RECRUIT IQ

A PROJECT REPORT

BY

LAKSHYA GARG (E23CSEU1841)

SATYA PRAKASH (E23CSEU1845)

A blue and red text on a black background

Description automatically generated

SUBMITTED TO

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY, BENNETT UNIVERSITY

GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

April 2025

# DECLARATION

I/We hereby declare that the work which is being presented in the report entitled “RecruitIQ”, is an authentic record of my/our own work carried out during the period from JAN, 2025 to April, 2025 at School of Computer Science and Engineering and Technology, Bennett University Greater Noida.

The matters and the results presented in this report has not been submitted by me/us for the award of any other degree elsewhere.

Lakshya Garg

(E23CSEU1841)

Satya Prakash

(E23CSEU1845)

# ACKNOWLEDGEMENT

I/We would like to take this opportunity to express my/our deepest gratitude to my/our mentor, **Dr. Nitin Arvind Shelke** for guiding, supporting, and helping me/us in every possible way. I/we was/were extremely fortunate to have him as my/our mentor as he provided insightful solutions to problems faced by me/us thus contributing immensely towards the completion of this capstone project. I/We would also like to express my/our deepest gratitude to VC, DEAN, HOD, faculty members and friends who helped me/us in successful completion of this capstone project.

Lakshya Garg

(E23CSEU1841)

Satya Prakash

(E23CSEU1845)

**TABLE OF CONTENTS**

[LIST OF TABLES vi](#_Toc85881407)

[LIST OF FIGURES vii](#_Toc85881408)

[LIST OF ABBREVIATIONS viii](#_Toc85881409)

[ABSTRACT ix](#_Toc85881410)

[1. INTRODUCTION 1](#_Toc85881411)

[1.1. Problem Statement 1](#_Toc85881412)

[2. Background Research 2](#_Toc85881413)

[2.1. Proposed System 2](#_Toc85881414)

[2.2. Goals and Objectives 2](#_Toc85881415)

[3. Project Planning 3](#_Toc85881416)

[3.1. Project Lifecycle 3](#_Toc85881417)

[3.2. Project Setup 3](#_Toc85881418)

[3.3. Stakeholders 4](#_Toc85881419)

[3.4. Project Resources 4](#_Toc85881420)

[3.5. Assumptions 5](#_Toc85881421)

[4. Project Tracking 5](#_Toc85881422)

[4.1. Tracking 5](#_Toc85881423)

[4.2. Communication Plan 6](#_Toc85881424)

[4.3. Deliverables 7](#_Toc85881425)

[5. SYSTEM ANALYSIS AND DESIGN 8](#_Toc85881426)

[5.1. Overall Description 8](#_Toc85881427)

[5.2. Users and Roles 8](#_Toc85881428)

[5.3. Design diagrams/ UML diagrams/ Flow Charts/ E-R diagrams 9](#_Toc85881429)

[5.3.1. Use Case Diagrams 9](#_Toc85881430)

[5.3.2. Class Diagram 10](#_Toc85881431)

[5.3.3. Activity Diagrams 11](#_Toc85881432)

[5.3.4. Sequence Diagram 12](#_Toc85881433)

[5.3.5. Data Architecture 13](#_Toc85881434)

[6. User Interface 14](#_Toc85881435)

[6.1. UI Description 14](#_Toc85881436)

[6.2. UI Mockup 14](#_Toc85881437)

[7. Algorithms/Pseudo Code 15](#_Toc85881438)

[8. Project Closure 16](#_Toc85881439)

[8.1. Goals / Vision 16](#_Toc85881440)

[8.2. Delivered Solution 16](#_Toc85881441)

[8.3. Remaining Work 16](#_Toc85881442)

[REFERENCES 17](#_Toc85881443)

LIST OF TABLES

Table Page

[Table 1: Goal and Objectives 2](#_Toc20994042)

[Table 2: Projec Setup Decisions 4](#_Toc20994042)

[Table 3: Stakeholders 5](#_Toc20994043)

[Table 4: Project Resources 5](#_Toc20994044)

[Table 5: Assumptions 6](#_Toc20994045)

[Table 6: Tracking 7](#_Toc20994046)

[Table 7: Regularly Scheduled Meetings 7](#_Toc20994047)

[Table 8: Information To Be Shared Within Our Group 8](#_Toc20994048)

[Table 9: Information To Be Provided To Other Groups 8](#_Toc20994049)

[Table 10: Information Needed From Other Groups 8](#_Toc20994050)

[Table 11: Deliverables 9](#_Toc20994051)

[Table 12: Users and Roles 10](#_Toc20994052)

LIST OF FIGURES

Figure Page

[Figure 1: Use-case diagram 11](#_Toc20994053)

[Figure 2: Class Diagram 12](#_Toc20994054)

[Figure 3: Activity Diagram 13](#_Toc20994055)

[Figure 4: Sequence Diagram 14](#_Toc20994056)

[Figure 5: UI Mockup 16](#_Toc20994057)

LIST OF ABBREVIATIONS

Abbreviation Explanation of the Abbreviation

|  |  |  |
| --- | --- | --- |
| AI | | Artificial Intelligence |
| NLP | Natural Language Processing | |
| NER | Named Entity Recognition | |
| UI | User Interface | |
| HR | Human Resources | |
| API | Application Programming Interface | |
| CSV | Comma-Separated Values | |
| BCA | Bachelor of Computer Applications | |
| MBA | Master of Business Administration | |
| PhD | Doctor of Philosophy | |
| HTML | HyperText Markup Language | |
| CSS | Cascading Style Sheets | |
| JSON | JavaScript Object Notation | |
| PDF | Portable Document Format | |
| spaCy | Industrial-strength NLP library in Python | |

ABSTRACT

Companies find it extremely difficult to sort through the vast number of resumes in today's competitive job market to find the best candidates, and job seekers find it difficult to modify their profiles to meet the requirements of employers.We created an AI-powered recruitment support platform that uses Natural Language Processing (NLP) to help HR professionals hire more quickly and give job seekers insightful feedback in order to solve this two-pronged issue. Using sophisticated natural language processing (NLP) techniques, this platform automatically parses resumes, extracts skills, recognizes experience and education, and matches candidate profiles to suitable job roles. To process and extract valuable information from unstructured resume and job description text, a specially trained natural language processing pipeline was created. We developed a strong resume analyzer that can identify technical and soft skills, degree requirements, and inferred years of experience even from free-form text input by using methods like tokenization, lemmatization, and named entity recognition. By using the processed data for candidate recommendation and job prediction, manual screening efforts are greatly reduced.   
Furthermore, the system gives job seekers information about underrepresented or missing skills on their resumes and recommends changes to better meet industry standards. It has a conversational chatbot to respond to user inquiries, API-integrated interview scheduling capabilities, and an analytics dashboard designed for candidates and HR managers to see hiring trends and opportunities. The platform is connected by an easy-to-use frontend that provides real-time feedback and simple navigation. By combining automation, intelligence, and interactivity into a single platform that benefits employers and job seekers alike, the project ultimately seeks to transform recruitment workflows.

1. INTRODUCTION

The hiring process is changing dramatically in the current fast-paced, fiercely competitive labor market. Manually reviewing hundreds or even thousands of resumes is a common step in traditional hiring procedures. This is time-consuming and subject to human bias and mistakes. Conversely, job seekers often question whether their resumes adequately highlight their qualifications, abilities, and strengths. The growing need for intelligent, automated solutions that can improve the hiring and job-search processes is highlighted by this discrepancy between candidate profiles and employer expectations. As natural language processing (NLP) and artificial intelligence (AI) gain traction, businesses are embracing smart technologies more frequently to optimize hiring processes. AI-powered systems for job recommendation, skill matching, and resume screening are quickly becoming indispensable resources for HR departments trying to shorten hiring times and enhance candidate quality. Candidates are simultaneously looking for feedback-driven platforms that assist them in customizing their resumes, identifying skills that are lacking, and locating positions that complement their qualifications and professional objectives. By creating a comprehensive platform for an AI-powered recruitment assistant, our project tackles these issues. This solution incorporates a specially designed natural language processing model that can extract structured data from unstructured resume text, including work experience, education, and skills. It also suggests the best-fitting positions by comparing candidate profiles with job descriptions and offers advice on how to make resumes better. The platform has features like an interview scheduling system, a chatbot for questions, and an analytics dashboard for candidates and HR to further improve the experience. By combining machine learning, human resources, and user-centric design, this project helps close the gap between opportunity and talent and provides a contemporary alternative to antiquated hiring procedures.

* 1. Problem Statement

Develop an AI-Driven Recruitment Bot to streamline HR recruitment processes by automating resume screening, interview scheduling, and candidate recommendations. The bot will enhance recruitment efficiency and improve the accuracy of short listing candidates by leveraging AI-based insight.

1. Background Research

Historically, hiring has been a difficult but intricate decision-making process that calls for a significant investment of time, money, and human judgment. Processing thousands of resumes and matching applicants with job descriptions quickly and effectively while maintaining fairness and avoiding bias is the new challenge for businesses in the digital age. Artificial Intelligence (AI) and Natural Language Processing (NLP) have emerged as revolutionary tools to automate this process in response to the growing trend of business automation.   
A thorough review of the literature found that AI in hiring has advanced significantly, especially in the areas of resume parsing, candidate ranking, and job recommendation. More than 67% of HR professionals think AI speeds up the candidate screening process, per a 2020 LinkedIn report. A transformer-based model was proposed by Malhotra et al. (2022) to improve accuracy and speed in resume classification and job role identification. Using Named Entity Recognition (NER) models to extract experience, education, and skills is another important development (Gupta & Verma, 2021).   
In addition to scholarly research, industry platforms like VMock and Rezi use natural language processing (NLP) to assess resume quality and offer immediate feedback. Their strategy motivated us to investigate comparable techniques within a platform that serves both recruiters and candidates.

* 1. Proposed System

RecruitIQ is a web-based platform for hiring that uses advanced artificial intelligence techniques to combine different hiring functions. It serves two groups of people: job seekers and HR professionals. It offers AI-powered resume parsing, matching between jobs and candidates, interview scheduling, feedback generation, and suggestions for resume enhancement.   
By comparing extracted resume skills, education, and experience with job descriptions, the system helps HRs rank candidates. It provides candidates with a dashboard of job matches and interview dates, as well as resume analysis with practical suggestions.   
An analytics dashboard provides information on hiring trends and candidate status, while a targeted chatbot facilitates communication.

* 1. Goals and Objectives

RecruitIQ's main objective is to use natural language processing (NLP) to automate and improve the hiring process while maintaining the system's scalability, equity, and interactivity.

Table 1: Goal and Objectives

|  |  |
| --- | --- |
| **Goal** | **Objective** |
| Resume Analysis | Use NLP to extract skills, experience, and education from unstructured resumes |
| Candidate Ranking | Match resumes with job descriptions using semantic similarity |
| Resume Feedback | Suggest skill enhancements based on market trends and role requirements |
| Interview Scheduling | Enable HRs to schedule and manage candidate interviews via API integrations |
| Chatbot and Dashboard | Improve user interaction and transparency through a UI-driven chatbot and analytics dashboard |

1. Project Planning
   1. Project Lifecycle

The project used a problem-driven development lifecycle, emphasizing intelligent automation through Natural Language Processing (NLP) and real-world applicability. The entire procedure was divided into distinct phases, each of which added to the overall efficacy of the system. The following are the stages of the lifecycle:

Understanding the Issue: The first step was to fully comprehend the difficulties associated with traditional hiring. Job seekers and HR professionals were the two main stakeholders that we identified. Managing interviews, finding the right candidates quickly, and sorting through thousands of resumes are challenges for HR teams. Candidates frequently don't get much feedback on their resumes and don't have access to resources for tailored enhancement. In order to address these problems, we developed a platform that would benefit both sides by using intelligent automation and analysis driven by natural language processing.

Frontend Development: Using contemporary frameworks, we began developing the frontend as the backend intelligence began to take shape. This included skill and job recommendations, resume upload and scoring pages, candidate and HR analytics dashboards, and an easy-to-use chatbot interface. Ensuring a user-friendly experience with smooth navigation was the aim.

Data Collection: We obtained structured datasets, including job descriptions and cleaned resumes, from websites such as Kaggle. Role-based job expectations, raw resume text, and job categories were all included in these datasets. Our models for job recommendation and resume analysis needed the data to be trained. In order to validate the model and conduct real-world testing, we also obtained sample resumes online.

Data Preprocessing: To transform unstructured resume and job description text into a machine-readable format, preprocessing was necessary. Lemmatization, tokenization, stopword and punctuation removal, and lowercase conversion were all part of this process. In order to preserve information such as years of experience, we preserved digits. Before supplying the data to our models, we also carried out deep cleaning to guarantee that there was little noise and redundancy.

NLP Model Building: We created a unique NLP pipeline to extract valuable data from resumes using spaCy, PhraseMatcher, and transformer-based models (en\_core\_web\_trf). This involved recognizing degree credentials, detecting hard and soft skills, and interpreting experience—even when it was presented in text format. In order to assist us in creating a recommendation engine, the model also carried out intelligent job matching by contrasting the features that were extracted from resumes with the requirements of the position. For robustness, fuzzy matching and contextual matching techniques were used to further enhance the skills.

Backend & API Integration: To link the frontend to the database and the NLP pipeline, we created RESTful APIs. These APIs managed chat support, resume analysis requests, job search and recommendation logic, user authentication, and interview scheduling via calendar APIs from third parties. User actions were synchronized with the underlying NLP logic thanks to backend logic.

Database Management: To store user information, resume text, features that were extracted, job postings, suggestions, comments, and interaction history, we built a structured database. To guarantee fast retrieval and scalable storage, the schema was in line with the requirements of both frontend user interface and backend processing.

Deployment & Testing: The last stage entailed combining the database, frontend, NLP backend, and APIs into a single platform. We thoroughly tested it for edge cases like mismatches, unclear resume formats, and missing data. The implemented platform was now able to process actual resumes, give candidates insightful feedback, and automate HR's screening procedure.

* 1. Project Setup

Important setup choices made throughout the project's development are compiled in the table below. These choices aided in directing the group's choices of technology, coding conventions, deployment plan, and integration of essential elements.

Table 2: Project Setup Decisions

|  |  |
| --- | --- |
| **Sno.** | **Decision Description** |
| 1 | The frontend was built using Next.js (React-based framework) for optimal performance and SEO capabilities. The backend was developed using Python with Flask for handling APIs and NLP processing. |
| 2 | Python 3.10 was used for NLP-related processing and modeling. Libraries used include spaCy, pandas, sklearn, pdfminer, and docx2txt. All code followed PEP-8 Python coding standards. |
| 3 | MongoDB was used as the primary database to store user details, job postings, and resume insights in a flexible and scalable manner. Mongoose was used for MongoDB integration in the backend. |
| 4 | The entire application was deployed using Vercel, which seamlessly integrated with the Next.js frontend. Backend endpoints (Flask APIs) were hosted separately and connected through REST APIs. |
| 5 | All resume and job data used for training and testing were sourced from publicly available Kaggle datasets. In production, resumes uploaded by candidates are stored securely with necessary preprocessing. |
| 6 | External APIs such as Google Calendar API were integrated for interview scheduling features. Secure OAuth authentication was implemented to ensure protected access to calendar services. |
| 7 | NLP models like spaCy’s en\_core\_web\_sm and en\_core\_web\_trf were used. Preprocessing, skill extraction, and named entity recognition (NER) were all managed through spaCy pipelines and PhraseMatcher. |
| 8 | Git was used for version control with the repository hosted on GitHub. Development was collaborative, following feature branching and pull request reviews. |

* 1. Stakeholders

Table 3: Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role** |
| Lakshya Garg | Team Leader |
| Satya Prakash | Team Member |

* 1. Project Resources

Table 4: Project Resources

|  |  |  |
| --- | --- | --- |
| **Resource** | **Resource Description** | **Quantity** |
| Development Team | The core team of student developers working on frontend, backend, and NLP model development | 3 |
| Vercel Hosting | Deployment platform used to host the frontend (Next.js application) | 1 |
| MongoDB Atlas | Cloud-based NoSQL database used to store resumes, job data, and user information | 1 |
| Laptops | Personal development machines used by each team member | 3 |
| Internet Connection | Required for accessing cloud services, APIs, and collaborative tools | 3 |
| Python & NLP Tools | Libraries and frameworks used for building the resume analysis engine (spaCy, pandas, etc.) | N/A |
| PDF and DOCX Parser | Tools like pdfminer.six and docx2txt for resume text extraction | N/A |
| GitHub Repository | Code repository for version control and collaboration | 1 |
| VS Code / Code Editor | IDEs used for code development | 3 |

* 1. Assumptions

Table 5: Assumptions

|  |  |
| --- | --- |
| **Sno.** | **Assumptions** |
| A1 | The project team (Lakshya, Param, Satya) will be able to coordinate and meet regularly to ensure steady progress. |
| A2 | Vercel will provide reliable and free hosting services for frontend deployment throughout the project lifecycle. |
| A3 | MongoDB Atlas will remain accessible under the free-tier limits for development and testing purposes. |
| A4 | Team members will gain sufficient understanding of NLP libraries like spaCy, scikit-learn, and PDF parsers. |
| A5 | Publicly available datasets will be sufficient for building a working and accurate resume parsing NLP model. |
| A6 | Third-party APIs for interview scheduling and chatbot functionalities will remain accessible and stable. |
| A7 | Adequate time will be available to integrate and test all modules (NLP, frontend, backend) before the final presentation. |

1. Project Tracking
   1. Tracking

Table 6: Tracking

|  |  |  |
| --- | --- | --- |
| **Information** | **Description** | **Link** |
| Code Storage | Project code is stored on GitHub | https://github.com/LAKSHYAG16/RecruitIQ-Intelligent-Recruitment-Redefined |
| Bug Tracking | Bugs and issues were tracked informally during development | Not applicable |
| Project Documents | Documents were maintained alongside the code in the GitHub repository | https://github.com/LAKSHYAG16/RecruitIQ-Intelligent-Recruitment-Redefined |
| Continuous Integration | No automated CI/CD pipeline was used | Not applicable |
| Regression Testing | Regression testing was done manually during development and stored on GitHub | https://github.com/LAKSHYAG16/RecruitIQ-Intelligent-Recruitment-Redefined |

* 1. Communication Plan

<< Identify all communications you will provide to other groups and all communications you need to receive from other groups. Share this information with affected groups. Verify that all stakeholders are included. >>

Table 7: Regularly Scheduled Meetings

|  |  |  |
| --- | --- | --- |
| **Meeting Type** | **Frequency/Schedule** | **Who Attends** |
| Team Sync-up | Weekly (in-person) | Project team (Lakshya, Satya) |
| Short Meeting | Weekly in class | Project team and mentor(Dr. Nitin Arvind Shelke) |
| Development Sprint Review | End of each development phase | Project team |
| Final Review | End of project | Project team |

Table 8: Information To Be Shared Within Our Group

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Project Team | Task updates, bug fixes, deployment status | Weekly | WhatsApp, in-Person |

Table 9: Information To Be Provided To Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Mentor(Dr. Nitin Arvind Shelke) | Weekly feedback, Evaluation Criteria, Final project submission | Every week, At project end | Classroom, GitHub |
| Evaluation Panel | Project demo & codebase | Final evaluation | Deployed site, GitHub, presentation |

Table 10: Information Needed From Other Groups

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Mentor(Dr. Nitin Arvind Shelke) | Weekly feedback, Evaluation Criteria | Every Week | Classroom |

* 1. Deliverables

A wide range of technical and user-focused deliverables are produced by the RecruitIQ project with the goal of supporting an intelligent, scalable hiring process. A highly effective resume parsing engine based on natural language processing (NLP) that can extract experience, education, and skills from unstructured text data is one of the main deliverables. Candidates are ranked by job relevance using a semantic matching engine that compares the extracted features to job descriptions. Additionally, the system offers a resume feedback module that suggests areas for improvement and skills based on current job requirements and industry trends. The other important deliverable is the user interface, a responsive web application with a dashboard specifically designed for candidates and HR professionals that was created using Next.js and Tailwind CSS. An analytics dashboard to track candidate engagement and recruitment metrics is included. The platform's integrated conversational chatbot improves user engagement and provides immediate assistance and information retrieval. Additionally, HRs and shortlisted candidates can schedule interviews more easily thanks to the integration of APIs. These deliverables ensure a user-friendly, accessible, and performance-optimized experience for all stakeholders in addition to meeting the system's functional requirements.

Table 11: Deliverables

|  |  |
| --- | --- |
| **Deliverable** | **Description** |
| NLP Model | Trained model to parse resumes and extract key entities |
| Matching Engine | Algorithm to match candidate profiles with job descriptions |
| Resume Feedback Module | Uses skill gap analysis to offer suggestions |
| Frontend Interface | Built with Next.js and Tailwind CSS for candidate and HR dashboards |
| Chatbot + Interview API Integration | Helps automate user queries and schedule interviews |

1. SYSTEM ANALYSIS AND DESIGN
   1. Overall Description

The goal of this end-to-end recruitment support platform is to give job candidates insightful feedback while also improving the hiring process for Human Resource (HR) departments. The system reads resumes and job postings using Natural Language Processing (NLP) and AI-based models to extract structured data such as years of experience, education, and skills. The system ranks and recommends the most qualified applicants for open positions by comparing this data with job requirements. Additionally, the system provides resume feedback to applicants, highlighting areas in which they lack qualifications or skills and making recommendations for improvement on an individual basis. A user interface web application and a backend AI/NLP service comprise the platform's two primary components. The backend, which performs tasks like skill extraction, degree identification, and experience parsing, is built using Python, spaCy, and specially trained NLP models. A combination of entity recognition and phrase matching was used to extract skills. Regular expressions and language models were used to identify degrees and experience data. HR teams can post jobs, schedule interviews, review ranked candidate recommendations, and monitor analytics dashboards using the platform's frontend, which is a contemporary, responsive web application. Candidates can search for available positions, upload their resumes, and get feedback. Integrated APIs are used to manage chatbot interfaces, notification services, and interview scheduling. This system improves candidate engagement and transparency in addition to streamlining the hiring process.

* 1. Users and Roles

Table 12: User and Roles

|  |  |
| --- | --- |
| **User Role** | **Description** |
| HR Manager | A user responsible for creating job descriptions, screening candidate referrals, interviewing, checking hiring statistics, and making the final hiring decisions. |
| Job Applicant | A user who uploads their resume, receives skill and qualification feedback, sees job suggestions, applies for jobs, and monitors their application status. |
| Developer Project Team(Lakshya, Param, Satya) | Responsible team members who create and maintain the platform, develop and train NLP models, handle API integration, and manage deployment. |
| Mentor | An advisor who tracks project progress, provides technical design feedback, and ensures that the project aligns with its defined objectives. |
| Interviewer (Optional) | A user who conducts interviews, gives feedback to candidates, and communicates with HR managers via the system. |
| AI NLP Service | An independent backend service that processes resumes and job descriptions, extracts structured data, and generates candidate rankings and recommendations. |
| Chatbot Service (API) | A program that answers frequent user queries, helps with interview scheduling, and guides applicants on next steps. |

* 1. Design diagrams/Architecture/ UML diagrams/ Flow Charts/ E-R diagrams
     1. Product Backlog Items
* In my capacity as HR manager, I would like to upload job descriptions so that I can match them with the resumes of the top candidates.
* In order for the system to assess my qualifications and pair me with appropriate positions, I would like to upload my resume as a job applicant.
* As a job seeker, I would like to see recommendations on how to make my resume better in order to improve my chances of getting hired.
* In order to swiftly find the best applicants, as an HR manager, I would like to see a ranked list of applicants.
* In my capacity as HR manager, I would like to set up interviews with chosen applicants in order to expedite the hiring procedure.
* As a user, I want to communicate with a chatbot to easily schedule interviews and receive prompt responses.
* In order to avoid missing out on pertinent opportunities, as a job applicant, I would like to receive tailored job recommendations.
* I want to monitor applicant trends and job analytics as an administrator so that we can comprehend our hiring trends.
* As a developer, I want to make sure that data moves smoothly between components by integrating a backend API with the frontend.
  + 1. Use Case Diagram

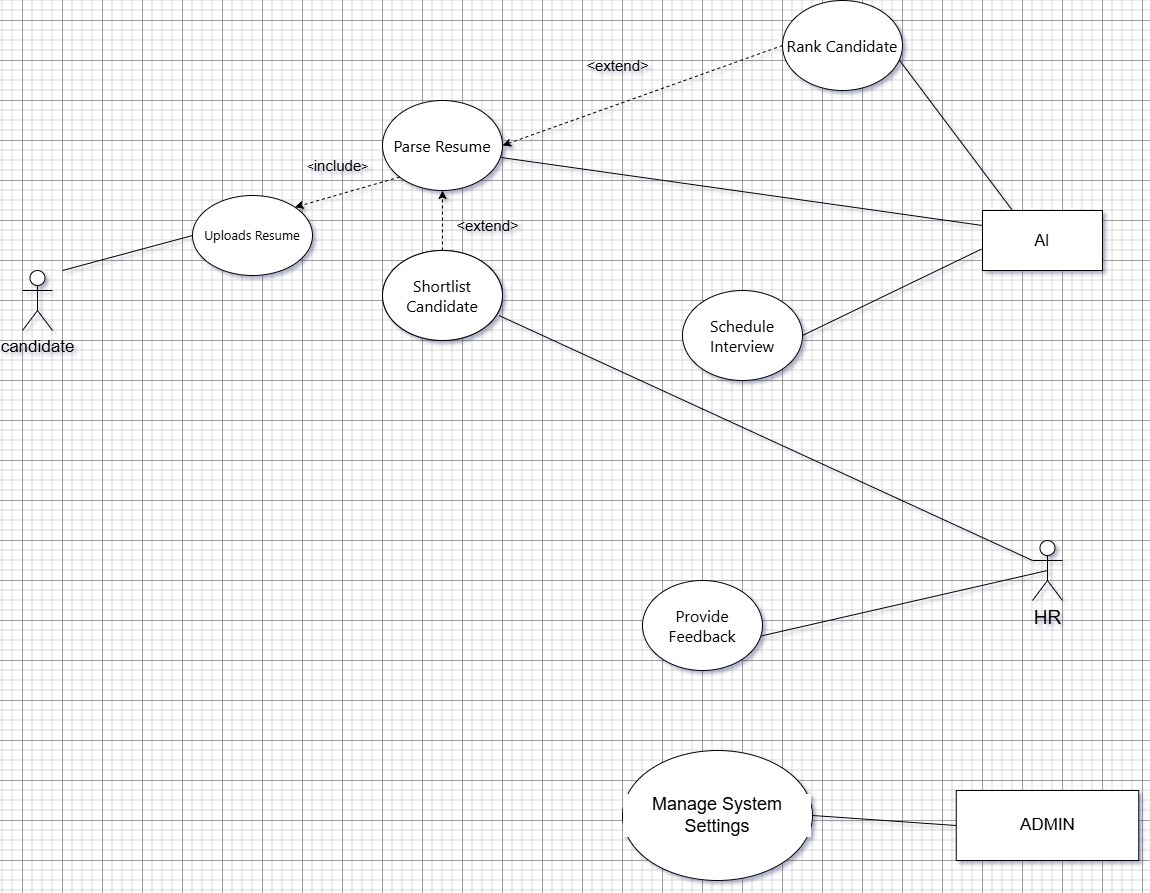


Figure 1: Use-case diagram

* + 1. Class Diagram

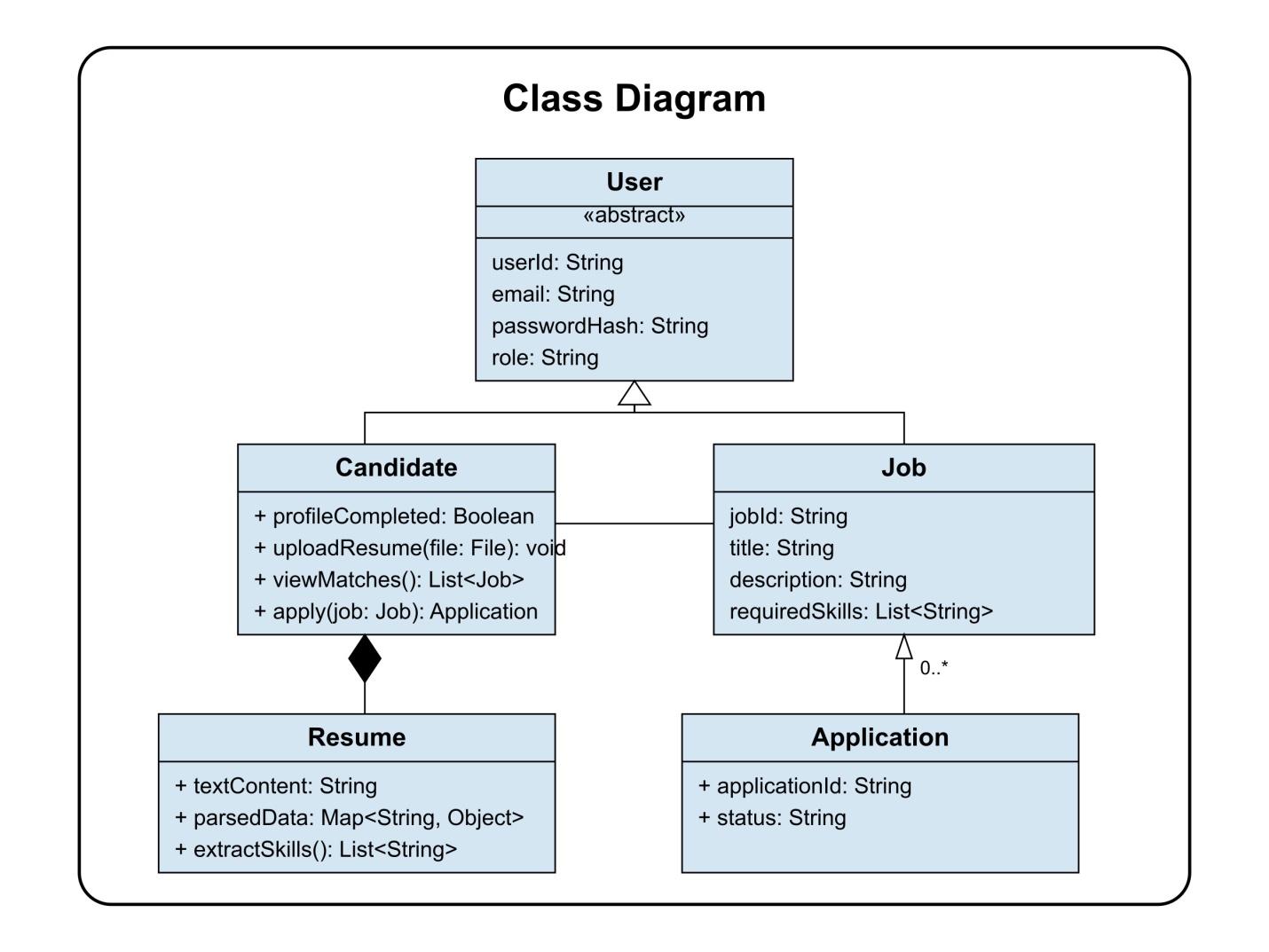


Figure 2: Class diagram

* + 1. Activity Diagrams

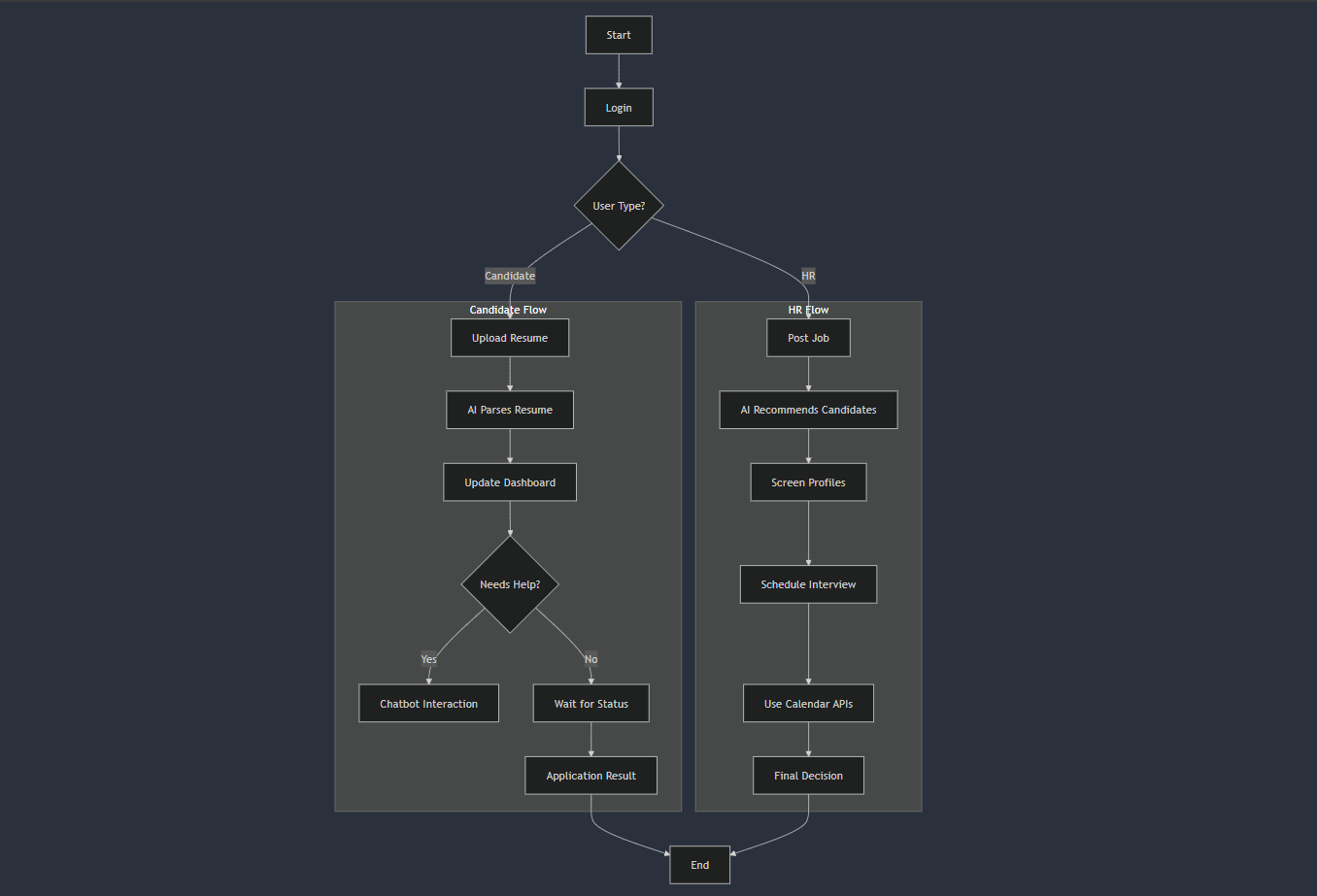


Figure 3: Activity diagram

* + 1. Sequence Diagram

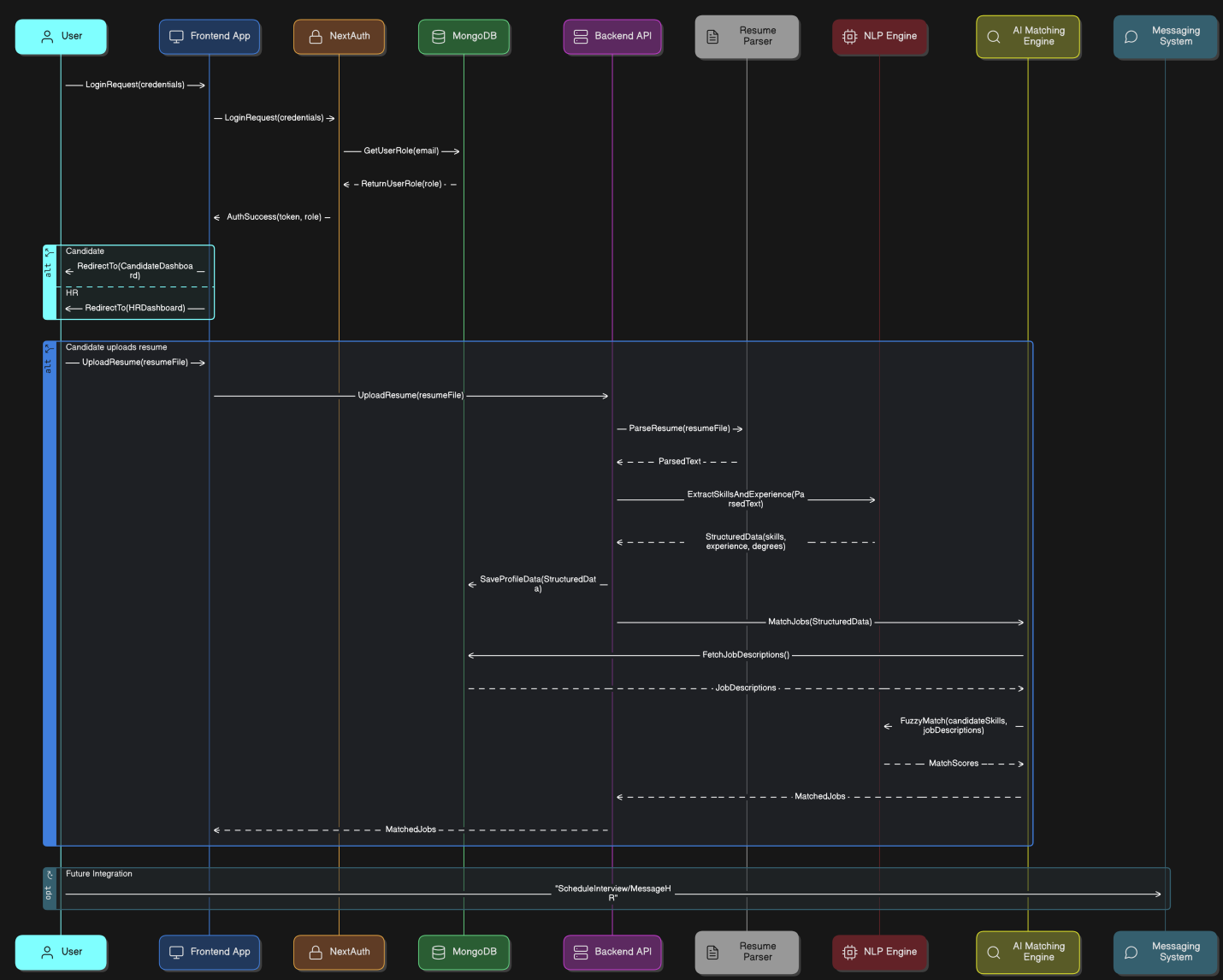


Figure 4: Sequence diagram

1. User Interface
   1. UI Description

A responsive web application created with Next.js and styled with Tailwind CSS serves as RecruitIQ's main user interface. Users first authenticate using NextAuth (Google or GitHub SSO) on a \*Login Page\*. After successfully logging in, they are taken to a \*Dashboard\* with a menu that changes according to their role:

Dashboard for Candidates:

* Resume Upload Module: a straightforward file-select or drag-and-drop interface where applicants can upload their resumes.
* AI Feedback Panel: Provides in-line, instantaneous formatting, keyword, and skill-related suggestions.
* Application Tracker: A progress bar and platform-based status updates for every job application.
* Chatbot Widget: Candidates can inquire about job openings or the status of their applications using a floating chat icon.

HR Dashboard:

* Navigation Sidebar: Connections to "Analytics," "Candidate Management," "AI Screening," and "Interview Scheduling."
* Candidate Management Table: AI-ranked candidates are displayed in this interactive list with search and filter options; selecting a row opens a detailed profile view.
* AI Screening View: Visual summaries (cards/charts) of match scores, sentiment-derived soft traits, and resume-parsed skills.
* Interview Scheduler: HR can choose custom times or suggested slots using the calendar integration; automated confirmation emails are sent.
* Analytics Pane: Dashboards with chart elements (bar graphs, pie charts) that display metrics such as source performance, diversity ratios, and time-to-hire.

Every screen has accessible labels and feedback (loading spinners, success/error toasts), is mobile-friendly, and uses consistent typography and spacing. Candidates and HR users can accomplish important tasks, such as uploading resumes, reviewing AI insights, and scheduling interviews, quickly and easily with this user interface design, which eliminates the need for extra desktop software.

* 1. UI Mockup

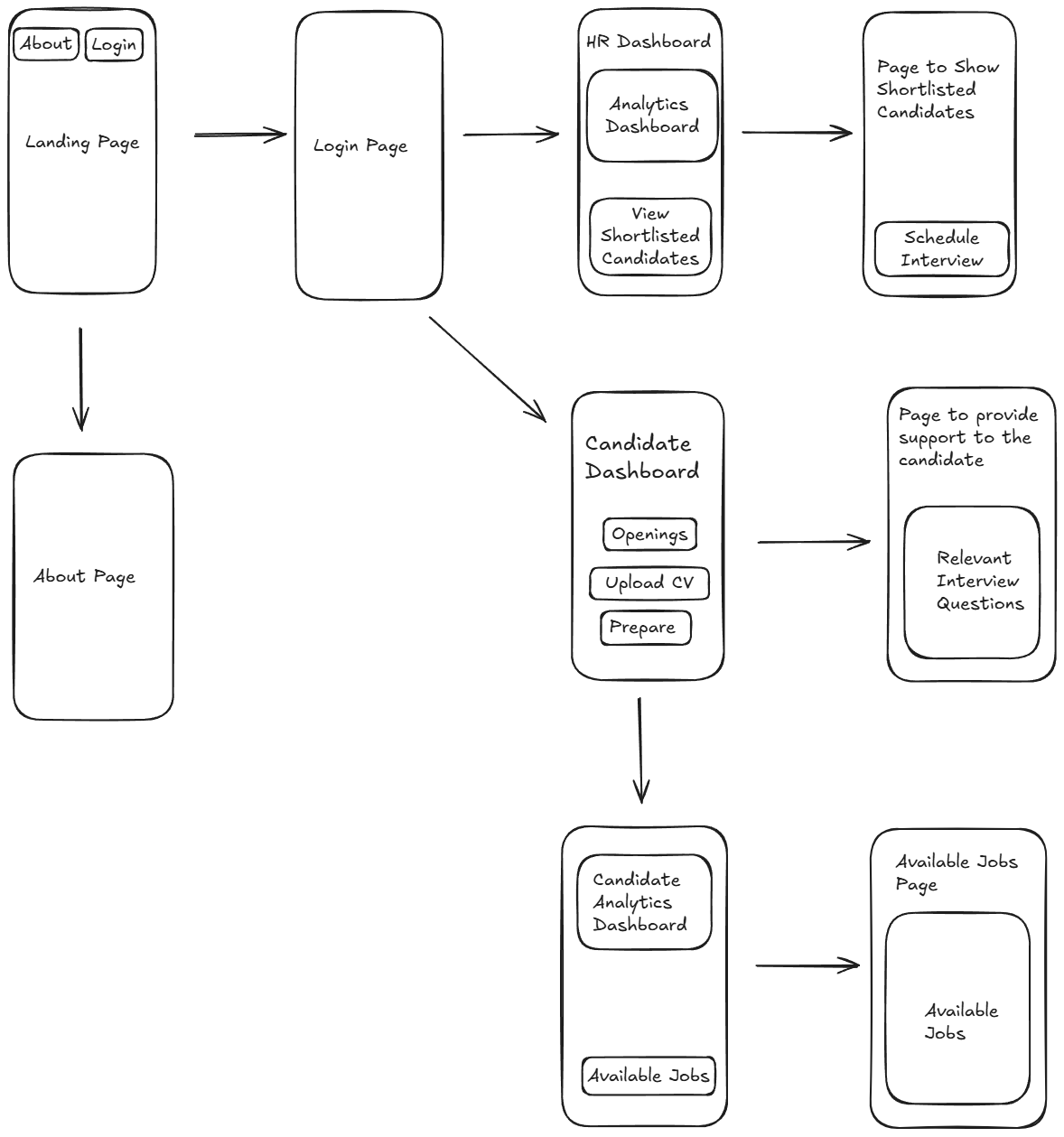
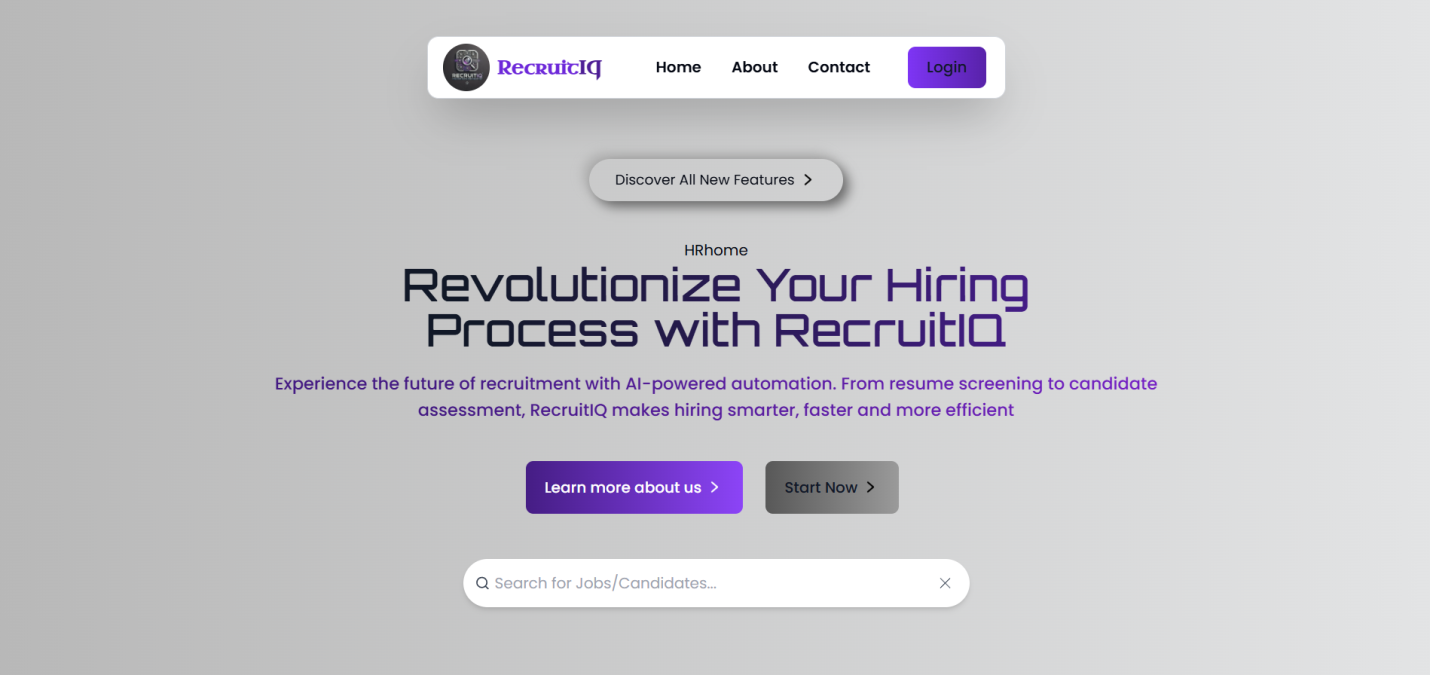


Figure 5: UI Mockup



1. Algorithms/Pseudo Code OF CORE FUNCTIONALITY

* NLP extraction and resume parsing   
  This module uses natural language processing (NLP) to extract structured data from a resume, such as experience, degree, and skills.   
  The algorithm Resume\_NLP\_Extractor   
  Resume (Text, PDF, or DOCX) entered Results: Parsed resume information (experience, education, and skills)   
  Start Open the spaCy language model (either en\_core\_web\_lg or en\_core\_web\_sm). List of recognized technical and soft skills to load Set up PhraseMatcher for the skills you already know.

pgsql

CopyEdit

Function parse\_resume(resume\_file):

text ← extract\_text\_from\_file(resume\_file)

doc ← NLP(text)

skills ← match\_phrases(text, skills\_list)

degrees ← match\_keywords(text, degree\_keywords)

experience ← extract\_experience\_years(text)

Return { "skills": skills, "degrees": degrees, "experience": experience }

End

* Job Description and Resume Matching   
  This module determines a matching score by comparing the extracted resume data with the job descriptions.   
  The algorithm Resume Matching Jobs   
  Input: Job descriptions and resume information (degree, experience, and skills). Output: Each job's Match Score   
  Start For every job listed in job\_descriptions: job\_skills ← extract\_keywords(job["description"]) job\_degree ← extract\_degree(job["description"]) job\_experience ← extract\_experience(job["description"])

lua

CopyEdit

score ← 0

If resume.degree matches job\_degree:

score += 1

If resume.experience ≥ job\_experience:

score += 1

skill\_overlap ← intersection(resume.skills, job\_skills)

skill\_score ← len(skill\_overlap) / len(job\_skills)

score += skill\_score

Store (job\_id, score) in match\_scores

Return match\_scores sorted by descending score

End

* System for Recommendations for Candidates   
  The HR manager benefits from this feature, which ranks resumes according to job-fit scores.

The Recommend\_Candidates algorithm   
Input: List of Parsed Resumes and Job Description Results: Rank List of Suggested Candidates   
Start by extracting job\_skills, job\_degree, and job\_experience from the job\_description.

vbnet

CopyEdit

For each resume in resumes:

resume\_skills ← resume["skills"]

resume\_degree ← resume["degree"]

resume\_experience ← resume["experience"]

score ← 0

If resume\_degree matches job\_degree:

score += 1

If resume\_experience ≥ job\_experience:

score += 1

skill\_overlap ← intersection(resume\_skills, job\_skills)

skill\_score ← len(skill\_overlap) / len(job\_skills)

score += skill\_score

Store (candidate\_name, score) in candidate\_scores

Return candidate\_scores sorted by descending score

End

1. Project Closure

An overview of the study and its application:   
RecruitIQ was developed using an iterative design-thinking process. After defining the problem and conducting research, we created a prototype, integrated the NLP pipeline, and evaluated matching algorithms. After that, we deployed a new frontend and integrated APIs.   
We evaluated a small sample of resumes and job advertisements, extracting information about work history, education, and technical and soft skills. Our pipeline showed promising results in identifying and ranking qualified candidates with the aid of fuzzy matching, PhraseMatcher, and spaCy models.   
Our solution demonstrates the feasibility and scalability of an AI-driven recruitment platform despite time and financial constraints.

* 1. Goals / Vision

The goal of RecruitIQ was to create a scalable, intelligent, and discrimination-free hiring system that maximizes candidate experience while minimizing human HR work. We envisioned a system that would help job seekers improve their career prospects in addition to screening them.

* 1. Delivered Solution

The final product includes:

* A full-stack web application
* Preprocessed and labeled resume and job description datasets
* Skill, education, and experience extraction modules
* Job-candidate matching algorithms
* Resume feedback mechanism
* A Chatbot interface
* Interview scheduling APIs
* Candidate and HR dashboards
  1. Remaining Work

While the core functionality is complete, the following improvements are planned:

* Fine-tuning the skill extraction model using transformer-based NER models
* Soft Skill analysis
* Integration of block chain for data authentication
* Sentiment analysis
* Integrating external job APIs for live job listings
* Adding multilingual support for international candidates
* Expanding analytics features with visual reports for HR
* Developing mobile application support for easier access

REFERENCES

* spaCy – Industrial-Strength Natural Language Processing in Python. <https://spacy.io/>
* scikit-learn: Machine Learning in Python. <https://scikit-learn.org/>
* pandas: Powerful data structures for data analysis. https://pandas.pydata.org/
* NumPy – The fundamental package for numerical computation. <https://numpy.org/>
* PDFMiner.six – A PDF parser in Python. <https://github.com/pdfminer/pdfminer.six>
* docx2txt – Extract text from DOCX files. <https://pypi.org/project/docx2txt/>
* fuzzywuzzy – Fuzzy String Matching in Python. <https://github.com/seatgeek/fuzzywuzzy>
* MongoDB – NoSQL database. <https://www.mongodb.com/>
* Vercel – Frontend hosting and deployment platform. <https://vercel.com/>
* Next.js – React framework for web development. <https://nextjs.org/>
* GitHub – Source code repository and version control. <https://github.com/>
* Kaggle – Datasets for machine learning projects. <https://www.kaggle.com/>
* Python Official Documentation. <https://docs.python.org/>
* Streamlit – Tool for building and sharing data apps. <https://streamlit.io/> (if you used it for dashboards)
* Hugging Face Transformers – State-of-the-art NLP models. https://huggingface.co/transformers/ (if applicable)
* Anaconda Distribution – Python/R distribution for scientific computing. <https://www.anaconda.com/>
* Malhotra, A., Jain, S. (2022). AI-Driven Resume Classification Using Transformers. Journal of AI Applications.
* Gupta, R., Verma, M. (2021). Skill Extraction from Resumes using Named Entity Recognition. Procedia Computer Science.
* Saeed, N. (2023). Resume Dataset. Kaggle. https://www.kaggle.com/noorsaeed/resume-datasets
* LinkedIn Talent Blog. (2020). How AI is Changing Recruitment.
* Rezi Resume AI: https://www.rezi.ai/