LEBANESE INTERNATIONAL UNIVERSITY

HIRE SYNC

Senior Project

By

Mohammad Zbib

Submitted to the School of Arts & Science of the

Lebanese International University Beirut, Lebanon

In part of fulfillment of the requirements for the degree of

BACHERLOR OF SCIENCE IN

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Supervised by: Dr. Issa Kamar

Spring 2024-2025

Chapter 1

In this chapter, we'll outline the main problem in current recruitment workflows, how our project addresses it, our goals, scope, tech stack, and the technical boundaries we worked within. Each section explains our approach in detail.

1.1 – Introduction:

Hiring is broken.

Recruiters spend hours reviewing resumes. Candidates send applications into black holes. And most systems don't actually help—they just digitize old processes.

In a world where speed and accuracy matter, traditional recruitment methods can't keep up. That's why we built an AI-powered recruitment SaaS platform—to help companies hire smarter, not harder.

Our system improves the hiring experience for recruiters and candidates. No more account creation. No more manual sorting. Just clean, fast, data-driven hiring.

1.2 – Objective:

Our goal is to create a recruitment system that:

- Automatically filters and ranks applicants using AI.
- Simplifies candidate submissions—no accounts, just resumes or LinkedIn profiles.
- Gives recruiters a clean dashboard with all the info they need in one place.
- Handles the full cycle—job posting, CV import, filtering, shortlisting, interviews, and final decisions.
- Saves time, reduces hiring costs, and makes life easier for everyone involved.

In short, we want to replace bloated, outdated hiring portals with something fast, smart, and efficient.

1.3 – Scope:

Our system is built for:

- Recruitment teams, hiring managers, and HR departments.
- Startups, mid-sized firms, and large organizations.
- Any company looking to streamline and modernize their hiring pipeline.

It can be used to:

- Post jobs publicly or privately

- Import CVs from LinkedIn, local files, or spreadsheets
- Use AI (Gemma3-4B) to instantly match candidates to job requirements
- Schedule interviews and manage candidate stages
- View performance reports and applicant analytics

It's web-based, available 24/7, and designed to work across industries—from tech to healthcare to education.

1.4 – Technology Constraints:

Frontend:

- TypeScript with React.js
- NestJs (NodeJs Framework)

Backend Services:

- Spring Boot (Java)
- Python for AI integration

AI:

- Gemma-3B model for smart candidate matching

Database:

- PostgreSQL for persistent storage

Infrastructure:

- Docker for containerization
- Firebase (Storage Bucket) to manage uploaded CVs as PDFs
- Azure (or alternative) for deployment and authentication

Limitations:

- AI inference must be optimized for cost and performance
- Proper integration needed between multi-language services (Node, Java, Python)
- Scaling real-time search and filtering required efficient indexing

1.5 – Problem:

Recruitment is full of pain points:

- Recruiters waste time reviewing irrelevant CVs manually
- Job seekers are forced to register and fill out long forms repeatedly
- Many platforms are expensive, slow, or too complicated
- Valuable candidates get ignored due to poor filtering
- Managers lack real-time insights into hiring performance

This leads to missed talent, burned time, and frustrated users on both ends.

1.6 – Solution:

Our project solves all these issues:

- AI-powered ranking using the Gemma-3B model gives recruiters the top matches instantly
- No sign-up for applicants—just upload and go
- Recruiters can import candidates in bulk from LinkedIn exports or spreadsheets
- Modern NestJS and Spring Boot APIs handle all core services efficiently
- Resumes are stored securely using Firebase, and Dockerized microservices make the platform scalable
- Built-in interview tracking and performance reports help HR make informed choices

The result? Faster hiring. Better matches. Happier teams.

Chapter 2 – Requirements & Use Cases

This chapter breaks down what our system must do (functional), how it should behave (non-functional), and maps those needs into clear UML use cases and scenarios.

2.1 – Functional and Conceptual Study

2.1.1 – Functional Requirements

Here’s what our recruitment platform can do:

1. Submit Application:
    - Candidates don’t need to sign up.
    - They can upload a resume or paste a LinkedIn profile directly.
2. AI-Powered Ranking:

- Every submission is instantly matched to job roles using our Gemma-3B model.
- Recruiters see top-ranked candidates first.

3. Browse Applicants:
    - Recruiters can filter, search, and sort through applicants easily.
    - Filters include experience, education, skills, and AI-matching score.
4. Job Posting Management:
    - Admins can create, update, and close job postings.
    - Job visibility can be public or internal.
5. Candidate Tracking:
    - View candidate history, resume, status (shortlisted, interviewed, rejected), and notes.
    - Move candidates across stages (New → Shortlisted → Interview → Offer → Hired).
6. Interview Scheduling:
    - Recruiters can set up interview slots.
    - Calendar integration is available.
7. Feedback & Comments:
    - Team members can leave notes or evaluations per candidate.
8. Favorites / Shortlist:
    - Bookmark candidates for later review.
9. Contact & Support:
    - Users can reach out to the admin team for help or suggestions.

2.1.2 – Non-Functional Requirements

Security:

- Authentication: Recruiters and admins must log in using secure credentials.

- Role-Based Access: Only admins can edit jobs, delete candidates, or view analytics.

Performance:

- Login: Should respond within 3 seconds max.
- Page Load: Search results and filters return results in under 1 second.
- Concurrency: System supports at least 1000 active users at once.
- Connectivity: If offline, a fallback page notifies users of the issue.

Maintainability:

- Error Logging: Every error is logged for dev review.
- Modular Codebase: Built with separate services for easy updates.

Reliability:

- Uptime: Available 24/7 with auto-scaling support.
- Recovery: Backup of data runs daily. Firebase stores all submitted PDFs.

2.2 – UML Use Case Diagram & Scenarios

2.2.1 – Use Case Diagram Overview

Actors:

- Candidate: External user applying for jobs.
- Recruiter: Logged-in user reviewing and managing applications.
- Admin: Internal staff managing the entire system.

Candidate Use Cases:

- Submit Resume (main)
- View Job Listings
- Paste LinkedIn Profile
- Upload Resume (PDF or Docx)
- No Signup Required

Recruiter Use Cases:

- Login
- View Dashboard
- Browse Candidates

Use AI Matching

- Shortlist / Reject
- Schedule Interview
- Add Comments

Extended Use Cases:

- Download Resume
- Search with Filters
- Add Candidate to Favorites

Admin Use Cases:

- Manage Users
- Create/Edit Job Posts
- View System Logs
- Respond to Feedback
- Manage AI Matching Rules

Includes:

- Login
- View Reports
- Access Candidate Analytics

Candidate Usecase Flow:

Use Case	Flow Steps
Submit Resume	1. Visit landing page → 2. Click "Apply Now" → 3. Upload resume or paste LinkedIn → 4. Submit → 5. Confirmation screen
View Job Listings	1. Visit jobs page → 2. Browse listings → 3. Click on job card for more info
Upload Resume	1. Choose file → 2. File parsed → 3. User reviews preview → 4. Submit
Paste LinkedIn Profile	1. Paste URL → 2. System fetches HTML → 3. Data extracted → 4. Preview and confirm
Get Matched to	1. Resume parsed → 2. Matching jobs shown → 3. User applies or

Use Case	Flow Steps
Jobs	bookmarks
Apply Without Account	1. Submit resume → 2. Enter email → 3. Track status via link sent to email
Track Applications	1. Visit unique tracking link → 2. View status updates → 3. Optional contact option
View Job Fit Score	1. Resume parsed → 2. Job score shown on each listing → 3. Click for details
Interview Feedback	1. Receive email alert → 2. Login or click link → 3. View recruiter notes or ratings
Upload Video Intro	1. Click "Add Video Intro" → 2. Record or upload → 3. Submit
Edit Application	1. Visit submission page → 2. Click “Edit” → 3. Update info → 4. Re-submit

Recruiter Usecase Flow:

Use Case	Flow Steps
Post a Job	1. Log in → 2. Go to "Post Job" → 3. Fill form or use smart template → 4. Submit
View Candidate Matches	1. Open job post → 2. See ranked list of candidates → 3. Click to view profiles
Review Applications	1. Log in → 2. Go to dashboard → 3. Filter/sort candidates → 4. Add notes or shortlist
Schedule Interview	1. Select candidate → 2. Choose available times → 3. Send invite
Message Candidates	1. Go to candidate profile → 2. Click "Message" → 3. Start chat or send email

Use Case	Flow Steps
Evaluate Resume Matches	1. Upload job description → 2. See candidate recommendations
Bulk Message Candidates	1. Select multiple candidates → 2. Compose message → 3. Send
Export Candidate Data	1. Go to "Export" tab → 2. Select fields → 3. Download CSV
Create Custom Filters	1. Open filter builder → 2. Add conditions (skills, experience) → 3. Save and apply
View Application Insights	1. Open job post → 2. Click “Insights” tab → 3. View charts: source, time-to-apply, etc.
Rate Candidate Profiles	1. Open profile → 2. Select score or thumbs up/down → 3. Add internal notes

Admin Usecase Flow:

Use Case	Flow Steps
Approve Job Postings	1. Log in to admin panel → 2. View pending jobs → 3. Approve or reject
Manage Users	1. Access user list → 2. Search/filter → 3. Suspend or edit access
Monitor Platform Usage	1. Open analytics dashboard → 2. View charts, user activity, job stats
Manage Tags/Skills	1. Open taxonomy editor → 2. Add/edit/remove skills or tags
Handle Reported Content	1. Open reports panel → 2. Review flagged content → 3. Take action
Edit Site Content	1. Go to CMS section → 2. Choose content (FAQ, T&Cs) → 3. Edit and publish
Create Team Accounts	1. Open "Teams" → 2. Add new admin/recruiter → 3. Assign roles
Moderate Messages	1. Access messages log → 2. View flagged chats → 3. Block user or delete message
View Error Logs	1. Open system logs → 2. Filter by severity/date → 3. Export if

Use Case	Flow Steps needed
Manage Subscription Plans	1. Go to billing panel → 2. Add or update plans → 3. Set pricing rules
Broadcast Announcement	1. Go to announcements → 2. Compose update → 3. Schedule or send

Chapter 3: System Design

This chapter describes the system design, including the database of our Recruitment AI SaaS, the related tables, and the relationships between them.

3.1 – ERD

This ER Diagram represents the database model of the Recruitment AI SaaS system. It outlines the core entities and the relationships connecting them.

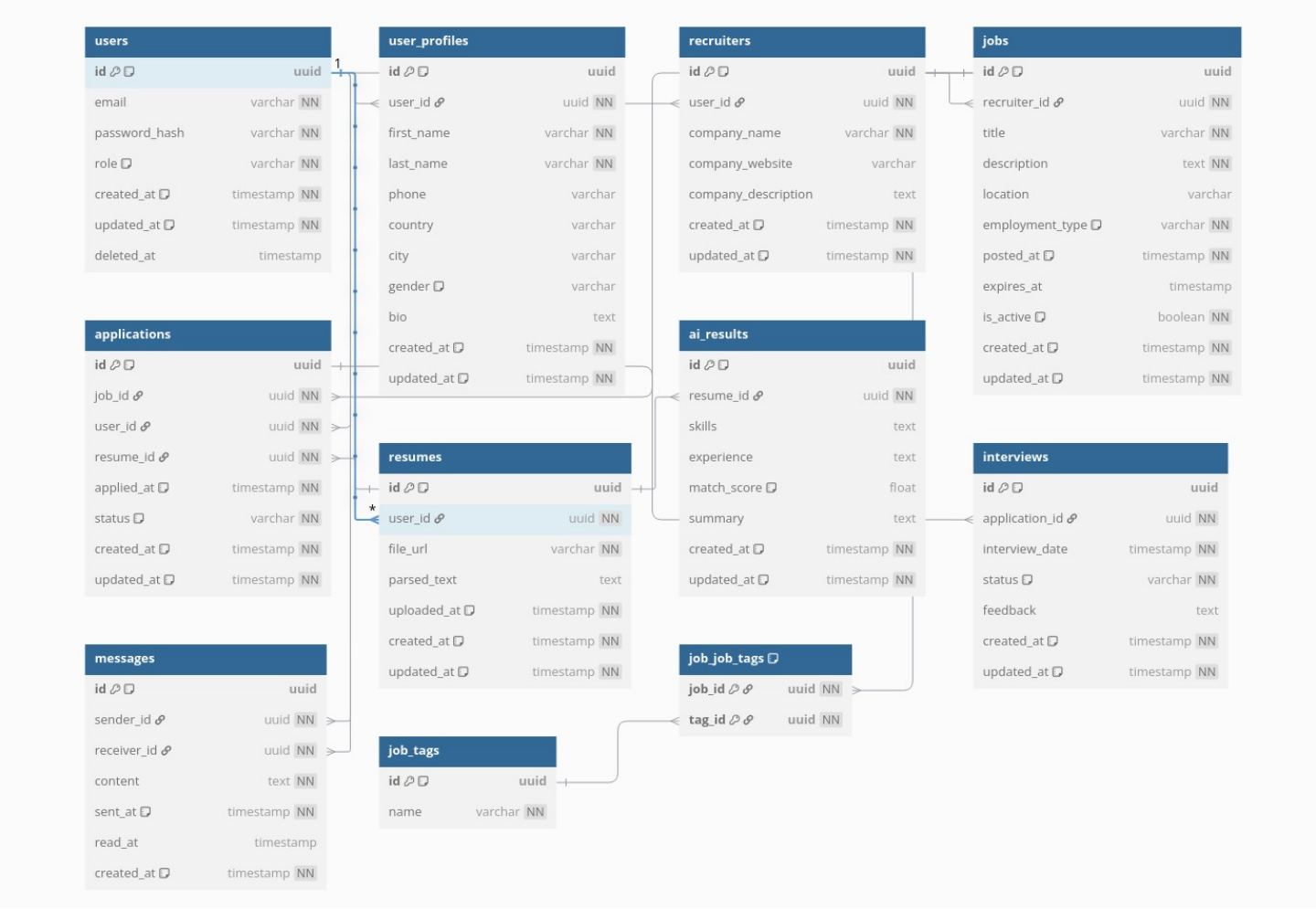
Main entities of the system:

- Users: Represent all system users including candidates and recruiters.
  Attributes: id (PK), email, password_hash, role, created_at, updated_at, deleted_at.
- User Profiles: Detailed info about users.
  Attributes: id (PK), user_id (FK), first_name, last_name, phone, country, city, gender, bio.
- Recruiters: Users managing job postings for companies.
  Attributes: id (PK), user_id (FK), company_name, company_website, company_description.
- Jobs: Job postings created by recruiters.
  Attributes: id (PK), recruiter_id (FK), title, description, location, employment_type, posted_at, expires_at, is_active.
- Applications: Job applications submitted by candidates.
  Attributes: id (PK), job_id (FK), user_id (FK), resume_id (FK), applied_at, status.
- Resumes: Candidate resumes uploaded to the system.
  Attributes: id (PK), user_id (FK), file_url, parsed_text, uploaded_at.
- AI Results: Automated analysis of resumes.
  Attributes: id (PK), resume_id (FK), skills, experience, match_score, summary.

- Interviews: Interview records linked to applications.
  Attributes: id (PK), application_id (FK), interview_date, status, feedback.
- Messages: Communication between users.
  Attributes: id (PK), sender_id (FK), receiver_id (FK), content, sent_at, read_at.
- Job Tags: Tags used to categorize job postings.
  Attributes: id (PK), name.
- Job_Tag Relationships: Many-to-many link between jobs and tags.
  Attributes: job_id (FK), tag_id (FK).

Each entity uses a UUID primary key for uniqueness and relationships are established via foreign keys. Timestamps track creation and updates to support auditing and history.

This design supports role-based users, detailed candidate profiles, recruiter job management, applications with resume linkage, AI-driven resume analysis, interview scheduling and feedback, plus direct messaging between users for seamless communication. The tagging system helps classify and filter job posts by relevant skills or categories.



3.2 – System Architecture

- The architecture of the Recruitment AI SaaS follows a modular and distributed design to ensure scalability, real-time capabilities, and smooth AI integration. The system is composed of multiple services and technologies, each with a specific role.

Components:

- Frontend (Next.js)
  Handles the user interface for recruiters. Communicates with the backend via REST APIs.
- Backend (Spring Boot)
  Core of the system. Exposes APIs, manages business logic, handles database access, and triggers background processing.

- Redis
  Used for caching frequently accessed data and storing temporary states like session tokens.
- RabbitMQ
  Acts as the message broker. Used for asynchronous communication between services like triggering Python pipelines after resume uploads.
- Python Microservice
  Processes resumes using NLP models. Scores candidates, parses resumes, and generates structured data.
- HuggingFace Models
  Embedded in the Python service for AI tasks: resume parsing, keyword extraction, and skill matching.
- PostgreSQL
  Main relational database. Stores all core entities: users, jobs, applications, resumes, interviews, etc.
- Supabase
  Used for handling storage (resume file uploads) and user auth/session management in specific parts of the app.

Flow Summary:

1. Recruiter posts a job via Next.js → API call hits Spring Boot backend.
2. Candidate applies → Resume is uploaded to Supabase, metadata saved in PostgreSQL.
3. RabbitMQ triggers Python AI job → NLP processing using HuggingFace.
4. AI service responds with parsed results → stored in DB.
5. Recruiter sees processed candidate data instantly → thanks to Redis caching.

Let me know if you want this illustrated in Mermaid code as well.

Chapter 4

This chapter outlines the structure of our AI Recruitment SaaS website, focusing on the design and purpose of each page.

Implementation:
4.1 - Introduction:

The website’s design aims for clarity and simplicity. Users should quickly understand what the platform offers and how it can help them, whether they’re recruiters or candidates.

4.2 - Project’s Parts:

- Landing Page:
  The very first page users see. Its role is to welcome visitors and introduce the platform. It explains who we are, what we do, and why the service matters.

  Key features:

  - A clear, concise welcome message
  - Brief overview of the platform’s benefits
  - Information about the company’s mission and values
  - Easy access to sign-up or log-in options

Landing Page:
The welcome page introducing the platform, company mission, and key benefits. Clear calls to action guide users to sign up or log in.

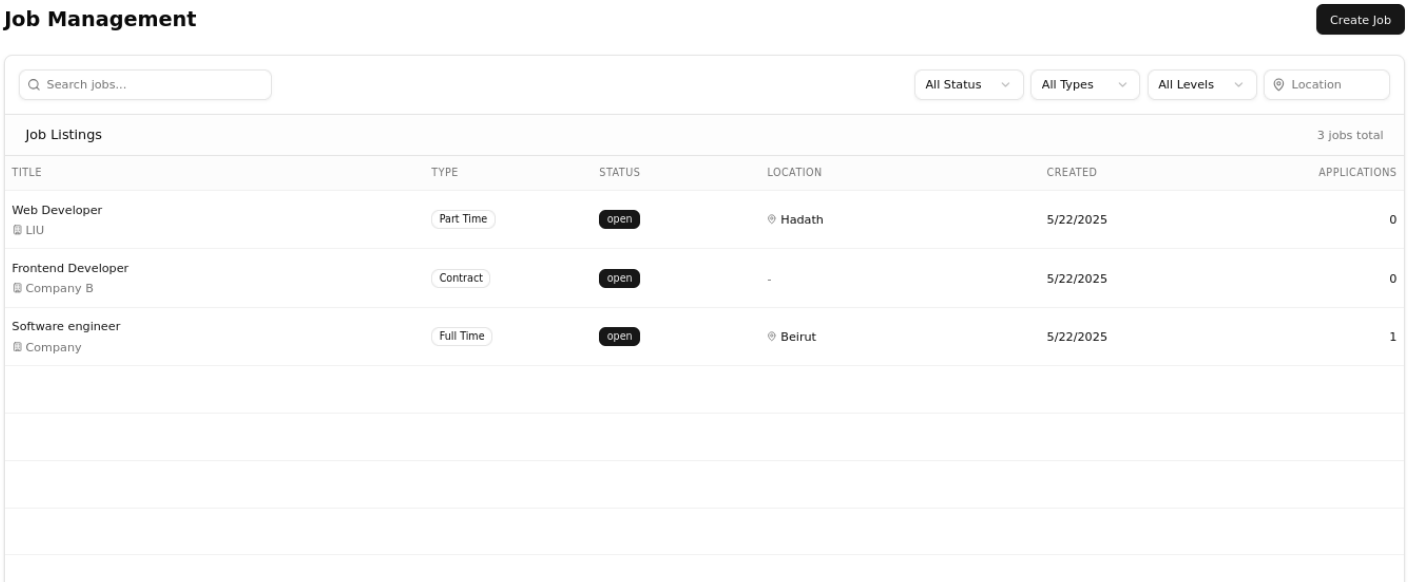
Revolutionizing Recruitment with AI

We've built an advanced AI system that transforms the hiring process by intelligently matching candidates to jobs through sophisticated resume analysis and skill assessment.

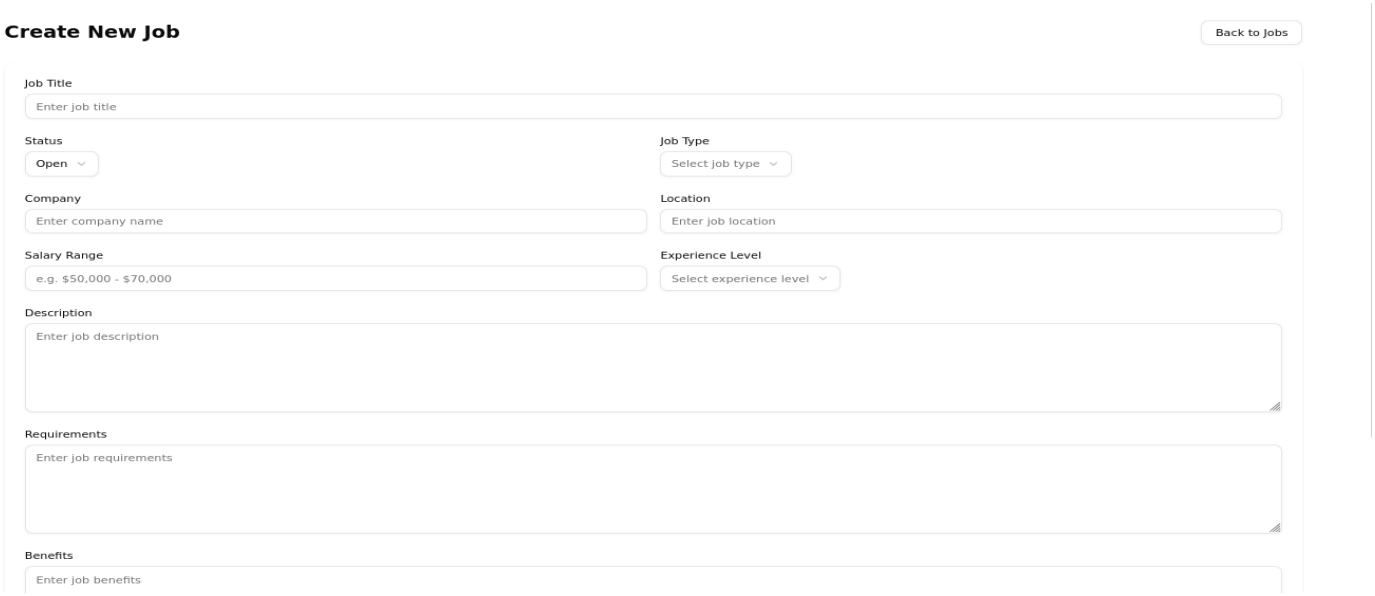
Explore Jobs

Developed by Mohammad Zbib | Final Year Project

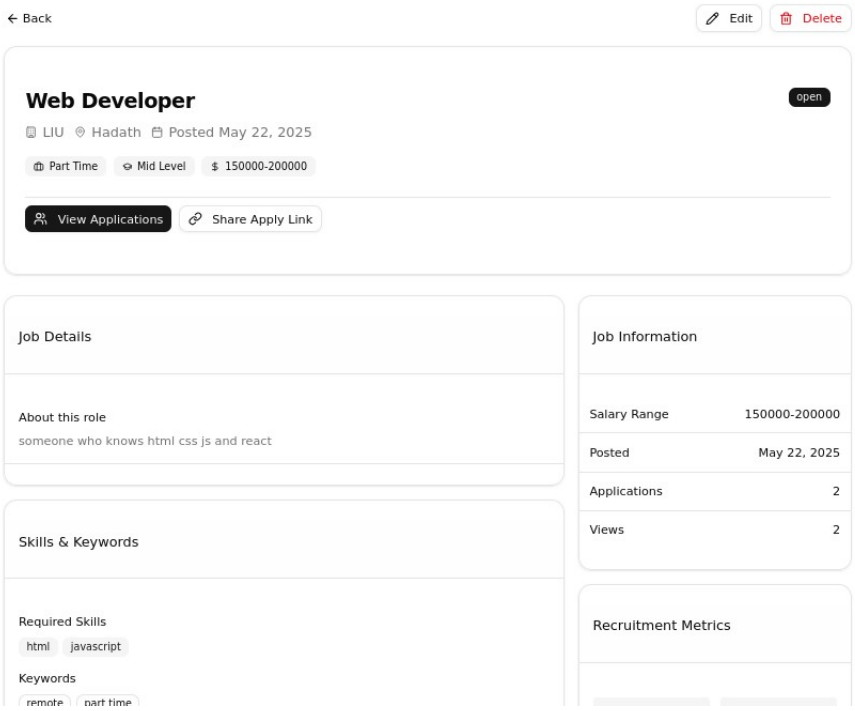
Jobs Page:
Displays all available job listings. Users can browse, filter, and search jobs. Recruiters see their active posts; candidates see all open roles.



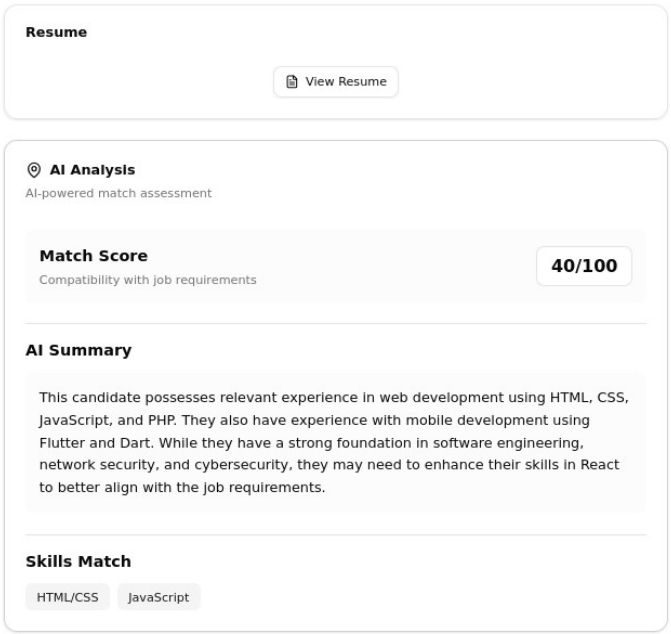
Create Job Page:
Allows recruiters to create new job postings. The form collects job title, description, skills required, location, employment type, and active status.



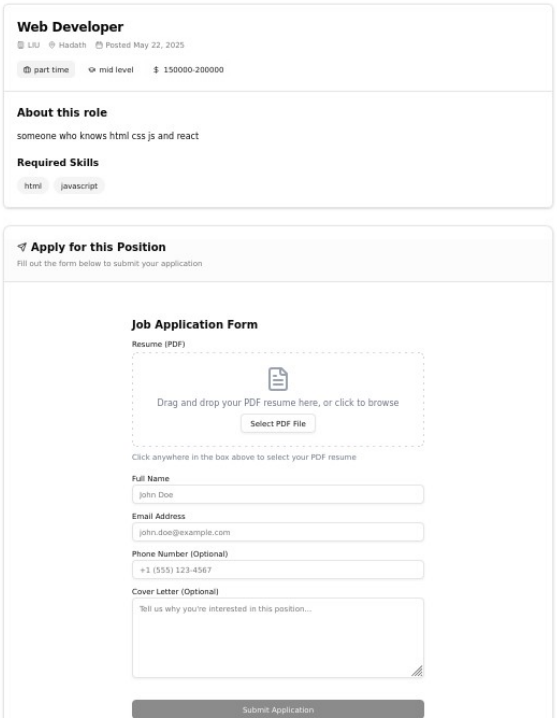
Job Details Page:
Displays full job info. Candidates can review the description and apply here. Recruiters can edit or deactivate the post.



Resume Review and AI Match Page:
Recruiters can view submitted resumes with parsed data and AI-generated match scores. The page highlights candidate skills, experience, and summary for quick evaluation.



Candidate Application Page:
The form where candidates submit their application. Includes resume upload and optional cover letter.



Future Work

The AI Recruitment SaaS platform has a solid foundation but leaves room for growth. Future improvements may include:

- **Advanced AI features:** Enhance resume parsing with deeper natural language understanding and predictive analytics to better match candidates and jobs.
- **Video interviews:** Integrate live or recorded video interview capabilities for seamless remote screening.
- **Collaborative hiring:** Add tools for teams to review applications, comment, and score candidates collectively.
- **Enhanced messaging:** Introduce real-time chat with notifications and message history.
- **Mobile app:** Develop a native app to improve accessibility and engagement for users on the go.
- **Analytics dashboard:** Provide recruiters with insights on hiring trends, application flow, and candidate quality.
- **Multi-language support:** Expand usability for a global audience with language localization.

- **Compliance & security:** Strengthen data protection, privacy features, and regulatory compliance.

Conclusion

This project delivers a streamlined AI-powered recruitment platform aimed at simplifying hiring for recruiters and candidates alike. The system provides essential features like job posting, application tracking, AI resume analysis, and communication tools wrapped in a clean, user-friendly interface.

The architecture supports scalability and future enhancements. By automating key processes and improving candidate-job matching, the platform reduces manual workload and speeds up recruitment cycles.

Overall, the solution meets the needs of modern hiring workflows while leaving a clear path for future growth and innovation.