



# Table of Contents

1. **Abstract**
2. **Introduction**
3. **Features and Functionality**
4. **System Architecture**
5. **Setup & Installation**
6. **User Guide**
  - For Job Seekers
  - For Recruiters
7. **API Documentation**
  - API Endpoints Overview
  - Sample Requests & Responses
  - Error Messages and Codes
  - API Testing Tools
8. **Database Schema**
9. **Security Guidelines**
10. **Testing Documentation**
11. **Deployment Guide**
12. **Troubleshooting / FAQs**
13. **Change Log / Version History**
14. **License**

## **15. Contributing Guide**

## **16. Conclusion**

# ABSTRACT

---

The **Online Job Recruitment Platform** is a web-based application designed to connect job seekers with recruiters in a simple, efficient, and modern way. It helps individuals find and apply for jobs, while allowing companies and recruiters to post job openings, review applications, and conduct interviews—all in one centralized platform.

This system supports both traditional user registration and Google login for easy access. Job seekers can create a profile, upload their resume, and apply to jobs using a user-friendly interface. They can also track the status of their applications in real time.

On the other side, recruiters can log in to post job vacancies, view applications, and shortlist candidates. For selected applicants, recruiters can schedule and conduct online interviews using secure video conferencing links. These links are private and only visible to the invited candidates.

A key feature of this platform is its integration with **AI-powered video/audio interview tools**, which allow recruiters to conduct live interviews and optionally use AI services to transcribe, summarize, or analyze the conversation for better decision-making.

The platform is built using Django for the backend, PostgreSQL for data storage, and includes REST APIs, secure authentication (JWT and OAuth), and third-party services like Google OAuth and Twilio or Jitsi for video calls. The project is scalable, secure, and suitable for real-world deployment.

In summary, this project provides a full digital hiring solution—from registration and job search to resume submission, recruiter management, and intelligent video interviews—making it a powerful tool for modern recruitment.

# INTRODUCTION

---

The **Online Job Recruitment Platform** is a full-featured web application created to simplify the recruitment process for both job seekers and employers. In today's digital world, searching for a job or finding the right candidate can be time-consuming and inefficient. This platform solves that problem by bringing everything into one place—job listings, resume submissions, applications, and interviews.

## Purpose of the Project

The goal of this project is to build a centralized, easy-to-use portal that allows:

- **Job Seekers** to register, upload resumes, search for jobs, apply online, and track their application status.
- **Recruiters** to post job opportunities, manage applicants, shortlist candidates, and conduct interviews—right from their dashboard.
- **Admins** to oversee all activity and manage the entire system from a single interface.

## Who Can Use This Platform?

- **Job Seekers:** Anyone looking for employment opportunities.
- **Recruiters/Employers:** Companies or HR professionals hiring new candidates.
- **Superuser/Admin:** The platform owner or system administrator who manages all users, jobs, and data.

## Key Components

This platform includes:

- **User Registration/Login** using email or Google
- **Job Posting and Job Application System**

- **Resume Upload and Profile Management**
- **Admin Dashboard for Monitoring Activities**
- **Email Notifications** for important updates
- **Video Interview Feature** for selected candidates
- **AI Support** to enhance interview evaluations

## **Technologies Used**

- **Backend:** Django (Python)
- **Frontend:** HTML, CSS, JavaScript (Django Templates)
- **Database:** PostgreSQL (or SQLite for development)
- **Authentication:** JWT and Google OAuth
- **APIs:** RESTful APIs for all data interactions
- **Video Calls:** Integrated with Twilio, Jitsi, or Daily.co
- **AI Tools (Optional):** Transcription, response summarization, interview analysis

## **Why This Project Matters**

Traditional hiring processes are often slow and fragmented. With this platform:

- Everything is online and in one place.
- Communication is faster through automation and real-time updates.
- Interviews can be done remotely and more efficiently using AI and video tools.

# FEATURES AND FUNCTIONALITY

---

## 1. User Registration/Login (Email + Google OAuth)

- Users can **sign up** using a standard email/password or **log in with their Google account** using secure OAuth integration.
- All login sessions are protected, and the platform uses **JWT** or Django's built-in session framework to maintain secure access.

## 2. Resume Upload and Management

- Job seekers can **upload their resumes** in PDF or DOCX format through their dashboard.
- The uploaded resume is securely stored on the server and is **automatically included** when the user applies for a job.
- Users can also **update their resume** at any time.

## 3. Job Listings and Search/Filter

- All job postings are visible to users.
- Candidates can use the **search bar** to find jobs by title or keyword.
- Additional **filters** such as location, company, experience level, and job type help users narrow down relevant opportunities.

## 4. Job Application and Tracking

- Once logged in, job seekers can click “**Apply**” to submit their application for any job.
- The system records each application and links it to the job and user.
- Users can **track the status** of their applications (e.g., Pending, Shortlisted, Interview Scheduled, Rejected) through their dashboard.

## 5. Admin Dashboard for Superuser

- A designated superuser (admin) has full access to manage:
  - All users (job seekers & recruiters)
  - Job posts
  - Applications
- Admins can also **post new jobs**, assign interviews, and monitor platform activity.

## 6. Email Notifications for Status Updates

- When key actions occur (e.g., successful registration, application submission, or interview scheduled), users receive **automated email alerts**.
- This keeps candidates informed without needing to log in frequently.

## 7. Recruiter-only Interview URL Sharing

- Once a recruiter shortlists a candidate, they can **generate a private interview URL** using a third-party video API.
- This **interview link is only visible to the selected candidate** in their dashboard.
- The recruiter maintains full control over who receives access to this URL.

## 8. AI-Powered Video/Audio Interview Feature

- The system integrates with platforms like **Twilio Video, Jitsi, or Daily.co** for real-time video/audio conferencing.
- Advanced AI features (optional) can:
  - **Transcribe conversations in real-time**
  - **Highlight important responses**
  - **Generate automated summaries**

- This makes interviews smarter and easier to evaluate, especially when managing many candidates.



# SYSTEM ARCHITECTURE

---

To explain how the system is technically structured, how each component interacts with others, and how it supports the features described above.

- **Frontend:** Built using **Django Templates** (HTML, CSS, JS). It provides a user-friendly interface for all users to register, log in, view/apply to jobs, and access interview links.
- **Backend:** Powered by **Django**, it handles all business logic, user authentication, data processing, resume handling, job posting management, and API integration for interviews.
- **Database:** Uses **PostgreSQL** (preferred for production) or **SQLite** (for local development). Stores structured data including users, jobs, applications, resumes, and interview links.
- **APIs:** Custom **REST APIs** manage job CRUD operations, application tracking, and interview scheduling. These APIs enable secure communication between frontend actions and backend data.
- **Third-party Integrations:**
  - **Google OAuth** for user login.
  - **SMTP Email Services** to send notifications.
  - **Video Conferencing APIs** (like Twilio, Jitsi, or Daily.co) for embedding real-time interview links.
  - (Optional) **AI services** for transcribing, analyzing, or summarizing interview data.

This modular and scalable architecture ensures the platform remains flexible, easy to maintain, and capable of integrating future features like analytics or recruiter insights.






# SETUP & INSTALLATION

---

This section provides a step-by-step guide to help developers set up and run the **Online Job Recruitment Platform** on their local machine for development or testing. It includes prerequisites, installation commands, and environment configuration

## Prerequisites

Before setting up the project, make sure the following tools are installed on your system:

-  **Python** (version 3.8 or above)
-  **pip** (Python package manager)
-  **PostgreSQL** (or SQLite for local testing)
-  **Git** (to clone the repository)
-  **Virtualenv** (for environment management)

## Installation Steps

### 1. Clone the Repository

bash

```
git clone https://github.com/your-username/online-job-recruitment.git
cd online-job-recruitment
```

### 2. Create and Activate a Virtual Environment

bash

```
python -m venv venv
venv\Scripts\activate    # On Windows
```

```
# or  
source venv/bin/activate # On macOS/Linux
```

### 3. Install Python Dependencies

bash

```
pip install -r requirements.txt
```

### Environment Variables Setup

Create a `.env` file in your project root and add the following:

env

```
DEBUG=True  
SECRET_KEY=your-django-secret-key  
DATABASE_NAME=your_db_name  
DATABASE_USER=your_db_user  
DATABASE_PASSWORD=your_db_password  
DATABASE_HOST=localhost  
DATABASE_PORT=5432  
  
EMAIL_HOST=smtp.gmail.com  
EMAIL_PORT=587  
EMAIL_HOST_USER=your-email@gmail.com  
EMAIL_HOST_PASSWORD=your-app-password  
  
GOOGLE_CLIENT_ID=your-google-client-id  
GOOGLE_CLIENT_SECRET=your-google-client-secret
```

 **Tip:** Add `.env` to your `.gitignore` to protect sensitive data.

### Database Setup

Run the following commands to set up the database schema:

```
bash
```

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Create a superuser for accessing the admin dashboard:

```
bash
```

```
python manage.py createsuperuser
```

### **Run the Development Server**

Start the server with:

```
bash
```

```
python manage.py runserver
```

Then open your browser and go to:

```
cpp
```

```
http://127.0.0.1:8000/
```

# USER GUIDE


---

This guide helps **job seekers** understand how to use each feature of the Online Job Recruitment Platform—from registering an account to attending interviews.

## 1. Register or Login

To start using the platform:

- Go to the website's **home page**.
- Click on “**Register**” if you're a new user.
- You can sign up using your **email and password**, or simply log in using your **Google account** for faster access.
- After login, you'll be taken to your personal **dashboard**, where you can manage your applications and profile.

 *Tip: Always remember your login details or use Google sign-in for ease.*

## 2. Upload Your Resume

Once you're logged in:


- Click on “**Profile**” or go to your **dashboard**.
- You'll see an option to **upload your resume**.
- Click **Browse** or **Choose File**, select your **PDF or DOCX** resume, and click **Upload**.
- This file is securely stored and will automatically be attached to any job you apply for.

 *Tip: Make sure your resume is updated and saved in PDF or Word format before uploading.*

### 3. Search & Apply for Jobs

To find jobs:


- Visit the “**Job Listings**” or “**All Jobs**” section from the menu.
- You can use the **search bar** to type a job title (like “Web Developer”) or use **filters** like location or category to narrow your results.
- Click on a job title to read the full description.
- If you're interested and have uploaded your resume, just click the “**Apply**” button—your application will be submitted instantly.

 *Tip: Apply only to jobs that match your skills and experience for better chances of selection.*

### 4. Check Application Status

After applying:

- Go to your **dashboard** and click “**My Applications**”.
- You'll see a list of jobs you've applied to and the **status** of each application, like:
  - **Pending** – waiting for review
  - **Shortlisted** – you've been selected for the next step
  - **Rejected** – unfortunately, not selected
  - **Interview Scheduled** – you're invited for an interview


 *Tip: Check your dashboard regularly or enable email notifications to stay updated.*

### 5. Join Interview via URL

If you're **shortlisted**:

- The recruiter will share a **secure video/audio interview link**.

- You'll see a notification on your dashboard and an email with the **interview link and time**.
- Just click the link at the scheduled time—it will open a video call interface (like Zoom or Google Meet).
- The platform may use **Twilio, Jitsi, or Daily.co** for the call, and may also include **AI features** like real-time transcription or auto-recording (if enabled).

 *Tip: Join the interview from a quiet place with a good internet connection and test your microphone/camera before joining.*

# API DOCUMENTATION

---

This section helps developers understand how the frontend and backend of the platform communicate using APIs (Application Programming Interfaces). These APIs allow actions like logging in, posting jobs, applying for jobs, and managing interviews through secure and structured HTTP requests.

## 1. User Login & Register API

- **Purpose:** Allow users (job seekers or recruiters) to sign up or log in.
- **Methods:**
  - `POST /api/register/` → Create a new user account.
  - `POST /api/login/` → Log in and receive a JWT token for authentication.
- **Returns:** JSON response with a success message and token if credentials are valid.
- **Authentication:** Not required to access these endpoints initially, but the token is required for further actions.

## 2. Job CRUD APIs (Create, Read, Update, Delete)

- **Purpose:** Allow recruiters or admins to manage job postings.
- **Endpoints:**
  - `GET /api/jobs/` → List all job postings.
  - `POST /api/jobs/` → Create a new job (admin/recruiter only).
  - `PUT /api/jobs/<job_id>/` → Update job details.



- `DELETE /api/jobs/<job_id>/` → Remove a job post.
- **Authentication:** Token required, only recruiters or superusers have create/edit/delete rights.

### 3. Application Submission API

- **Purpose:** Let job seekers apply to jobs and check application status.
- **Endpoints:**
  - `POST /api/apply/` → Submit an application to a job with resume attachment.
  - `GET /api/my-applications/` → View all jobs the user has applied for.
- **Authentication:** Required (JWT token) for identifying the applicant.

### 4. Fetch Candidate List API

- **Purpose:** Allow recruiters to view who applied to their jobs.
- **Endpoint:**
  - `GET /api/job/<job_id>/applicants/` → Returns list of candidates with resume links and status.
- **Authentication:** Required. Only job owners (recruiters) can access this data.

### 5. Assign/View Interview URL API

- **Purpose:** Enables recruiters to assign interview links to shortlisted candidates, and allows candidates to view their interview URL.
- **Endpoints:**

- `POST /api/interview/create/` → Recruiter assigns an interview URL to a candidate.
- `GET /api/interview/<user_id>/` → Candidate fetches their assigned interview URL.
- **Authentication:** Required for both recruiter and job seeker. Role-based access ensures only authorized users see or assign URLs.

### All API Endpoints Use:

- **Request Format:** JSON (standard key-value format).
- **Authentication:** JWT tokens passed in headers like this:

makefile

`Authorization: Bearer <your_token_here>`

### API Endpoints Overview

Endpoint	Method	Description	Auth Required
<code>/api/register/</code>	POST	Register a new user	✗
<code>/api/login/</code>	POST	Log in and receive a JWT token	✗
<code>/api/jobs/</code>	GET	List all job postings	✓
<code>/api/jobs/&lt;id&gt;/</code>	GET	View a specific job post	✓
<code>/api/jobs/create/</code>	POST	Create a new job post (admin only)	✓ (Admin)
<code>/api/jobs/&lt;id&gt;/update/</code>	PUT	Update a job post (admin only)	✓ (Admin)
<code>/api/jobs/&lt;id&gt;/delete/</code>	DELETE	Delete a job post (admin only)	✓ (Admin)

<code>/api/apply/</code>	POST	Apply to a job	✓
<code>/api/resume/upload/</code>	POST	Upload or update resume	✓
<code>/api/profile/</code>	GET	Get current user's profile info	✓
<code>/api/profile/update/</code>	PUT	Update profile	✓

## Sample Request & Response

### 1. Register

- **POST** `/api/register/`
- **Request:**

```
json
{
  "username": "john_doe",
  "email": "john@example.com",
  "password": "12345678"
}
```

- **Response:**

```
json
{
  "message": "User registered successfully",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50aWZ5In0..."
}
```

### 2. Login

- **POST** `/api/login/`

- **Request:**

json

```
{  
  "email": "john@example.com",  
  "password": "12345678"  
}
```

- **Response:**

json

```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50b3R5IiwiZXhwIjowfQ=="  
}
```

### 3. Get Job Listings

- **GET** `/api/jobs/`

- **Headers:**

`Authorization: Bearer <token>`

- **Response:**

json

```
[  
  {  
    "id": 1,  
    "title": "Frontend Developer",  
  },  
  {  
    "id": 2,  
    "title": "Backend Developer",  
  },  
  {  
    "id": 3,  
    "title": "Fullstack Developer",  
  },  
  {  
    "id": 4,  
    "title": "Data Scientist",  
  },  
  {  
    "id": 5,  
    "title": "Product Manager",  
  },  
  {  
    "id": 6,  
    "title": "Marketing Specialist",  
  },  
  {  
    "id": 7,  
    "title": "Sales Representative",  
  },  
  {  
    "id": 8,  
    "title": "Customer Support",  
  },  
  {  
    "id": 9,  
    "title": "Human Resources",  
  },  
  {  
    "id": 10,  
    "title": "Finance Analyst",  
  },  
  {  
    "id": 11,  
    "title": "Operations Manager",  
  },  
  {  
    "id": 12,  
    "title": "Business Development",  
  },  
  {  
    "id": 13,  
    "title": "Quality Assurance",  
  },  
  {  
    "id": 14,  
    "title": "Systems Administrator",  
  },  
  {  
    "id": 15,  
    "title": "Network Engineer",  
  },  
  {  
    "id": 16,  
    "title": "Database Administrator",  
  },  
  {  
    "id": 17,  
    "title": "Software Engineer",  
  },  
  {  
    "id": 18,  
    "title": "DevOps Engineer",  
  },  
  {  
    "id": 19,  
    "title": "Data Engineer",  
  },  
  {  
    "id": 20,  
    "title": "Machine Learning Engineer",  
  },  
  {  
    "id": 21,  
    "title": "AI Research Scientist",  
  },  
  {  
    "id": 22,  
    "title": "Robotics Engineer",  
  },  
  {  
    "id": 23,  
    "title": "Bioinformatics Scientist",  
  },  
  {  
    "id": 24,  
    "title": "Genetic Engineer",  
  },  
  {  
    "id": 25,  
    "title": "Nanotechnology Researcher",  
  },  
  {  
    "id": 26,  
    "title": "Space Scientist",  
  },  
  {  
    "id": 27,  
    "title": "Environmental Engineer",  
  },  
  {  
    "id": 28,  
    "title": "Civil Engineer",  
  },  
  {  
    "id": 29,  
    "title": "Mechanical Engineer",  
  },  
  {  
    "id": 30,  
    "title": "Electrical Engineer",  
  },  
  {  
    "id": 31,  
    "title": "Chemical Engineer",  
  },  
  {  
    "id": 32,  
    "title": "Aerospace Engineer",  
  },  
  {  
    "id": 33,  
    "title": "Automotive Engineer",  
  },  
  {  
    "id": 34,  
    "title": "Marine Engineer",  
  },  
  {  
    "id": 35,  
    "title": "Agricultural Engineer",  
  },  
  {  
    "id": 36,  
    "title": "Food Scientist",  
  },  
  {  
    "id": 37,  
    "title": "Pharmaceutical Researcher",  
  },  
  {  
    "id": 38,  
    "title": "Biotechnology Researcher",  
  },  
  {  
    "id": 39,  
    "title": "Environmental Scientist",  
  },  
  {  
    "id": 40,  
    "title": "Public Health Specialist",  
  },  
  {  
    "id": 41,  
    "title": "Healthcare Administrator",  
  },  
  {  
    "id": 42,  
    "title": "Medical Researcher",  
  },  
  {  
    "id": 43,  
    "title": "Biomedical Engineer",  
  },  
  {  
    "id": 44,  
    "title": "Healthcare Consultant",  
  },  
  {  
    "id": 45,  
    "title": "Healthcare Analyst",  
  },  
  {  
    "id": 46,  
    "title": "Healthcare Manager",  
  },  
  {  
    "id": 47,  
    "title": "Healthcare Specialist",  
  },  
  {  
    "id": 48,  
    "title": "Healthcare Researcher",  
  },  
  {  
    "id": 49,  
    "title": "Healthcare Scientist",  
  },  
  {  
    "id": 50,  
    "title": "Healthcare Engineer",  
  },  
  {  
    "id": 51,  
    "title": "Healthcare Technician",  
  },  
  {  
    "id": 52,  
    "title": "Healthcare Assistant",  
  },  
  {  
    "id": 53,  
    "title": "Healthcare Support",  
  },  
  {  
    "id": 54,  
    "title": "Healthcare Representative",  
  },  
  {  
    "id": 55,  
    "title": "Healthcare Sales",  
  },  
  {  
    "id": 56,  
    "title": "Healthcare Marketing",  
  },  
  {  
    "id": 57,  
    "title": "Healthcare Development",  
  },  
  {  
    "id": 58,  
    "title": "Healthcare Operations",  
  },  
  {  
    "id": 59,  
    "title": "Healthcare Quality",  
  },  
  {  
    "id": 60,  
    "title": "Healthcare Compliance",  
  },  
  {  
    "id": 61,  
    "title": "Healthcare Security",  
  },  
  {  
    "id": 62,  
    "title": "Healthcare Privacy",  
  },  
  {  
    "id": 63,  
    "title": "Healthcare Ethics",  
  },  
  {  
    "id": 64,  
    "title": "Healthcare Law",  
  },  
  {  
    "id": 65,  
    "title": "Healthcare Policy",  
  },  
  {  
    "id": 66,  
    "title": "Healthcare Regulation",  
  },  
  {  
    "id": 67,  
    "title": "Healthcare Standards",  
  },  
  {  
    "id": 68,  
    "title": "Healthcare Guidelines",  
  },  
  {  
    "id": 69,  
    "title": "Healthcare Best Practices",  
  },  
  {  
    "id": 70,  
    "title": "Healthcare Innovation",  
  },  
  {  
    "id": 71,  
    "title": "Healthcare Research",  
  },  
  {  
    "id": 72,  
    "title": "Healthcare Development",  
  },  
  {  
    "id": 73,  
    "title": "Healthcare Production",  
  },  
  {  
    "id": 74,  
    "title": "Healthcare Distribution",  
  },  
  {  
    "id": 75,  
    "title": "Healthcare Sales",  
  },  
  {  
    "id": 76,  
    "title": "Healthcare Marketing",  
  },  
  {  
    "id": 77,  
    "title": "Healthcare Support",  
  },  
  {  
    "id": 78,  
    "title": "Healthcare Training",  
  },  
  {  
    "id": 79,  
    "title": "Healthcare Education",  
  },  
  {  
    "id": 80,  
    "title": "Healthcare Consulting",  
  },  
  {  
    "id": 81,  
    "title": "Healthcare Research",  
  },  
  {  
    "id": 82,  
    "title": "Healthcare Development",  
  },  
  {  
    "id": 83,  
    "title": "Healthcare Production",  
  },  
  {  
    "id": 84,  
    "title": "Healthcare Distribution",  
  },  
  {  
    "id": 85,  
    "title": "Healthcare Sales",  
  },  
  {  
    "id": 86,  
    "title": "Healthcare Marketing",  
  },  
  {  
    "id": 87,  
    "title": "Healthcare Support",  
  },  
  {  
    "id": 88,  
    "title": "Healthcare Training",  
  },  
  {  
    "id": 89,  
    "title": "Healthcare Education",  
  },  
  {  
    "id": 90,  
    "title": "Healthcare Consulting",  
  },  
  {  
    "id": 91,  
    "title": "Healthcare Research",  
  },  
  {  
    "id": 92,  
    "title": "Healthcare Development",  
  },  
  {  
    "id": 93,  
    "title": "Healthcare Production",  
  },  
  {  
    "id": 94,  
    "title": "Healthcare Distribution",  
  },  
  {  
    "id": 95,  
    "title": "Healthcare Sales",  
  },  
  {  
    "id": 96,  
    "title": "Healthcare Marketing",  
  },  
  {  
    "id": 97,  
    "title": "Healthcare Support",  
  },  
  {  
    "id": 98,  
    "title": "Healthcare Training",  
  },  
  {  
    "id": 99,  
    "title": "Healthcare Education",  
  },  
  {  
    "id": 100,  
    "title": "Healthcare Consulting",  
  },  
]
```

```
    "company": "TechSoft",
    "location": "Remote",
    "description": "React developer needed...",
    "deadline": "2025-07-01"
  },
  ...
]
```

## ✗ Error Messages and Codes

Error	Code	Message
Invalid Credentials	401	"Invalid email or password"
Unauthorized Access	403	"You do not have permission"
Not Found	404	"Requested resource not found"
Bad Request	400	"Invalid input or missing parameters"
Server Error	500	"An unexpected error occurred"

## 🔧 API Testing Tools

To simplify testing and exploration of the APIs:

- **Swagger UI:** Auto-generates interactive API documentation based on your Django DRF setup.
- **Postman:** You can import a collection of all endpoints and test using different users (admin, job seeker, guest).

# DATABASE SCHEMA

---

This section explains how the platform's data is organized in the database using Django models. Each model corresponds to a table that stores relevant information and connects users, jobs, applications, and interviews.



## Main Tables / Models:

### 1. User (CustomUser)

- Inherits from Django's `AbstractUser`.
- Stores login credentials, email, and role (job seeker or recruiter).
- May include additional fields like `is_recruiter` to differentiate user roles.

#### Fields:

- `id` (Primary Key)
- `username`
- `email`
- `password` (hashed)
- `is_recruiter` (Boolean)
- `date_joined`

## 2. Profile

- Linked to `CustomUser` through a One-to-One relationship.
- Stores additional user details such as:
  - `resume` (PDF/DOCX file upload)
  - `contact number`, `bio`, `location`, etc.
- Allows easy separation of core auth data and user-specific info.

### Fields:

- `user` (OneToOne → User)
- `full_name`
- `contact_number`
- `profile_picture`

## 3. Job

Stores job posting data by recruiters or admins.

### Fields:

- `id` (Primary Key)
- `title`
- `company_name`
- `location`
- `description`
- `experience_required`
- `salary_range`

- `posted_by` (ForeignKey → User)
- `deadline`
- `date_posted`

### Relationships:

One-to-Many: One recruiter can post many jobs.

## 4. Application

Stores job applications submitted by job seekers.

### Fields:

- `id` (Primary Key)
- `applicant` (ForeignKey → User)
- `job` (ForeignKey → Job)
- `resume` (ForeignKey → Resume)
- `cover_letter` (optional TextField)
- `status` (Applied / Reviewed / Rejected / Hired)
- `applied_on`

## 5. Interview

- Stores scheduled interview information.
- Fields include:
  - `candidate` (ForeignKey to User)
  - `recruiter` (ForeignKey to User)
  - `job` (ForeignKey to Job)
  - `interview_url` (string/link to video room)
  - `scheduled_time` (datetime)
- Only shortlisted candidates are linked to interviews.



This schema ensures logical relationships and efficient data access, allowing seamless interaction between users, jobs, applications, and AI-driven interview scheduling.

# SECURITY GUIDELINES

---

To protect the application, its users, and sensitive data by implementing standard security practices.

## 1. JWT & OAuth

- **JWT (JSON Web Tokens)** are used to securely log in users and verify their identity with every API request.
- Once a user logs in, a signed token is issued and stored on the client side (e.g., in local storage or cookies).
- **Google OAuth 2.0** lets users log in with their Google account using secure authorization flow.

## 2. Role-Based Access

- Users are assigned roles like **Job Seeker**, **Recruiter**, or **Admin**.
- The system uses these roles to control access:
  - Recruiters can create and manage job postings.
  - Job seekers can only view and apply.
  - Admins can access all data and manage users.

## 3. CSRF Protection

- CSRF (Cross-Site Request Forgery) protection is enabled by Django's middleware.
- Every form includes a CSRF token to prevent unauthorized form submissions from external websites.

#### 4. Resume Validation

- Resume uploads are validated for format and size.
- Only `.pdf` and `.docx` formats are allowed, with a reasonable file size limit to prevent abuse.

#### 5. Password Hashing

- Passwords are never stored as plain text.
- Django automatically hashes passwords using secure algorithms like PBKDF2 before saving them in the database.

# TESTING DOCUMENTATION

---

Testing is a critical part of the development lifecycle that ensures all components of the **Online Job Recruitment Platform** function as intended. This section outlines the types of tests performed, how they are written and executed, and how to verify the expected outcomes.

## 1. Unit Tests

- These are tests for individual components like models, forms, and views.
- Examples:
  - Checking that a **Job** or **User** model saves correctly.
  - Ensuring form validation works for resume uploads and job applications.
  - Testing view responses (e.g., homepage loads, form errors appear correctly).
- **Tool used:** Django's built-in **TestCase**.

## 2. Integration Tests

- These check that different parts of the app work together.
- Examples:
  - User registration → login → resume upload → job application → application status update.
  - Recruiter login → job posting → application received.
- Confirms that the entire flow behaves as expected across multiple modules.

### 👁️ 3. Manual Testing (for Interview Feature)

- Focused testing for the video/audio interview functionality.
- Steps include:
  - Assigning an interview URL as a recruiter.
  - Checking if the candidate sees the URL on their dashboard.
  - Verifying the video/audio room opens properly (Jitsi, Twilio, etc.).
  - Ensuring access is restricted to selected candidates only.

### 📋 Sample Test Cases

Feature	Test Case	Expected Outcome
User Registration	Valid form data	User is created and redirected to dashboard
Login	Invalid password	Error message is shown
Job Posting	Recruiter creates new job	Job appears in listing immediately
Resume Upload	Upload PDF	Resume saved and linked to profile

Apply to Job	Seeker clicks apply with resume uploaded	Application saved and visible to recruiter
Permissions	Seeker tries to post a job	Access denied / redirected

## How to Run Tests

To run all tests using Django's test runner:

bash

```
python manage.py test
```

## Tools Used

- **Django Test Framework** – for unit and integration testing.
- **Postman** – for manual API testing and verification.
- **Browser Developer Tools** – for UI testing and inspection.

# DEPLOYMENT GUIDE

---

This section provides a step-by-step explanation of how to deploy the Online Job Recruitment Platform to a live server so that users can access it online. It includes using services like Render or Heroku, setting up environment variables, configuring static/media files, and enabling a secure HTTPS domain.

## 1. Hosting (Render or Heroku)

- Use **Render** or **Heroku** to host your Django project.
- Connect your GitHub repo to the platform

Set the build and start commands:

```
bash
```

```
pip install -r requirements.txt
gunicorn projectname.wsgi:application
```

## 2. Environment Variables Setup

Store sensitive info in environment settings (or `.env` file locally):

```
env
```

```
SECRET_KEY=your-secret-key
DEBUG=False
DATABASE_URL=your-database-url
GOOGLE_CLIENT_ID=your-google-client-id
GOOGLE_CLIENT_SECRET=your-google-client-secret
EMAIL_HOST_USER=your-email
EMAIL_HOST_PASSWORD=your-app-password
```

---

### 3. Static and Media Files Configuration

- Use **Whitenoise** for serving static files.

In `settings.py`, set:

python

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Before deploying:

bash

CopyEdit

```
python manage.py collectstatic
```

---

### 4. Domain & HTTPS Setup

- Connect a custom domain (like `www.jobportal.com`) through Render/Heroku.
- Update DNS records in your domain registrar.
- Enable **SSL (HTTPS)** — platforms like Render provide free SSL via Let's Encrypt.



# TROUBLESHOOTING /FAQs

---

To help users and developers identify and solve common issues that may arise while using the platform.

## 1. Login/Email Errors

- **Problem:** Can't log in or didn't receive an email.
- **Solutions:**
  - Make sure your email and password are correct.
  - Check if you already have an account.
  - Look in your spam folder for verification or notification emails.
  - If using Google login, ensure browser popups are enabled.

## 2. Resume Upload Issues

- **Problem:** Upload fails or file is rejected.
- **Solutions:**
  - Only upload **.pdf** or **.docx** files.
  - Keep file size under 2MB.
  - Remove special characters from the file name.
  - Try using another browser if upload keeps failing.

### 3. Video Link Not Opening

- **Problem:** Clicking the interview link doesn't work or shows an error.
- **Solutions:**
  - Make sure you're logged in as the selected candidate.
  - Check if the link has expired or is broken.
  - Use a supported browser (Chrome/Firefox) and ensure internet access.
  - Contact the recruiter if the problem continues.

# CHANGE LOG / VERSION HISTORY

---

To track the progress and key updates made to the Online Job Recruitment Platform throughout its development.

## Version Breakdown:

- **v1.0 – Base Job Features**  
Basic platform with user login, job listings, job application functionality, and admin panel.
- **v1.1 – Resume Upload + Google Login**  
Added support for uploading resumes (PDF/DOCX) and enabled Google OAuth login for faster registration.
- **v1.2 – Email Notifications**  
System now sends automatic emails to users on status updates and key activities like application submission or interview scheduling.
- **v1.3 – Interview URL Sharing**  
Recruiters can generate and assign private video interview links to selected candidates.
- **v1.4 – AI-Powered Video Interview Integration**  
Introduced integration with third-party video tools (Twilio, Jitsi, etc.) and added optional AI features like transcription, candidate scoring, or summary generation.

# LICENSE





---

To let users and developers know how they can legally use, modify, and share the source code of this project.

## License Type: MIT License

The Online Job Recruitment Platform is released under the **MIT License**, which is a very open and permissive software license.

### What You Can Do:

-  Use the code for personal, academic, or commercial purposes.
-  Modify the code to suit your needs.
-  Share or distribute your version freely.
-  Reuse parts of the code in your own projects.

### What You Must Do:

- Include a copy of the original MIT License when sharing the project.
- Credit the original authors of the code.

### What the License Doesn't Cover:

- There is **no warranty**. The software is provided “as is.”
- If the project causes problems, the authors **are not liable** for any issues.

This license promotes open-source contributions and allows both students and businesses to build upon the project freely.

# CONTRIBUTING GUIDE

---

## **Purpose:**

To guide new developers who want to contribute to the Online Job Recruitment Platform by following a clean and collaborative process.

## **1. Fork the Repository**

- Go to the GitHub repository.
- Click the **Fork** button (top right).
- This creates a copy of the project under your GitHub account.

## **2. Create a Feature Branch**

On your local machine:

bash

```
git clone https://github.com/your-username/online-job-recruitment.git
```

```
cd online-job-recruitment
```

```
git checkout -b feature/your-feature-name
```

---

## **3. Follow Commit Format**

Write clear, meaningful commit messages:

bash

```
git add .
```

```
git commit -m "Add: feature to filter jobs by location"
```

Use consistent prefixes like:

- **Add:** (for new features)
- **Fix:** (for bug fixes)
- **Update:** (for improvements)
- **Remove:** (for code cleanups)

#### 4. Submit a Pull Request (PR)

Push your branch to GitHub:

bash

```
git push origin feature/your-feature-name
```

- Visit your GitHub repo
- Click **“Compare & pull request”**
- Fill out the PR form and submit

After review and approval, your changes will be merged into the main project.

## 5. Testing Your Changes

Before submitting a PR:

Run the Django test suite:

```
bash  
python manage.py test
```

- Manually test the key functionality you've worked on.

## Notes

- Be respectful and patient during the review process.
- Always pull the latest changes from `main` before starting new work.
- For large changes, consider opening an issue to discuss first.

# CONCLUSION

---

The **Online Job Recruitment Platform** is a complete, user-friendly solution that simplifies and modernizes the hiring process for both job seekers and recruiters. It provides a smooth experience from job search to final interviews, all through a single web application.

**For Job Seekers**, it offers:

- Easy account registration (email or Google)
- Resume upload and job applications
- Application tracking and status updates
- Secure access to AI-powered video interview links

**For Recruiters**, it enables:

- Posting jobs and managing applicants
- Reviewing resumes and assigning interviews
- Sharing private video links with selected candidates
- Full access to admin dashboard features

---

Built with Django, PostgreSQL, and REST APIs, and enhanced with tools like Twilio and Google OAuth, the platform is secure, scalable, and production-ready. It combines modern web development practices with AI tools to provide an advanced recruitment experience.

This project showcases how digital technologies can make recruitment faster, smarter, and more accessible to everyone.



