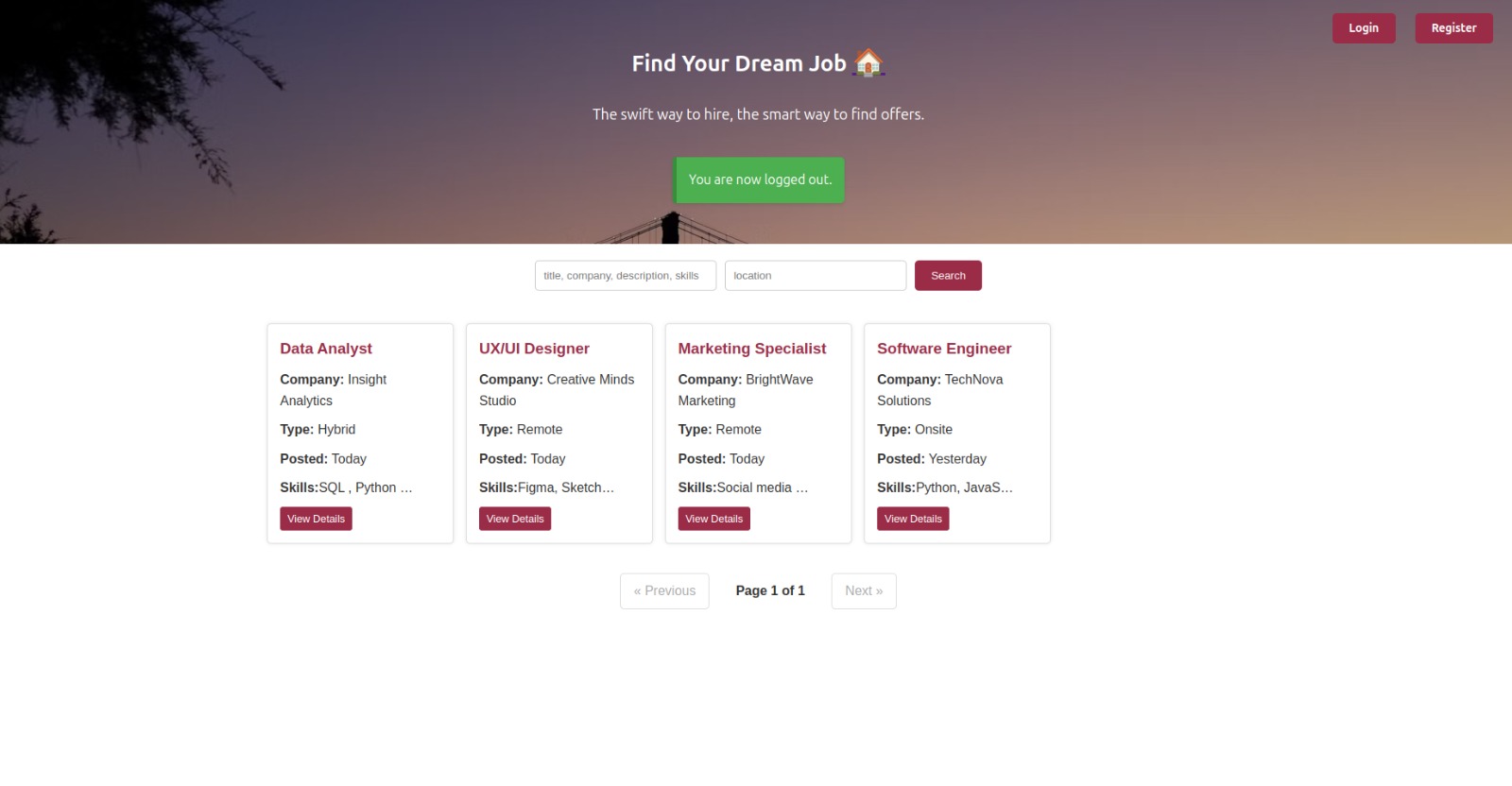
We have design user login facility to manage and update all of the information. It is fully secured page. Without appropriate username and password it cannot be accessed by anyone. For admin login after giving username and password we need to click a login button , when we click login button it is not directly entered in home page , it stay in login page. Then it starts a session and set two variables called username and password. If the username and password are matched with database, it can enter in home page. It is not possible without click login button. In case if username or password are not matched with database than Invalid username or password massage is shown. We can describe the login facility in admin login by using below flow chart given below--

**Admin Login Flow Chart:**

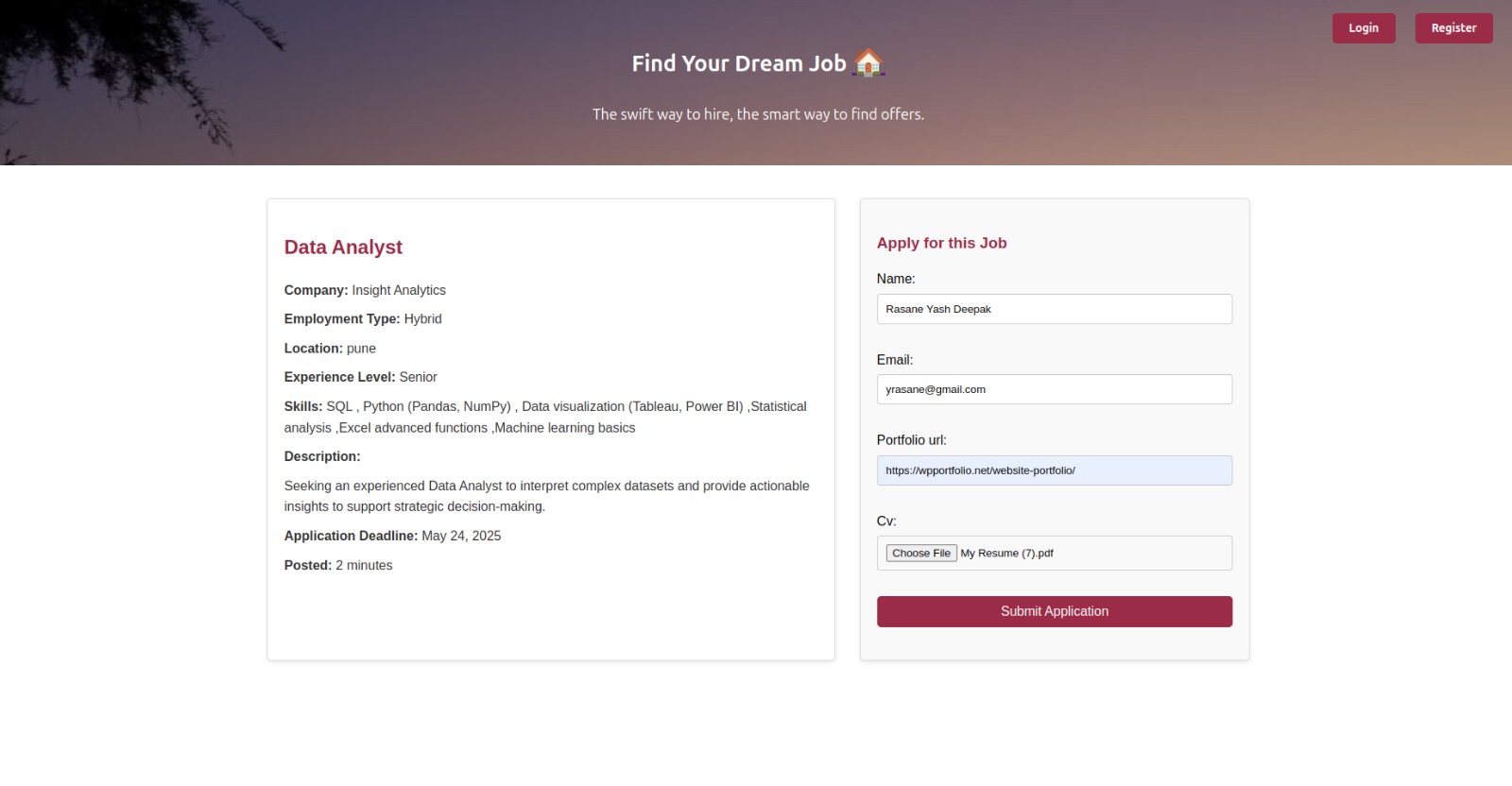
****

**Fig. 4.2: Admin Login Flow Chart.**

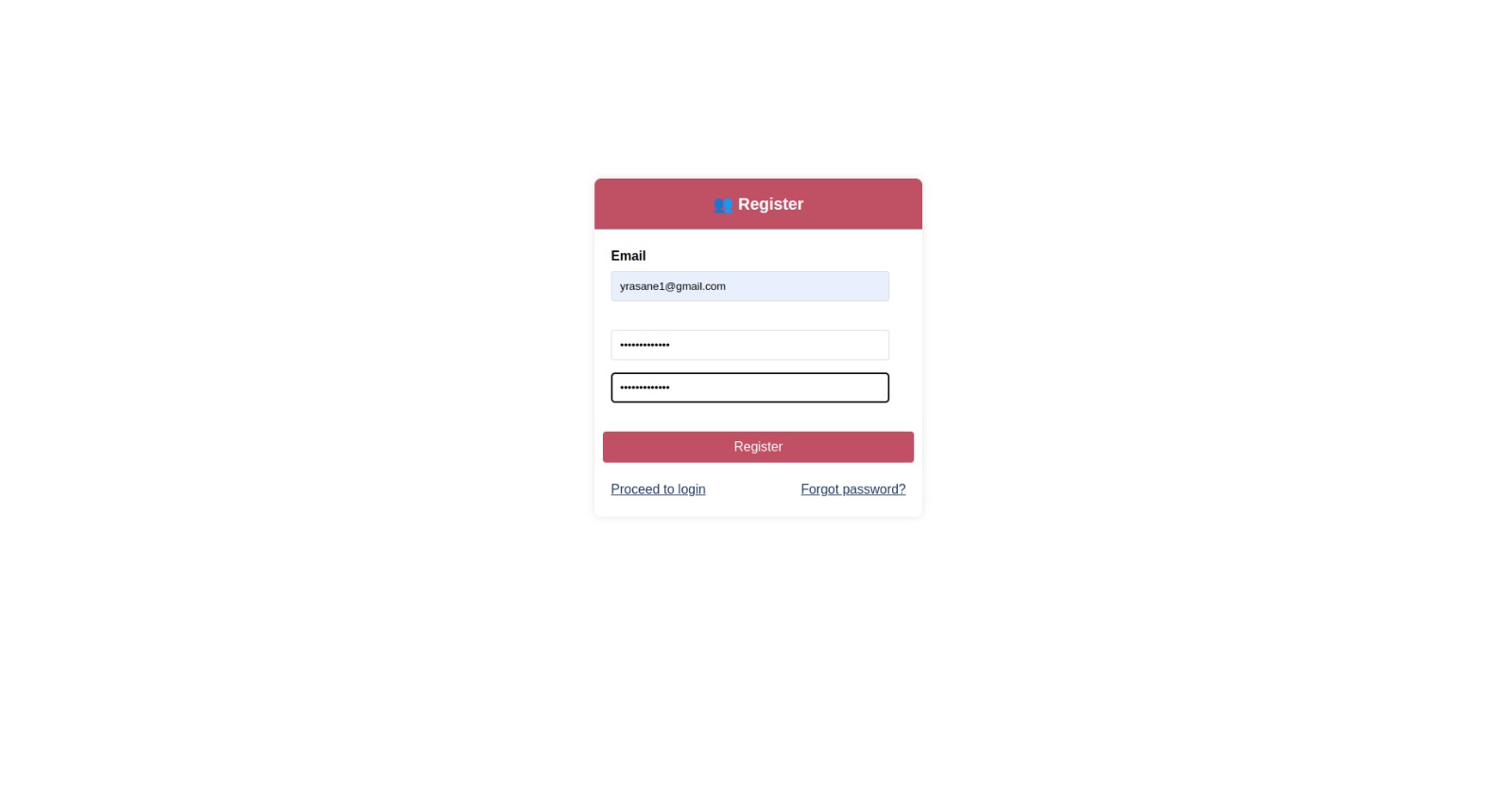
**Home Page**

****

**Job advertisement details**

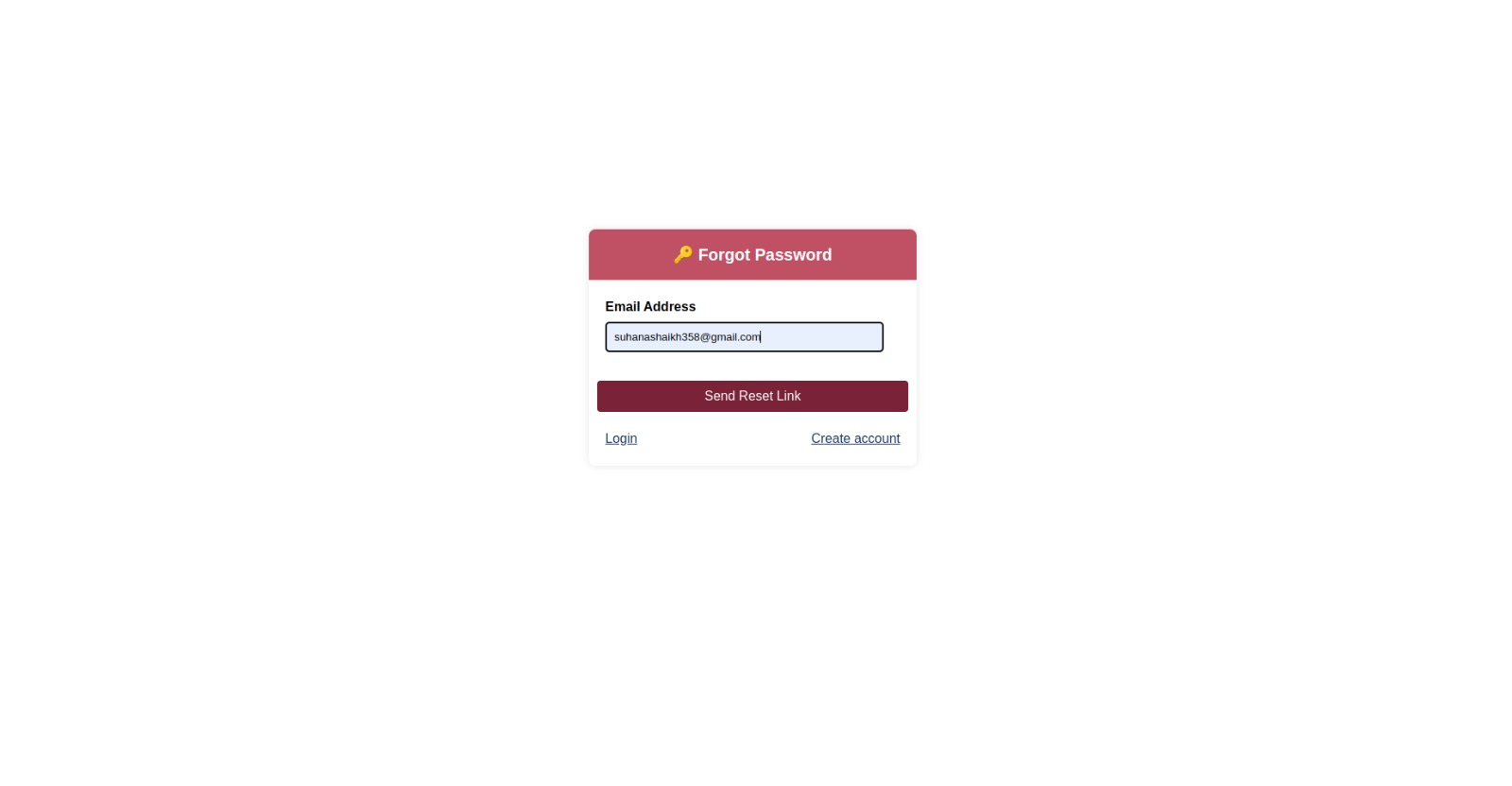
****

**Registration Page**

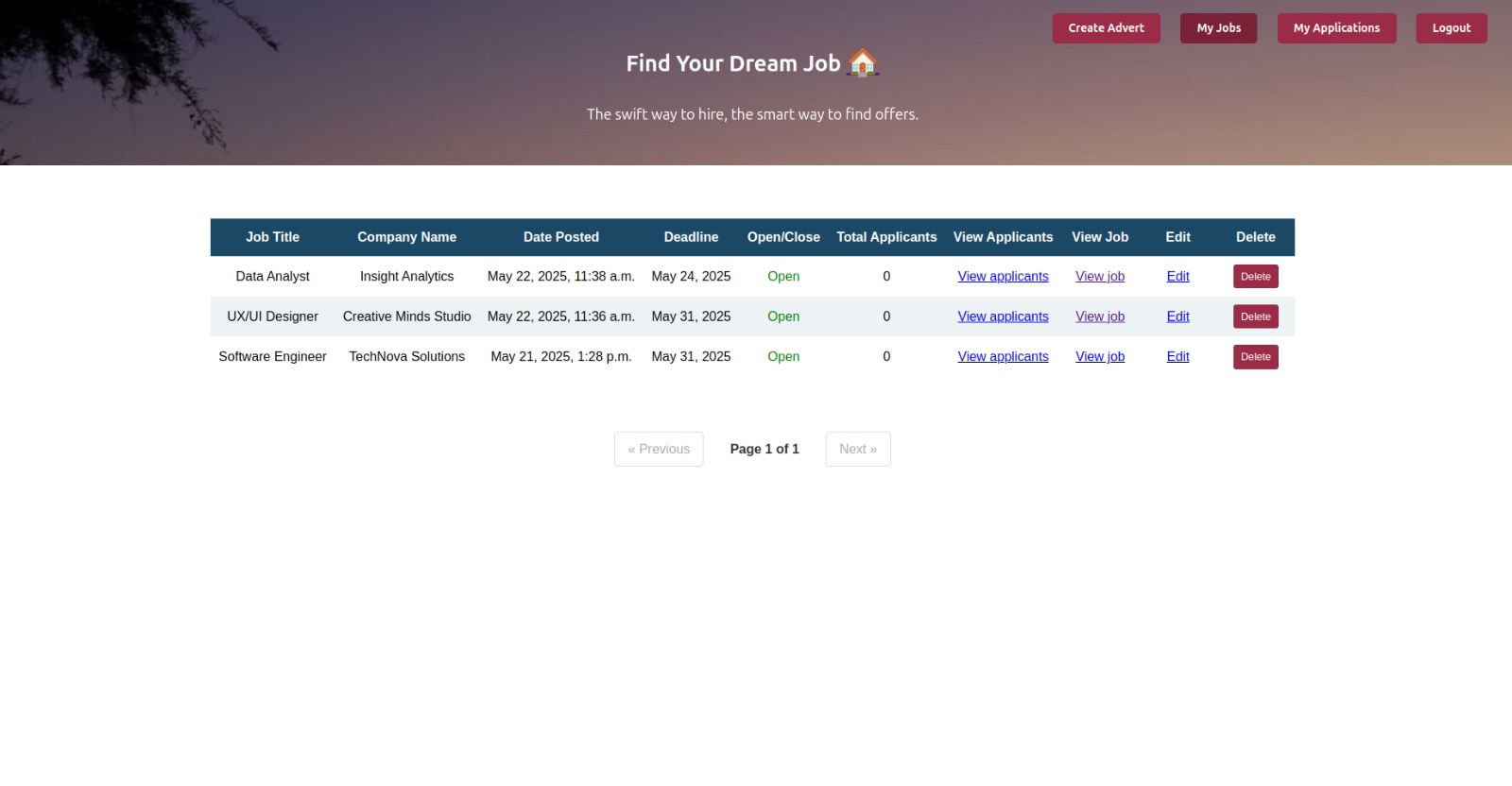
****

**Login Page**

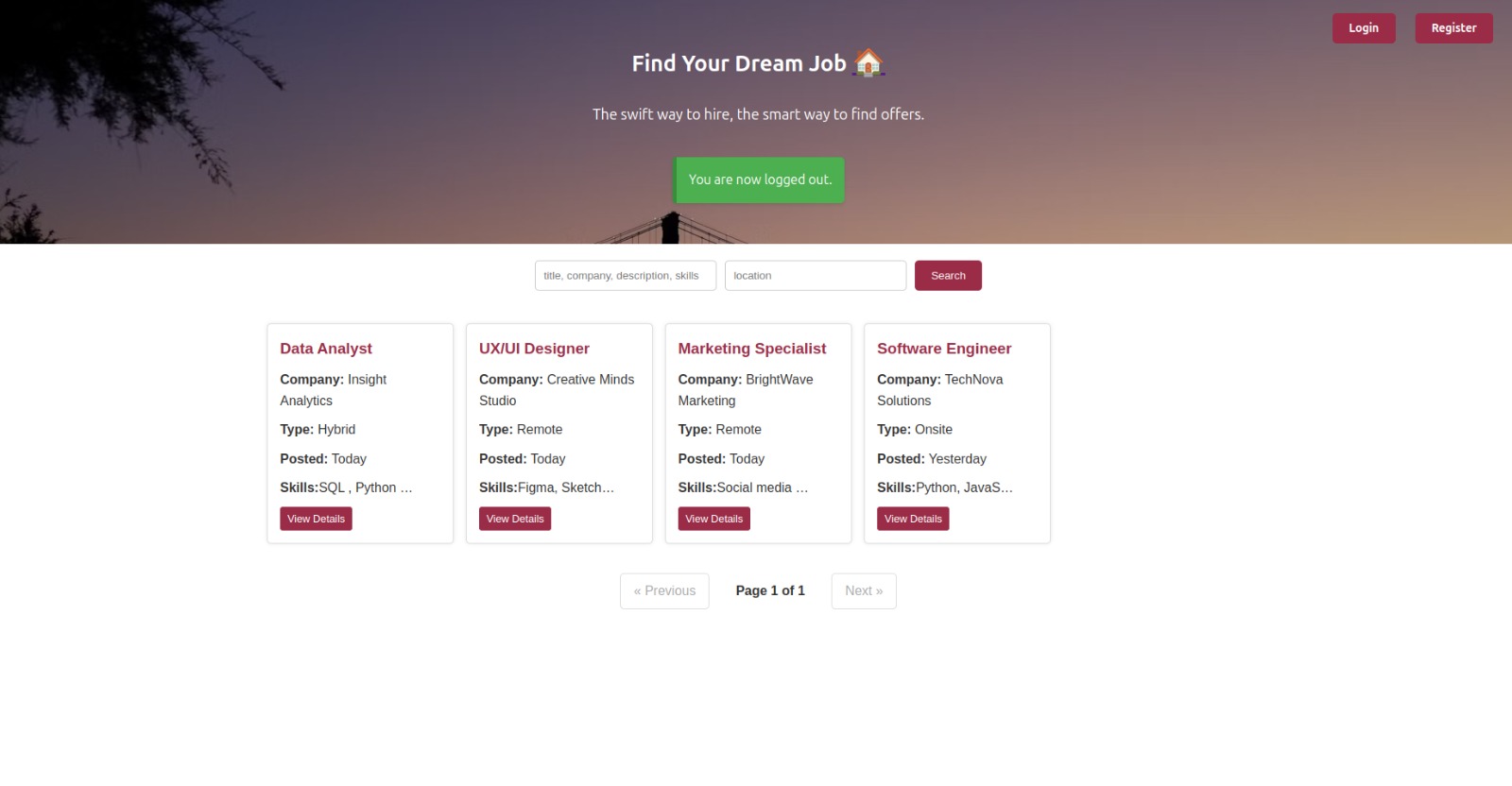
****

**Forgot Password**

**My Jobs**

****

**Job List**

****

**Future scope**

1. **AI-Based Job Matching**
   * Use artificial intelligence to match jobs with users based on their skills and experience.
2. **Mobile App**
   * Create Android and iOS apps so users can access the portal easily on their phones.
3. **Admin Dashboard with Analytics**
   * Add charts and reports to help the admin monitor user activity and job postings.
4. **Employer Branding**
   * Allow employers to add videos, reviews, and details about their company.
5. **Skill Tests and Certifications**
   * Provide online tests to help users prove their skills and get certified.
6. **Multi-language Support**
   * Add support for multiple languages to help more users access the portal.
7. **Stronger Security**
   * Add features like two-step verification and spam protection.
8. **API Integrations**
   * Connect the portal with LinkedIn, Google Jobs, and resume parsers.
9. **Chatbot Assistance**
   * Add a chatbot to help users with common questions and navigation.
10. **Internship and Freelance Jobs**

* Allow postings for internships and freelance projects, not just full-time jobs.

**Bibliography**

1. **Django Documentation**  
   <https://docs.djangoproject.com/>  
   → Used for understanding the Django framework and implementing backend functionalities.
2. **W3Schools**  
   <https://www.w3schools.com/>  
   → Referred for HTML, CSS, JavaScript, and basic web development tutorials.
3. **Stack Overflow**  
   <https://stackoverflow.com/>  
   → Helped troubleshoot issues and find solutions for Django, HTML, and Python-related problems.
4. **MDN Web Docs (Mozilla Developer Network)**  
   <https://developer.mozilla.org/>  
   → Used for frontend development references including HTML5, CSS3, and JavaScript.
5. **GeeksforGeeks**  
   <https://www.geeksforgeeks.org/>  
   → Referred for Python programming concepts and Django project examples.
6. **GitHub Repositories and Open Source Projects**  
   <https://github.com/>  
   → Used for exploring structure and code examples of similar job portal projects.
7. **PlantUML Documentation**  
   <https://plantuml.com/>  
   → Used to create use case diagrams, ER diagrams, and flowcharts.