Design and Development of AI-Powered Recruitment Platform – Wize (MyLamp.Ai)

A PROJECT REPORT

Submitted in partial fulfilment of the

requirement for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS(MCA)

2023-2025

by

Ashok Sharma

23FS20MCA00096

DEPARTMENT OF COMPUTER APPLICATIONS

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007

RAJASTHAN, INDIA

2025

DEPARTMENT OF COMPUTER APPLICATIONS

MANIPAL UNIVERSITY JAIPUR, JAIPUR – 303 007 (RAJASTHAN), INDIA

Date: 4 June, 2025

CERTIFICATE

This is to certify that the project titled **Design and Development of AI-Powered Recruitment Platform – Wize** is a record of the Bonafide work done by **ASHOK SHARMA** 23FS20MCA00096 submitted in partial fulfilment of the requirements for the award of the Degree of Master Of Computer Applications 2023-2025 **of Manipal University Jaipur, during the academic year 2024-2025.**

Under the guidance of

Dr. Veerpal Kaur

Department of Computer Applications
School of AIML, IoT&IS, CCE, DS and Computer Applications
Faculty of Science, Technology and Architecture
Manipal University Jaipur
Jaipur, Rajasthan

Dr. Shilpa Sharma

HOD, Dept of Computer Application
Manipal University Jarpur

ACKNOWLEDGMENT

I sincerely thank Manipal University Jaipur and the Department of Computer Applications for offering me the opportunity to work on a live industry project during my internship.

I extend my deep gratitude to Ms. Veerpal Kaur, my guide, for her valuable support and continuous encouragement throughout this project.

I also thank the team at MyLamp AI for their mentorship, especially in helping me learn practical backend and frontend development using Node.js and Tailwind CSS.

Last but not least, I thank my family and peers for their support and motivation.

ABSTRACT

In today's fast-paced digital world, recruitment processes are undergoing a significant transformation through artificial intelligence.
The need for faster, smarter, and more accurate hiring has become crucial for organizations, especially in the tech-driven job market.
To address these challenges, MyLamp AI has developed **Wize**, an AI-powered recruitment platform aimed at simplifying and optimizing the hiring lifecycle for both candidates and recruiters.
During my internship at MyLamp AI, I worked on the development of various user interface components and backend functionalities that support the candidate journey within Wize.
My contribution began after the **"Get Hired Instantly"** button, which leads candidates through key features such as **Talent Match**, **AI Interviews**, **Resume Analyzer**, and the **Career Opportunities** module.
Each section was carefully designed and implemented to improve user experience and enhance the platform's intelligence.
Key highlights of my work include building dynamic UI interfaces for candidate profile creation, smart resume upload with skill parsing, automatic AI-based interview generation, and resume analysis that provides actionable insights on hard skills, soft skills, and grammar corrections.
Additionally, I worked on backend features like managing resume data and handling interview performance records.
The project was developed using modern front-end and back-end technologies including Flutter, JavaScript, and REST APIs. Tools such as Git, VS Code, Firebase, and internal APIs provided by the team were used throughout the development phase.
This internship provided hands-on experience in building real-world applications and enhanced my skills in UI/UX design, backend integration, and agile collaboration.

LIST OF TABLES

LIST OF FIGURES

Index

Contents

1. Introduction

1.1 WOEK DONE / MOTIVATION:

The hiring process is time-consuming and inefficient when handled manually. MyLamp AI's platform Wize automates key stages using AI. My responsibility included frontend UI development and backend integration to improve the candidate's journey**.**

1.2 OBJECTIVE:

The primary objective of the internship project at Wize (MyLamp AI) was to contribute to the development of a smart and efficient recruitment platform by working on both frontend and backend components. The specific goals of the project were as follows:

Create user-friendly UI with Tailwind CSS:

Design and develop clean, responsive, and accessible user interfaces using Tailwind CSS to ensure a smooth and intuitive experience for both recruiters and candidates.

Implement backend logic with Node.js for parsing, scoring, and tracking:

Develop backend services using Node.js to handle resume parsing, candidate scoring based on job criteria, and tracking application progress efficiently.

Integrate resume and interview APIs:

Connect and integrate external or internal APIs related to resume handling and interview scheduling, enabling smooth data exchange and functional interactions within the platform.

Build a modular job search and application system:

Create a flexible and scalable system that allows users to search for jobs, apply seamlessly, and manage applications, ensuring modularity for future enhancements.

1.3 PROJECT OVERVIEW:

The internship project was undertaken at Wize, a product developed by the startup MyLamp AI, focused on revolutionizing the recruitment and hiring process through artificial intelligence.

Wize serves as a smart hiring platform that connects recruiters with job seekers, enabling efficient resume screening, candidate shortlisting, and job matching. The platform aims to reduce manual effort and improve hiring accuracy using AI-powered tools and intuitive user interfaces.

During the internship, I contributed to the development of the backend functionality for the resume management section using Node.js and designed user interfaces using Tailwind CSS. I collaborated with a team of developers to implement APIs, optimize performance, and ensure seamless frontend-backend integration.

The project also emphasized scalable architecture, secure data handling, and real-time interactions between recruiters and candidates

2. BACKGROUND MATERIAL

2.1. Conceptual Overview (Concepts / Theory Used)

Wize is an AI-powered recruitment platform developed by *MyLamp AI* to modernize and automate various stages of the hiring process. The platform leverages artificial intelligence (AI) and machine learning (ML) technologies to minimize manual intervention, enhance candidate matching accuracy, and improve overall recruiter efficiency.

Key functionalities of Wize include:

- **Automated Resume Processing:** The system uses parsing algorithms to extract relevant information such as skills, experience, and education from uploaded resumes.
- **Job Matching:** AI models analyze both job requirements and candidate profiles to suggest the most suitable matches based on predefined scoring and ranking metrics.
- **Interview Simulation:** Using candidate data, the platform simulates interview experiences by generating role-specific questions and evaluating candidate preparedness.

The platform ultimately aims to make hiring faster, smarter, and more data-driven, benefiting both recruiters and job seekers.

2.2 Technologies Involved

- **Frontend: Tailwind CSS**

  Tailwind CSS is a utility-first CSS framework that allows for rapid UI development by using predefined utility classes directly in HTML. Instead of writing custom CSS, developers apply classes for padding, margin, color, font size, and more. This leads to:

  - Faster development time with less boilerplate code
  - More consistent designs across components
  - Responsive layouts with minimal effort
  - Easier maintenance and readability of code

  Tailwind was used to design clean and intuitive interfaces for the recruiter dashboard, resume viewer, and job posting modules.

Backend: Node.js and Express.js

- **Node.js** is an open-source, cross-platform JavaScript runtime environment that executes code outside of a browser. It is known for its non-blocking, event-driven architecture, which is suitable for building scalable network applications.

- **Express.js** is a minimal and flexible Node.js web application framework. It simplifies the creation of RESTful APIs and handles routing, middleware, form submissions, and request handling efficiently.

In Wize, Node.js and Express.js were used to develop the backend logic for resume parsing, job matching algorithms, user authentication, and handling asynchronous operations like API calls and database transactions.

Database: Firebase Firestore

Firestore is a cloud-based NoSQL database from Google's Firebase platform. It stores data in a document-oriented structure, making it ideal for dynamic and real-time applications like Wize.

Key features:

- Real-time data synchronization between client and server
- Flexible, schema-less data modeling
- Secure and scalable storage
- Easy integration with frontend frameworks

In Wize, Firestore was used to store user data, resume content, job postings, and application status, all with real-time syncing for instant updates on the frontend.

Development Tools:

GitHub:

- A cloud-based version control platform used to manage source code. It enabled collaborative development, issue tracking, and version management throughout the internship.

Postman:

- A powerful tool for testing REST APIs. It was used extensively to verify API endpoints, check request/response formats, and debug backend logic.

Visual Studio Code (VS Code):

- The primary code editor used for writing and managing frontend and backend code. It offers powerful extensions, syntax highlighting, and Git integration.

Third-Party APIs:

Resume Parser API:

This API automates the extraction of structured information from resumes. It converts unstructured resume files (PDF/DOCX) into a structured format, identifying fields like name, contact details, skills, education, and work experience. This structured data is crucial for candidate scoring and job matching.

AI Interview API:

This API generates custom interview questions based on the job role and candidate resume. It simulates real-world interview scenarios by analyzing the candidate's background and tailoring questions accordingly. It is useful for pre-screening and improving the quality of candidate assessments.

3. METHODOLOGY

3.1 Detailed Methodology

The development of the Wize platform followed the Agile Software Development methodology, which emphasizes iterative progress, continuous feedback, and collaboration. The internship tasks were divided into multiple sprints, each focusing on a specific feature or module. The agile approach enabled quick adaptation to changes, seamless communication within the team, and regular evaluation of progress.

Each sprint typically followed this workflow:

- **Planning:**
  At the beginning of each sprint, tasks were discussed with the team. This included defining the goals of the sprint, identifying the module to be developed (e.g., resume upload, parsing logic, UI components), and assigning responsibilities.

Design & UI Development:

- The sprint started with the creation of the frontend using Tailwind CSS. Mockups or reference designs were followed to ensure UI consistency and responsiveness across devices.

Backend Logic Implementation:

- After the UI was prepared, backend functionality was built using Node.js and Express.js. This included writing RESTful APIs for resume upload, parsing, scoring, and fetching matched jobs.

Integration & API Binding:

- The frontend and backend were connected using REST APIs. Data exchange was tested and debugged to ensure accurate flow from the client interface to the server and back.

Testing & Feedback:

- Postman was used for testing API responses. Functional testing and peer reviews were performed before final deployment. Feedback from senior developers or project leads was incorporated in the next sprint cycle.

3.2 Data Flow Diagram (DFD)

3.2.1 Level 0 – System Overview:

The Level 0 diagram presents a **high-level view** of how user data flows through the main system components.

[User] → Resume Upload → [Parser Engine] → Extracted Data
→ [Match Engine] → Matched Jobs
→ [Interview Generator] → AI Questions
[Feedback Module]

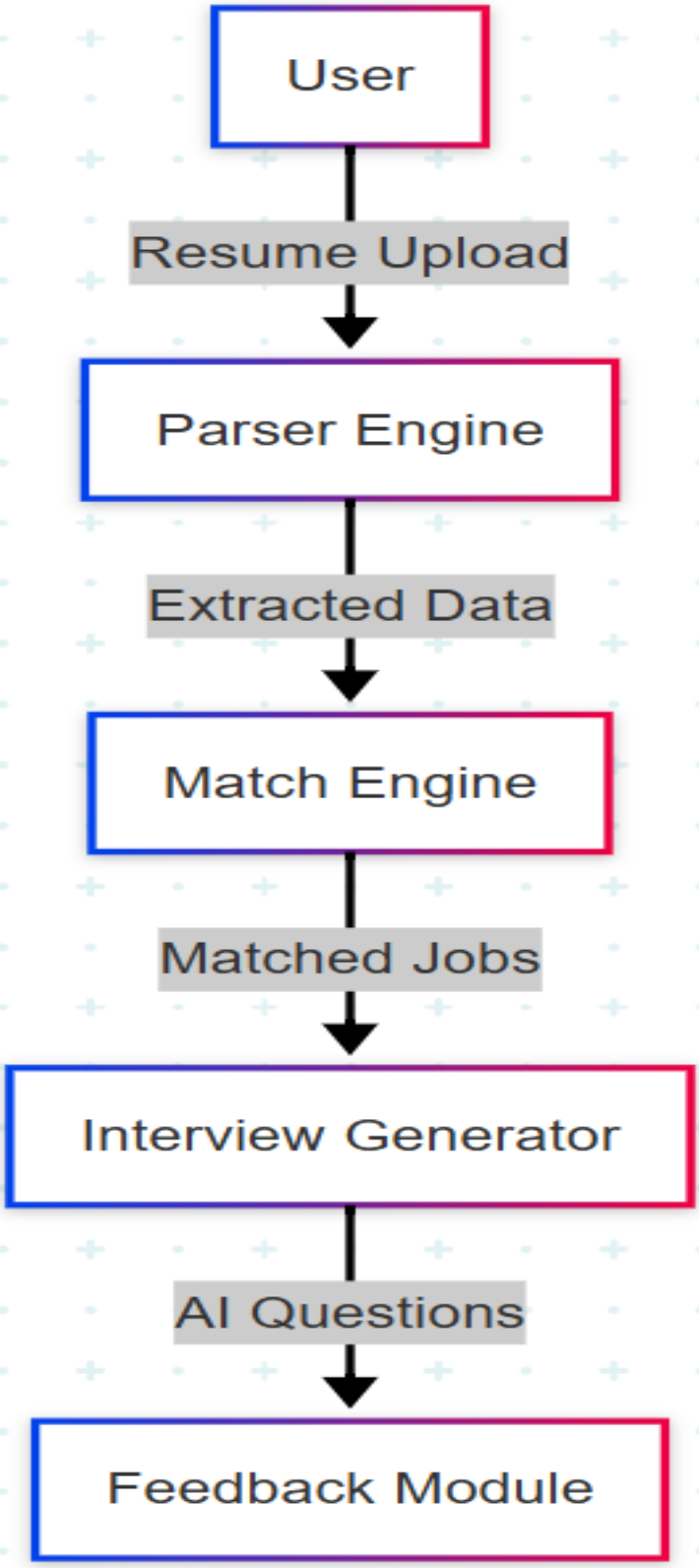
User
Resume Upload
Parser Engine
Extracted Data
Match Engine
Matched Jobs
Interview Generator
AI Questions
Feedback Module

Fig. No. 3.2.1

Explanation:

- The user uploads a resume through the platform.
- The Parser Engine processes the file to extract structured data like skills, education, and experience.
- The Match Engine compares the parsed data with available job listings to find relevant matches.
- Simultaneously, the Interview Generator creates personalized interview questions using AI based on the candidate's profile.
- All results are fed into the Feedback Module, which displays the outputs (matched jobs and interview questions) to the user.

Level 1 – Resume Parsing Module

This diagram zooms into the internal flow of the Resume Parsing Module.

User Uploads Resume
→ Resume Parser API
→ Extracted Skills, Education, Experience
→ Job Matching Module → Display Matched Jobs on UI
→ Resume Analyzer Module → Generate Feedback Report

Explanation:

- When the user uploads a resume, it is sent to the **Resume Parser API**.
- The API analyzes the resume and extracts relevant candidate information such as:
  - Skills
  - Educational qualifications
  - Work experience
- The extracted data is then:
  - Sent to the **Job Matching Module**, which suggests suitable job listings.
  - Sent to the **Resume Analyzer Module**, which provides feedback on resume quality, missing information, or improvement suggestions.
- Finally, both job results and analyzer feedback are rendered on the frontend for the user to review.

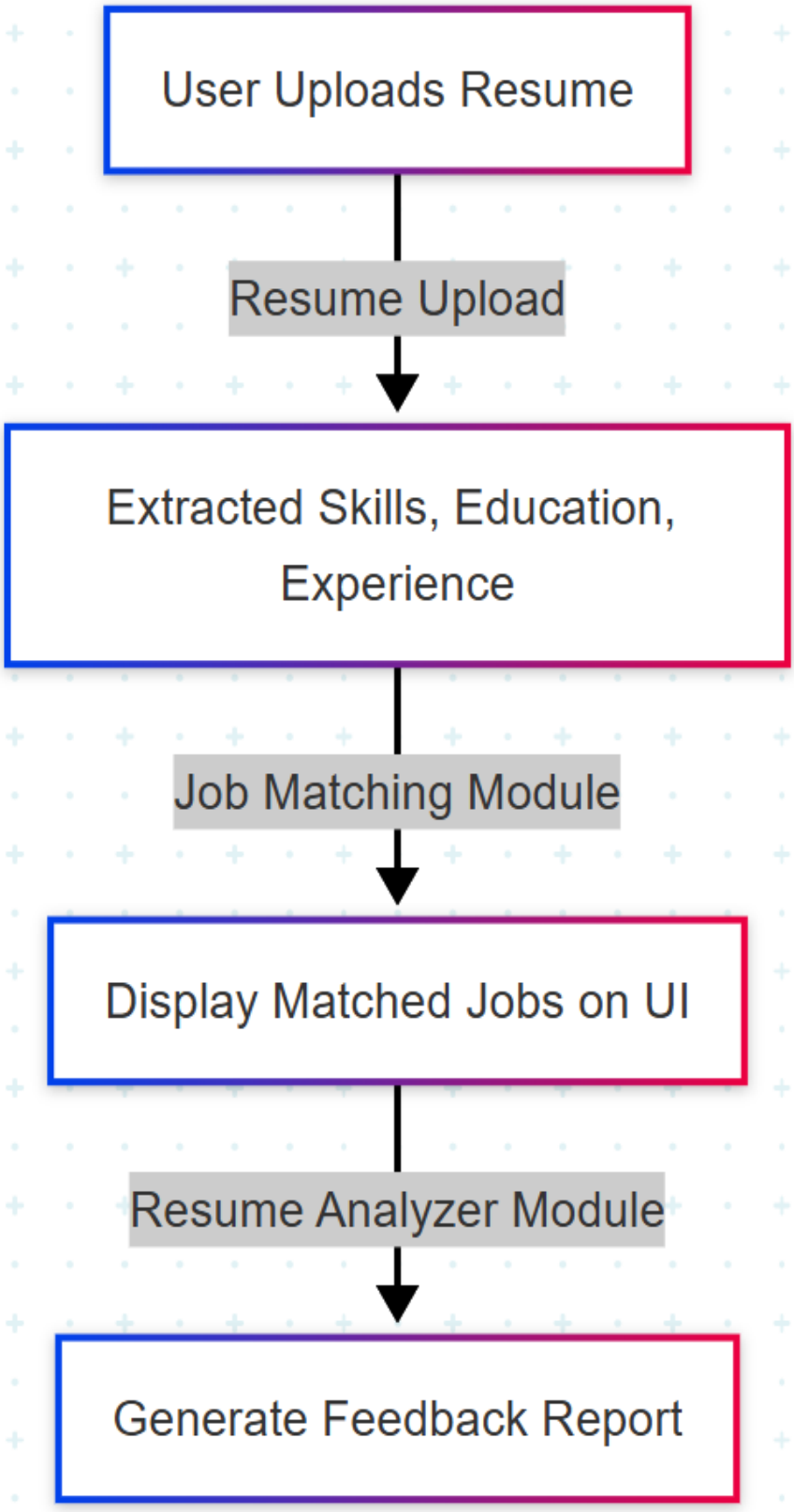
User Uploads Resume
Resume Upload
Extracted Skills, Education, Experience
Job Matching Module
Display Matched Jobs on UI
Resume Analyzer Module
Generate Feedback Report

Fig. No. 3.2.2

3.3 ER Diagram:

Entities:

- **User**: Represents a candidate who registers on the platform.
- **Resume**: Contains uploaded resume data linked to the user.
- **Job**: Represents job listings available on the platform.
- **Interview**: Stores interview details linked to the user.
- **Performance**: Holds feedback and scoring for completed interviews.

Relationships:

- A **User** can upload multiple **Resumes** (One-to-Many).
- A **User** can apply to many **Jobs**, and a **Job** can have many applicants (Many-to-Many).
- A **User** can attend multiple **Interviews** (One-to-Many).
- Each **Interview** has one **Performance** record (One-to-One).

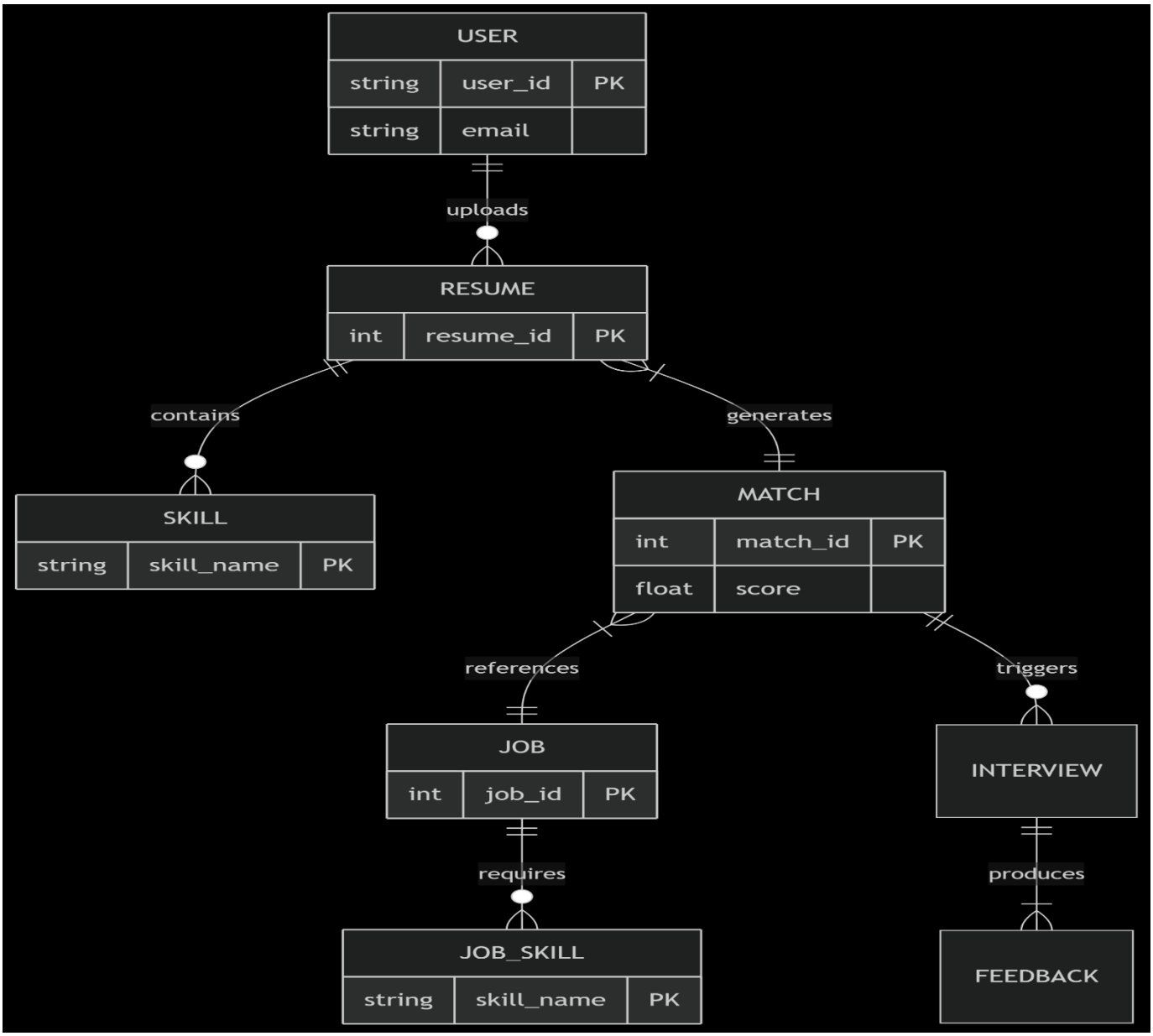

Fig. No. 3.3.1

4. IMPLEMENTATION

4.1 Modules

The implementation of the Wize platform was modular, with each module developed independently and integrated through well-defined APIs. Below is an in-depth explanation of each major module I contributed to during my internship:

The implementation followed an agile methodology, with frequent testing and iterative improvements based on user feedback and functionality tests.

1. Talent Match Module:

Purpose:
This module is designed to automatically match candidates with the most relevant job opportunities based on their resumes or profiles.

How It Works:
When a user uploads their resume, the backend leverages a Resume Parser API to extract key data like skills, previous job roles, education, certifications, and total experience.
This extracted data is then processed using a matching algorithm that compares candidate profiles against a curated database of job openings.

The algorithm scores each job based on how closely it aligns with the user's qualifications and preferences (e.g., desired role, location, domain).
Matched jobs are then sent to the frontend where they are displayed to the user in a ranked format.

Impact:

Significantly reduces the time spent searching for jobs.
Helps users discover opportunities they might not have considered.

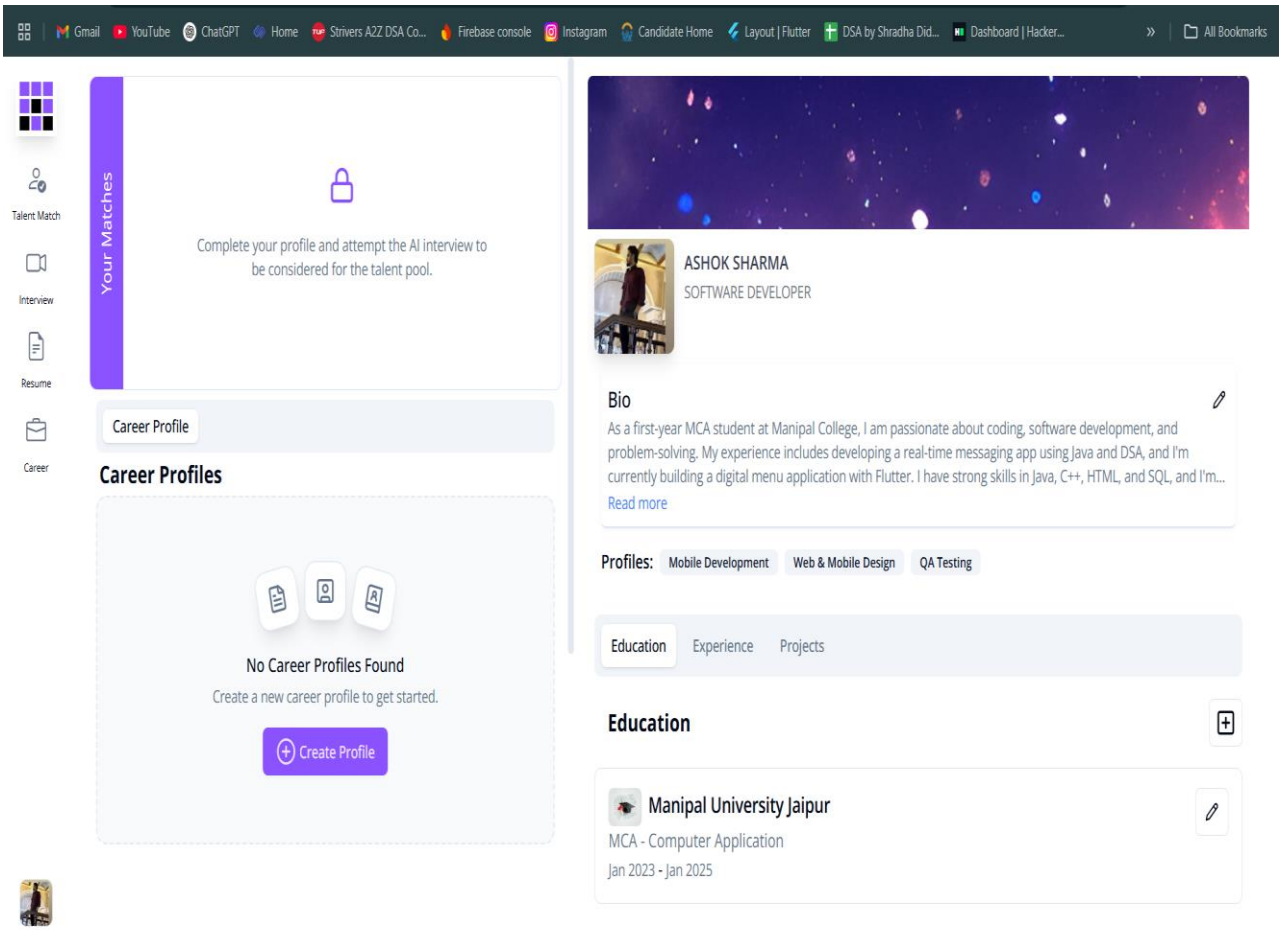

Fig. No. 4.1.1

2. Resume Analyzer Module:

Purpose:
To assess the quality and effectiveness of a user's resume from a recruiter and ATS (Applicant Tracking System) perspective.

How It Works:

- Once a resume is uploaded, it is passed through an **AI-powered Resume Analyzer** API.
- The system performs the following checks:
    - **Hard Skills**: Matches the technical skills mentioned in the resume with popular industry keywords.
    - **Soft Skills**: Detects communication, leadership, and teamwork-related traits.
    - **Grammar Check**: Uses language models to identify and highlight grammatical errors.

ATS Friendliness: Evaluates resume formatting, keyword density, and structure for compatibility with recruiter software.

- After analysis, a **detailed feedback report** is generated and returned to the user via the frontend.

Impact:

- Empowers users to improve their resumes before applying for jobs.
- Enhances job application success rates by ensuring ATS compliance.

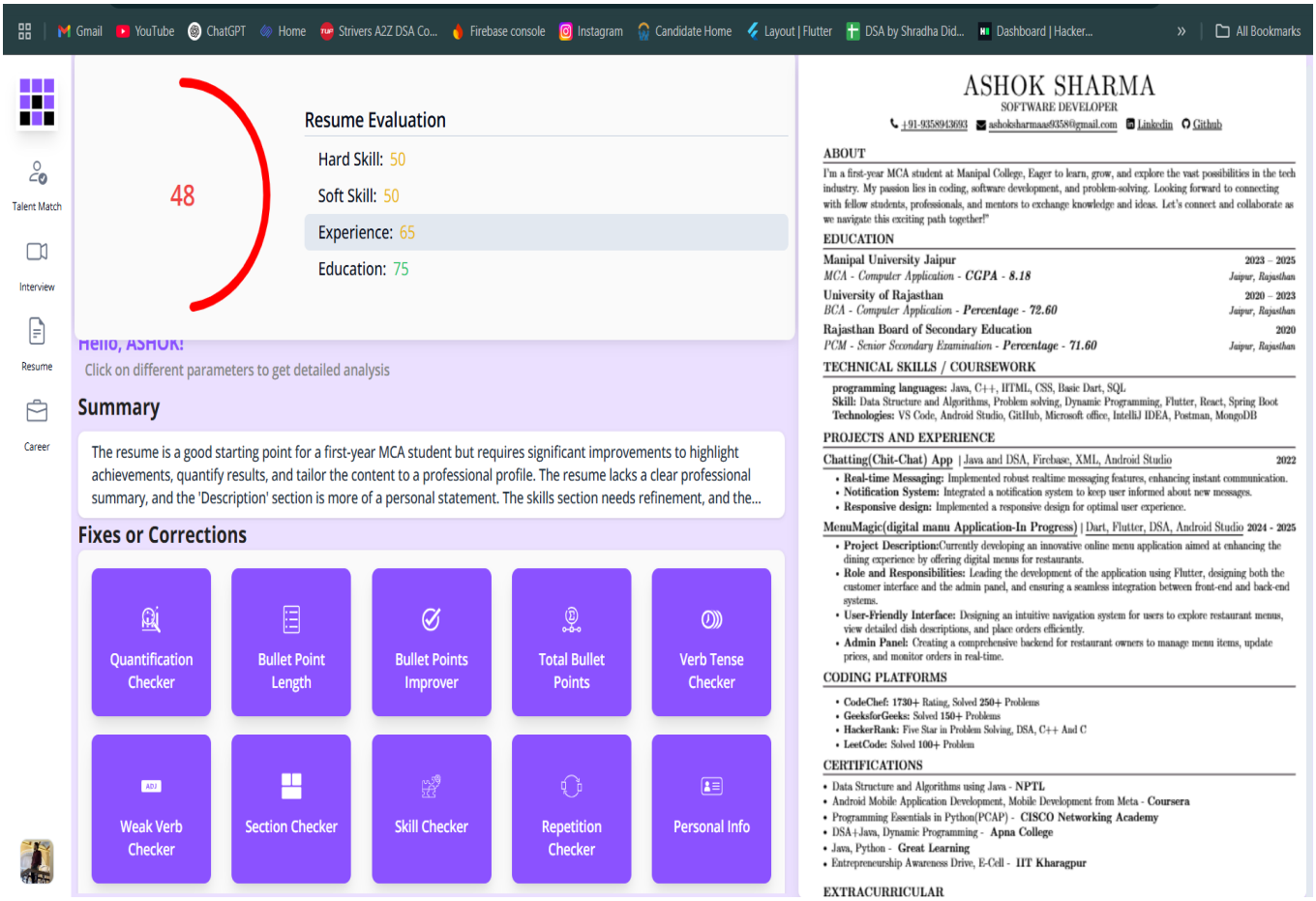

Fig. No. 4.1.2

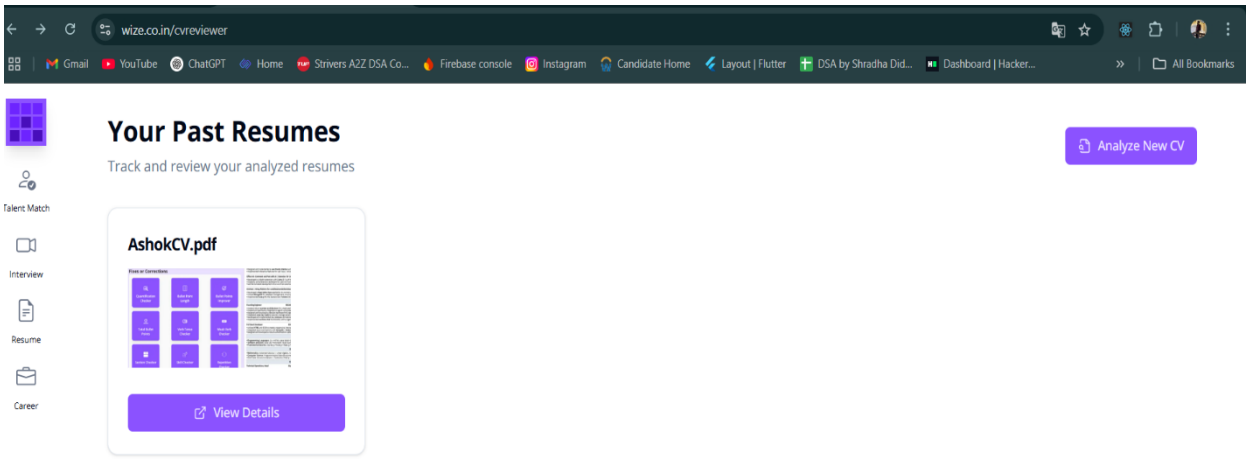

Fig. No. 4.1.3

3. Interview Module:

Purpose:
To help users prepare for job interviews using **AI-simulated interview practice sessions**.

How It Works:

- Based on the candidate's resume, the system generates **domain-specific interview questions** using an AI Interview Generator API.
- Users can choose to:
  - Answer questions by typing responses.
  - Simulate an actual interview experience with timed questions and structured sessions.
- The backend evaluates the responses using:
  - **Accuracy Analysis**: Measures how relevant and correct the response is.
  - **Response Timing**: Tracks how quickly the user answers.
  - **Confidence Indicator**: Optional tone analysis for spoken inputs (if supported in future upgrades).
- A summary report is provided with scores and recommendations.

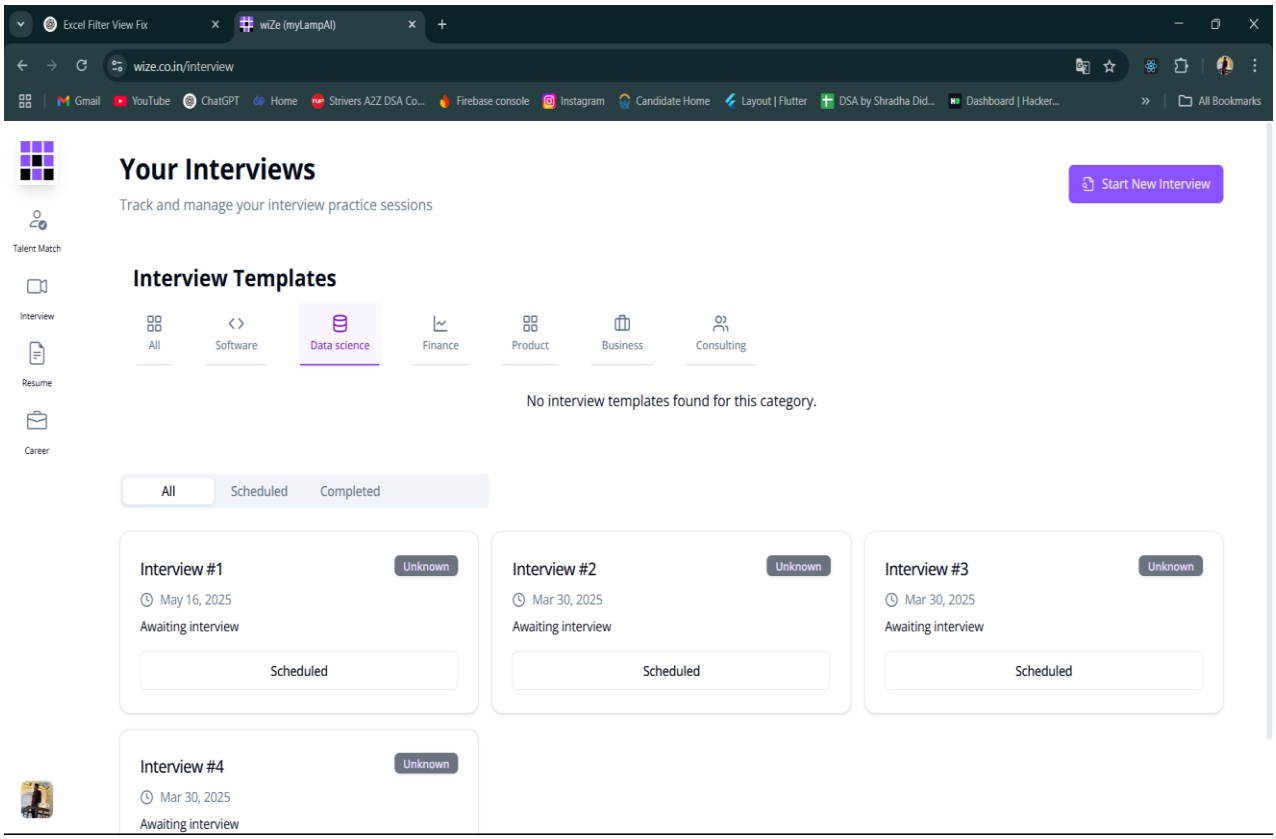

Fig. No. 4.1.4

- Boosts user confidence before real interviews.
- Provides targeted preparation material based on the user's job role.

4. Upload Resume Module:

Purpose:

This module initiates the job-matching process by allowing users to upload their resumes, which are then parsed to extract meaningful information such as skills, experience, and education.

How It Works:

- Users navigate to the "Upload Resume" page via the dashboard.
- The resume is uploaded through a form input which accepts .pdf, .doc, or .docx formats.
- Once uploaded, the resume is sent to the backend where:
    - It is passed to a Resume Parser API.
    - The API extracts structured data (name, contact info, skills, education, work experience).
    - Parsed data is stored in the Firestore database under the user's profile.
      This extracted data is used later for job matching, AI interviews, and analytics

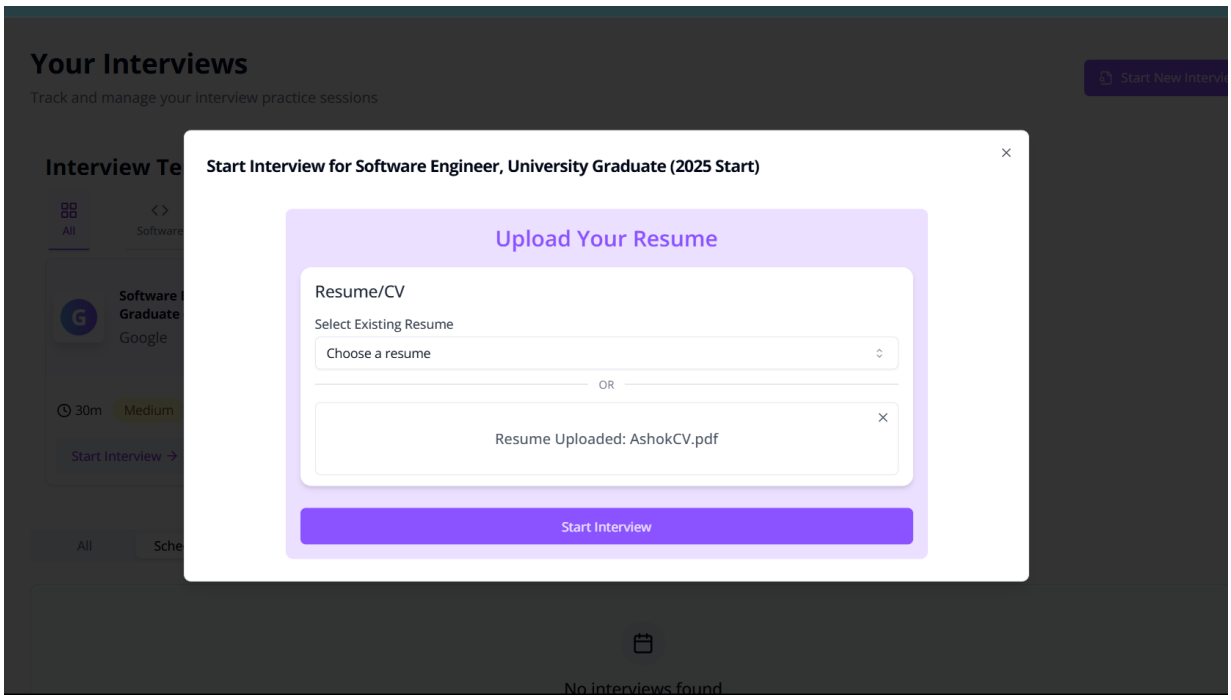

Fig. No. 4.1.5

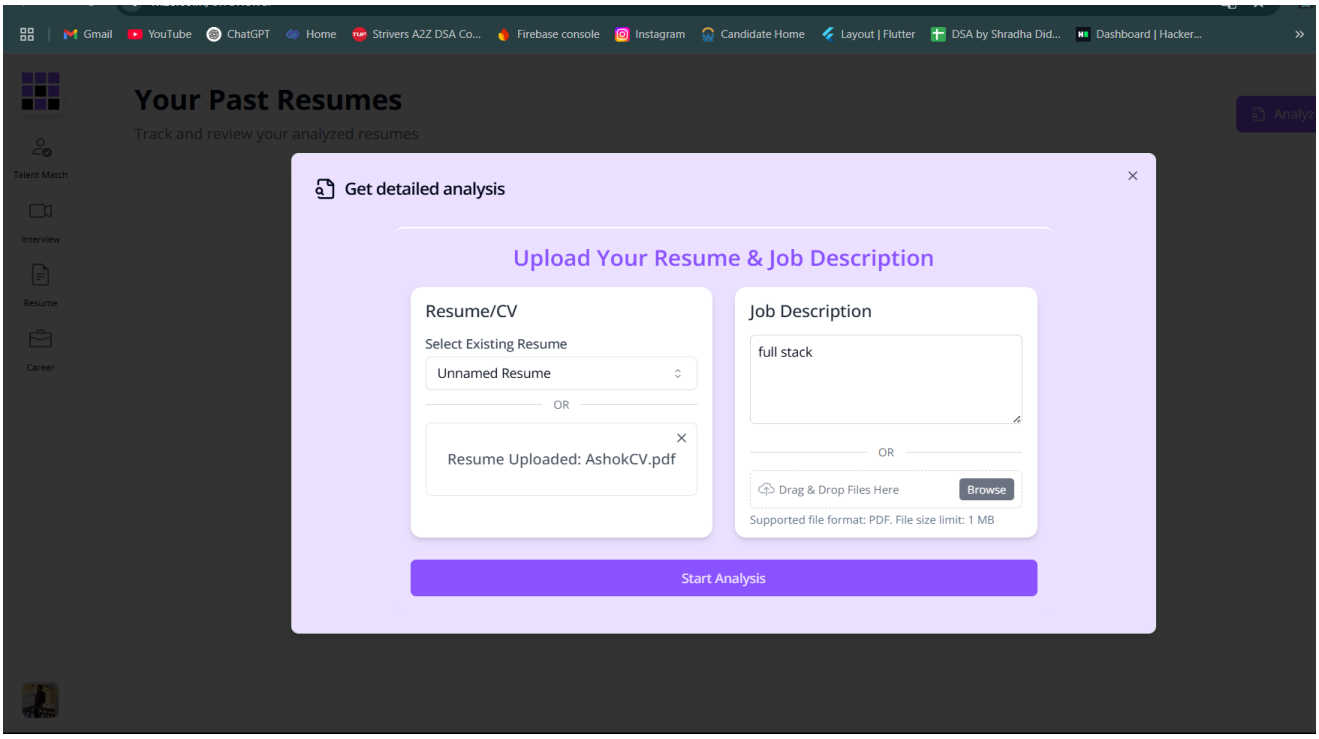

Fig. No. 4.1.6

- Automates the manual effort of filling out forms.
- Lays the foundation for job recommendation and interview preparation.

5. Career Page Module

Purpose:
To act as a central hub for users to browse, filter, and apply to jobs tailored to their profiles.
How It Works:

- After resume parsing, the user's profile data (skills, preferred job role, and domain) is stored in Firestore.
- The backend then uses this data to query a job database and retrieve listings most relevant to the user.
- The career page dynamically displays:
    - Job Title
    - Company Name
    - Required Skills
    - Job Description
    - Apply Now Button
- Users can filter jobs using:
    - Domain/Industry
    - Location
    - Experience level
- Clicking the "Apply" button triggers a backend route that stores the application record and sends a confirmation to the user.

Impact:

- Reduces friction in the job application process.
- Provides a personalized job discovery experience.

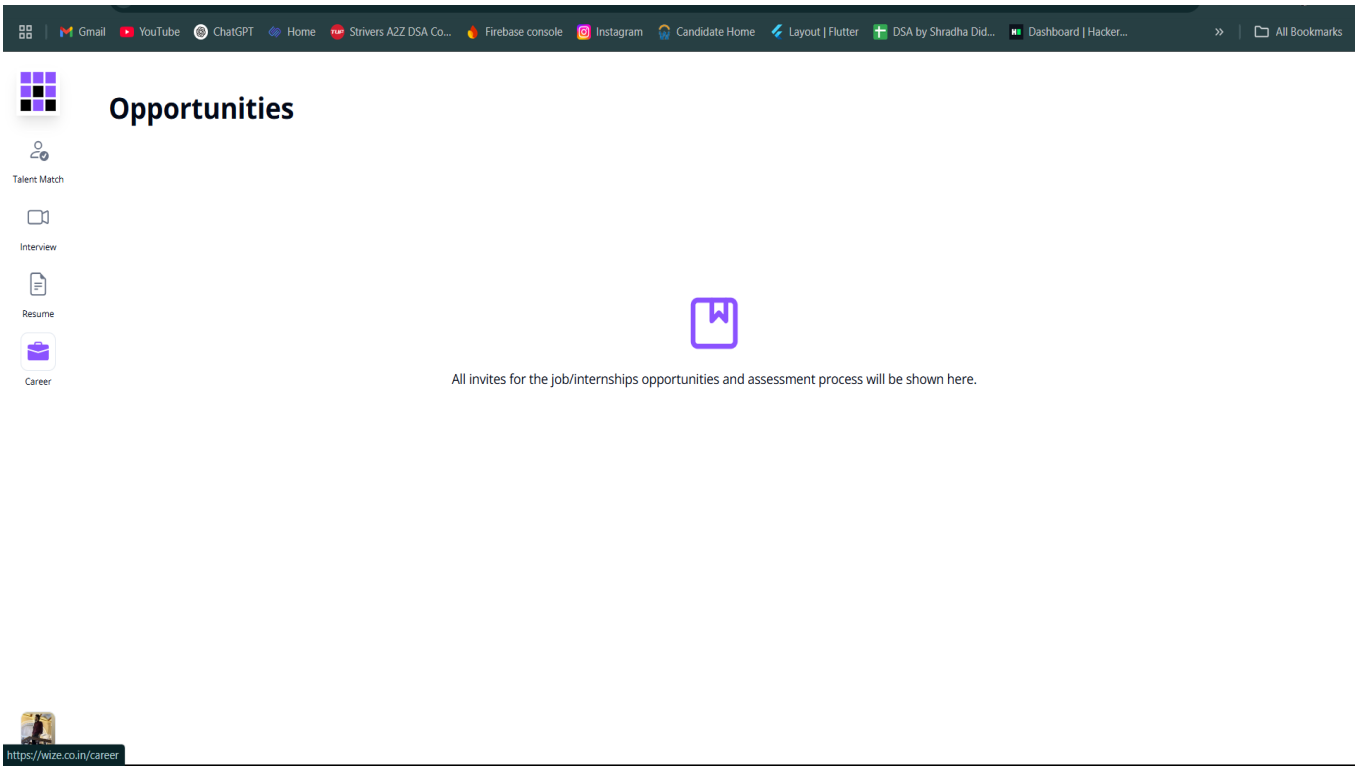

Fig. No. 4.1.7

4.2 Prototypes:

All user interface modules in the project were initially developed as prototypes using Tailwind CSS, focusing on responsiveness, simplicity, and usability. Each screen was designed to represent a specific functional module of the application, such as resume upload, job matching, resume analysis, and interview simulation.

The prototypes were built with a mobile-first approach and later adapted for desktop views. Tailwind CSS utility classes made it easy to manage layout, spacing, and component styling without writing custom CSS.

Screens were tested on various resolutions to ensure a consistent experience across devices.

These UI prototypes served as the foundation for backend integration, enabling developers to connect user interactions with functional APIs smoothly. The key screens include:

- Resume Upload Page – Allows users to upload their resume in various formats.
- Home (Get Hired) Page – Acts as the dashboard for users to start their journey.
- Resume Analyzer Result Page – Displays analyzed feedback on uploaded resumes.
- Career Page – Lists job suggestions based on the user's profile.
- Interview Simulation Page – Hosts domain-specific questions and response tracking.

5. RESULTS AND ANALYSIS

5.1 Functional Testing

Functional testing was conducted to validate the core modules of the application and ensure that each performed according to its defined requirements. Each module was tested with real user inputs and API responses to verify correctness, reliability, and performance.

Module	Test Case	Expected Outcome	Result
Resume Upload	Upload a valid .pdf resume	Resume is successfully parsed and stored in the database	Pass
Resume Upload	Upload an unsupported file format (.txt)	Error message shown and file rejected	Pass
Resume Analyzer	Resume missing essential sections	Feedback includes warnings about missing summary or skills	Pass
Resume Analyzer	Resume with strong formatting and keywords	High score and positive suggestions displayed	Pass
Interview Module	Resume for "Data Science" role	AI-generated domain-specific questions appear	Pass
Interview Module	Submit interview answers	System evaluates accuracy, domain relevance, and timing	Pass
Career Page	Candidate with Java skills	Job listings related to Java shown in user dashboard	Pass
Career Page	No resume uploaded	Prompt to upload resume shown; no job suggestions loaded	Pass

Table No. 5.1.1

5.2 Resume Parsing Results

The Resume Parsing Module was designed to extract key information from uploaded resumes and evaluate them based on predefined metrics. During testing, multiple resumes were uploaded to evaluate the system's accuracy and consistency.

Key Evaluation Criteria:

Formatting: Checks for clear headings, consistent structure, and professional layout.

Grammar and Language: Identifies spelling or sentence-level issues.

Keyword Strength: Scans for role-specific keywords based on the selected field (e.g., Java, Data Science, React).

Section Presence: Verifies that key sections like Objective, Experience, Education, Skills, and Certifications are present.

ATS Optimization: Determines whether the resume follows standard formats compatible with Applicant Tracking Systems.

Result:

Each resume was scored out of 100. Well-structured resumes with domain-specific keywords scored above 85, while resumes lacking key sections or using poor formatting scored below 60. The Resume Analyzer provided actionable tips to help users improve their resumes and meet industry standards.

Feature	Status	Notes
User Registration & Login	✅ Completed	Firebase Auth integration works seamlessly
Google Maps Integration	✅ Completed	Accurate location tracking and route rendering
Booking Flow	✅ Completed	End-to-end booking

		flow operational (select > book > track)
Driver Module	✅ Completed	Functional driver-side interface with ride acceptance
Notifications	✅ Completed	Real-time ride updates sent via Firebase Cloud Messaging
Ratings & Reviews	✅ Completed	Post-ride feedback

Table No. 5.2.1

5.3 Interview Feedback

The AI-based Interview Module provided candidate-specific questions and evaluated answers using three key metrics:

- **Answer Accuracy** – Measures how well the answer addresses the question.
- **Domain Relevance** – Assesses whether the answer aligns with the selected job field.
- **Response Timing** – Evaluates how quickly and confidently the user answers.

Candidate	Role	Accuracy	Relevance	Timing	Remarks

A	**Data Scientist**	**9/10**	**10/10**	**9/10**	**Confident and precise responses**
B	**Front end Developer**	**7/10**	**8/10**	**6/10**	**Relevant answers, slower delivery**
C	**Java Developer**	**6/10**	**5/10**	**4/10**	**Needs improvement in content and timing**

Table No. 5.3.1

6. CONCLUSIONS & FUTURE SCOPE

6.1 Conclusions

The internship experience at Wize, a product of MyLamp AI, was both enriching and transformative. Over the course of the internship, I gained significant exposure to real-world development workflows, tools, and technologies that go far beyond classroom knowledge. This opportunity allowed me to practically apply my understanding of web development, backend engineering, and system design within a fast-paced, AI-powered recruitment platform.

Working on the Talent Match, Resume Analyzer, Interview Module, and Career Page, I was able to contribute to building functional and scalable components that directly impact job seekers and recruiters. My primary responsibilities included backend API integration using Node.js and Express.js, user interface development with Tailwind CSS, and working with cloud services such as Firebase Firestore for real-time data storage. In addition, I collaborated closely with UI/UX designers, backend engineers, and AI specialists, learning how to coordinate tasks efficiently within a team environment using tools like GitHub, Postman, and VS Code.

The internship helped solidify my technical foundation, especially in:

Structuring modular backend systems
Handling API requests and data flows
Designing and testing dynamic frontend layouts
Ensuring real-time UI responsiveness with Firebase

Interpreting AI-generated outputs and integrating third-party services like Resume Parser APIs and Interview Bots

I also learned the importance of agile development, continuous feedback, and modular design thinking. Every sprint involved planning, building, testing, and reviewing work, which helped in continuous improvement and better time management.

Perhaps the most impactful part of this internship was understanding how AI is reshaping traditional recruiting. By building tools that automatically score resumes, generate interview questions, and match candidates to job roles, I gained insight into how intelligent systems are enhancing user experiences and decision-making processes.

In summary, this internship not only strengthened my coding skills but also developed my problem-solving abilities, teamwork, and understanding of production-level application development. The confidence and knowledge I've gained from this experience will serve as a strong foundation for my future professional journey.

6.2Future Scope of Work

Although the platform is robust and functional, several enhancements can be implemented to elevate the product experience further. Here are a few proposed extensions that can improve both usability and scalability:

1. Recruiter Dashboard and Analytics

Currently, the system is primarily candidate-focused. A future extension could include a dedicated dashboard for recruiters that allows them to:

- View candidate profiles and AI-based scores
- Review interview responses and feedback
- Filter and shortlist based on skills, experience, or fit
- Track hiring analytics (conversion rates, response times, etc.)

Adding this feature would complete the two-sided marketplace functionality and attract more companies to use Wize as a full recruitment platform.

2. AI Career Guidance Chatbot

Integrating a real-time chatbot trained on career-related data can help users:

- Choose job roles based on their resume strength and interest
- Suggest improvements in their resume or profile
- Guide them on interview preparation
- Provide links to learning resources and certification paths

This chatbot could be powered by a language model like ChatGPT and tailored with domain-specific datasets for improved accuracy.

3. Video Interviewing and Real-Time Analytics

The current interview module is text-based. Future updates could include:

- One-way and two-way video interview recording
- Real-time facial expression tracking and sentiment analysis
- AI-generated scoring for confidence, tone, and speech clarity

Such a feature would simulate real interviews and prepare candidates better, while giving recruiters a more human-like impression of applicants.

4. Gamification Features for Engagement

To increase user engagement, gamification elements can be added:

- Badges for completing profiles or taking mock interviews
- Leaderboards based on resume scores or quiz results
- Daily challenges related to industry knowledge or job tests
- Streaks and points to unlock premium features

Gamification can make the platform more fun and interactive, especially for students and freshers

5. Resume Builder and Skill Gap Suggestions

A built-in resume builder can help users design ATS-friendly resumes. This can include:

- Real-time suggestions for missing sections
- Keyword analysis based on targeted job roles
- Skill gap analysis and course recommendations

This module can integrate with existing learning platforms (e.g., Coursera, Udemy) for upskilling users directly from the app.

6. Scalability and Cross-Platform Support

Future versions of the platform should support:

- Mobile applications (Android/iOS) for job searching on the go
- Language localization for regional accessibility
- Scalability to support enterprise-level recruitment

With a scalable and inclusive architecture, Wize can become a one-stop platform for AI-driven hiring and career development.

References

- Tailwind CSS Docs – tailwindcss.com
- Node.js Docs – nodejs.org
- Firebase Firestore Docs – firebase.google.com
- Postman API Testing Tool – https://www.postman.com/
- Resume Parsing API Documentation – RapidAPI platform
- Artificial Intelligence in Human Resources: A Review and Future Directions, Journal of Business Research, 2022
- Natural Language Processing for Resume Parsing, International Journal of Computer Applications, 2021
- Automated Interview Systems Using AI, IEEE Access, 2023