Career Gap Job App Documentation

Career Gap Job App Documentation

Career Gap Job App Documentation

Overview

A job platform designed specially for individuals with career gaps. The app helps align employer

skill requirements with job seekers current skills, offering upskilling guidance, emotional support,

and tailored job opportunities.

Goals

-Provide job opportunities to people with career gaps.

-Help candidates groom their skills through short courses and tasks.

-Educate employers to value skill over resume gaps.

-Foster a supportive community for job seekers.

User Roles

1. Job Seeker

-Create and manage profile.

-Take skill tests.

-View and apply for jobs.

-Receive grooming suggestions.

-Join support community.

2. Employer

-Post job requirements.

-Indicate if job is gap-friendly.

-Review skill-based candidate profiles.

-Offer interview opportunities.

3. Admin (Optional)

-Manage users and reports.

-Curate training resources.

Core Features

Onboarding

-Choose role (Job Seeker / Employer).

-Basic registration/login (Firebase Auth).

Job Seeker Features

-Skill-based profile creation.

-Career gap information field.

-Upload resume/video pitch.

-Take micro skill assessments.

-View job listings with skill match.

-Get recommended grooming paths.

-Join community forum.

Employer Features

-Job posting with skill requirements.

-Mark job as gap-friendly.

-View groomed, skill-ready profiles.

-Shortlist based on test badges and video pitch.

Tech Stack

Frontend:

-React.js

(User interfaces for job seekers & employers)

Backend:

-Django

(Handles APIs, business logic, authentication)

Database:

-MySQL

(Stores user profiles, job listings, skills, etc.)

Hosting/Cloud:

-AWS EC2

(Host your Django backend)

-AWS RDS

(Managed MySQL)

-S3

(For storing resumes, profile images, videos)

Dev Tools You Can Use

-Jira

: For planning and task tracking

-Postman

: For testing Django APIs

-GitHub

: For version control and collaboration

PHASE 1: PLANNING & REQUIREMENTS

Week 1: Define the Scope

-Finalize key features:

-Job seeker profile

-Employer job posting

-Skill-based job matching

-Upskilling recommendations

-Community section (optional)

-Decide MVP: Minimal features to launch version 1

Deliverables:

-Feature list

-User roles +

flows

-Database design (ER Diagram)

-UI wireframes (basic layout for each screen)

PHASE 2: BACKEND SETUP (Django + MySQL)

Week 2-3: Build APIs

-Set up Django project

-Create MySQL database

-Build models:

-User (Job Seeker, Employer)

-Profile, Skills, JobPost, Application, SkillTest

-Build REST APIs (using Django REST Framework):

-Register/Login

-Post job

-Apply to job

-Match users to job by skill

-Get skill suggestions

PHASE 3: FRONTEND (React.js)

Week 4-5: Build UI Pages

-Sign up/Login page

-Dashboard for job seekers

-Profile creation/edit page

-Job listing +filter by skills

-Apply to job + status

-Grooming suggestions

PHASE 4: INTEGRATION + TESTING

Week 6:

-Connect React.js frontend to Django APIs

-Test data

flow: Register

-Create Profile

-Match Job

-Apply

-Fix bugs, handle errors, authentication logic

PHASE 5: DEPLOYMENT

Week 7:

-Host Django app on

AWS EC2

-Use

AWS RDS

for MySQL

-Host frontend on

Ve r c e l

or

AWS S3 + CloudFront

-Add domain name

Bonus (Optional but Powerful)

-Email verification

(Django + AWS SES)

-Admin dashboard

(Django admin)

-Basic analytics

(Track how many apply, success rate)