INTELLIHIRE: AN AI-POWERED INTERVIEWER FOR AUTOMATED CANDIDATE SELECTION

BY

TONG QIAN RU

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Jasmina Khaw Yen Min, for giving me the invaluable opportunity to work on my final year project titled "IntelliHire: An AI-Powered Interviewer for Automated Candidate Selection." Her guidance, support, and encouragement have been crucial in helping me navigate through the challenges of this project and take my first steps toward a future in AI-driven solutions.

I am also deeply grateful to my parents and family for their endless love, support, and encouragement throughout this journey.

ABSTRACT

The recruitment process has gone a long way to determine the success of organisations in today's highly competitive job market. Traditional interview techniques conducted by human recruiters can be time-consuming, require a huge number of resources, and often suffer from scheduling challenges and inconsistent evaluation criteria, which can influence the way decisions are made. These struggles can cause an inconsistency in the way candidates are evaluated and ultimately result poor hiring decision making. Artificial intelligence (AI) is the up-and-coming technological process that addresses these problems in recruitment. IntelliHire: an AI Interviewer for automated candidate selection is a project that envisions building an extensive Audio-visual enabled machine understanding engine to automate the shortlisting from resumes till scoring interview sessions. IntelliHire provides an inexpensive, time-effective and unbiased way to replace traditional interview methods. This innovation attempts to minimize the time and resources expected from a recruitment process while improving precision in selection as well as providing fairness for job applicants. Ultimately, IntelliHire has the potential to revolutionize the hiring process, providing organizations with a powerful tool to make more informed and objective hiring decisions.

Area of Study: Artificial Intelligence, Web Application Development

Keywords: AI Interview System, Resume Screening, Job Recommendation, Natural Language Processing, Candidate Evaluation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
RDBMS	Relational Database Management System
UI	User Interface
GUI	Graphical user interface
ERD	Entity Relationship Diagram
VS Code	Visual Studio Code
IDE	Integrated Development Environment
TTS	Text-to-Speech
STT	Speech-to-Text
TC	Test Case
SLA	Service-Level Agreement
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript and XML
WebRTC	Web Real-Time Communication
RBAC	Role-Based Access Control
CSRF	Cross-Site Request Forgery
XSS	Cross-Site Scripting
CRUD	Create, Read, Update, Delete
MVC	Model-View-Controller
ORM	Object-Relational Mapping
SQL	Structured Query Language
UX	User Experience
CDN	Content Delivery Network
PDF	Portable Document Format
DOCX	Office Open XML Document

Chapter 1 Introduction

In this chapter, we present the background and motivation of our research, the contributions of this project to the field, and the overall organization of this report.

With the rapidly evolving technological landscape nowadays, Artificial Intelligence (AI) has been widely integrated in various industries and sectors, improving processes compared to traditional methods. One such industry is recruitment, where conventional practices previously relied a lot on human interviewers to screen and select suitable candidates. These such outdated methods would typically be plagued with drawbacks such as high costs, slow processing, and inconsistent evaluation criteria, which have a tendency to influence the efficiency and impartiality of candidate screening in a negative manner.

To address these issues, this project introduces IntelliHire: An AI-Powered Interviewer for Automated Candidate Selection. IntelliHire system uses AI to match candidate resumes, interview the candidates with a blend of generic and position-specific questions, and generate evaluation reports that suggest candidate suitability for specific jobs. This approach saves time and effort compared to traditional methods, eliminates scheduling conflicts, and ensures that all candidates are assessed on a level playing field [1].

With this work, we demonstrate how the recruitment process can be enhanced by AI to make the hiring process more consistent, efficient, and fair. The following sections present the problem statement and motivation, project goals, scope, contributions, and overall organization of this report.

1.1 Problem Statement and Motivation

1.1.1 Problem Statement

1.1.1.1 Resume Screening Challenges

One of the problems with the recruitment process is screening resumes manually [2], which is time-consuming and prone to human error. Recruiters may overlook potential candidates due to exhaustion, heavy workload, or inconsistency in criteria of evaluation. This lack of standardization creates inefficiencies and possibly overlooks top talent.

1.1.1.2 Inefficient and Resource-Intensive Interview Process

Traditional interview process is always inconsistent and highly reliant on human effort, requiring much time and human resources to arrange and conduct interviews [3]. Coordinating multiple candidate interviews manually may be resource-draining, lead to schedule conflicts, and prolong the overall recruitment process.

1.1.1.3 Delays in Interview Scheduling and Feedback

Manual scheduling of interviews and providing feedback are likely to cause delays, which can frustrate candidates and result in losing good talent [4]. Lack of an automated scheduling system and feedback mechanism causes inefficiency, which slows down the hiring process.

1.1.2 Motivation

The main motivation behind developing IntelliHire is to make the hiring process more efficient and unbiased. Traditional methods are time and cost intensive, especially when handling a large number of applicants. Moreover, manual evaluations are prone to vary from interviewer to interviewer, and it may be difficult to maintain consistency while assessing the candidates, which may eventually lead to less than optimal hiring decisions.

1.2 Objectives

The project objectives of IntelliHire project are:

1.2.1 Develop an AI-Powered Resume Screening System

To make the resume screening process automated using AI that analyses and filters resumes based on predefined criteria, only the most suitable candidates proceeding to the interview stage. This will save time and effort required in manual evaluation and create a standardized process to resume screening.

1.2.2 Automate the Interviewing Process for Increased Efficiency and Consistency

To create an AI-powered interviewer that can conduct structured virtual interviews, generate relevant questions from the job description and the candidate background, and provide real-time responses analysis. This will save time and human effort, increase consistency in candidate assessments, and streamline the entire recruitment process.

1.2.3 Real-Time Scheduling and Feedback System

To develop an automated scheduling module that allows candidates to schedule interviews in real-time and receive timely, automated feedback after each interview. It would simplify the recruitment process, enhance the candidate's experience, and reduce delays in decision-making.

1.3 Project Scope and Direction

The proposed IntelliHire system is designed to transform the traditional hiring process through the use of AI in resume screening, interviewing, and even evaluating candidates based on the same parameters. The scope of the project involves the development of an AI-powered interviewer that can scan resumes, interview, and render a fair judgement of the candidates. The system will promote the interersts of various industries with a perspective towards improving the fairness, objectivity, and efficiency of the recruitment process.

The IntelliHire system will be developed as a web application, which can be accessed by both recruiters and candidates. It will have support for multiple job roles across different fields and hence be applicable to any size of company. The system will also include features such as scheduling interviews in real-time, AI-based question generation, and automated feedback mechanisms. The project will also involve the development of a user-friendly interface that ensures ease of use for all users.

1.4 Contributions

The contributions of this project are:

- **Development of an AI interviewer**: This project will develop an AI-powered system that is able to conduct candidate interviews autonomously, analyze their responses, and provide a comprehensive evaluation.
- **Resume analysis and interview evaluation integration**: The system will combine resume data with interview performance to provide a full view of a candidate's suitability for a job.
- **Enhancement of recruitment efficiency**: By automating time-consuming tasks, the system will significantly reduce the time and expense involved in the hiring process.
- **Improvement of consistency in candidate evaluation**: The AI system is designed to provide uniform assessments across all candidates, minimizing variability caused by manual evaluations.

This project represents a significant advancement in the use of AI in human resource management and providing a more efficient, cost-effective, and fairer means of candidate screening.

1.5 Report Organization

This report is organized into seven chapters to systematically present the development of the IntelliHire system. Chapter 1 introduces the background of the project, the problem statement, objectives, scope, contributions, and the overall structure of the report. Chapter 2 provides a literature review of existing AI-powered recruitment systems, highlighting their features, limitations, and relevance to this study, followed by a comparative analysis of the reviewed systems and the proposed solution. Chapter 3 explains the system methodology, which includes design diagrams, architecture, system flows, use cases, activity diagrams, and the Agile development approach, before concluding with the project timeline. Chapter 4 details the system design by presenting the block diagrams, component specifications, database design, and system interactions. Chapter 5 focuses on the implementation phase, covering hardware and software setup, configurations, system operations, and the challenges encountered during implementation. Chapter 6 presents the evaluation and discussion of the system, including testing objectives, techniques, acceptance criteria, decision table testing, state transition testing, and project challenges. Finally, Chapter 7 concludes the report with a summary of findings and recommendations for future improvements to IntelliHire.

Chapter 2 Literature Review

2.1 Previous Works on AI Interviewers

The use of AI-powered platforms in recruitment has significantly changed the way companies conduct interviews, offering more efficient and data-driven processes. Interviewer.ai, Talently.ai, and Apriora are three prominent platforms that have contributed to this evolution. Each platform offers unique features designed to streamline the interview process, improve candidate evaluation, and reduce the workload for recruiters.

2.1.1 Interviewer.ai

Interviewer.ai [5] stands out for its well-rounded approach to assessing job candidates. It doesn't just screen resumes—it also analyzes interview responses using AI, producing detailed reports that help recruiters make informed decisions. These reports include scores for key traits like communication, energy, professionalism, and sociability. Each score is generated by the platform based on the candidate's video interview performance. One of its strong points is the report generation system. For every interview, Interviewer.ai creates an Excel file that stores important candidate details. This includes their name, contact info, application status, interview date, and their overall and individual scores. This structure is especially helpful for large recruitment drives, where organizing and comparing candidate data quickly becomes essential.

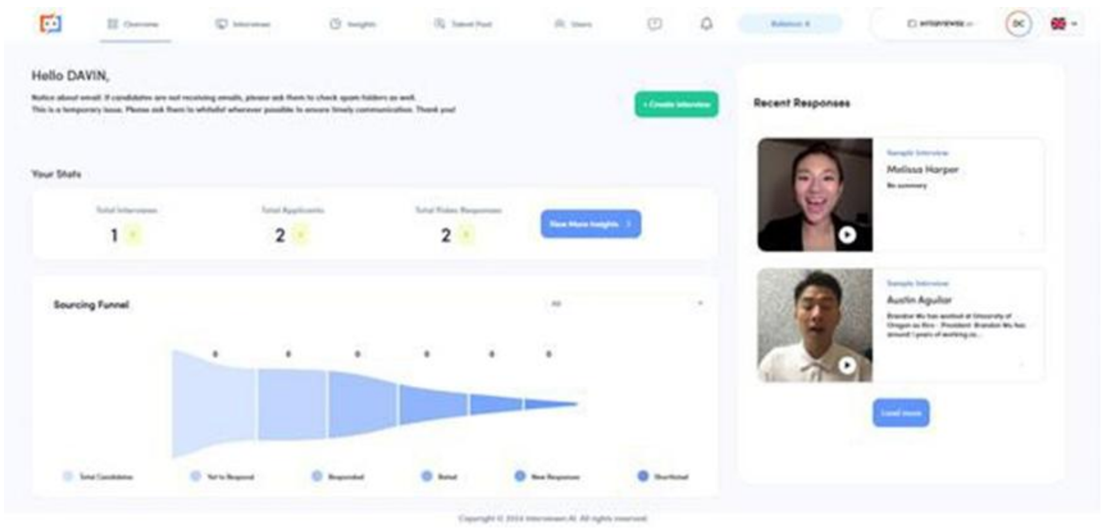
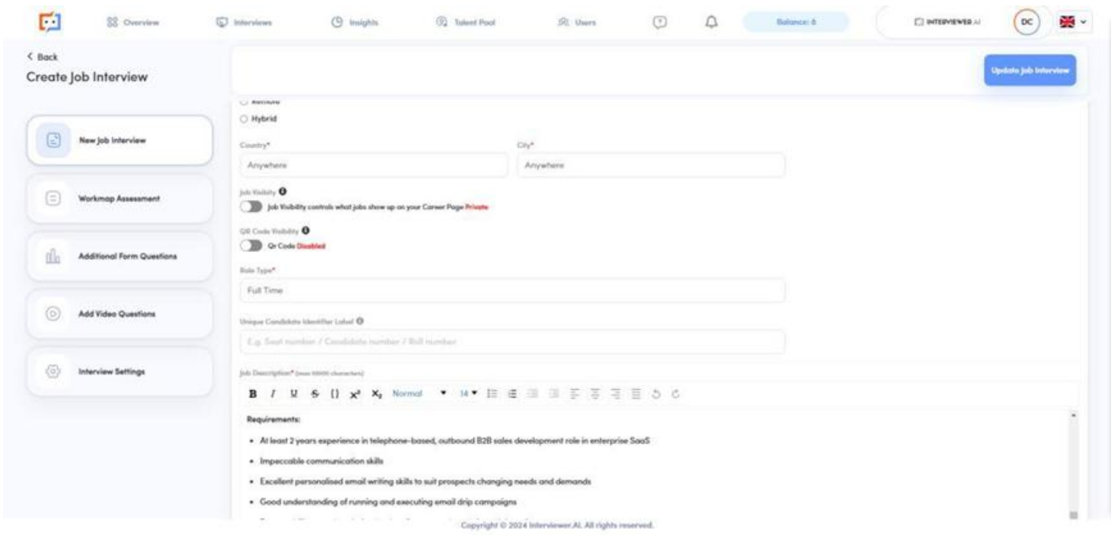


Figure 2.1.1.1 Overview of interviewer.ai

Figure 2.1.1.1 shows the overview page of Interviewer.ai, which provides companies with a comprehensive analysis of the number of candidates applying for various positions in the organization. This feature offers a centralized view of the recruitment pipeline, enabling recruiters to quickly assess the volume and distribution of applicants across different job postings. Such an overview aids in identifying which positions attract the most interest and may require more attention or adjustment in the recruitment strategy.



Create Job Interview

Hybrid

Country*: Anywhere

City*: Anywhere

Job Visibility: Job Visibility controls what jobs show up on your Career Page Private

QR Code Visibility: Qr Code Disabled

Role Type*: Full Time

Unique Candidate Identifier Label: E.g. Seat number / Candidate number / Roll number

Job Description* (max 10000 characters)

Requirements:

- At least 2 years experience in telephone-based, outbound B2B sales development role in enterprise SaaS
- Impeccable communication skills
- Excellent personalised email writing skills to suit prospects changing needs and demands
- Good understanding of running and executing email drip campaigns

Figure 2.1.1.2 Cerate job interview page

Figure 2.1.1.2 depicts the initial stage of the interview process within Interviewer.ai, where the company creates a job interview profile. This step involves inputting essential details such as the job title, company location, and specific requirements for the role. Additionally, the platform allows users to include a detailed job description, ensuring that candidates fully understand the expectations before proceeding with the interview. A key feature at this stage is the ability to set interview questions and assign a time limit for each, providing structure to the interview process and ensuring that all candidates are assessed under consistent conditions.

Additional Form Question Bank

Video Question Bank

Figure 2.1.1.3 Question bank of interviewer.ai

As seen in Figure 2.1.1.3, users can pick questions from the platform's question banks or write their own. There are two main sections: the "Additional Form Question Bank," which includes community-submitted questions, and the "Video Question Bank," which is more structured and organized by category. While the form bank sometimes contains irrelevant or spam content, the video bank is more reliable and easier to use.

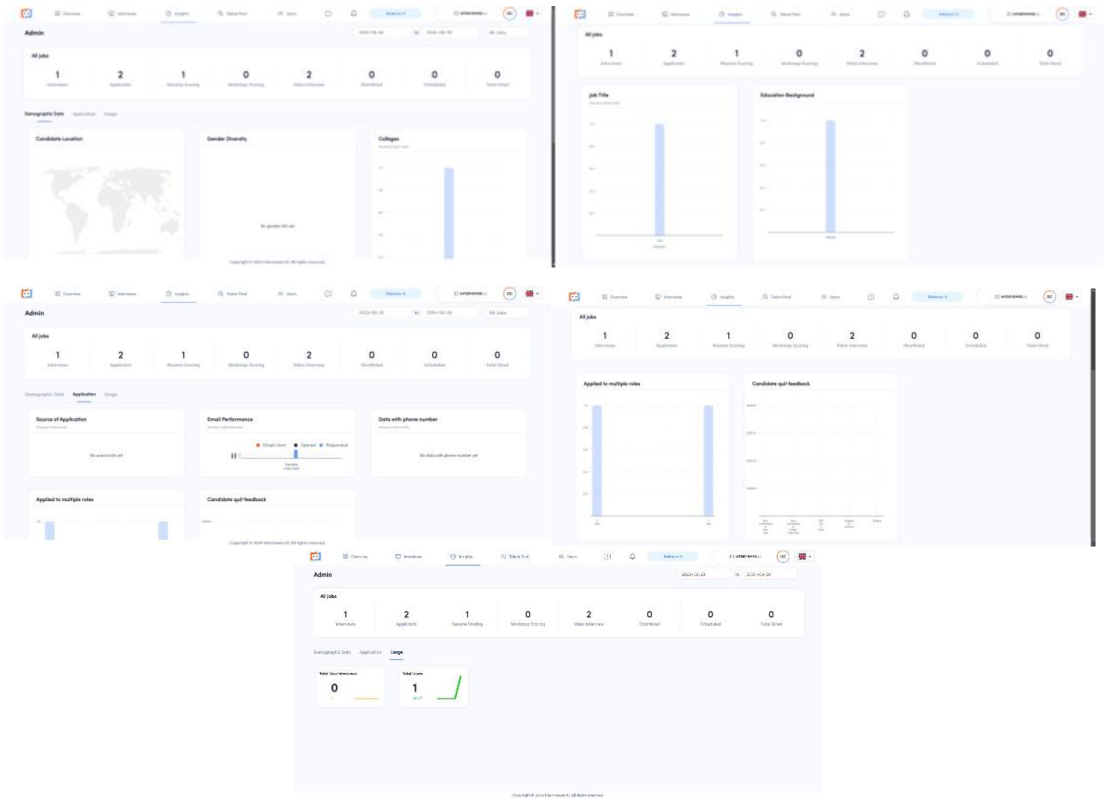


Figure 2.1.1.4 Insights page of interviewer.ai

After the interview setup is complete, users can access overall insights for that specific interview, as seen in Figure 2.1.1.4. This page provides detailed demographic and educational data about the candidates, including their location, gender, and academic background. These insights are invaluable for recruiters as they offer a quick yet comprehensive overview of the applicant pool, helping to identify trends or patterns that may influence the hiring decision. For example, understanding the geographical distribution of candidates could inform future recruitment strategies, while educational backgrounds might highlight the need for additional training or support in certain areas.

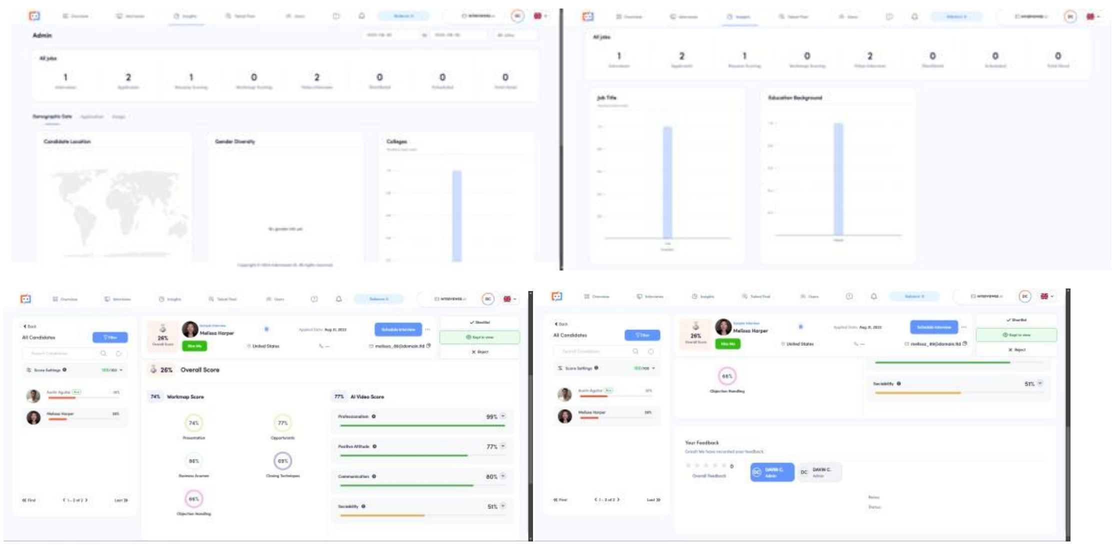
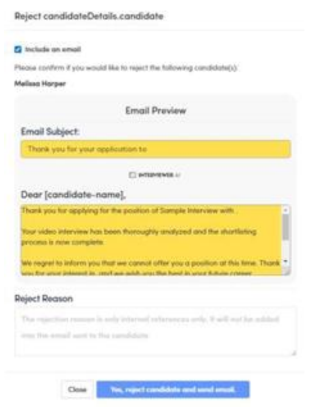


Figure 2.1.1.5 Overall review of created job interview

Figure 2.1.1.5 presents the overall review page for an individual candidate. Here, recruiters can evaluate the candidate's interview performance based on Interviewer.ai's automated scoring and analysis. The platform provides a transcript of the interview alongside the recorded video, which is conveniently segmented according to each question asked. This feature significantly enhances the review process, as it allows recruiters to focus on specific responses without having to sift through the entire interview. After reviewing, recruiters can decide whether to shortlist the candidate, keep them in view for future opportunities, or reject them.


Reject candidateDetails.candidate

☑ Include an email

Please confirm if you would like to reject the following candidate(s):

Melissa Harper

Email Preview

Email Subject:

Thank you for your application to

INTERVIEWER.AI

Dear [candidate-name],

Thank you for applying for the position of Sample Interview with .

Your video interview has been thoroughly analyzed and the shortlisting process is now complete.

We regret to inform you that we cannot offer you a position at this time. Thank you for your interest in , and we wish you the best in your future career.

Reject Reason

The rejection reason is only internal references only. It will not be added into the email sent to the candidate.

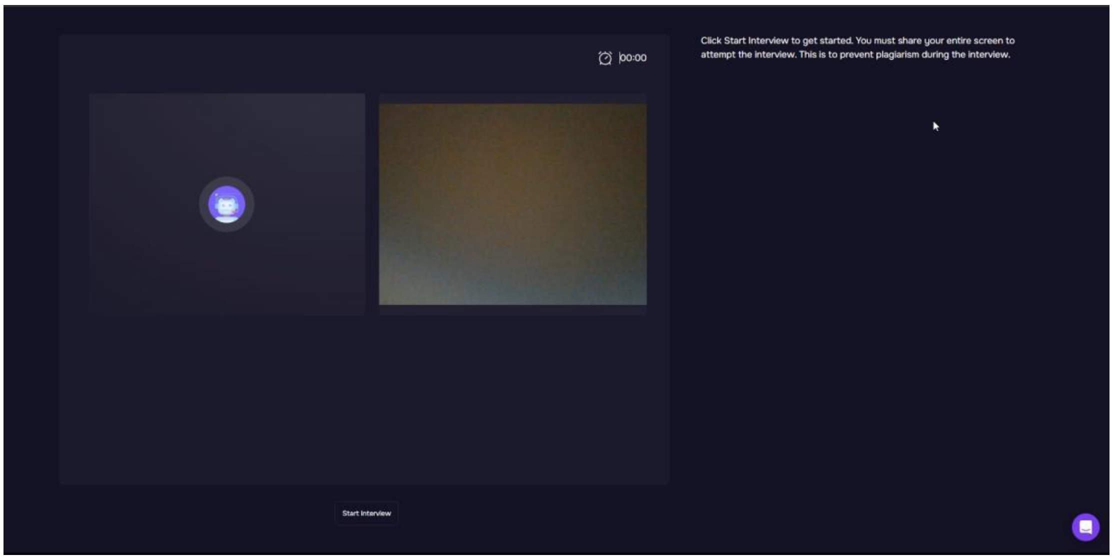
Figure 2.1.1.6 Auto generate email page

For candidates who are not selected, Interviewer.ai will send an auto-generated email, as shown in Figure 2.1.1.6 which allows recruiters to efficiently communicate rejections, streamlining the process and ensuring that all candidates receive timely feedback. This feature not only saves time but also maintains a professional and courteous communication flow with all applicants, enhancing the company's employer brand.

In summary, Interviewer.ai offers a robust platform with a variety of features designed to streamline and enhance the recruitment process. From creating detailed job profiles and selecting relevant interview questions to analysing candidate data and automating communication, the platform addresses many common challenges faced by recruiters. By providing structured tools and insights, Interviewer.ai enables more informed decision-making and contributes to a more efficient and effective hiring process. However, improvements in user-generated content management and further enhancement of candidate interaction could elevate the platform's utility even further.

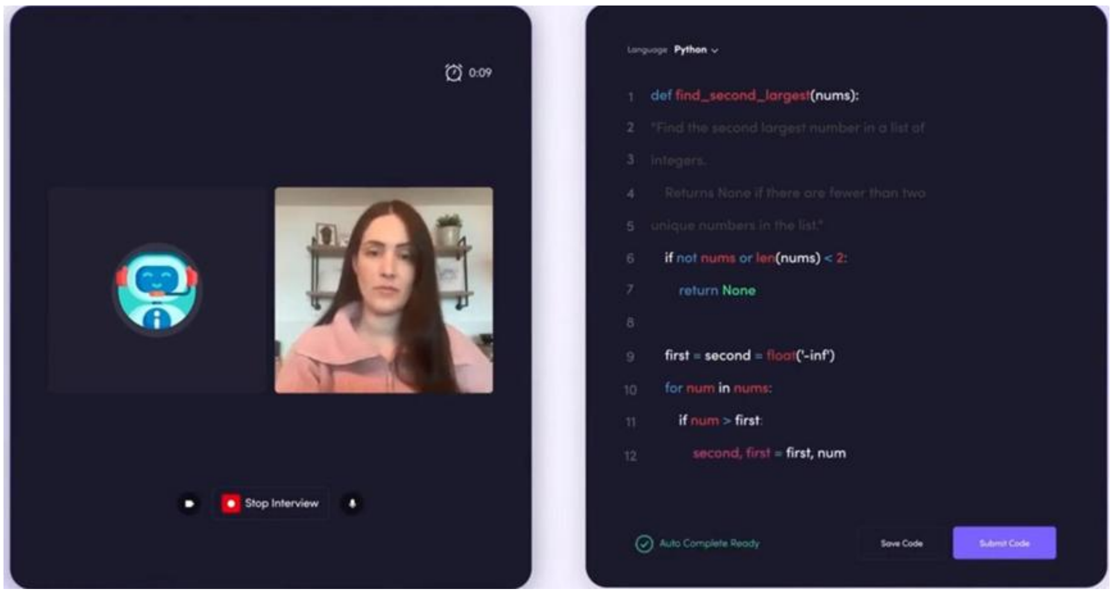
2.1.2 Talently.ai

Talently.ai [6] is a platform designed with a strong emphasis on flexibility, integrity, and efficiency in the interview process. Unlike traditional scheduling systems, Talently.ai allows candidates to begin their interview sessions at their convenience without needing to set an appointment. This on-demand approach ensures that the interview process is accessible and accommodating to candidates from various time zones and with different schedules, thereby broadening the talent pool. One of Talently.ai's standout features is its rigorous anti-cheating mechanism, which requires candidates to share their entire screen before starting the interview, as shown in Figure 2.1.2.1. This feature is particularly effective in preventing candidates from browsing unauthorized websites or using external resources during the interview, thereby maintaining the integrity of the assessment.



Click Start Interview to get started. You must share your entire screen to attempt the interview. This is to prevent plagiarism during the interview.

Figure 2.1.2.1 Share screen reminder before interview start

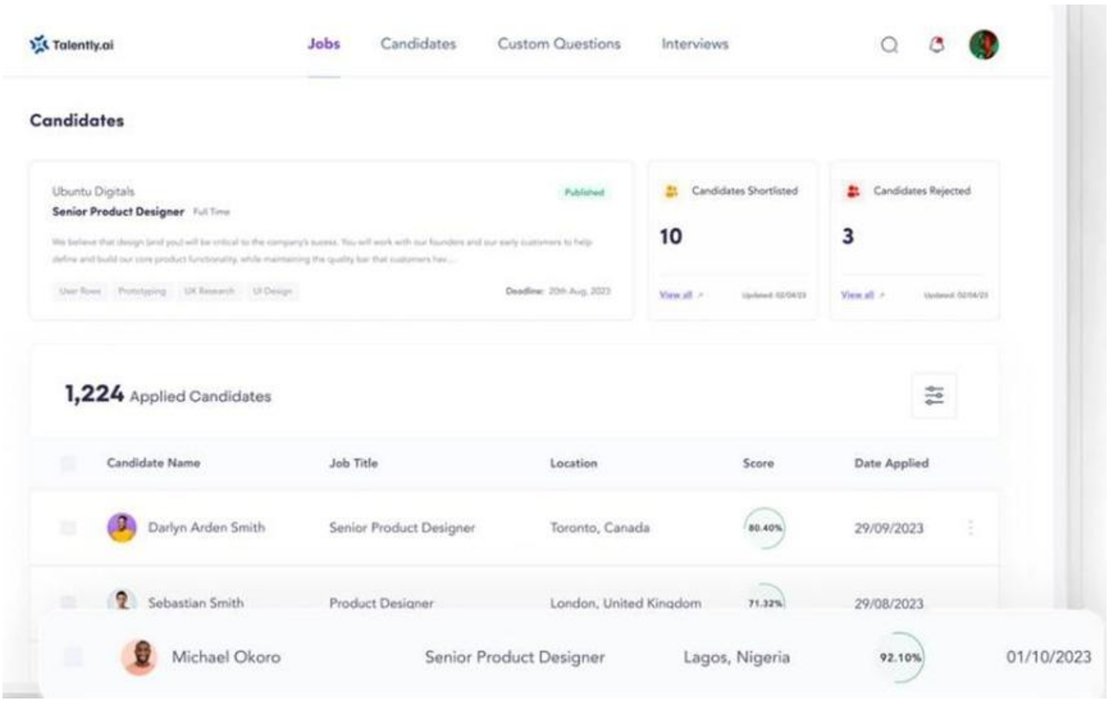


```python
def find_second_largest(nums):
"Find the second largest number in a list of
integers.
  Returns None if there are fewer than two
unique numbers in the list."
  if not nums or len(nums) < 2:
    return None

  first = second = float('-inf')
  for num in nums:
    if num > first:
      second, first = first, num
```

Figure 2.1.2.2 Shared screen technical test in talently.ai

As shown in Figure 2.1.2.2, the platform includes coding or logic-based challenges, depending on the job role. The AI evaluates answers in real-time, checking syntax, logic flow, and problem-solving approach.

This method shifts the focus from surface-level responses to real performance. It reduces bias by prioritizing skill over personality or presentation style. Recruiters can then view the top-performing candidates through a ranked list, as seen in Figure 2.1.2.3, which simplifies decision-making.



Candidates

Ubuntu Digitals
Published
Senior Product Designer Full Time
We believe that design (and you) will be critical to the company's success. You will work with our founders and our early customers to help define and build our core product functionality, while maintaining the quality bar that customers hav...
User flows · Prototyping · UX Research · UI Design
Deadline: 20th Aug, 2023

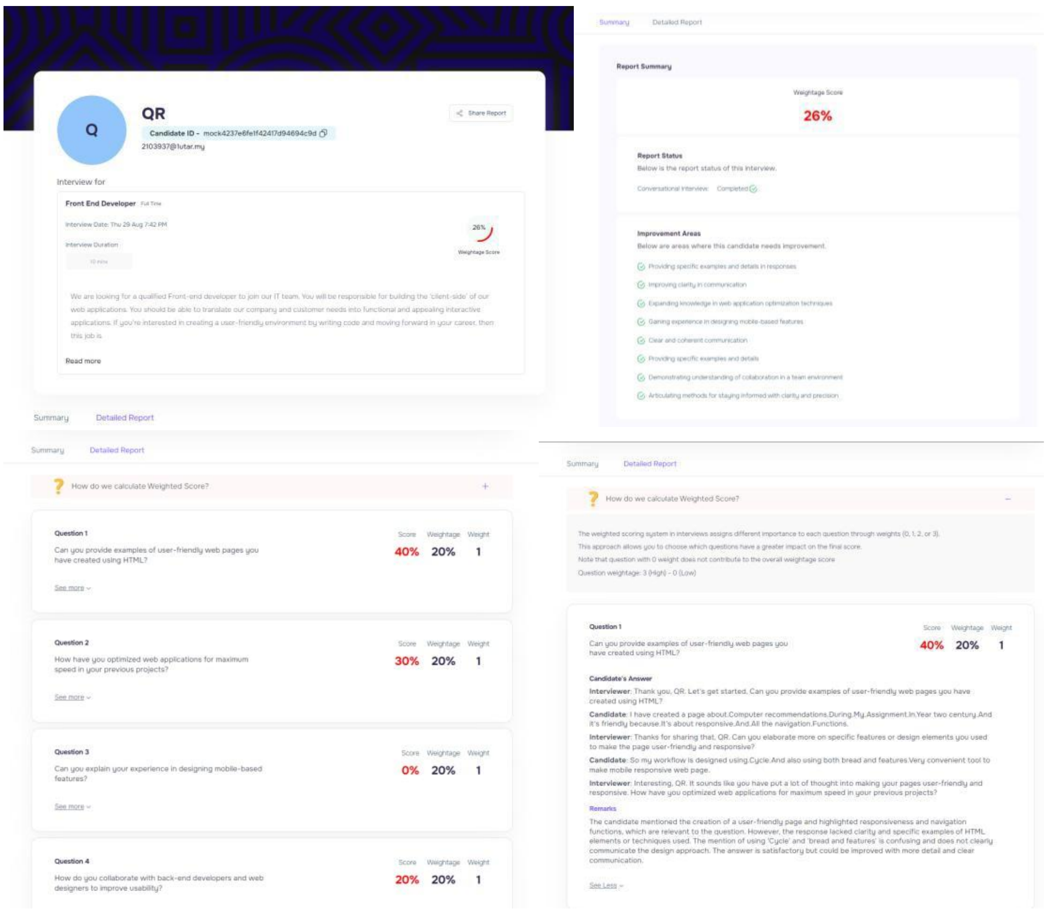
Candidates Shortlisted: 10 — View all — Updated 02/04/23
Candidates Rejected: 3 — View all — Updated 02/04/23

1,224 Applied Candidates

Candidate Name	Job Title	Location	Score	Date Applied
Darlyn Arden Smith	Senior Product Designer	Toronto, Canada	80.40%	29/09/2023
Sebastian Smith	Product Designer	London, United Kingdom	71.32%	29/08/2023
Michael Okoro	Senior Product Designer	Lagos, Nigeria	92.10%	01/10/2023

Figure 2.1.2.3 Shortlisted candidate overview in talently.ai

While Talently.ai excels in testing job-related skills and upholding fairness, it lacks some of the deeper insights that platforms like Interviewer.ai provide. For example, Interview.ai breaks down traits such as communication, sociability, and energy. These soft skills are important in many roles and help recruiters form a complete view of a candidate. Talently.ai, in contrast, mainly scores based on task responses. This approach works well for technical roles but may fall short when evaluating interpersonal strengths or leadership potential.


QR

Candidate ID - mock4237e6fe1f42417d94694c9d

2103937@1utar.my

Share Report

Interview for

Front End Developer Full Time

Interview Date: Thu 29 Aug 7:42 PM

Interview Duration

10 mins

26% Weightage Score

We are looking for a qualified Front-end developer to join our IT team. You will be responsible for building the 'client-side' of our web applications. You should be able to translate our company and customer needs into functional and appealing interactive applications. If you're interested in creating a user-friendly environment by writing code and moving forward in your career, then this job is

Read more

Summary Detailed Report

Report Summary

Weightage Score

26%

Report Status

Below is the report status of this interview.

Conversational Interview: Completed ✓

Improvement Areas

Below are areas where this candidate needs improvement.

- ✓ Providing specific examples and details in responses
- ✓ Improving clarity in communication
- ✓ Expanding knowledge in web application optimization techniques
- ✓ Gaining experience in designing mobile-based features
- ✓ Clear and coherent communication
- ✓ Providing specific examples and details
- ✓ Demonstrating understanding of collaboration in a team environment
- ✓ Articulating methods for staying informed with clarity and precision

Summary Detailed Report

How do we calculate Weighted Score?

Question	Score	Weightage	Weight
Question 1: Can you provide examples of user-friendly web pages you have created using HTML?	40%	20%	1
Question 2: How have you optimized web applications for maximum speed in your previous projects?	30%	20%	1
Question 3: Can you explain your experience in designing mobile-based features?	0%	20%	1
Question 4: How do you collaborate with back-end developers and web designers to improve usability?	20%	20%	1

See more

How do we calculate Weighted Score?

The weighted scoring system in interviews assigns different importance to each question through weights (0, 1, 2, or 3).

This approach allows you to choose which questions have a greater impact on the final score.

Note that question with 0 weight does not contribute to the overall weightage score

Question weightage: 3 (High) - 0 (Low)

Question 1

Can you provide examples of user-friendly web pages you have created using HTML?

Score 40% Weightage 20% Weight 1

Candidate's Answer

Interviewer: Thank you, QR. Let's get started. Can you provide examples of user-friendly web pages you have created using HTML?

Candidate: I have created a page about.Computer recommendations.During.My.Assignment.In.Year two century.And it's friendly because.It's about responsive.And.All the navigation.Functions.

Interviewer: Thanks for sharing that, QR. Can you elaborate more on specific features or design elements you used to make the page user-friendly and responsive?

Candidate: So my workflow is designed using.Cycle.And also using both bread and features.Very convenient tool to make mobile responsive web page.

Interviewer: Interesting, QR. It sounds like you have put a lot of thought into making your pages user-friendly and responsive. How have you optimized web applications for maximum speed in your previous projects?

Remarks

The candidate mentioned the creation of a user-friendly page and highlighted responsiveness and navigation functions, which are relevant to the question. However, the response lacked clarity and specific examples of HTML elements or techniques used. The mention of using 'Cycle' and 'bread and features' is confusing and does not clearly communicate the design approach. The answer is satisfactory but could be improved with more detail and clear communication.

See Less

Figure 2.1.2.4 talently.ai interview report

Another key difference is in how the platforms handle candidate feedback. Unlike Interviewer.ai, which requires recruiters to manually share interview reports, Feedback in Talently.ai is instant. As soon as the interview ends, candidates receive a report by email, as shown in Figure 2.1.2.4. This allows them to reflect on their performance right away, helping them prepare better for future interviews. However, these reports are not as detailed as those from Interviewer.ai. Instead of giving scores across multiple traits, Talently.ai either scores individual questions or gives a general performance summary. This limits how much insight the recruiters and candidates can gain from the results.

In summary, Talently.ai is a strong platform for roles that require technical accuracy and independent problem-solving. Its flexible, on-demand interview system and real-time assessment make it a convenient choice for both recruiters and applicants. Its anti-cheating measures and fast feedback loop are major advantages. However, for roles that rely on soft skills or require a deeper personality analysis, it may not be the most comprehensive option. Even so, Talently.ai plays an important role in streamlining hiring processes—especially where technical performance is the top priority.

2.1.3 Apriora

Apriora [7] is an AI-powered interview platform that emphasizes both efficiency and candidate engagement, offering a range of features designed to streamline the interview process while maintaining a high level of interaction. This platform aims to create a more dynamic and responsive interview experience by integrating unique functionalities that distinguish it from other AI interview platforms such as Interviewer.ai and Talently.ai.

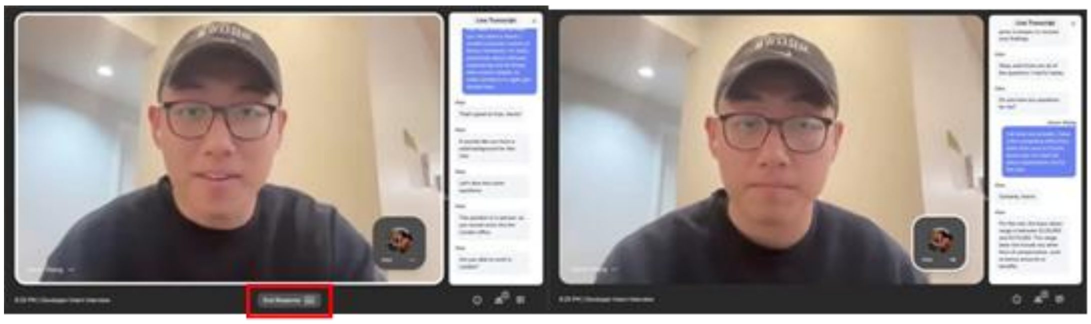


Figure 2.1.3.1 Apriora interview page

One of Apriora's standout features, as shown in Figure 2.1.3.1, is the convenience it offers during the interview session. Unlike other platforms, Apriora allows candidates to control the recording process by simply pressing the space bar to stop recording after they finish answering a question. This functionality is not available in the other platforms mentioned earlier, where the system automatically detects when a candidate has finished speaking. By giving candidates direct control, Apriora not only saves time during the interview process but also reduces the risk of the system prematurely ending a recording due to misdetection. This small yet significant feature contributes to a smoother and more efficient interview experience.

In addition to this, Apriora promotes a more interactive interview environment by allowing candidates to ask questions before the interview concludes. This feature fosters a two-way communication channel, enabling candidates to seek clarification or express any concerns they might have, which can be particularly beneficial for roles that require strong communication and interpersonal skills.

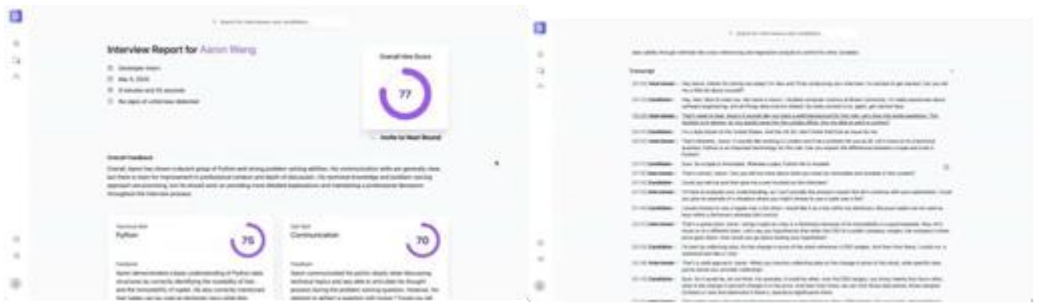


Figure 2.1.3.2 Interview report of Apriora

Similar to Talently.ai, Apriora provides candidates with their interview report immediately after the session concludes. This instant feedback mechanism allows candidates to quickly review their performance and gain insights into areas where they can improve. However, Apriora goes a step further by delivering a full transcript of the interview session, which makes it easier for recruiters to review the content in detail. It looks at both job-specific skills and broader qualities like communication and analytical thinking. This targeted evaluation helps recruiters focus on the candidates who are most likely to succeed in the role.

In short, Apriora combines smart automation with human-friendly features. It gives candidates more control, promotes interaction, and offers valuable feedback. These qualities help set it apart from other AI interview platforms.

2.1.4 Braintrust

Braintrust [8] is a decentralized talent network that uses blockchain technology to connect freelancers with companies in a transparent, efficient, and user-empowered ecosystem. Unlike traditional AI interview platforms, Braintrust redefines recruitment by offering a decentralized model that prioritizes trust, control, and community participation.

One of the standout features of Braintrust is its sophisticated talent-matching algorithm. The platform uses advanced algorithms to pair freelancers with job opportunities that align with their skills and experience. This AI-driven matching process is particularly beneficial for ensuring that candidates are evaluated based on their fit for the role, streamlining the hiring process and improving outcomes for both freelancers and companies. The use of AI in this context could extend to matching candidates with interview questions and assessments that best evaluate their suitability for the role, further enhancing the efficiency of the recruitment process.

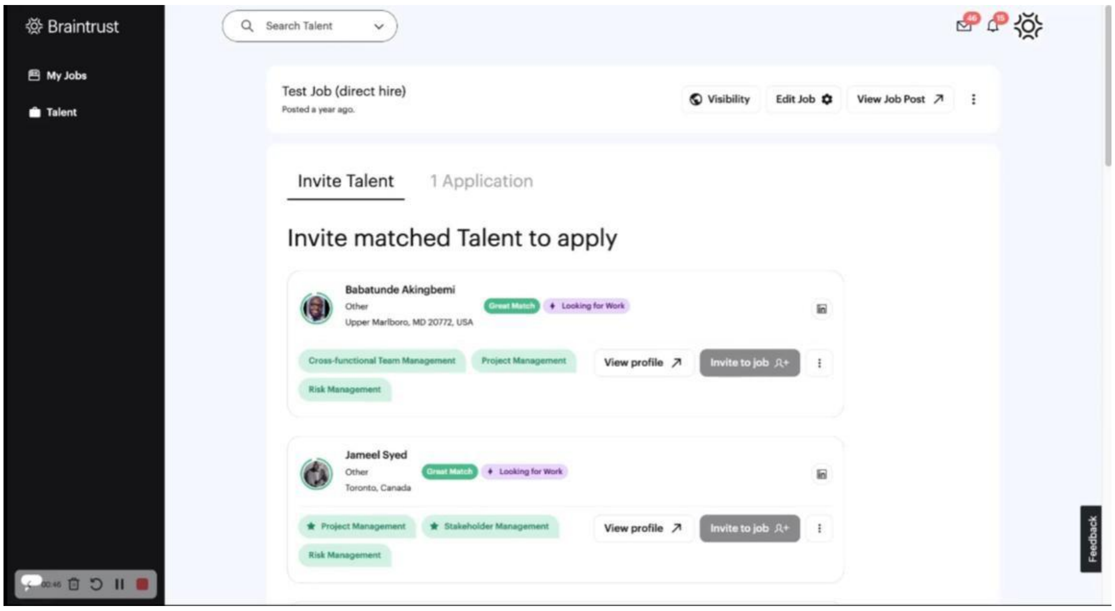
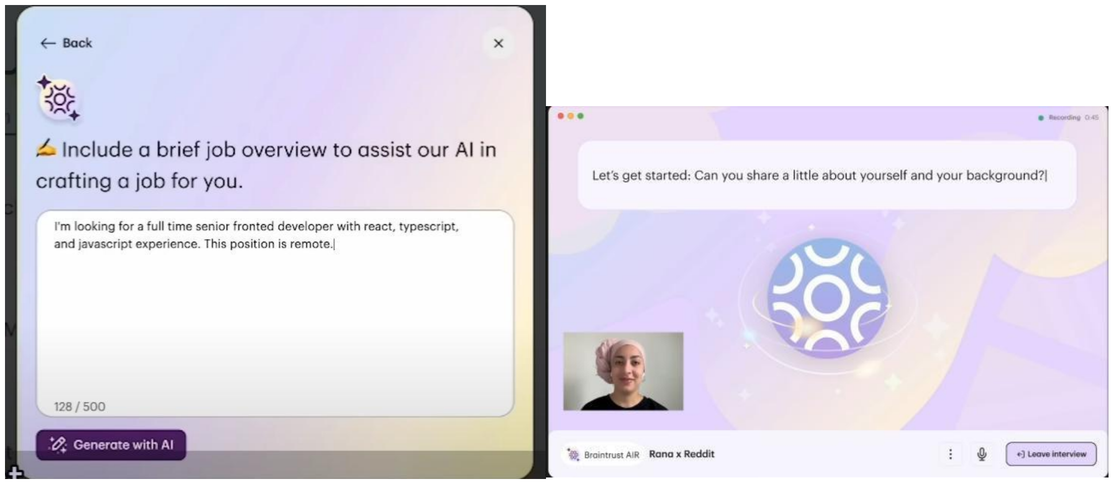


Figure 2.1.4.1 Talent matching algorithm in Braintrust

Braintrust also emphasizes user empowerment, giving freelancers greater control over their work environment. Freelancers can set their own rates, select projects that resonate with their interests, and participate in the platform's governance. This level of personalization not only enhances user satisfaction but could also make the interview process more engaging and candidate friendly.

However, access to Braintrust's AI-driven interview capabilities is available only through a subscription to Braintrust AIR. As depicted in Figure 2.1.4.1, hiring managers can utilize Braintrust AIR to automate the job description creation process, with AI writing the full job description and posting it to the Braintrust job board. Once applications are received, Braintrust AIR analyses them to determine which candidates should proceed to the interview stage. It then schedules and conducts live interviews with the selected applicants.



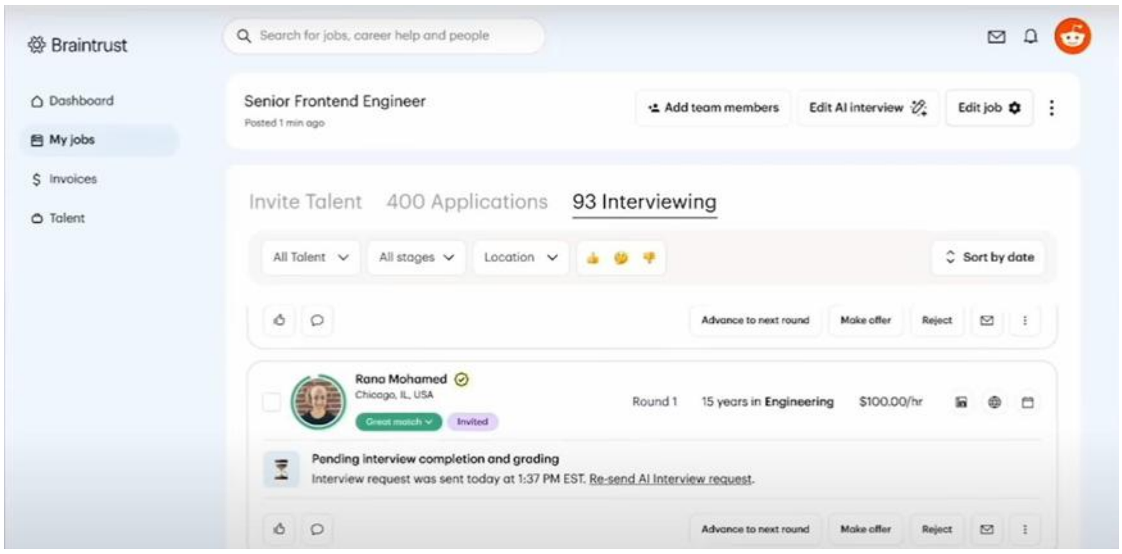
✍ Include a brief job overview to assist our AI in crafting a job for you.

I'm looking for a full time senior fronted developer with react, typescript, and javascript experience. This position is remote.

Let's get started: Can you share a little about yourself and your background?

Figure 2.1.4.2 AI generate job with user description and the AI interview interface in Braintrust AIR

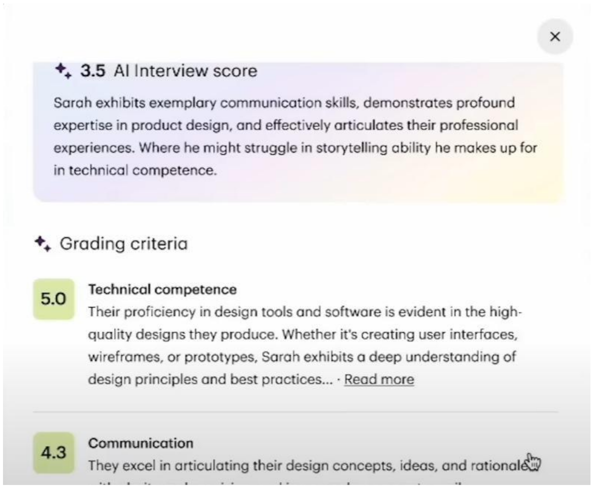
During the interview, Braintrust AIR generates questions tailored to the specific role and the applicant's background. This ensures that the interview is not only relevant but also thorough, covering all necessary aspects of the candidate's qualifications.



Pending interview completion and grading
Interview request was sent today at 1:37 PM EST. Re-send AI Interview request.

Figure 2.1.4.3 AI filtering feature in Braintrust AIR

After the interview session, Braintrust AIR generates a scorecard for each candidate, as shown in Figure 2.1.4.4. This scorecard is based on the applicant's performance during the interview and provides hiring managers with a clear, data- driven assessment of each candidate. The scorecard includes a grading criterion and a short description of the applicant's answers, helping hiring managers make informed decisions about advancing candidates to the next round, extending an offer, or rejecting the applicant.



3.5 AI Interview score

Sarah exhibits exemplary communication skills, demonstrates profound expertise in product design, and effectively articulates their professional experiences. Where he might struggle in storytelling ability he makes up for in technical competence.

Grading criteria

5.0 **Technical competence**
Their proficiency in design tools and software is evident in the high-quality designs they produce. Whether it's creating user interfaces, wireframes, or prototypes, Sarah exhibits a deep understanding of design principles and best practices... · Read more

4.3 **Communication**
They excel in articulating their design concepts, ideas, and rationale

Figure 2.1.4.4 Generated scorecard after interview session in Braintrust AIR

In summary, Braintrust stands out as a forward-thinking platform that leverages blockchain technology and AI to create a transparent, efficient, and user-centric talent network. Its decentralized approach and advanced talent-matching algorithms make it a powerful tool for connecting freelancers with suitable job opportunities. However, its reliance on blockchain, focus on the tech industry, and subscription-based access to AI- driven features present challenges that may limit its broader applicability. Despite these limitations, Braintrust's innovative model offers a glimpse into the future of decentralized talent networks.

2.2 Limitation of Previous Studies

AI-driven recruitment tools have made major strides, but they still fall short in several key areas. Each platform brings something useful to the table, but they also carry limitations that reduce their overall effectiveness. Below is a breakdown of the most common issues found across some popular platforms.

Interviewer.ai has several limitations, particularly in its interaction and transparency with candidates. The platform does not allow candidates to ask questions during the interview, which could limit the depth of the interaction. Additionally, candidates are not able to view their interview results, which could lead to a lack of transparency and feedback. Besides, the transcript is only available alongside the recorded video, which can make it challenging for recruiters to review the content thoroughly. Moreover, the platform lacks features to prevent dishonest practices, such as screen sharing or tab switching checkers.

Talently.ai does a good job with cheating prevention and flexible scheduling. However, it falls short in documentation and feedback. Recruiters can't generate detailed reports from the platform, which makes tracking and comparing candidates harder. It also doesn't assess resumes—only the interview performance is considered. This limits how well the platform captures a full picture of the applicant. Candidates also face several limitations. They can't ask follow-up questions, view their scores, or get any feedback through email. Another issue is that the platform doesn't allow recruiters to set time limits for individual questions, which could lead to inconsistencies during evaluations.

Apriora has limitations in terms of video review and overall candidate analysis. The platform does not segment video recordings by question, which can make it difficult for recruiters to navigate and review specific parts of the interview. It also does not offer an overall analysis of all candidates' results, focusing instead on individual performance without providing a comprehensive overview. The platform does not support the auto-generation of emails to inform candidates about their results, nor does it allow for setting time limits on questions, which might affect the consistency of evaluations.

While Braintrust offers a decentralized and transparent approach to talent matching, it is not without its challenges. The complexity of its decentralized system may be intimidating for users unfamiliar with blockchain technology, potentially deterring them from fully utilizing the platform. Additionally, Braintrust's current focus on the tech industry limits its applicability across other sectors, reducing its versatility as a comprehensive talent-matching solution. Moreover, Braintrust's reliance on user participation in governance is a double-edged sword. While it empowers users by giving them a voice in platform decisions, the success of this model depends heavily on active and informed participation. If users are not fully engaged or lack the necessary knowledge to participate effectively, the platform's governance and decision-making processes could be hindered. Furthermore, while the AI-powered features of Braintrust AIR are impressive, they come at a cost, as access to these advanced tools requires a subscription, which could be a barrier for smaller companies or independent freelancers.

These limitations across the platforms underline the importance of a well- rounded, user-friendly, and transparent AI-powered interview system, which the proposed IntelliHire project aims to address by integrating and improving upon the strengths of these existing tools.

2.3 Comparison of Reviewed System

Table 2.3.1 Comparison Result

Feature	**Interviewer.ai**	**Talently.ai**	**Apriora**	**Braintrust**
User-friendly	Yes	Yes	Yes	Yes
Free to use	No	No	No	No
Avatar	N/A	N/A	N/A	N/A
Device	Desktop/ Laptop	Desktop/ Laptop	Desktop/ Laptop	Desktop/ Laptop
Effective Interaction	No (candidates cannot ask questions)	No (candidates cannot ask questions)	Yes (candidates can ask questions)	Yes (user control and participation)
Strength	- Detailed candidate reports - Automated data export - Comprehensive interviewanalysis	- Flexible interview scheduling - Anti-cheating measures - Immediate feedback	- Candidate controls recording - Interactive interview environment - Immediate feedback with full transcript	- Transparency and empowerment - Efficient talent matching
Weakness	- Limited interaction - Transcript only availablewith video - No anti-cheating measures	- Lack of comprehensivereports - No resume scoring - Limited transparency forcandidates	- No segmented video review - Focuses on individual analysis only - No automated emails	- Complex for non-tech-savvy user - Limited industry scope - User participation required

2.4 Proposed Solutions

This project introduces a new AI-powered interview platform designed to bring together the best parts of existing systems—like Interviewer.ai, Talently.ai, Apriora, and Braintrust—while fixing their weaknesses. The goal is to create a complete, smart, and user-friendly solution that works better for both recruiters and candidates. Here is how the proposed system improves upon current tools:

1. **Enhanced Candidate Interaction and Transparency**: The platform will promote better interaction between candidates and the system. Like Apriora, candidates will be allowed to ask questions during the interview. This supports a more engaging, two-way experience. In addition, candidates will be able to view their results afterward. This adds transparency and helps them understand how they performed.
2. **Detailed Reports and Easy-to-Read Transcripts**: To go beyond what Talently.ai and Braintrust offer, the platform will provide detailed documentation for both resume screening and interview performance. After each interview, candidates will be able to view their performance. Transcripts will also be shared in a clear format, separated from the video files, making it easier for both candidates and recruiters to review them.
3. **Improved Video Review Capabilities**: Interviewer.ai allows video interviews to be broken into parts. This makes reviewing easier. This platform takes that idea further. Recruiters will be able to jump directly to specific questions in the video. They can also read the full transcript at the same time. By combining video playback with transcript viewing, Apriora does, recruiters can evaluate candidates more efficiently and with better focus.
4. **Flexible Scoring and Candidate Insights**: The scoring system will be fully customizable. Recruiters can set criteria that match both job-specific skills and broader qualities like communication or problem-solving. After interviews are complete, the system will offer an overall analysis of all applicants. This will help recruiters compare candidates more effectively and make smarter, data-driven hiring decisions.

By integrating these features, the proposed solution aims to create a more comprehensive, transparent, and effective AI-powered interview platform that addresses the limitations of existing tools while leveraging their strengths.

2.5 Summary

The review has passed through four AI interview platforms: Interviewer.ai, Talently.ai, Apriora, and Braintrust. Each platform has unique features while exposing some limitations. Interviewer.ai is good at detailed candidate reports and analysis but poor in interactive elements. Talently.ai is good at anti-cheating features and customizable scheduling but poor in extensive documentation. Apriora is good at superior candidate control and interaction but lacking in segmented video reviews. Braintrust employs blockchain technology for truthful talent matching but exposes complexity issues for non-tech users.

The comparative review also revealed key shortcomings shared by the existing systems, such as restrictions on candidate communication, limited transparency levels, inadequate documentation, and insufficient anti-cheating capabilities. Against these background, the proposed IntelliHire platform aims to complement the functionality of existing platforms by addressing their flaws through enhanced candidate interaction, extensive reports, advanced video review functionality, robust anti-cheating capabilities, and flexible score mechanism. This synergistic approach intends to create an improved, transparent, and easy-to-use AI-powered interview platform that can both benefit recruiters and candidates.

Chapter 3 System Methodology and Model

3.1 System Design

The architecture employed is a client-server model where the client (web browser) engages with the server-side application created with Laravel. Within this model, there are two categories of clients: the Recruiter portal and the Candidate portal. Each portal serves as an interface that permits users to interact with the system according to their specific roles.

The Recruiter Portal allows recruiters to create job postings, review candidate applications, set interview schedules, and monitor interview results. The Candidate Portal allows candidates to browse available jobs, submit applications with uploaded resumes and cover letters, and attempt interviews once scheduled.

Both portals communicate with a backend server, which processes client requests, applies the required business logic, and interacts with a local MySQL database (running on XAMPP) for data storage and retrieval. In addition, the backend server interacts with external APIs, especially the Gemini API, which is used to generate interview questions based on the provided job information.

The server-side application uses the Model-View-Controller (MVC) design pattern internally, which is a fundamental characteristic of the Laravel framework. Models handle data management and interact with the database, views display information to users via interfaces, and controllers serve as middlemen, managing user input, executing business logic, and producing suitable responses.

This separation of concerns improves the maintainability and extensibility of the code and simplifies the update and debugging process.

By combining the client-server architecture for external communication and the MVC design pattern for internal code structure, the system ensures a clear, orderly, and efficient development and operation process.

3.1.1 System Design Diagram

The system design diagram shows the interaction between the recruiter portal, candidate portal, backend server, database, and external API.

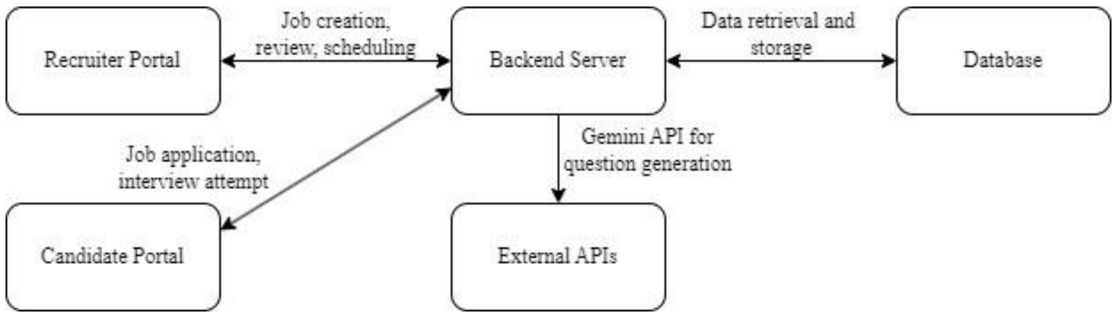

Recruiter Portal
Job creation, review, scheduling
Backend Server
Data retrieval and storage
Database
Job application, interview attempt
Gemini API for question generation
Candidate Portal
External APIs

Figure 3.1.1.1 System Design Diagram illustrating the interaction between Recruiter Portal, Candidate Portal, Backend Server, Database, and External APIs in IntelliHire.

3.2 System Architecture

IntelliHire follows a well-structured client-server architecture with different responsibilities assigned to various system layers. This approach ensures modularity, extensibility, and ease of maintenance, which are important for support future upgrades and a larger user base. Although IntelliHire is built on Laravel, a framework that follows the Model-View-Controller (MVC) pattern internally, its overall architecture operates on a client-server model at a broader level, incorporating database management and external API services.

The IntelliHire system is made up of five main components:

- Users
- Frontend
- Backend Server (Laravel Framework)
- Database Layer (MySQL)
- External APIs (Gemini API)

Each component plays different role, working together to support a smooth and efficient recruitment process. These elements connect users with application logic, storage, and external services.

3.2.1 User

IntelliHire serves two main types of users: recruiters and candidates. Recruiters are in charge of the hiring process. They create job listings, review applications, shortlist candidates, schedule interviews, and update application statuses. Candidates use the system to look for jobs, submit their applications, attend AI-powered interviews, and track their application results. The system tailors the experience based on the role of the user so that both groups get tools suitable to their needs.

3.2.2 Frontend

Frontend layer comprises two distinct portals, each designed to cater the requirements of a particular set of users:

1. Recruiter Portal
2. Candidate Portal

Both portals are the primary means for users to interact with the IntelliHire system.

In the recruiter portal, users can post jobs, monitor applications, schedule interviews, and view interview feedback. Recruiters also get insights by way of performance summaries and analytics.

The candidate portal allows the candidates to register, complete their profiles, search for jobs, apply for jobs, and attend automated interviews. They can also view the results of their applications.

The frontend communicates with the backend server using HTTP-based API calls. It sends data collected from users (e.g., applications and interview responses) and retrieves information (e.g., job listings and interview scheduling) by securely interacting with the Laravel backend. Separating the frontend and backend ensures a more responsive and dynamic user experience.

3.2.3 Backend Server (Laravel Framework)

The core of application logic, the Laravel backend server, is used to manage the interaction between the frontend, database, and external APIs.

Internally, Laravel employs the MVC (Model-View-Controller) pattern, in which the application logic is divided into three primary layers: data is managed by models, presentation is managed by views, and logic that binds models and views together is managed by controllers. This internal structure makes the backend highly organized and manageable.

At the system level, the Laravel backend acts as a middleman, handling client requests from both portals, applying business logic, and communicating with the database for data storage and retrieval.

The specific responsibilities of the backend include:

1. Manage job postings created by recruiters, ensuring they are properly stored and updated in the database.
2. Process candidate applications, record submitted information, and update application status.
3. Store and retrieve candidate interview responses and feedback data.
4. Manage interview scheduling between recruiters and candidates.
5. Prepare and format data to be sent to the Gemini API in order to generate customized interview questions based on specific job postings.

The backend ensures that all business rules are enforced, validates data, manages authentication and authorization, and secure communication. It is the intellectual component of the system, tasked with directing all the activities to ensure consistency and reliability.

3.2.4 Database layer (MySQL)

The IntelliHire system implements a MySQL database to keep the core data of the system in the storage. The database is hosted locally using XAMPP during the development phase so that it's readily accessible along with schema and data management.

Each of the tables gets associated with other tables through structured relationship to ensure data integrity along with optimizing processes for queries. The database layer plays a major role in providing real-time facilities for front-end and back-end activities.

3.2.5 External APIs (Gemini API)

One of the standout features of IntelliHire is its deep integration with the Gemini API, which facilitates multiple stages of the hiring process. The most visible application is in the generation of interview questions, where Gemini receives job descriptions and returns customized sets of questions that are match for each role. This automation saves recruiter time, ensures consistency, and offers a more engaging candidate experience.

Aside from question generation, the Gemini API also powers an interview chat system in real time. By maintaining conversational context, the system can ask follow-up questions and provide dynamic responses, making the interview process more natural. Gemini also performs resume parsing and analysis, extracting text from different file formats and translating it into structured information such as skills, education, and experience. This makes candidates easier to compare for recruiters.

On the employer side, the API is used for job requirements parsing, whereby the content of job postings is parsed to find skills, qualifications, and certifications. The structured data feeds AI-driven job matching, whereby Gemini goes beyond simple keyword comparisons to align applicants with jobs based on semantic understanding, context, and skill equivalence.

At the assessment and interview stages, Gemini supports response evaluation via analysis of candidate answers, completeness assessment, clarity, and communication, followed by the generation of feedback, scores, and suggestions. It also evaluates answers to assessment questions and discriminates between content issues and possible speech-to-text processing transcription errors.

Finally, all the AI insights, like scores from evaluations, feedback, and suggestions—remains in AI data models within the system. The records maintain structured metrics like confidence levels, improvement suggestions, and suggestions for hiring, providing recruiters with solid decision support throughout the recruitment process.

3.2.6 System Architecture Diagram

The following diagram illustrates how the different system components are connected and interact with each other.

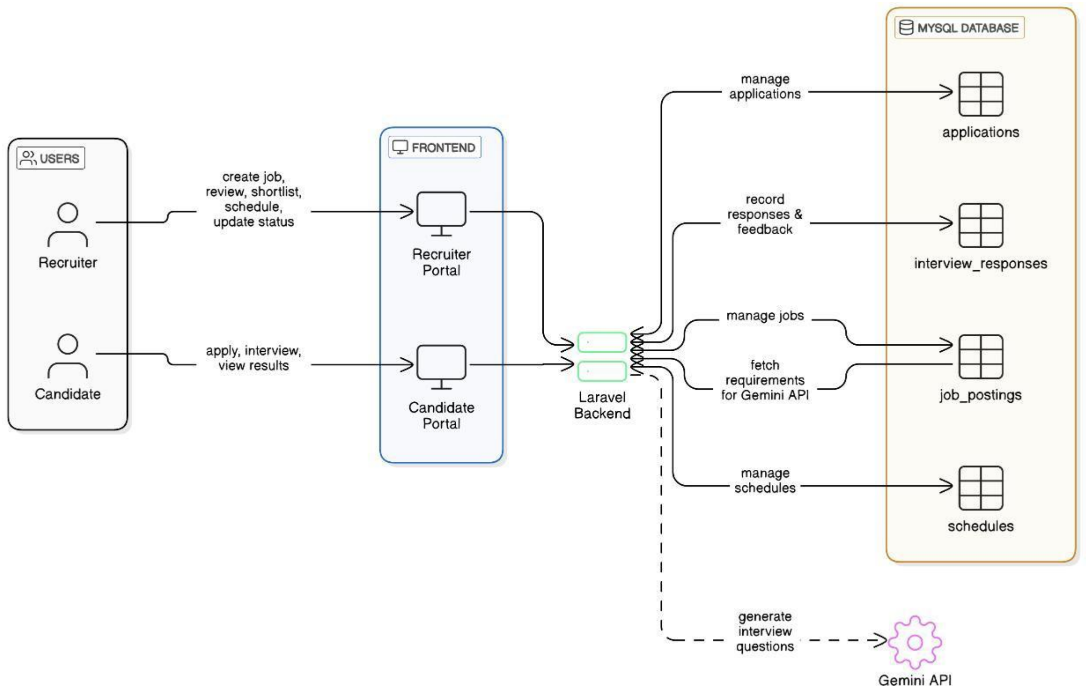


Figure 3.2.6.1 System architecture diagram

3.3 System Flow

The IntelliHire platform guides users through the recruitment process based on their roles. When users first enter the system, they log in or register. They also select whether they are a recruiter or a candidate. Based on this, they are taken to the matching portal.

In the recruiter portal, users can perform a variety of tasks including create job postings by entering all relevant details and optionally setting preset interview questions and assessment parameters. Recruiters can also manage existing job listings and review applications for specific positions. Within the application management module, recruiters can update application statuses to “shortlisted,” “rejected,” or “offered.” When a candidate is shortlisted, the recruiter sets a date range for interviews, which automatically generates a unique interview code.

On the candidate side, users can browse job listings, submit applications with a resume, cover letter, and optional notes, and track the status of their applications. If shortlisted, candidates gain access to a "Join Interview" function, which requires the generated code and must fall within the recruiter's scheduled date range. The system then checks for code validity and interview eligibility before launching the AI-powered interview interface. During the interview, questions are generated by Gemini API, either solely based on job and candidate data or with preset questions if configured. After the interview, recruiters review the candidate's responses and update the status accordingly. The workflow offers a fully automated, yet customizable, recruitment process from job posting to final hiring decision.

3.3.1 System Flow Chart

The system flowchart provides a clear indication of the user interactions with the platform, categorized into two main user types: candidates and recruiters. Upon entering the system, the user is prompted to register or log in. Based on the selected user type, users are redirected to the appropriate portal. Users can search for posted jobs, apply for jobs with supporting documents, and track the status of their application through the candidate portal. If shortlisted, they can proceed to an interview session using a unique code within a scheduled date range. The recruiter portal enables users to post new jobs, define interview questions and assessments, and manage applications. Recruiters can review applications, update statuses, and assess interview responses. The interview process is automated using the Gemini API, which generates interview questions dynamically. If preset questions are provided, these are passed into the API to guide the question generation; otherwise, the API relies on the job description and candidate profile to generate appropriate questions. This structured flow ensures seamless interaction and automation between application submission, interview handling, and final selection.

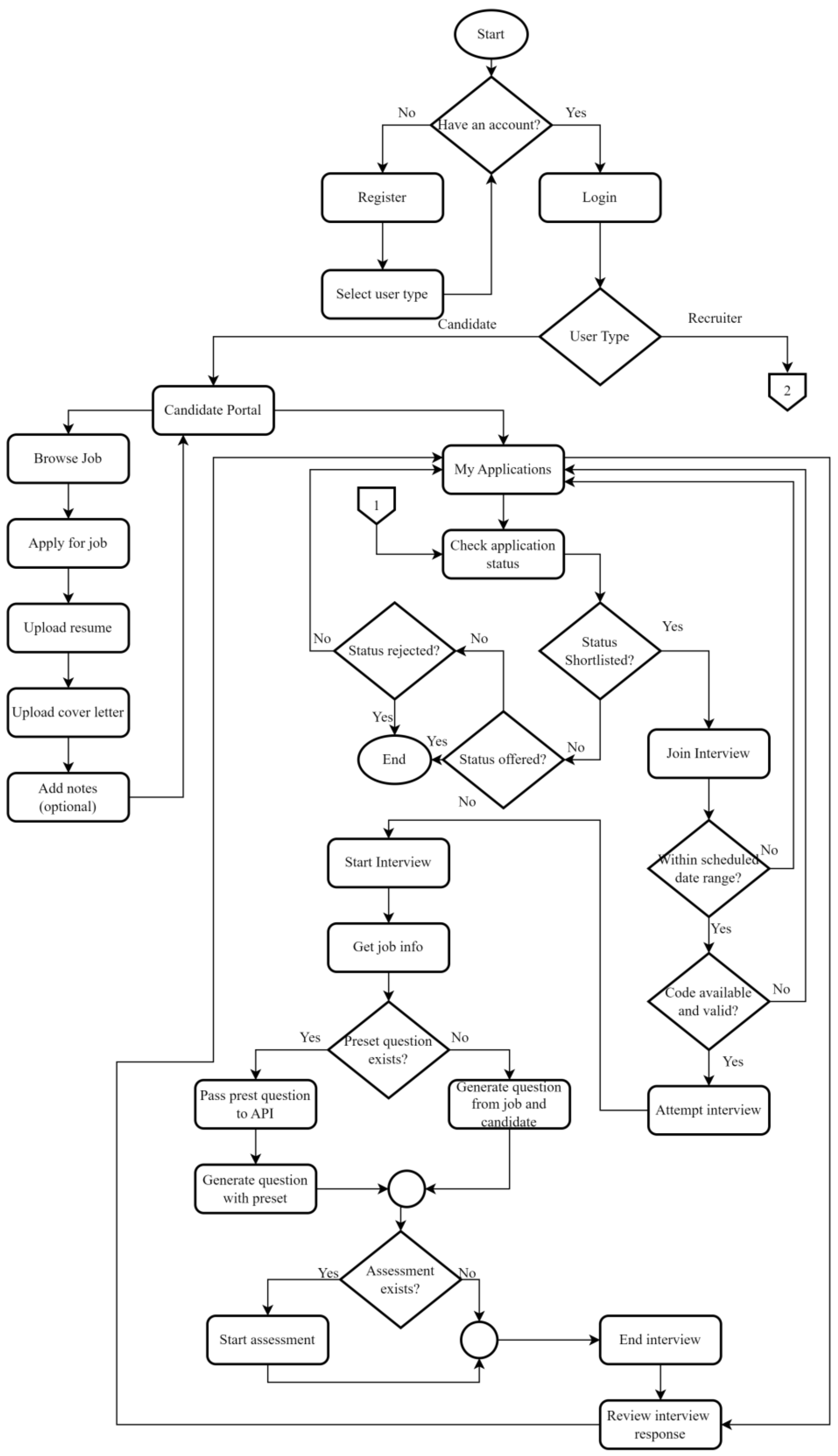


Figure 3.3.1.1 System flow chart (part 1)

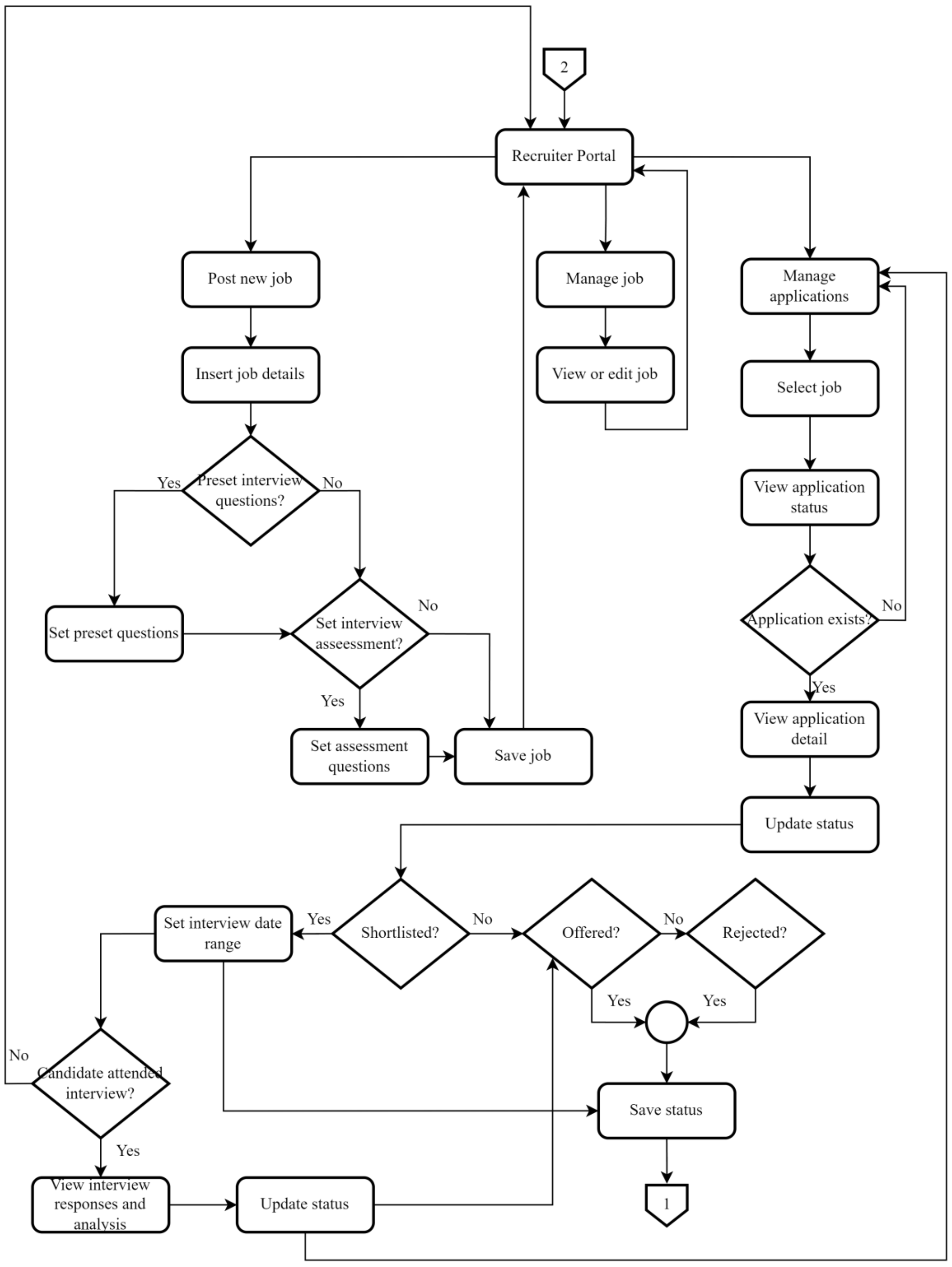


Figure 3.3.1.2 System flow chart (part 2)

3.4 System Use Case

There are two primary actors in the system. Both actors can login and register and select their user type, which determines the access path.

Candidate Use Cases:

After logging in, candidates can view available jobs, apply for jobs, and upload their resume and cover letter as part of the application process. They can also join interview sessions, during which questions may be generated automatically via AI. After the interview, candidates can view their interview status. Candidate can also receive job recommendations based on jobs they have not applied for. These features aim to streamline the application and interview process, while the inclusion of future capabilities like AI feedback and real-time transcription will further improve the user experience.

Recruiter Use Cases:

Recruiters can post, view the job applications, and view resumes and cover letters of the applicants. Recruiters can also define the application status and create or preset interview questions, which are used during interview sessions. These procedures help screen candidates more efficiently.

3.4.1 System Use Case Diagram

The use case diagram shows the interactions between the two primary actors, Candidate and Recruiter. Each actor is associated with a range of system functionalities reflecting real-world hiring workflows.

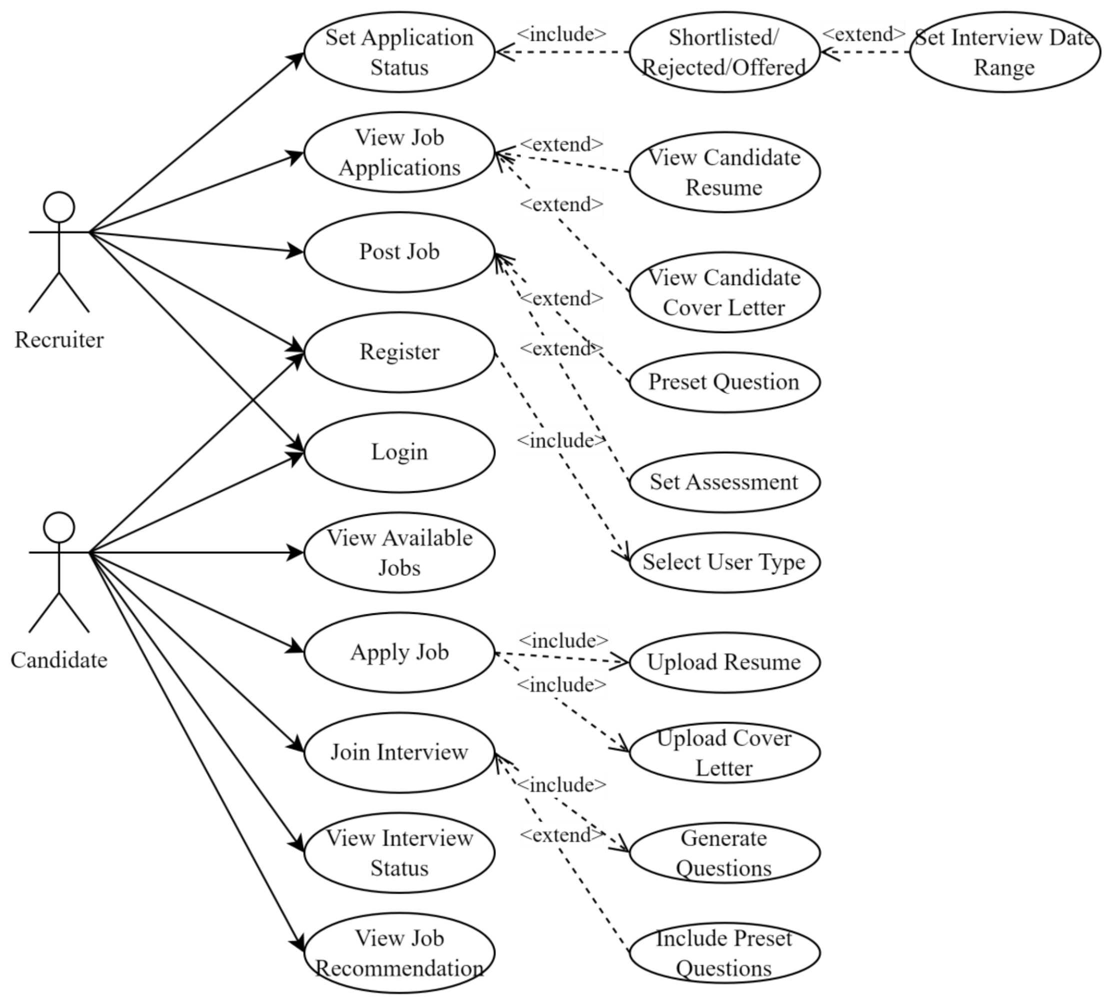


Figure 3.4.1.1 System use case diagram

3.4.2 Use Case Description

3.4.2.1 Use Case 1 – User Registration

Table 3.4.1 Use case description for User Registration

Use Case ID	UC001	**Version**	1.0
Use Case	User Registration		
Purpose	Allow a new user to create an account and access the Candidate/Recruiter portal.		
Actor	User (primary), System (supporting)		
Trigger	User clicks Register on the landing page.		

Precondition	User is not logged in and does not already have an account.	
Scenario Name	**Step**	**Action**
Main Flow	1	System displays the registration form.
	2	User enters name, email, password, and selects role.
	3	User submits the form.
	4	System validates required fields and formats (email, password length).
	5	System checks for duplicate email/username.
	6	System creates the account and initializes default profile.
	7	System confirms registration and redirects to Login.
Alternate Flow – Duplicate email	5.1	Duplicate is found.
	5.2	System blocks creation and shows “Email already registered” message.
	5.3	User is prompted to Login or use a different email.
Alternate Flow – Weak/Invalid Password	4.1	Password policy fails.
	4.2	System highlights the field and shows password rules.
	4.3	Candidate re-enters a valid password and resubmits.
Rules	− Email must be unique. − Password must meet minimum policy (length/complexity).	
Author	Tong Qian Ru	

3.4.2.2 Use Case 2 – Candidate Applies for Job

Table 3.4.2 Use case description for Candidate Applies for Job

Use Case ID	UC002	**Version**	1.0
Use Case	Candidate Applies for Job		
Purpose	Enable a candidate to submit an application with supporting documents for a selected job.		
Actor	Candidate (primary), Recruiter (secondary reviewer), System		
Trigger	Candidate selects a job and clicks Apply.		
Precondition	Candidate is logged in as Candidate; the job is open for applications.		

Scenario Name	**Step**	**Action**
Main Flow	1	System shows the application form for the selected job.
	2	Candidate uploads resume and (optional) cover letter.
	3	Candidate submits the application.
	4	System validates file types/sizes and required fields.
	5	System stores the application with status Pending.
	6	**Resume Parsing:** System extracts text and structured data (skills, education, experience).
	7	**Job Matching:** System compares parsed resume with parsed job requirements and computes a matching score.
	8	System records application, parsing output, and matching score.
	9	System notifies the candidate that the application was submitted successfully.
Alternate Flow – Duplicate Application	3.1	System detects an existing application by the same candidate for the same job.
	3.2	System blocks submission and shows "You have already applied to this job."
Alternate Flow – Invalid File Format	4.1	File format is unsupported.
	4.2	System rejects the file and prompts for PDF/DOCX.
	4.3	Candidate re-uploads and resubmits.
Alternate Flow – Parsing/Matching Fails	6.1/7.1	Parsing/matching temporarily fails.
	6.2/7.2	System queues a retry; application is still stored as Pending.
	6.3/7.3	System completes parsing/matching asynchronously and updates the record.
Rules	- One active application per candidate per job. - Resume must be PDF/DOCX within allowed size. - Applications default to **Pending** until recruiter action.	
Author	Tong Qian Ru	

3.4.2.3 Use Case 3 – Candidate Joins Interview

Table 3.4.3 Use case description for Candidate Joins Interview

Use Case ID	UC003		**Version**	1.0
Use Case	Candidate Joins Interview			
Purpose	Allow a shortlisted candidate to join a scheduled interview and complete the AI-led session (and assessment if configured).			
Actor	Candidate (primary), System, Gemini API (external service)			
Trigger	Candidate clicks Join Interview and enters the Interview Code.			
Precondition	Candidate is Shortlisted for the job; an interview date range and code have been set by the recruiter; current time is within the range.			
Scenario Name	**Step**	**Action**		
Main Flow	1	System prompts for Interview Code.		
	2	Candidate enters the code and submits.		
	3	System validates the code and checks the scheduled date/time window.		
	4	System retrieves job information, any preset questions, and candidate profile.		
	5	**Question Generation:** System calls Gemini to generate tailored questions (uses preset if present; otherwise job + candidate data).		
	6	**Interview Chat:** System starts the real-time interview; context is maintained; follow-up questions may be generated automatically.		
	7	Candidate provides responses (text/audio; audio may be transcribed).		
	8	**Evaluation:** System sends responses to Gemini for multi-criteria scoring and feedback.		
	9	**Assessment (if configured):** System delivers assessment questions, collects answers, and evaluates them.		
	10	System stores the transcript, scores, and feedback in the database.		

	11	System shows completion status to the candidate.
Alternate Flow – Invalid/Expired Code	3.1	Code is invalid or outside the scheduled range.
	3.2	System blocks access and displays the reason; Candidate may retry or contact recruiter.
Alternate Flow – Candidate Abandons	4.1	Candidate closes the interview mid-session.
	4.2	System saves partial progress and marks the attempt as completed.
Rules	− Interview can only be started within the scheduled date/time window. − Each code is unique per interview session. − Multiple attempts may be restricted per job (institution policy).	
Author	Tong Qian Ru	

3.4.2.4 Use Case 4 – Recruiter Posts Job

Table 3.4.4 Use case description for Recruiter Posts Job

Use Case ID	UC004	**Version**	1.0
Use Case	Recruiter Posts Job		
Purpose	Allow a recruiter to create and publish a job posting with structured requirements and optional interview/assessment presets.		
Actor	Recruiter (primary), System		
Trigger	Recruiter clicks Post New Job.		
Precondition	Recruiter is logged in with Recruiter role.		
Scenario Name	**Step**	**Action**	
Main Flow	1	System displays the job creation form.	
	2	Recruiter enters job title, description, location, employment type, and other details.	
	3	Recruiter defines requirements (skills, education, experience, certifications).	
	4	(Optional) Recruiter sets preset interview questions.	
	5	(Optional) Recruiter sets assessment questions/parameters.	

	6	Recruiter saves the job.
	7	System validates fields and required data.
	8	Job Requirements Parsing: System analyzes the job content and stores a structured requirement profile.
	9	System publishes the job, creates a Job ID, and confirms creation.
Alternate Flow – Missing Required Fields	7.1	Validation fails (e.g., title/description missing).
	7.2	System highlights errors and prevents publishing until fixed.
Rules	− One active application per candidate per job. − Resume must be PDF/DOCX within allowed size. − Applications default to **Pending** until recruiter action.	
Author	Tong Qian Ru	

3.4.2.5 Use Case 5 – Recruiter Manages Application

Table 3.4.5 Use case description for Recruiter Manages Application

Use Case ID	UC005	**Version**	1.0
Use Case	Recruiter Manages Application		
Purpose	Allow a recruiter to review applications, view candidate documents, shortlist/reject/offer, and schedule interviews.		
Actor	Recruiter (primary), System		
Trigger	Recruiter opens Manage Applications for a specific job.		
Precondition	The job exists and has at least one submitted application.		
Scenario Name	**Step**	**Action**	
Main Flow	1	System lists all applications for the selected job with key indicators (matching score, submission time).	
	2	Recruiter selects an application to review.	
	3	System displays candidate profile, parsed resume summary, and cover letter.	

	4	Recruiter updates Application Status to Shortlisted/Rejected/Offered.
	5	If Shortlisted, recruiter sets an interview date range; System generates a unique Interview Code and associates it with the candidate + job.
	6	System saves changes and (optionally) notifies the candidate.
Alternate Flow – No Application	1.1	There are no applications.
	1.2	System displays an empty state.
Rules	− Interview code must be unique per candidate+job and time-bound.	
Author	Tong Qian Ru	

3.4.2.6 Use Case 6 – Recruiter Reviews Interview Results

Table 3.4.6 Use case description for Recruiter Reviews Interview Result

Use Case ID	UC006	**Version**	1.0
Use Case	Recruiter Reviews Interview Results		
Purpose	Allow a recruiter to view interview transcripts, AI feedback/scores, and make a hiring decision.		
Actor	Recruiter (primary), System, Gemini API		
Trigger	Recruiter opens Application Details for a specific candidate and job.		
Precondition	The candidate has completed the interview (and assessment if applicable); evaluation data is available.		
Scenario Name	**Step**	**Action**	
Main Flow	1	System retrieves the interview transcript, AI feedback, and scoring breakdown.	
	2	System shows any assessment results and overall recommendations.	
	3	Recruiter reviews results and (optionally) adds notes.	
	4	Recruiter updates the application status to Rejected, Shortlisted (next round), or Offered.	

	5	System stores the final decision and (optionally) notifies the candidate.
Alternate Flow – Evaluation Pending	1.1	AI evaluation is still processing.
	1.2	System indicates Pending and offers a refresh option.
Rules	− Only authorized recruiters for the job can view results. − Final status change is audit-logged with timestamp and user.	
Author	Tong Qian Ru	

3.5 System Activity

The IntelliHire system involves various activities performed by the candidates, recruiters, and system. To better illustrate how these activities work with each other and progress, the activity diagram is used. While the flowchart is focused on sequential steps, the activity diagram emphasizes role coordination of activity, decision nodes, and automated activities such as resume parsing, job requirements analysis, AI-based job matching, and interview assessment. This provides a clearer picture of how human actions and system work together to streamline the recruitment process. It maps the interactions between candidates, recruiters, and the system across different stages of recruitment, from job posting and application submission to interview handling and AI-driven evaluation.

3.5.1 System Activity Diagram

The system activity diagram below shows the overall workflow of IntelliHire.

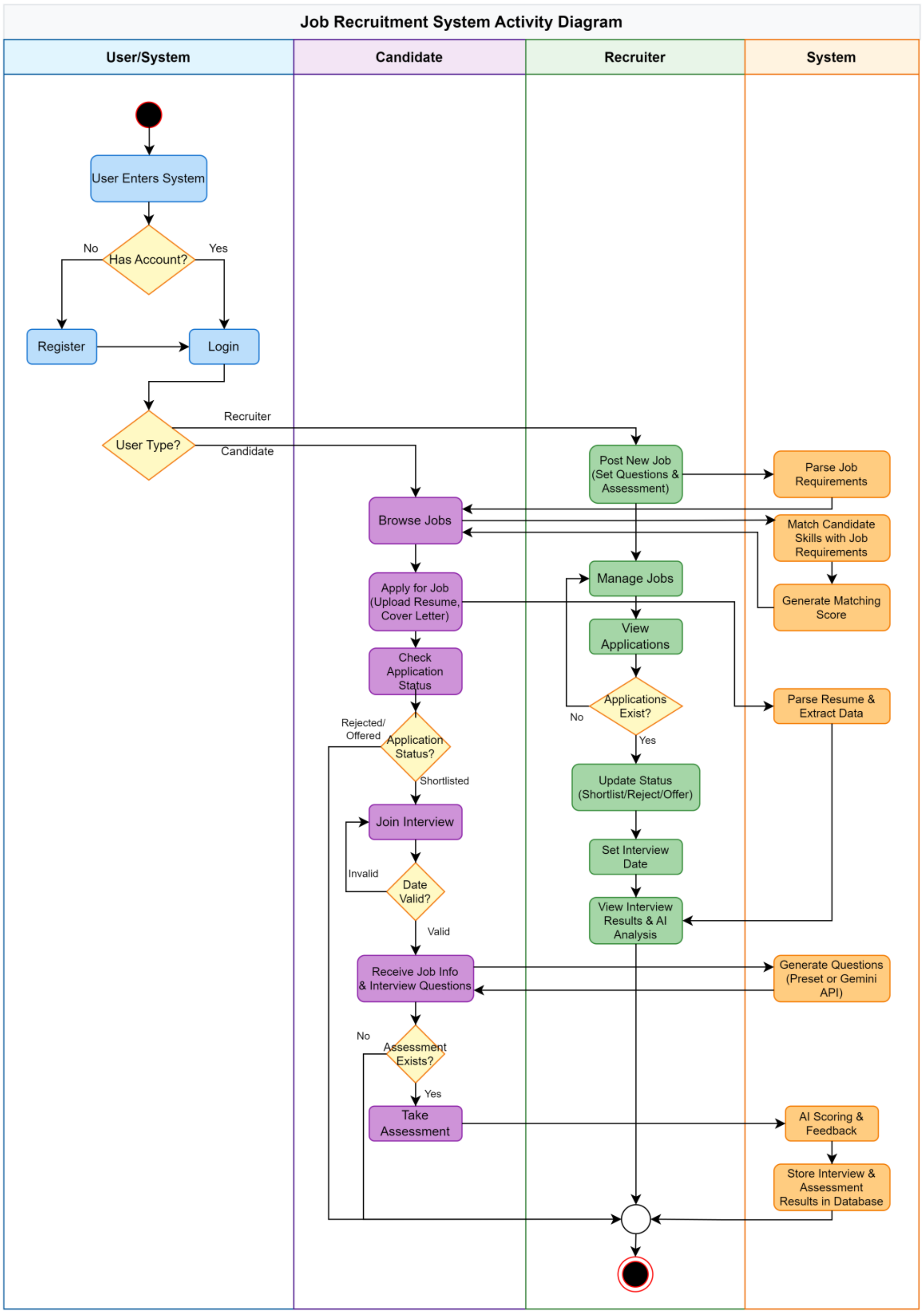


Figure 3.5.1.1 System Activity Diagram

3.6 Methodology of the System

The IntelliHire system will be developed using the Agile methodology [9], which is well-suited for projects that require flexibility and rapid iterations. Agile methodology emphasizes continuous feedback, iterative development, and adaptability to change, making it ideal for developing a complex system like IntelliHire.

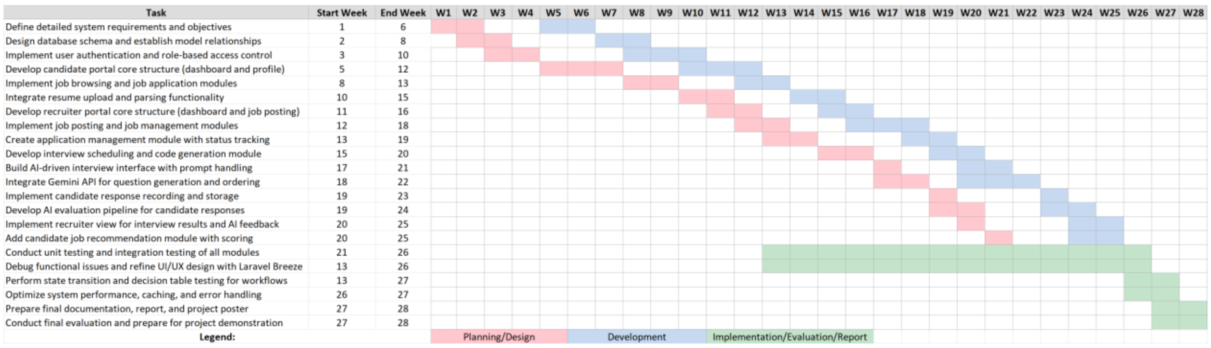
The development process will be divided into several sprints, each focusing on a specific aspect of the system [10]. The sprints will be typically lasting two to four weeks, allowing the development team to deliver functional components of the system incrementally. This approach ensures that any issues or changes in requirements can be addressed promptly, reducing the risk of project delays.

Key Phases of the Agile Methodology for IntelliHire:

1. **Sprint Planning:** For each sprint, tasks are identified and prioritized based on project requirements and milestones. Sprint goals are determined by aligning the most important features with the project timeline and available resources.
2. **Design and Prototyping:** Before implementation, wireframes and prototypes of the system interface and components are created. These help in visualizing the workflow and ensuring the design meets the intended objectives before development begins.
3. **Development:** Each sprint focuses on implementing the planned features through coding. As the sole developer, best practices such as modular coding, version tracking, and consistent documentation are applied to maintain quality and allow for easier debugging and enhancements.
4. **Testing:** After development, the implemented features are tested systematically. This includes unit testing to verify individual functions, integration testing to ensure components work together, and functional testing to confirm that features perform as expected.
5. **Review and Retrospective:** At the end of each sprint, progress is reviewed against the sprint goals. Reflection is carried out to identify what worked well, challenges faced, and adjustments that can be made to improve productivity in the next sprint.

6. **Finalization:** Once all sprints are completed, the system will be finalized in a controlled environment for demonstration and evaluation purposes. Final testing ensures that the project meets the defined scope and objectives.
7. **Maintenance and Updates:** Opportunities for improvement or additional features can be identified after evaluation. Future updates may include refining existing modules or extending functionality based on project feedback.

3.7 Timeline



Task	Start Week	End Week	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22	W23	W24	W25	W26	W27	W28
Define detailed system requirements and objectives	1	6	🟥	🟥			🟦	🟦																						
Design database schema and establish model relationships	2	8		🟥	🟥				🟦	🟦																				
Implement user authentication and role-based access control	3	10			🟥	🟥				🟦	🟦	🟦																		
Develop candidate portal core structure (dashboard and profile)	5	12					🟥	🟥	🟥			🟦	🟦	🟦																
Implement job browsing and job application modules	8	13								🟥	🟥			🟦	🟦															
Integrate resume upload and parsing functionality	10	15										🟥	🟥			🟦	🟦													
Develop recruiter portal core structure (dashboard and job posting)	11	16											🟥	🟥			🟦	🟦												
Implement job posting and job management modules	12	18												🟥	🟥			🟦	🟦	🟦										
Create application management module with status tracking	13	19													🟥	🟥				🟦	🟦									
Develop interview scheduling and code generation module	15	20															🟥	🟥			🟦	🟦								
Build AI-driven interview interface with prompt handling	17	21																	🟥			🟦	🟦							
Integrate Gemini API for question generation and ordering	18	22																	🟥	🟥		🟦	🟦	🟦						
Implement candidate response recording and storage	19	23																			🟥				🟦					
Develop AI evaluation pipeline for candidate responses	19	24																			🟥	🟥			🟦	🟦				
Implement recruiter view for interview results and AI feedback	20	25																				🟥				🟦	🟦			
Add candidate job recommendation module with scoring	20	25																					🟥			🟦	🟦			
Conduct unit testing and integration testing of all modules	21	26													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Debug functional issues and refine UI/UX design with Laravel Breeze	13	26													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Perform state transition and decision table testing for workflows	13	27																										🟩	🟩	
Optimize system performance, caching, and error handling	26	27																										🟩	🟩	
Prepare final documentation, report, and project poster	27	28																											🟩	🟩
Conduct final evaluation and prepare for project demonstration	27	28																											🟩	🟩
Legend:			🟥 Planning/Design					🟦 Development					🟩 Implementation/Evaluation/Report																	

Figure 3.7.1 Gantt Chart

Chapter 4 System Design

4.1 System Block Diagram

The system block diagram provides a high-level description of IntelliHire architecture, describing how the different components and modules interact with each other to achieve the system goals. It illustrates the process of data input from users (candidates and recruiters) into the system, how inputs are routed through the backend and external AI services, how data is stored inside the database, and finally how outputs are shown back to the users. By concealing low-level implementation details, the block diagram unveils the main building blocks of IntelliHire and their relations to each other, which makes the overall system easier to understand.

4.1.1 High-Level System Block Diagram

The high-level system block diagram depicts the main entities in IntelliHire and how they relate to one another. The system is accessed by the recruiter and candidate through the frontend that communicates with the Laravel backend. The authentication, applications, scheduling, notifications, and file handling and transfer AI-based tasks like resume extraction, job matching, and interview assessment are delegated to the Gemini API in the backend. The MySQL database is used to store data persistently, while real-time audio/video recording and text-to-speech features are dealt with using browser APIs on the client side. Here is a clear indication of the system boundaries and external interactions.

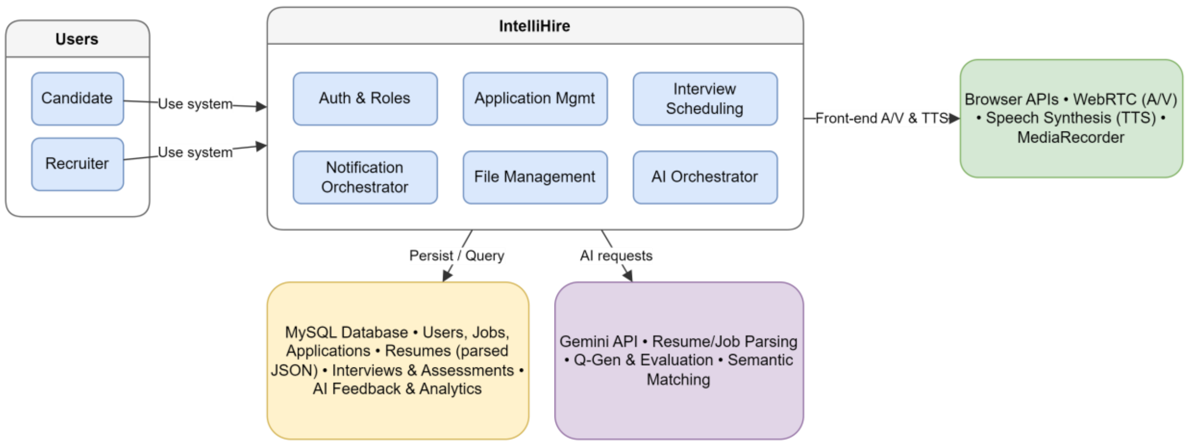

Users
Candidate
Recruiter
Use system
Use system
IntelliHire
Auth & Roles
Application Mgmt
Interview Scheduling
Notification Orchestrator
File Management
AI Orchestrator
Front-end A/V & TTS
Browser APIs • WebRTC (A/V) • Speech Synthesis (TTS) • MediaRecorder
Persist / Query
AI requests
MySQL Database • Users, Jobs, Applications • Resumes (parsed JSON) • Interviews & Assessments • AI Feedback & Analytics
Gemini API • Resume/Job Parsing • Q-Gen & Evaluation • Semantic Matching

Figure 4.1.1.1 High-level system block diagram

4.1.2 Internal Subsystems Block Diagram

The block diagram of internal subsystems is more precise in depicting IntelliHire's structure. It breaks the backend into distinct modules such as Core Services, AI Orchestrator, and Persistence Layer. Core Services have application logic (authentication, scheduling, file management, notifications), while the AI Orchestrator regulates Gemini API operations such as resume parsing, job requirements extraction, interview question generation, and candidate response assessment. The MySQL persistence layer stores user, job, resume, interview, and assessment data. The frontend includes portals for candidates and recruiters, as well as dedicated interfaces for interviews and assessments. Browser APIs complement the frontend by enabling real-time multimedia processing. This diagram emphasizes the internal responsibilities and communication between system modules.

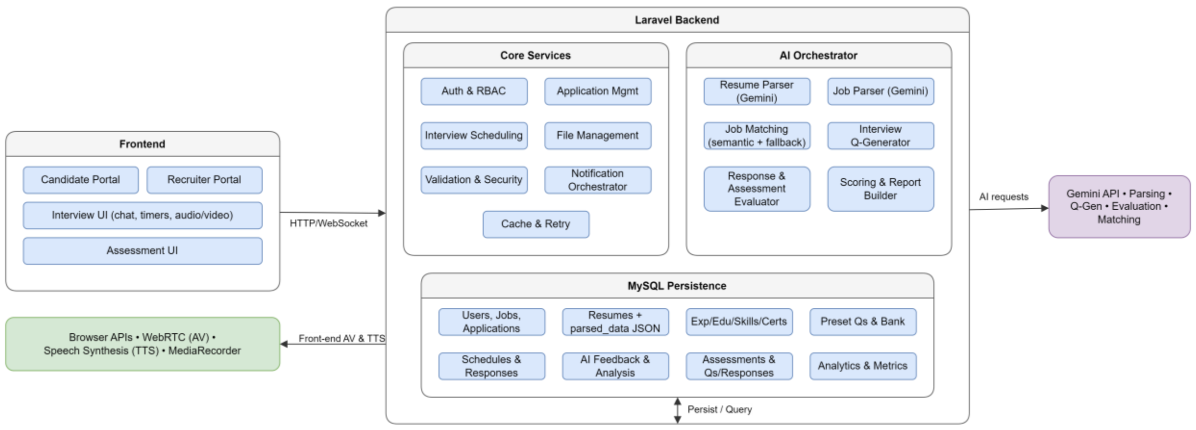


Figure 4.1.2.1 Internal subsystem block diagram

4.2 System Components Specifications

This section outlines the main parts of the IntelliHire system, divided into frontend interfaces, backend services, external integrations, and system-wide patterns of communication. Each component plays an essential role in ensuring the smooth communication between candidates, recruiters, and the platform as the backend's role lies in handling data, AI-driven analysis, and storage.

4.2.1 Frontend Components

The frontend provides all user-facing interfaces for candidates and recruiters. It has been designed with responsiveness, usability, and real-time interaction, offering ease of access to job postings, interviewing procedures, and assessments.

4.2.1.1 Candidate Portal

The frontend provides all user-facing interfaces for candidates and recruiters. It has been designed with responsiveness, usability, and real-time interaction, offering ease of access to job postings, interviewing procedures, and assessments.

- **Dashboard interface**: The dashboard consolidates all essential information such as AI-generated job recommendations, current application statuses, and scheduled interviews, allowing candidates to view everything at a glance.
- **Job search and application**: Candidates can browse and filter available job postings, then apply by uploading resumes or cover letters directly through the system, streamlining the application process.
- **Profile management**: Allows candidates to manage personal information, and keep track of their application history, ensuring their profile stays updated.
- **Interactive job cards with AI matching scores**: Each job listing is presented with a compatibility score, helping candidates quickly gauge suitability based on their skills and experiences.

4.2.1.2 Recruiter Portal

The recruiter portal enables HR personnel and hiring managers to create, manage, and monitor job postings, applications, and interviews.

- **Job posting management**: Recruiters can create, edit, and publish postings with AI-powered parsing of job requirements, which ensures job descriptions are consistent and accurate.
- **Application management**: This interface allows recruiters to review candidate submissions, update statuses, and manage the overall hiring funnel efficiently.

- **Interview scheduling**: Recruiters can set up interview sessions and automatically generate unique interview codes, simplifying session management.
- **Analytics dashboard**: AI-driven analytics provide recruiters with insights into candidate performance, evaluation scores, and hiring statistics to support evidence-based decisions.
- **Rich text editor for job descriptions**: Recruiters can compose detailed job postings using an editor that supports formatting and rich content.
- **Exportable reports**: Candidate data and evaluation reports can be exported for further review, record-keeping, or sharing with stakeholders.

4.2.1.3 Interview User Interface

The interview interface provides a real-time, AI-enhanced experience for both candidates and recruiters, incorporating conversational features and multimedia integration.

- **Chat-based interview experience**: The interface presents questions in a conversational flow, simulating a natural interview session.
- **Speech recognition integrati**on: Candidates can answer using voice input, which is processed via WebRTC and transcribed for AI evaluation in real time.
- **Video recording**: The system records interview sessions using camera and microphone, enabling richer evaluation through video playback.
- **AI-generated text-to-speech**: Questions are read aloud using text-to-speech, making the interview more interactive and accessible.
- **Real-time conversation flow management:** The module ensures seamless coordination between AI prompts, candidate responses, and evaluation feedback.

4.2.1.4 Assessment User Interface

The assessment interface supports structured evaluations, allowing recruiters to test candidates with both text and audio-based answers.

- **Dynamic question display**: The assessment interface supports structured evaluations, allowing recruiters to test candidates with both text and audio-based answers.

- **Multi-modal input**: The interface accepts text responses and audio recordings, giving candidates multiple ways to respond.
- **Progress tracking**: Candidates are guided with counters and timers, helping them manage time effectively during assessments.
- **Immediate scoring and feedback**: The AI system evaluates responses instantly and provides feedback, speeding up the selection proces.
- **Audio recording capabilities**: Spoken answers are recorded for later review and cross-verification of AI scoring.

4.2.2 Backend Components

The backend powers the core business logic, AI orchestration, and data management of IntelliHire. It is developed on the Laravel framework and designed for scalability, security, and seamless integration with external services.

4.2.2.1 Core Services Architecture

- **Laravel MVC framework**: The system is built on the model-view-controller pattern, which separates concerns and simplifies maintainability.
- **Authentication and role-based access**: Access rights are controlled based on user type (candidate or recruiter), ensuring secure role-specific operations.
- **File management service**: Handles the uploading, processing, and retrieval of candidate documents such as resumes and cover letters.
- **Interview session management**: Maintains interview state and session data to ensure reliability during live assessments.
- **RESTful API design**: Facilitates structured communication between frontend and backend using JSON-based APIs.

4.2.2.2 AI Orchestrator

- **Resume parser service**: Extracts and structures text from candidate resumes in PDF or DOCX formats for easy analysis.

- **Job requirements parser**: Identifies and organizes required skills, qualifications, and experience from job postings.
- **Job matching engine**: Compares parsed resume data with job requirements to produce compatibility scores.
- **Interview response evaluator**: Processes candidate responses (text or speech) and provides detailed AI-driven feedback.
- **Assessment scorer**: Automatically grades candidate answers against expected solutions, ensuring objective evaluation.
- **Error handling and retry logic**: Protects against API failures by retrying requests with exponential backoff strategies.
- **Caching mechanisms**: Improves performance by temporarily storing frequently accessed results from the AI service.

4.2.2.3 Data Processing Services

- **Resume processing pipeline**: Automates the sequence from uploading resumes to extracting text, parsing with AI, and storing in the database.
- **Job matching algorithm**: Performs semantic analysis across multiple criteria to recommend best-fit jobs for candidates.
- **Response analysis pipeline**: Converts spoken responses into text, evaluates with AI, and formats structured feedback.
- **Asynchronous processing queues**: Offloads heavy tasks like AI evaluation into background jobs to improve system responsiveness.
- **Error logging and monitoring**: Captures system errors with detailed logs for debugging and continuous improvement.

4.2.2.4 Persistence Layer

- **Database design**: Employs a normalized schema to represent entities such as users, jobs, applications, and interviews.
- **Eloquent ORM relationships**: Maps database entities into Laravel models, maintaining associations and constraints automatically.

- **Migration system**: Provides schema version control, allowing updates without disrupting operations.
- **Caching strategy**: Uses Redis or memory caching to reduce repeated API requests and heavy calculations.
- **Foreign key constraints and indexing**: Safeguards data integrity and speeds up query performance.

4.2.3 External Services Integration

IntelliHire integrates with several external services to deliver AI-powered analysis, multimedia processing, and secure file management.

4.2.3.1 Gemini AI API

- **Model configuration**: Integrates Gemini 2.0 Flash for resume parsing, interview question generation, and evaluation tasks.
- **Integration patterns**: Uses structured API requests with formatted payloads to communicate effectively with the AI service.
- **Error handling**: Applies retry strategies and fallback options to maintain resilience when requests fail.
- **Performance optimization**: Uses batching and caching to minimize costs and reduce response times.

4.2.3.2 Browser APIs

- **WebRTC API**: Enables real-time voice and video streaming during interviews and assessments.
- **Speech Synthesis API**: Generates spoken versions of text-based interview questions for natural interaction.
- **MediaRecorder API**: Captures audio and video from candidate devices for storage and later review.
- **File API**: Manages secure file upload and validation directly in the browser.

4.2.3.3 Database Management

- **MySQL database**: Serves as the main storage solution for all system data, including users, jobs, and interviews.
- **Connection pooling**: Enhances performance by reusing database connections efficiently.
- **Backup strategy**: Implements regular backups to prevent data loss and ensure recovery.

4.2.3.4 File Storage System

- **Laravel storage service**: Manages secure file storage for resumes and other documents.
- **File validation**: Ensures uploaded files meet type, size, and security requirements.
- **Storage optimization**: Compresses files and integrates with CDN for faster delivery and scalability.
- **Access control**: Restricts file access based on user roles and permissions.

4.2.4 System Integration Patterns

This section describes the system-wide mechanisms that connect different components together and ensure secure, reliable communication.

4.2.4.1 API Communication

- **HTTP client implementation**: Uses Laravel's HTTP facade to make structured API requests.
- **Request/response handling**: Manages JSON serialization and error mapping consistently.
- **Authentication**: Protects API communication with keys and digital signatures.
- **Rate limiting**: Applies request throttling and quotas to prevent abuse.

4.2.4.2 Real-time Communication

- **WebSocket integration**: Provides immediate updates during live interviews and assessments.
- **Event broadcasting**: Sends notifications and system status updates to connected users.
- **Session management**: Maintains session state across interview interactions.
- **Concurrent user handling**: Supports multi-user participation without service interruption.

4.2.4.3 Security Implementation

- **Data encryption**: Protects sensitive information both at rest and in transit.
- **Access control**: Enforces role-based permissions to separate candidate and recruiter functions.
- **Input validation**: Prevents malicious input such as XSS or SQL injection.
- **Audit logging**: Records user activities for accountability and security monitoring.

4.3 Database Design

The entity-relationship diagram (ERD) illustrates the logical structure of the IntelliHire database. It defines the main entities such as Users, Jobs, Applications, Resumes, Interview Sessions, and Assessments, along with their attributes and relationships. This diagram provides a clear overview of how data is organized and linked within the system. For instance, a User entity can represent either a Candidate or Recruiter, while Applications link Candidates to specific Jobs. Resumes store parsed information such as education, experience, and skills, and are associated with Candidates. Assessment entities capture test questions, candidate responses, and AI evaluation scores. By presenting these relationships, the ERD highlights how IntelliHire maintains consistency, supports efficient queries, and enables seamless integration between different functional modules.

4.3.1 Entity-Relationship Diagram (ERD)

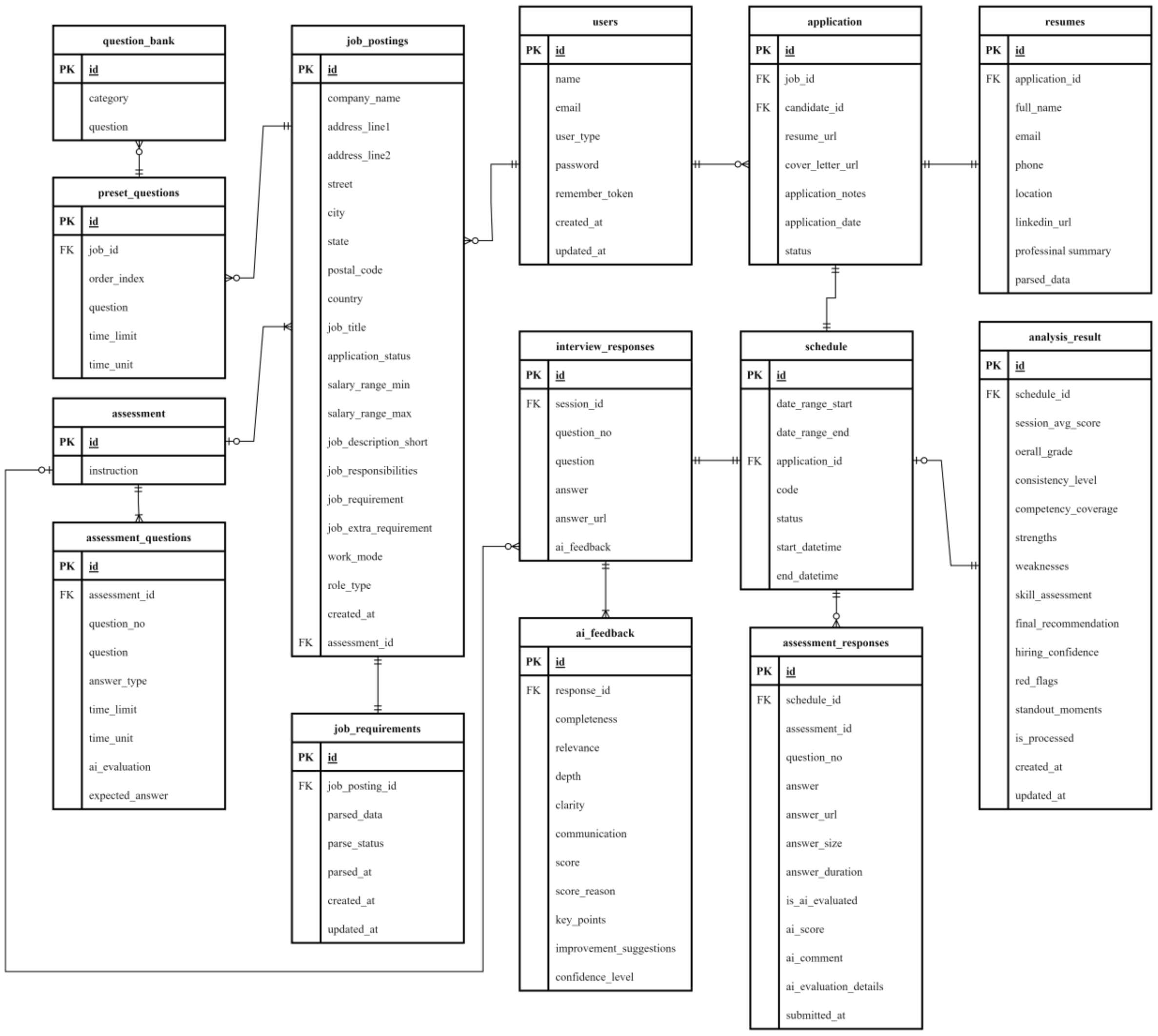


Figure 4.3.1.1 Entity Relationship Diagram (ERD)

The following description outlines the purpose of each major table included in the database design:

- **Users**: Stores information for all system users, including both candidates and recruiters. The table also manages role assignments, authentication details, and unique identifiers such as email addresses.
- **Job Postings**: Holds data related to job opportunities posted by recruiters. Attributes include company details, job title, job description, work mode, salary range, and parsed requirements.

- **Job Requirements**: Breaks down each posting into structured requirements such as skills, education, work experience, and certifications, enabling precise candidate-job matching.
- **Applications**: Captures candidate submissions for specific job postings. Each record links a candidate to a job and tracks application status (e.g., applied, shortlisted, interviewed, offered, rejected).
- **Resumes**: Stores uploaded resumes and their parsed content in structured format. Associated tables (resume_skills, resume_education, resume_experiences, resume_certifications) extend this entity with detailed candidate background information.
- **Schedules**: Manages interview time ranges and unique session codes. It ensures that interviews are securely scheduled, tracked, and linked to the appropriate applications or assessments.
- **Interview Responses**: Records candidate answers during interviews, including text, audio, or video responses. Each response is linked to a session and can be further evaluated by the AI system.
- **AI Feedback**: Stores AI-generated evaluation results for each interview response, including indicators such as completeness, clarity, and communication skills, to support recruiter decision-making.
- **Analysis Results**: Summarizes the overall outcome of interview sessions by compiling feedback into structured evaluation reports and hiring recommendations.
- **Assessments**: Represents structured tests that recruiters can assign to candidates. Attributes include assessment configuration, instructions, and links to relevant questions.
- **Assessment Questions**: Defines individual questions within each assessment, including expected answers and time constraints.
- **Assessment Responses**: Stores candidate responses to assessment questions, together with AI evaluation results and scoring.
- **Preset Questions**: Allows recruiters to predefine interview questions tied to a specific job posting, making interview preparation more efficient.

- **Question Bank**: A reusable repository of predefined questions that recruiters can select from when creating job postings or assessments, ensuring consistency across different recruitment processes.

4.4 System Components Interaction Operations

This section describes how different modules of IntelliHire interact during system operation. While Chapter 3 introduced the static structure of the system architecture, this section provides a dynamic perspective of how components work together in real execution. The interaction operations cover controller responsibilities, request handling, AI communication, persistence through the database, and feedback to the frontend.

4.4.1 Main Laravel Controllers and Routes

The IntelliHire system follows Laravel's MVC structure, where controllers manage requests, invoke services, and return responses to the frontend. Each controller is responsible for a specific domain of operations:

- **JobPostingController** (/jobs): Handles CRUD operations for job postings. The store() method creates job postings and immediately triggers AI-powered parsing of requirements, while the update() method allows recruiters to modify postings and re-parse requirements. The toggleStatus() endpoint manages the active/inactive state of job applications.
- **CandidateJobController** (/candidate/jobs): Provides candidates with job browsing and application functionality. The index() method displays available jobs with AI-calculated matching scores, while show() retrieves job details. The apply() method manages application submissions, including resume uploads handled by uploadResume().
- **InterviewController** (/candidate/interview): Manages interview sessions. The validateCode() method authenticates session access, startInterview() initializes interviews, and chat() coordinates AI-driven interview conversations. The clearChat() function resets a session for a fresh start.

- **RecruiterApplicationController** (/recruiter/applications): Supports recruiters in managing candidate applications. The index() method lists applications, applicationDetails() provides detailed candidate data, and analyzeInterview() invokes AI evaluation of responses. Recruiters can update statuses through updateStatus() and arrange sessions via scheduleInterview().
- **ResumeParserController** (internal): Dedicated to processing uploaded resumes. Its methods extract text using Gemini AI (extractTextWithGemini()) and save structured data into the database (parseAndSave()).
- **JobRequirementsParserController** (internal): Analyzes job postings for structured requirements. It parses recruiter inputs with Gemini AI and saves results for the job matching module.
- **JobMatchingController** (internal): Implements the AI-powered job recommendation process. The calculateMatch() and calculateSkillsMatchOptimized() methods compute compatibility scores between candidates and job requirements.
- **AssessmentController** (/assessment/{schedule}): Handles assessment management. showAssessment() displays test interfaces, while submitAnswer() records candidate responses. The evaluateResponse() method integrates with Gemini to automatically score answers.
- **CandidateDashboardController** (/candidate/dashboard): Displays personalized dashboards, including job recommendations, application progress, and upcoming interviews.
- **RecruiterDashboardController** (/recruiter/dashboard): Provides recruiters with an overview of job postings, applications, and analytics in one interface.
- **RecruiterResumeController** (internal): Allows recruiters to access and review candidate resumes, complementing the parsing process.
- **ProfileController** (/profile): Manages user accounts and profile data. Methods include edit(), update(), and destroy() for profile editing and deletion.

4.4.2 Background Jobs and Queues

Although IntelliHire is built with Laravel’s queue infrastructure enabled, the current implementation does not actively dispatch background jobs. The system is configured to use the database as the queue driver, and tables such as jobs, job_batches, and failed_jobs are already created in the schema. This ensures that the technical foundation for asynchronous processing exists, but at present all interactions with the Gemini API, resume parsing, job requirement analysis, and interview evaluations are executed synchronously. In practice, this means that whenever a candidate uploads a resume or submits an interview response, the backend controller immediately calls the Gemini API and waits for a response before sending feedback back to the frontend. This synchronous design has the advantage of simplicity and predictability, because candidates and recruiters receive instant responses within the same request cycle without needing to check back later for results.

However, the synchronous approach also comes with limitations. When large resumes are uploaded or when many candidates are attempting interviews at the same time, the system may experience delays while waiting for AI responses to return. For example, parsing a resume may take several seconds, and during this time the candidate must wait for the server to complete its interaction with Gemini before receiving confirmation. Similarly, recruiters who attempt to analyze interview sessions may encounter slower response times because the evaluation is carried out in real time. In high-traffic scenarios, this synchronous execution could become a bottleneck, affecting the perceived responsiveness of the platform.

Laravel's queue system provides a solution to these challenges by allowing certain workloads to be offloaded into background jobs. If IntelliHire were to adopt asynchronous queues, operations such as resume parsing, requirement extraction, AI interview evaluations, and automated notifications could be dispatched into the job queue instead of being executed directly in the request cycle. The queue worker, running in the background, would process each job independently and update the database once results are ready. Candidates and recruiters could then retrieve processed results through dashboard updates or notifications, while the main application remains responsive. For example, a candidate might upload a resume and receive immediate confirmation that the file was accepted, while the parsing and AI analysis occur in the background, with the parsed results made available shortly thereafter.

At present, IntelliHire has chosen not to implement full queue-based processing in order to keep the system architecture straightforward for deployment and testing. Nevertheless, the underlying support for queues is already present, meaning the system can evolve to adopt asynchronous operations in future iterations. This would allow the platform to handle higher traffic loads, reduce waiting times during AI-heavy operations, and make better use of system resources. In particular, enabling queues would be beneficial for scheduled batch operations, such as nightly job matching calculations across thousands of candidates or sending bulk interview reminders to applicants. By laying this foundation early, IntelliHire demonstrates forward compatibility, ensuring that scalability improvements can be introduced when required without a major redesign of the system.

4.4.3 External API Calls

The IntelliHire system integrates extensively with the Gemini AI API to deliver resume parsing, job requirement analysis, interview evaluation, and assessment scoring. The integration pattern is relatively straightforward, relying on Laravel's built-in Http facade to construct requests and process responses. Controllers are directly responsible for formatting the request payloads, sending them to the API, and parsing the returned JSON into structured data. Although this approach creates a tighter coupling between controllers and the external API, it simplifies the development process by avoiding additional abstraction layers such as dedicated service classes. Each request includes candidate or recruiter data prepared in structured format, and the responses are stored directly in the database for further use.

To enhance reliability, IntelliHire implements a basic retry logic with exponential backoff whenever Gemini API requests fail. Instead of failing immediately, the system automatically attempts to resend the request up to three times. Each retry doubles the waiting period, which allows temporary network delays or service interruptions to resolve before the next attempt. This mechanism ensures that the system is more resilient against transient errors, improving the overall reliability of AI-based operations. However, since the system does not currently use asynchronous queues, all retries still occur within the same user request cycle, which may occasionally increase response time during heavy operations such as large resume parsing.

The API calls are used in different parts of the system, and the main examples include:

- **InterviewController::chat()**: Handles conversational AI during interview sessions by sending candidate responses and retrieving AI-generated follow-up questions.
- **ResumeParserController::parseWithGemini()**: Extracts structured data from resumes uploaded in PDF or DOCX format.
- **JobRequirementsParserController::parseWithGemini()**: Analyzes job descriptions to extract standardized requirements.
- **JobMatchingController::makeAIRequest()**: Computes compatibility scores between candidates and job postings.
- **RecruiterApplicationController::analyzeInterview()**: Submits recorded interview responses to Gemini for structured evaluation and feedback.
- **AssessmentController::evaluateResponse()**: Automatically grades candidate assessment responses against expected answers.

4.4.4 Data Persistence

The IntelliHire system relies on Laravel's **Eloquent ORM** to manage all data persistence operations, ensuring a structured and maintainable interaction with the underlying **MySQL database**. Each entity in the database is mapped to a corresponding **model class**, which defines the **relationships** between tables and enforces **referential integrity** at the application level. For example, a **User** model is associated with multiple **Applications** through a `hasMany` relationship, while an **Application** is linked to a specific **Job Posting** and may also have a one-to-one relationship with a **Resume**. These mappings allow the system to express complex queries in an object-oriented manner without writing raw SQL, thereby improving both readability and maintainability.

During runtime, controllers interact with the persistence layer by creating, retrieving, updating, and deleting records through Eloquent models. For instance, when a candidate submits an application, the **Application model** automatically stores the job reference, candidate reference, and status in the database, while linked **Resume models** handle resume uploads and parsed data storage. The use of **foreign keys** and **indexed relationships** ensures that database operations remain consistent and efficient, reducing redundancy and preventing orphaned records. Additionally, the system implements **migrations** to manage schema changes over time, enabling smooth evolution of the database design without disrupting live data.

Caching mechanisms are also integrated into the persistence layer. Frequently accessed data, such as parsed resume results or AI matching scores, can be temporarily stored in **Redis** or in-memory cache. This reduces repetitive API calls to Gemini and minimizes query load on the database, enhancing the system's responsiveness under heavy traffic. By combining **ORM abstractions**, **data integrity enforcement**, and **caching strategies**, IntelliHire achieves a persistence layer that is both robust and scalable, capable of supporting the complex workflows of recruitment operations.

4.4.5 Real-time Features

Real-time interactivity is a critical component of IntelliHire, particularly in the **AI-driven interview and assessment modules**. Although the system does not currently implement **Laravel Echo** or dedicated **WebSocket servers**, it leverages a combination of **AJAX polling** and **browser APIs** to achieve near real-time communication. In the **Interview User Interface**, candidate responses are captured through **WebRTC**, transcribed into text, and transmitted to the backend, where they are immediately evaluated by the Gemini API. The backend then returns AI feedback, which is displayed on the frontend chat interface with minimal delay. This creates a conversational flow that closely simulates a live interviewer experience, even though the underlying mechanism relies on request-response cycles rather than persistent connections.

Multimedia integration also contributes to real-time functionality. The **MediaRecorder API** enables the capture of candidate video and audio during interviews and assessments, while the **Speech Synthesis API** delivers AI-generated questions audibly to candidates. These browser-native features reduce dependency on external services and ensure that interactions feel immediate and engaging. Similarly, **text-to-speech** and **speech-to-text** capabilities enhance accessibility by allowing candidates to interact naturally through voice commands and spoken answers.

While some updates, such as **application status changes** or **dashboard refreshes**, still require manual page reloads or timed polling, the foundation for more advanced real-time features has already been established. The architecture is designed to be **extensible**, meaning future iterations of IntelliHire could integrate WebSockets or push notifications to replace polling with true event-driven communication. Nevertheless, even in its current form, the system demonstrates the ability to support **real-time interviews, dynamic assessments, and instant feedback**, which collectively represent one of the most innovative aspects of the platform.

4.4.6 Security and Authentication

The **security architecture** of IntelliHire is designed around Laravel's built-in authentication framework, which provides a robust foundation for managing **user sessions**, **role-based access**, and **data protection**. Users authenticate through a standard **login system**, where credentials are verified and stored using **bcrypt hashing** for passwords. Once authenticated, session tokens maintain persistent access across the application until logout or expiration. Unlike token-based authentication systems such as **Laravel Sanctum** or **Passport**, IntelliHire employs a **session-based model**, which is simpler to implement and sufficient for the scope of a web-based recruitment platform. This design choice prioritizes ease of use while maintaining a secure boundary between candidate and recruiter roles.

Role management is enforced through the `user_type` attribute stored in the **Users table**, which distinguishes candidates from recruiters. Laravel **middleware** checks this attribute before granting access to restricted routes, ensuring that recruiters cannot impersonate candidates and vice versa. For example, job posting endpoints are protected by recruiter-only middleware, while application submission endpoints are limited to candidate accounts. This **role-based access control (RBAC)** guarantees that users interact only with data and functions relevant to their role, preventing unauthorized operations.

Additional security layers are applied to protect against common web vulnerabilities. **Cross-Site Request Forgery (CSRF) tokens** are automatically embedded in all form submissions to prevent malicious requests, while **input validation rules** sanitize user-provided data before it reaches the database. File uploads, particularly resumes and cover letters, undergo **validation for file type, size, and MIME type** to ensure that harmful scripts cannot be executed through document submissions. Furthermore, **resource ownership checks** are consistently enforced in controllers; for example, a candidate can only view or modify their own applications and resumes, and a recruiter can only access applications associated with their job postings.

In addition to preventative measures, IntelliHire incorporates **audit logging** features that record critical user activities such as logins, application submissions, interview scheduling, and profile updates. These logs serve as a valuable tool for monitoring suspicious activity, tracing errors, and ensuring accountability within the system. Combined with the use of **HTTPS encryption** for all data transmissions, the platform ensures that sensitive information such as personal details, resumes, and interview data remain secure both in transit and at rest.

Through this layered approach—spanning **authentication**, **authorization**, **input validation**, **file handling**, and **audit monitoring**—IntelliHire establishes a comprehensive security model that protects both candidates and recruiters while maintaining system reliability and trustworthiness.

Chapter 5 System Implementation

5.1 Hardware Setup

The hardware involved in this project includes a laptop computer. The laptop, equipped with an Intel Core i5-11400H processor, 16GB DDR4 RAM, NVIDIA GeForce RTX 3050 GPU, and a 500GB M.2 PCIe NVMe SSD, is used for system development, UI/UX design, and implementation of AI integration tasks. It provides the necessary computational power for building the IntelliHire platform and conducting simulated interview sessions within a Windows 11 operating environment.

Table 5.1.1 Specification of laptop

Description	**Specifications**
Model	Illegear Onyx V series
Processor	Intel Core i5-11400H
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 3050 Laptop GPU
Memory	16GB DDR4 RAM
Storage	500GB M.2 PCIe NVMe Solid State Drive

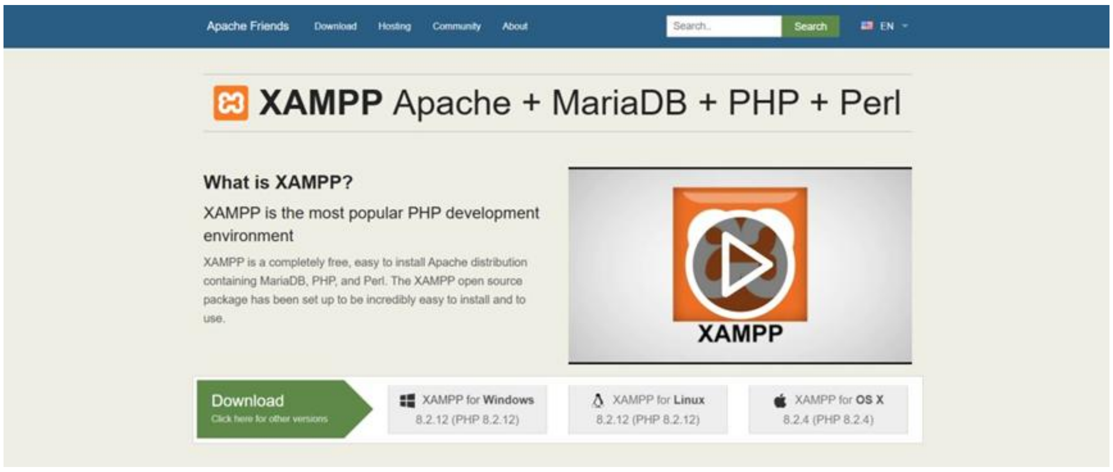
5.2 Software Setup

In order to begin the development of the IntelliHire system, several essential software tools are required to be installed and configured. These tools provided the foundational environment for local development, database management, package handling, and code editing.

5.2.1 XAMPP (for local server environment)

XAMPP [13] is an open-source cross-platform web server solution stack package developed by Apache Friends. It includes Apache, MariaDB, PHP, and Perl, which allows developers to create a local server environment to develop and test PHP applications efficiently.

In this project, XAMPP is utilized to run the Apache server and MySQL database locally, simulating the functioning of an actual web server without deploying it to the cloud.


Apache Friends
Download
Hosting
Community
About
Search..
Search
EN
XAMPP
Download
Click here for other versions
XAMPP for Windows 8.2.12 (PHP 8.2.12)
XAMPP for Linux 8.2.12 (PHP 8.2.12)
XAMPP for OS X 8.2.4 (PHP 8.2.4)

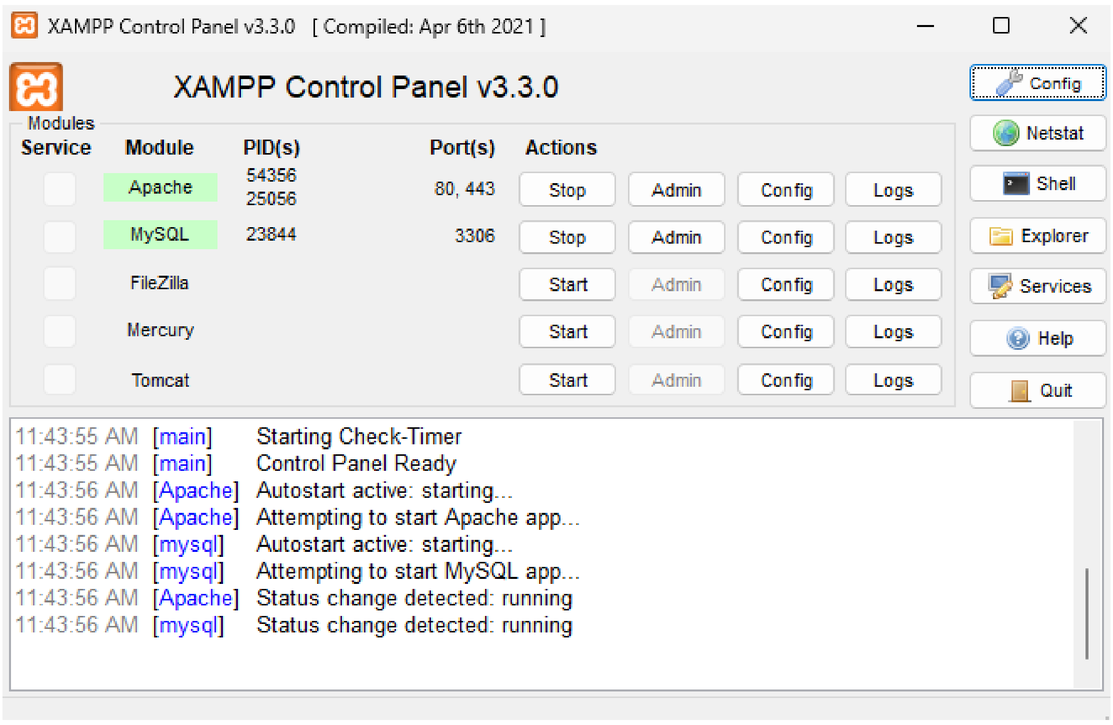
XAMPP Apache + MariaDB + PHP + Perl

What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

Figure 5.2.1.1 XAMPP download page



XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Modules

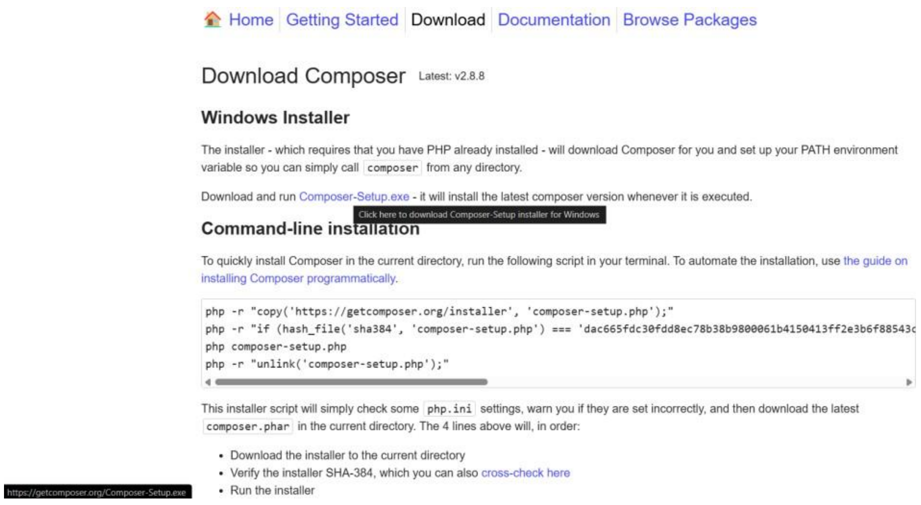
Service	Module	PID(s)	Port(s)	Actions			
	Apache	54356 25056	80, 443	Stop	Admin	Config	Logs
	MySQL	23844	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
11:43:55 AM  [main]    Starting Check-Timer
11:43:55 AM  [main]    Control Panel Ready
11:43:56 AM  [Apache]  Autostart active: starting...
11:43:56 AM  [Apache]  Attempting to start Apache app...
11:43:56 AM  [mysql]   Autostart active: starting...
11:43:56 AM  [mysql]   Attempting to start MySQL app...
11:43:56 AM  [Apache]  Status change detected: running
11:43:56 AM  [mysql]   Status change detected: running
```

Figure 5.2.1.2 XAMPP control panel

1. Composer (PHP dependency manager)

Composer [14] is amongst the most sought-after dependency managers for PHP. With Composer, one can easily install and run libraries or packages required in PHP projects. Composer is utilized in managing Laravel dependencies, and so it is required in any Laravel project. Composer is installed and configured to globally access Laravel installation.



Download Composer Latest: v2.8.8

Windows Installer

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run Composer-Setup.exe - it will install the latest composer version whenever it is executed.

Command-line installation

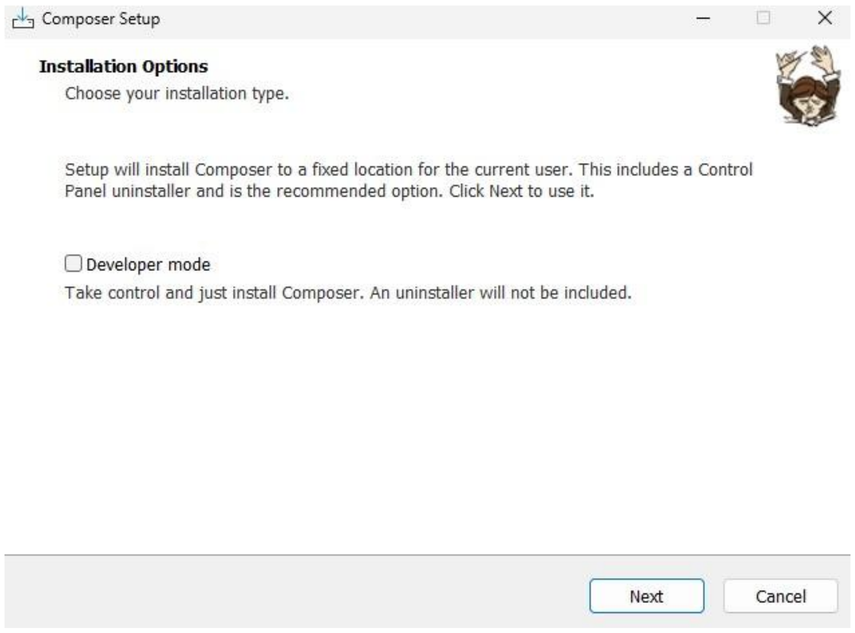
To quickly install Composer in the current directory, run the following script in your terminal. To automate the installation, use the guide on installing Composer programmatically.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === 'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

This installer script will simply check some `php.ini` settings, warn you if they are set incorrectly, and then download the latest `composer.phar` in the current directory. The 4 lines above will, in order:

- Download the installer to the current directory
- Verify the installer SHA-384, which you can also cross-check here
- Run the installer

Figure 5.2.1.3 Composer download page



Installation Options

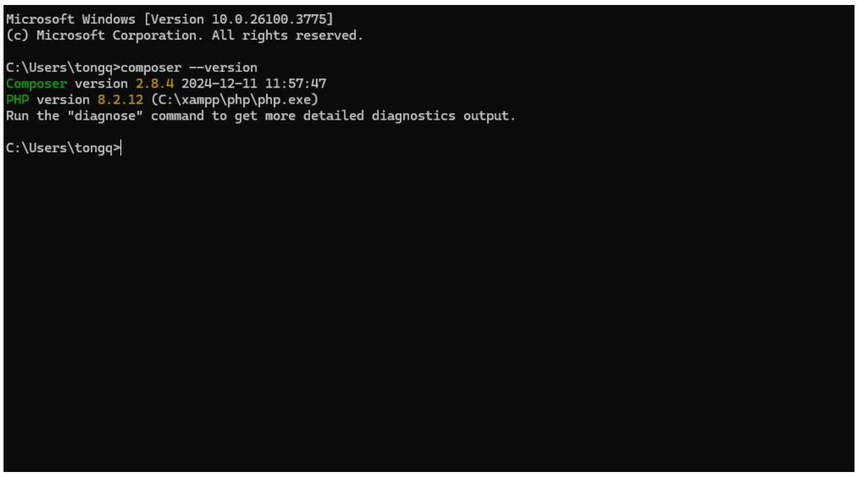
Choose your installation type.

Setup will install Composer to a fixed location for the current user. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.

☐ Developer mode

Take control and just install Composer. An uninstaller will not be included.

Figure 5.2.1.4 Composer setup page

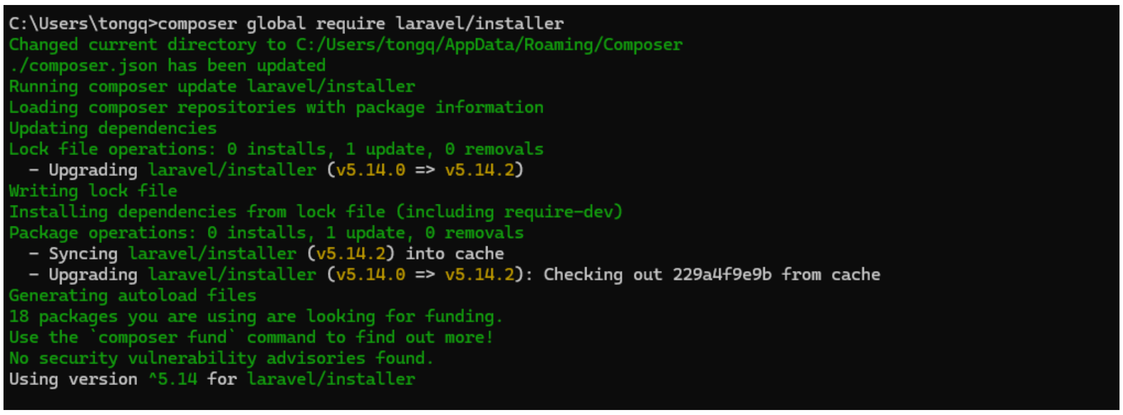


```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tongq>composer --version
Composer version 2.8.4 2024-12-11 11:57:47
PHP version 8.2.12 (C:\xampp\php\php.exe)
Run the "diagnose" command to get more detailed diagnostics output.

C:\Users\tongq>
```

Figure 5.2.1.5 Composer version check

5.2.2 Installing Laravel

Laravel is a PHP framework that simplifies web development. It follows the MVC(Model-View-Controller) structure and offers built-in features like routing, templating, and database integration. To install Laravel, the command composer global require laravel/installer is used. This command downloads the Laravel installer, making it easyto create new Laravel projects.



```
C:\Users\tongq>composer global require laravel/installer
Changed current directory to C:/Users/tongq/AppData/Roaming/Composer
./composer.json has been updated
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading laravel/installer (v5.14.0 => v5.14.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
  - Syncing laravel/installer (v5.14.2) into cache
  - Upgrading laravel/installer (v5.14.0 => v5.14.2): Checking out 229a4f9e9b from cache
Generating autoload files
18 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^5.14 for laravel/installer
```

Figure 5.2.2.1 Laravel installation

5.2.3 Visual Studio Code (IDE)

Visual Studio Code (VS Code) [15] is used as the main code editor for IntelliHire.It is lightweight, fast, and supports many useful extensions. It works well with Laravelprojects and allows developers to customize the environment as needed.

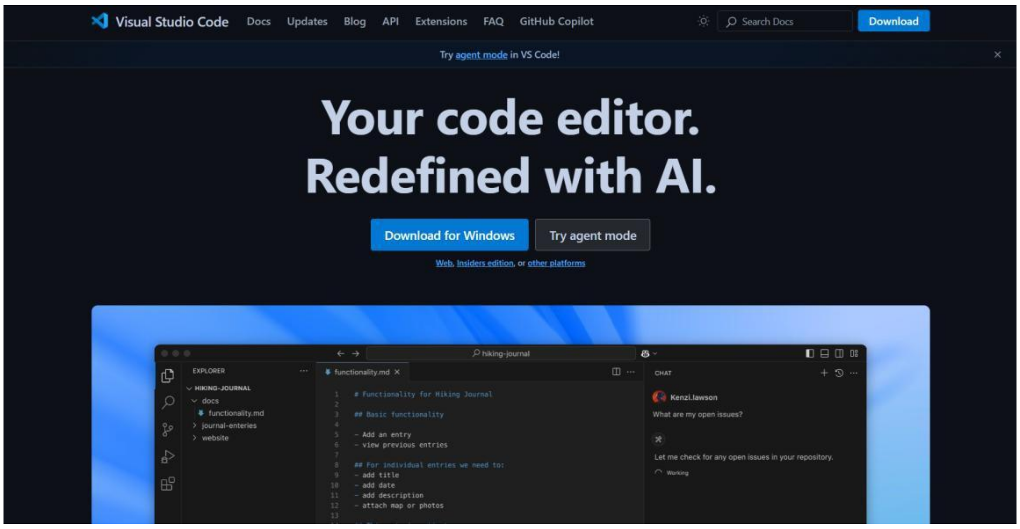
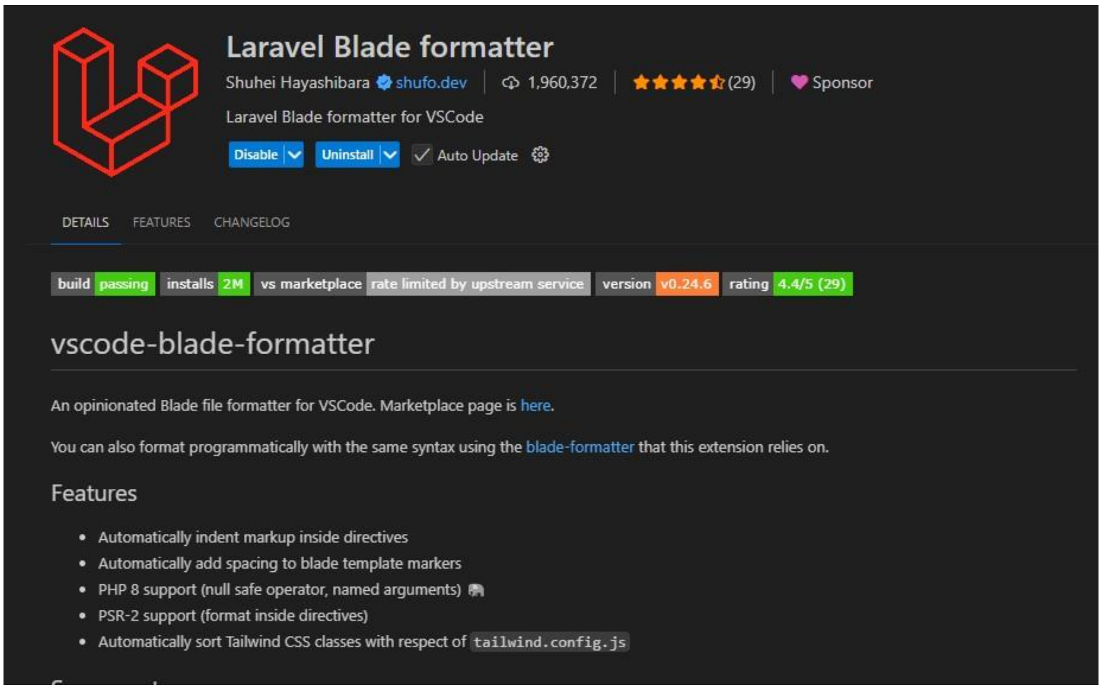


Figure 5.2.3.1 Visual Studio Code download page

Several VS Code extensions are also being installed to enhance development withLaravel:

Laravel Blade Formatter: This extension formats .blade.php files, makingthe code cleaner and easier to read.



Laravel Blade formatter

Shuhei Hayashibara shufo.dev | 1,960,372 | (29) | Sponsor

Laravel Blade formatter for VSCode

Disable | Uninstall | Auto Update

DETAILS FEATURES CHANGELOG

build passing | installs 2M | vs marketplace rate limited by upstream service | version v0.24.6 | rating 4.4/5 (29)

vscode-blade-formatter

An opinionated Blade file formatter for VSCode. Marketplace page is here.

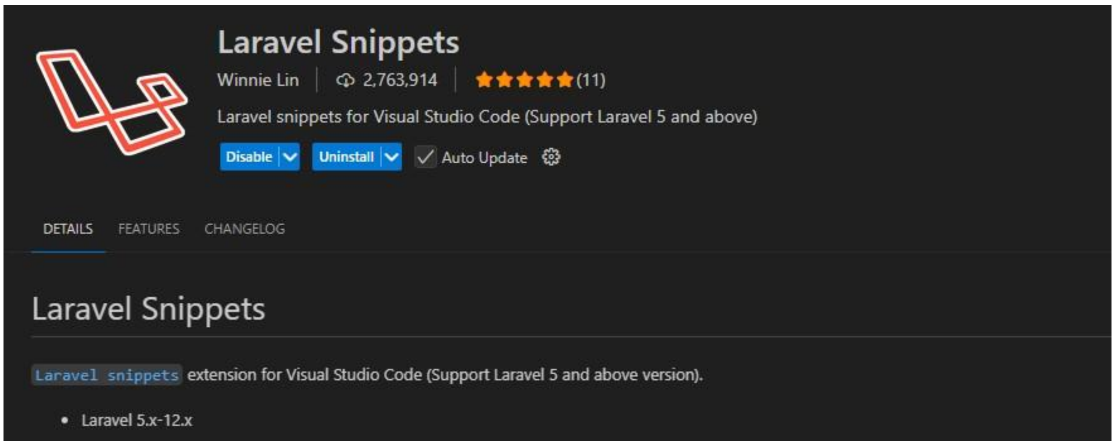
You can also format programmatically with the same syntax using the blade-formatter that this extension relies on.

Features

- Automatically indent markup inside directives
- Automatically add spacing to blade template markers
- PHP 8 support (null safe operator, named arguments)
- PSR-2 support (format inside directives)
- Automatically sort Tailwind CSS classes with respect of `tailwind.config.js`

Figure 5.2.3.2 Laravel Blade Formatter

Laravel Snippets: Provides code snippets for common Laravel constructs, saving development time.



Laravel snippets for Visual Studio Code (Support Laravel 5 and above)

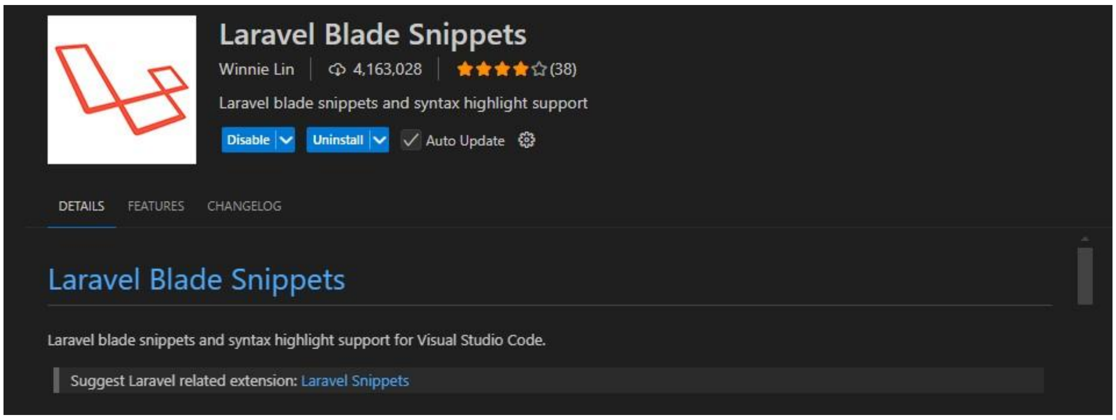
Laravel Snippets

`Laravel snippets` extension for Visual Studio Code (Support Laravel 5 and above version).

- Laravel 5.x-12.x

Figure 5.2.3.3 Laravel Snippets

Laravel Blade Snippets: Offers specific support for Blade templating syntax,making it easier to work within view files.



Laravel blade snippets and syntax highlight support

Laravel Blade Snippets

Laravel blade snippets and syntax highlight support for Visual Studio Code.

Suggest Laravel related extension: Laravel Snippets

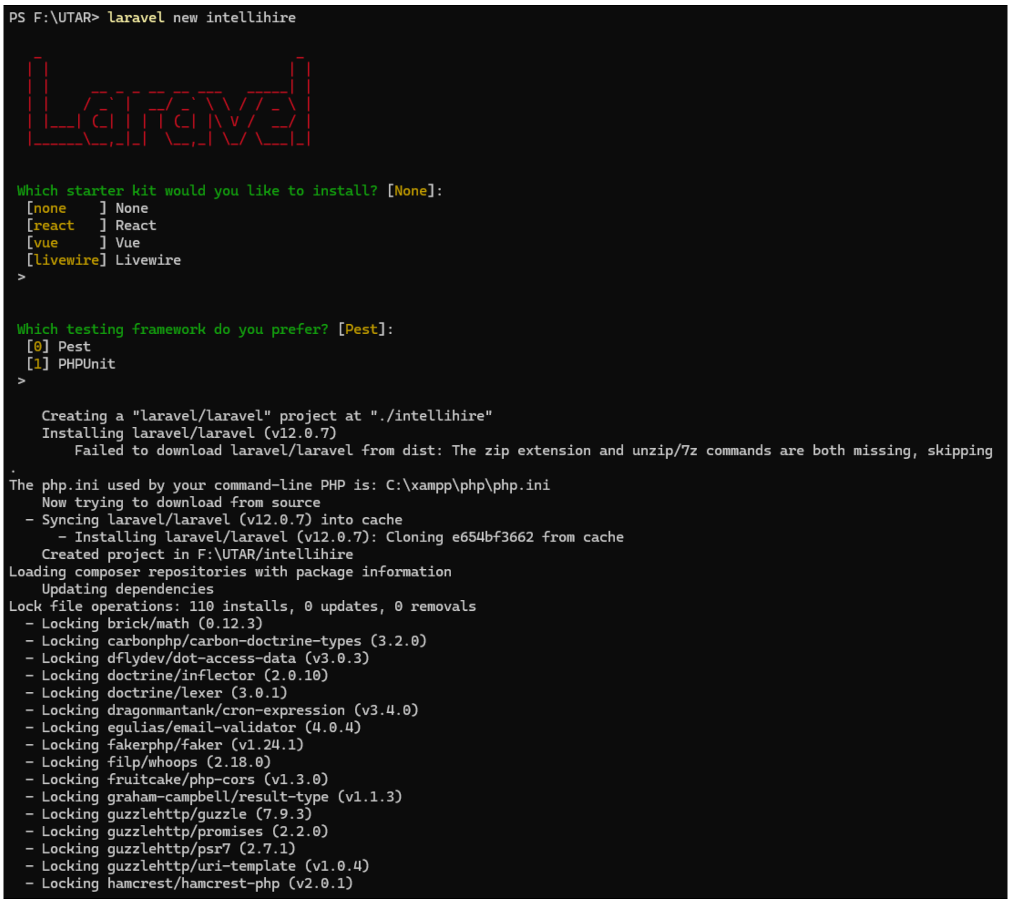
Figure 5.2.3.4 Laravel Blade Snippets

5.3 Setting and Configuration

Once the required tools are installed, the development setup process for IntelliHire began.

5.3.1 Create Laravel Project

A new Laravel project is created using the command “laravel new intellihire”, which created the directory structure and essential files for the application. Thiscommand sets up Laravel’s core components including routes, controllers, models, views, and configuration files.

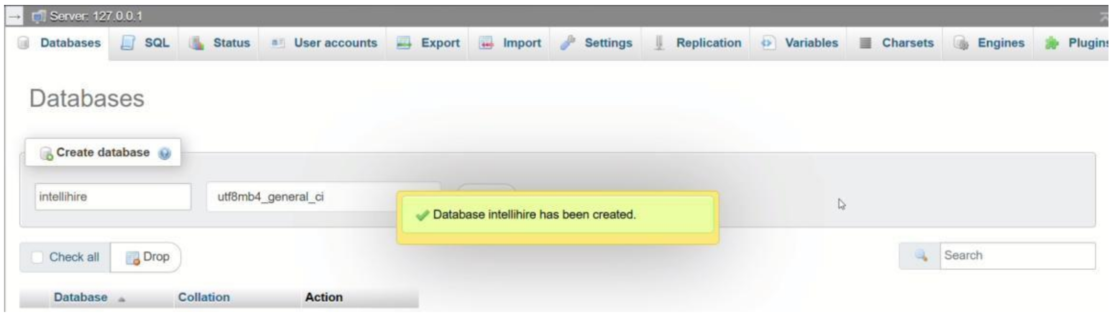


```
PS F:\UTAR> laravel new intellihire

  _                               _
 | |                             | |
 | |     __ _ _ __ __ ___   _____| |
 | |    / _` | '__/ _` \ \ / / _ \ |
 | |___| (_| | | | (_| |\ V /  __/ |
 |______\__,_|_|  \__,_| \_/ \___|_|


 Which starter kit would you like to install? [None]:
  [none    ] None
  [react   ] React
  [vue     ] Vue
  [livewire] Livewire
 >


 Which testing framework do you prefer? [Pest]:
  [0] Pest
  [1] PHPUnit
 >

    Creating a "laravel/laravel" project at "./intellihire"
    Installing laravel/laravel (v12.0.7)
        Failed to download laravel/laravel from dist: The zip extension and unzip/7z commands are both missing, skipping
.
The php.ini used by your command-line PHP is: C:\xampp\php\php.ini
    Now trying to download from source
  - Syncing laravel/laravel (v12.0.7) into cache
      - Installing laravel/laravel (v12.0.7): Cloning e654bf3662 from cache
    Created project in F:\UTAR/intellihire
Loading composer repositories with package information
    Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
  - Locking brick/math (0.12.3)
  - Locking carbonphp/carbon-doctrine-types (3.2.0)
  - Locking dflydev/dot-access-data (v3.0.3)
  - Locking doctrine/inflector (2.0.10)
  - Locking doctrine/lexer (3.0.1)
  - Locking dragonmantank/cron-expression (v3.4.0)
  - Locking egulias/email-validator (4.0.4)
  - Locking fakerphp/faker (v1.24.1)
  - Locking filp/whoops (2.18.0)
  - Locking fruitcake/php-cors (v1.3.0)
  - Locking graham-campbell/result-type (v1.1.3)
  - Locking guzzlehttp/guzzle (7.9.3)
  - Locking guzzlehttp/promises (2.2.0)
  - Locking guzzlehttp/psr7 (2.7.1)
  - Locking guzzlehttp/uri-template (v1.0.4)
  - Locking hamcrest/hamcrest-php (v2.0.1)
```

Figure 5.3.1.1 Laravel project creation

5.3.2 Database Setup

The MySQL database is set up using phpMyAdmin, which comes with XAMPP. Anew database named intellihire is created. This database will store all the system's tablesand data used during development. To connect between Laravel and MySQL, the .envfile in the Laravel project is edited with appropriate database credentials, such as DB_DATABASE, DB_USERNAME, and DB_PASSWORD. The password is not setsince the database is only used locally.

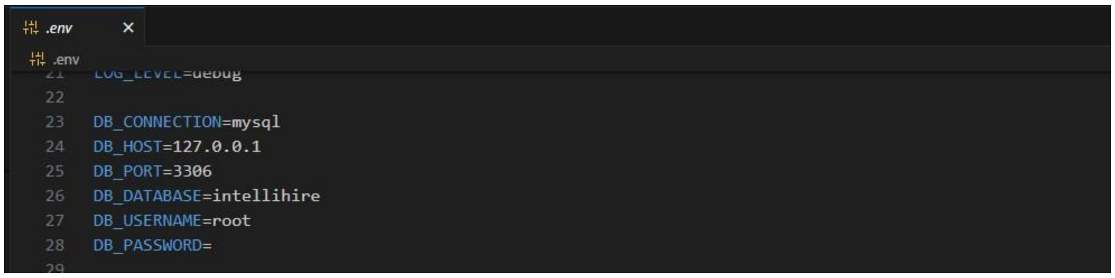


intellihire

utf8mb4_general_ci

Database intellihire has been created.

Figure 5.3.2.1 Database creation page

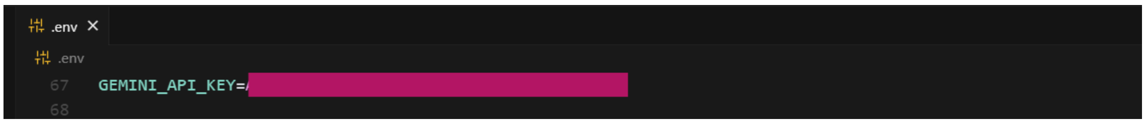


```
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=intellihire
DB_USERNAME=root
DB_PASSWORD=
```

Figure 5.3.2.2 Database setup page in .env of Laravel project

5.3.3 API Key Configuration

The Gemini API is configured by storing its API key securely in the Laravel project's .env file. A new environment variable named GEMINI_API_KEY is added, which holds the secret key provided by Google. This approach ensures that sensitive credentials are not hard-coded into the source code. The Laravel configuration files and controllers access the API key using the env() helper function, enabling authenticated requests to the Gemini API during resume parsing, job requirement analysis, and interview evaluations.

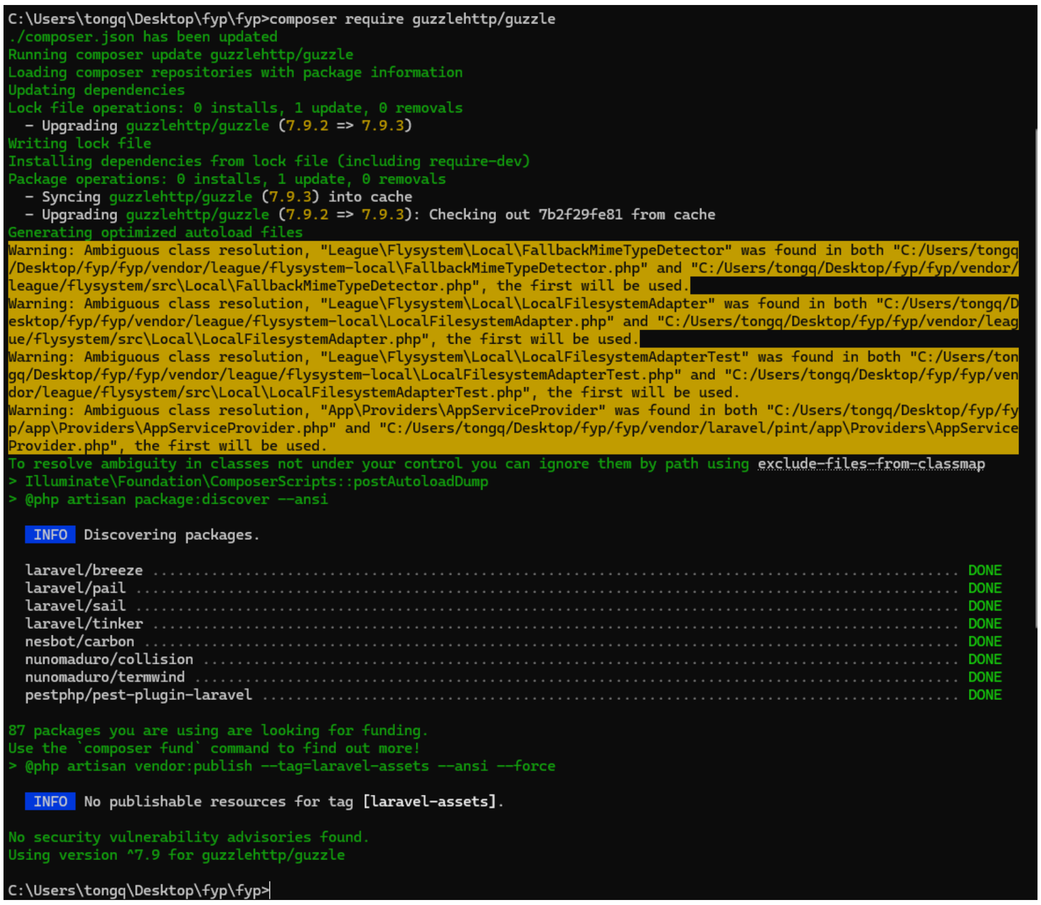


```
GEMINI_API_KEY=[redacted]
```

Figure 5.3.3.1 Gemini API key setup in .env of Laravel project

5.3.4 Installing HTTP Client Package

The GuzzleHTTP package is installed using the command "composer require guzzlehttp/guzzle". This package enables the system to send HTTP requests to external services such as the Gemini API. Guzzle acts as a PHP HTTP client and is crucial for sending POST requests with payloads containing job and candidate information to receive dynamically generated interview questions.



```
C:\Users\tongq\Desktop\fyp\fyp>composer require guzzlehttp/guzzle
./composer.json has been updated
Running composer update guzzlehttp/guzzle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading guzzlehttp/guzzle (7.9.2 => 7.9.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
  - Syncing guzzlehttp/guzzle (7.9.3) into cache
  - Upgrading guzzlehttp/guzzle (7.9.2 => 7.9.3): Checking out 7b2f29fe81 from cache
Generating optimized autoload files
Warning: Ambiguous class resolution, "League\Flysystem\Local\FallbackMimeTypeDetector" was found in both "C:/Users/tongq
/Desktop/fyp/fyp/vendor/league/flysystem-local\FallbackMimeTypeDetector.php" and "C:/Users/tongq/Desktop/fyp/fyp/vendor/
league/flysystem/src\Local\FallbackMimeTypeDetector.php", the first will be used.
Warning: Ambiguous class resolution, "League\Flysystem\Local\LocalFilesystemAdapter" was found in both "C:/Users/tongq/D
esktop/fyp/fyp/vendor/league/flysystem-local\LocalFilesystemAdapter.php" and "C:/Users/tongq/Desktop/fyp/fyp/vendor/leag
ue/flysystem/src\Local\LocalFilesystemAdapter.php", the first will be used.
Warning: Ambiguous class resolution, "League\Flysystem\Local\LocalFilesystemAdapterTest" was found in both "C:/Users/ton
gq/Desktop/fyp/fyp/vendor/league/flysystem-local\LocalFilesystemAdapterTest.php" and "C:/Users/tongq/Desktop/fyp/fyp/ven
dor/league/flysystem/src\Local\LocalFilesystemAdapterTest.php", the first will be used.
Warning: Ambiguous class resolution, "App\Providers\AppServiceProvider" was found in both "C:/Users/tongq/Desktop/fyp/fy
p/app\Providers\AppServiceProvider.php" and "C:/Users/tongq/Desktop/fyp/fyp/vendor/laravel/pint/app\Providers\AppService
Provider.php", the first will be used.
To resolve ambiguity in classes not under your control you can ignore them by path using exclude-files-from-classmap
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

  laravel/breeze ................................................................................. DONE
  laravel/pail ................................................................................... DONE
  laravel/sail ................................................................................... DONE
  laravel/tinker ................................................................................. DONE
  nesbot/carbon .................................................................................. DONE
  nunomaduro/collision ........................................................................... DONE
  nunomaduro/termwind ............................................................................ DONE
  pestphp/pest-plugin-laravel .................................................................... DONE

87 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^7.9 for guzzlehttp/guzzle

C:\Users\tongq\Desktop\fyp\fyp>
```

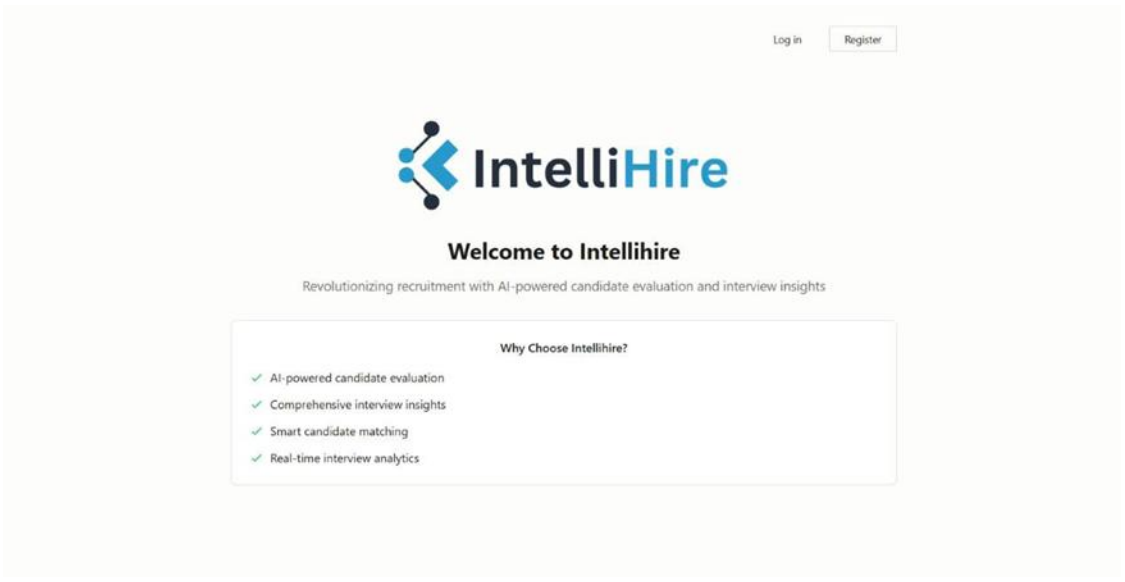
Figure 5.3.4.1 GuzzleHTTP installation

5.4 System Operation (with Screenshot)

This section demonstrates how the IntelliHire system operates from both candidate and recruiter perspectives. Screenshots of the user interface are provided together with explanations of system flows. For complex features such as AI-driven operations, code snippets are also included to show the underlying implementation.

5.4.1 Landing Page & Authentication

Upon accessing the IntelliHire platform, users are greeted with a landing page that introduces the system in a clean and intuitive layout (Figure 5.4.1). From here, users may proceed to authentication features. The login page (Figure 5.4.2) allows both candidates and recruiters to access their accounts using registered email and password. For new users, the registration page (Figure 5.4.3) provides account creation with an additional role selection option, where users must indicate whether they are registering as a candidate or a recruiter. This ensures that upon successful login, each user is directed to the correct dashboard tailored to their role.



Welcome to Intellihire

Revolutionizing recruitment with AI-powered candidate evaluation and interview insights

Why Choose Intellihire?

- ✓ AI-powered candidate evaluation
- ✓ Comprehensive interview insights
- ✓ Smart candidate matching
- ✓ Real-time interview analytics

Figure 5.4.1.1 IntelliHire landing page

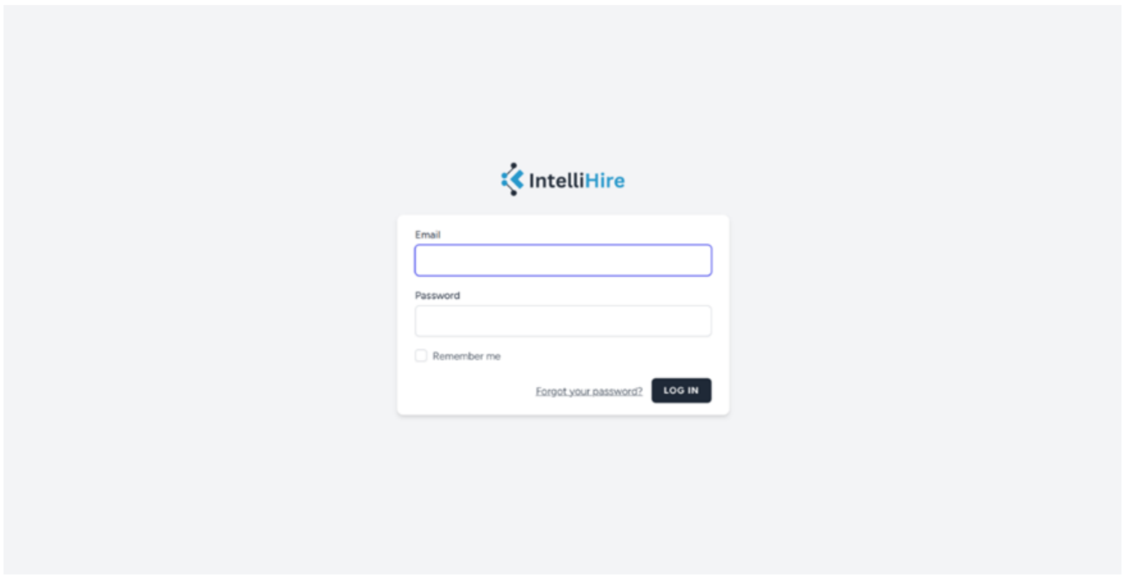


Figure 5.4.1.2 IntelliHire login page

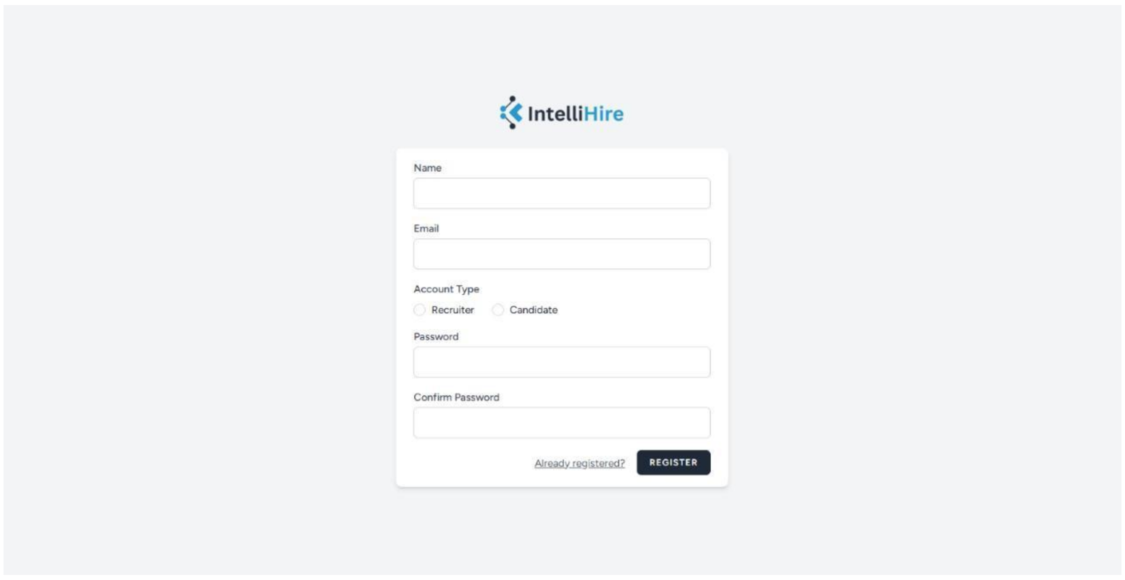


Figure 5.4.1.3 IntelliHire registration page

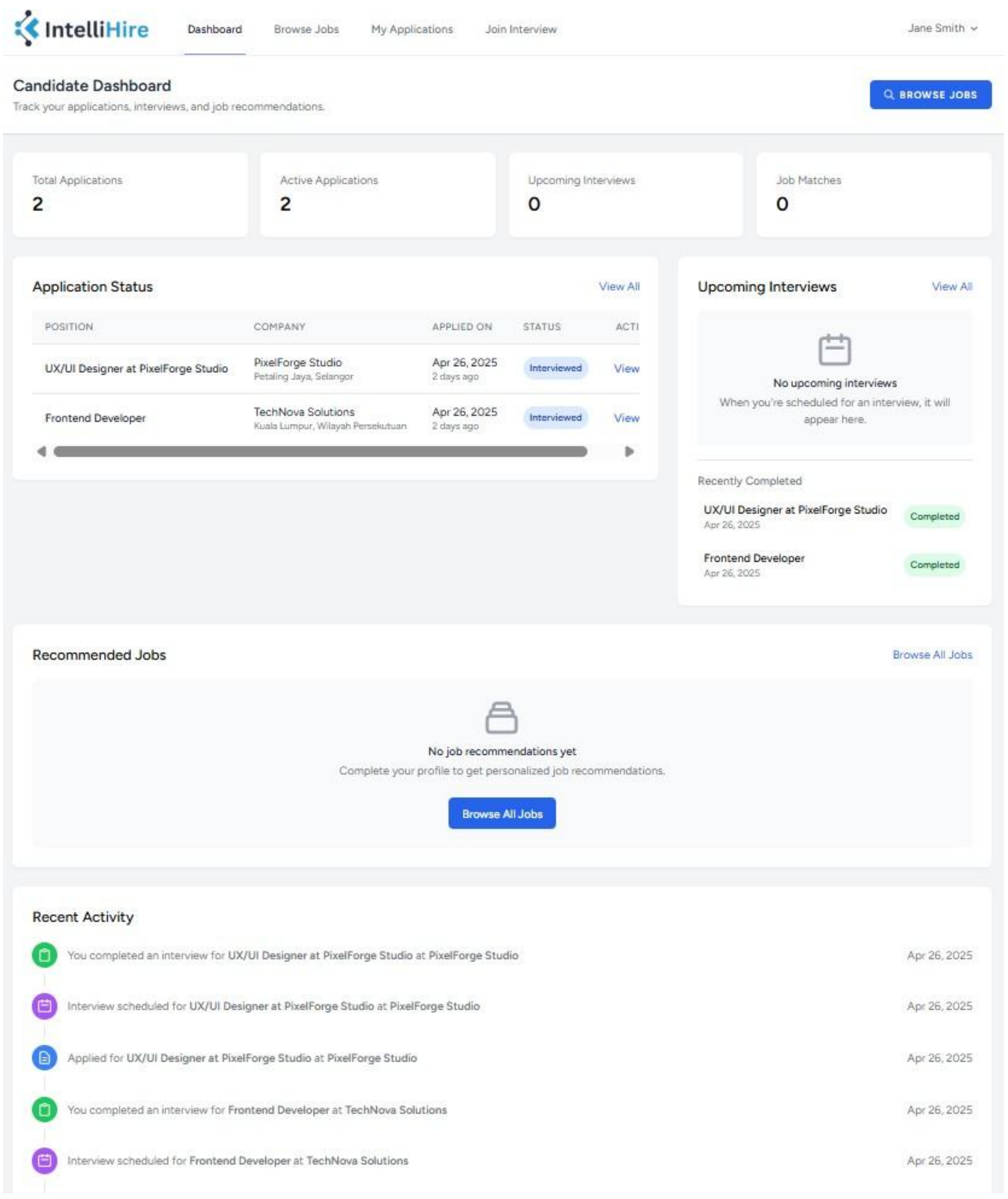
5.4.2 Candidate Dashboard

The landing dashboard for candidates is composed of multiple dynamic components aimed at improving the job search and application experience. At the top of the page, a summary section presents key statistics such as the total number of applications, active applications, upcoming interviews, and job matches. These figures are displayed in styled blocks to give users a quick overview of their current status.

Below this, the system provides personalized job recommendations. While the current implementation suggests jobs randomly, this feature is designed to encourage continued engagement with the platform, and future enhancements may allow recommendations to be filtered based on the candidate’s skills, preferences, or past applications for greater accuracy.

The dashboard also includes an activity timeline, which records significant events such as job applications, interview schedules, and completed interviews. This chronological log helps candidates stay informed of their progress throughout the hiring process. Additionally, an upcoming interviews section highlights any sessions that have been scheduled, along with direct links to access them when the time comes. This ensures that candidates remain aware of important deadlines and do not miss scheduled interviews.

Finally, the application status component displays a detailed table of all submitted applications, including the job title, company name, application date, and current status. Each record also provides a shortcut to the full application details page, making it easy for candidates to review and manage their submissions. Together, these components make the dashboard a central hub for tracking applications, managing interviews, and discovering new opportunities.



Candidate Dashboard
Track your applications, interviews, and job recommendations.

Total Applications	Active Applications	Upcoming Interviews	Job Matches
2	2	0	0

Application Status — View All

POSITION	COMPANY	APPLIED ON	STATUS	ACTI
UX/UI Designer at PixelForge Studio	PixelForge Studio Petaling Jaya, Selangor	Apr 26, 2025 2 days ago	Interviewed	View
Frontend Developer	TechNova Solutions Kuala Lumpur, Wilayah Persekutuan	Apr 26, 2025 2 days ago	Interviewed	View

Upcoming Interviews — View All

No upcoming interviews
When you're scheduled for an interview, it will appear here.

Recently Completed

UX/UI Designer at PixelForge Studio Apr 26, 2025	Completed
Frontend Developer Apr 26, 2025	Completed

Recommended Jobs — Browse All Jobs

No job recommendations yet
Complete your profile to get personalized job recommendations.
Browse All Jobs

Recent Activity

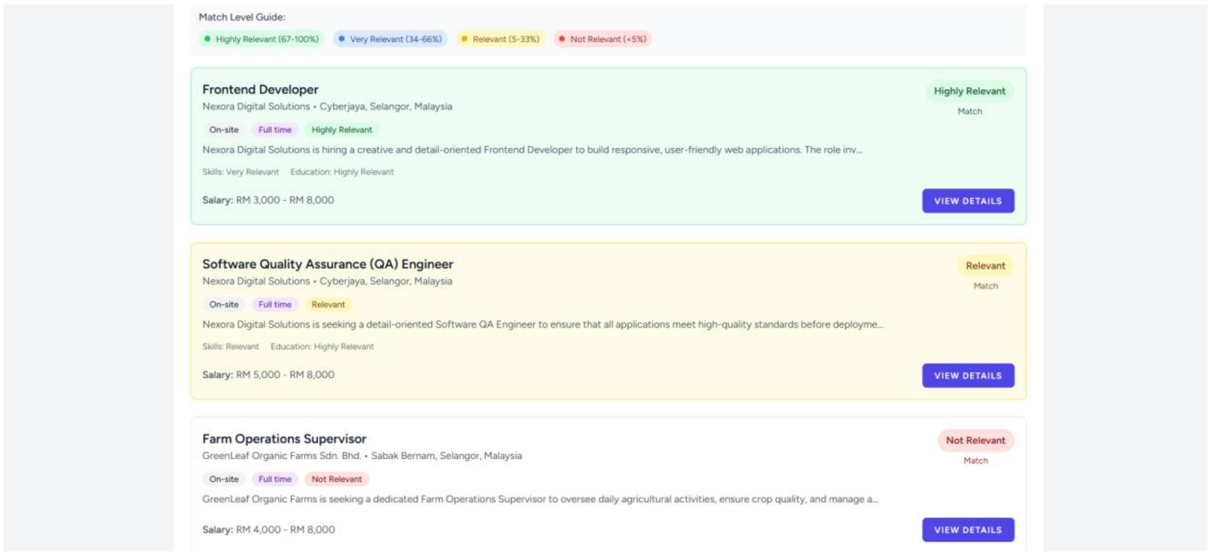
You completed an interview for UX/UI Designer at PixelForge Studio at PixelForge Studio	Apr 26, 2025
Interview scheduled for UX/UI Designer at PixelForge Studio at PixelForge Studio	Apr 26, 2025
Applied for UX/UI Designer at PixelForge Studio at PixelForge Studio	Apr 26, 2025
You completed an interview for Frontend Developer at TechNova Solutions	Apr 26, 2025
Interview scheduled for Frontend Developer at TechNova Solutions	Apr 26, 2025

Figure 5.4.2.1 IntelliHire candidate dashboard

5.4.3 Job Browsing & Application

In the jobs module for candidates, the system provides two distinct browsing experiences depending on whether resume data exists in the database.

When resume information is available, meaning the candidate has uploaded and parsed a resume during a previous application, the job browsing page leverages the AI-powered matching module to enhance the listings. Each job card includes a compatibility score that represents how closely the candidate's qualifications align with the job requirements. These scores are generated through the AI matching algorithm, which compares extracted resume data—such as skills, education, experience, and certifications—against the requirements parsed from job postings. Jobs are then displayed in ranked order, with the most compatible positions appearing first.



Match Level Guide:

● Highly Relevant (67-100%) ● Very Relevant (34-66%) ● Relevant (5-33%) ● Not Relevant (<5%)

Frontend Developer — Highly Relevant Match

Nexora Digital Solutions • Cyberjaya, Selangor, Malaysia

On-site | Full time | Highly Relevant

Nexora Digital Solutions is hiring a creative and detail-oriented Frontend Developer to build responsive, user-friendly web applications. The role inv...

Skills: Very Relevant Education: Highly Relevant

Salary: RM 3,000 - RM 8,000

VIEW DETAILS

Software Quality Assurance (QA) Engineer — Relevant Match

Nexora Digital Solutions • Cyberjaya, Selangor, Malaysia

On-site | Full time | Relevant

Nexora Digital Solutions is seeking a detail-oriented Software QA Engineer to ensure that all applications meet high-quality standards before deployme...

Skills: Relevant Education: Highly Relevant

Salary: RM 5,000 - RM 8,000

VIEW DETAILS

Farm Operations Supervisor — Not Relevant Match

GreenLeaf Organic Farms Sdn. Bhd. • Sabak Bernam, Selangor, Malaysia

On-site | Full time | Not Relevant

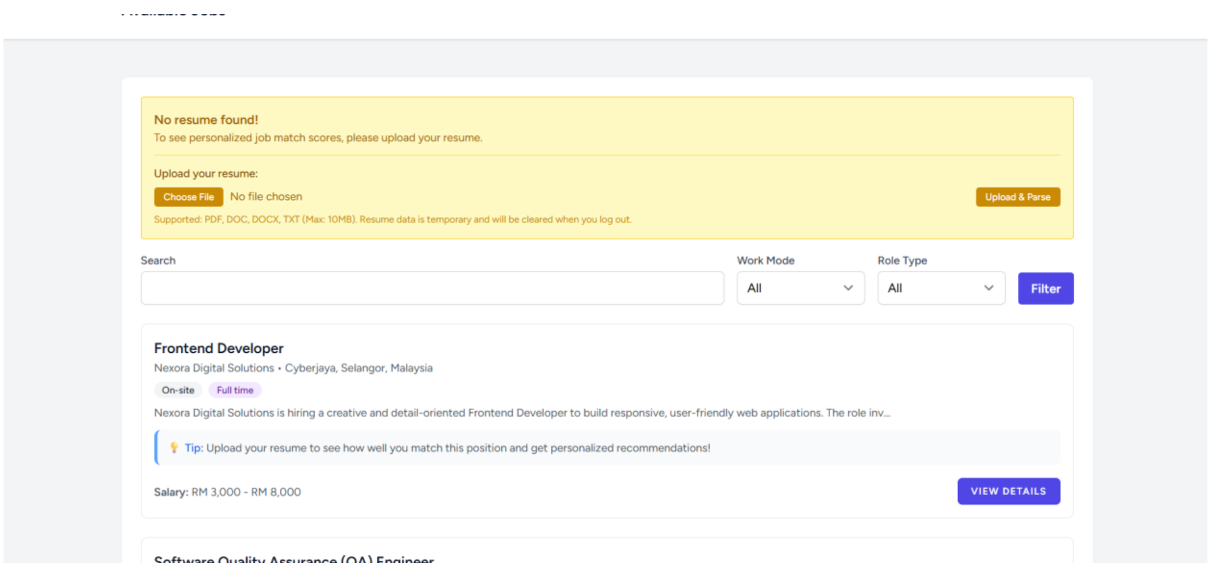
GreenLeaf Organic Farms is seeking a dedicated Farm Operations Supervisor to oversee daily agricultural activities, ensure crop quality, and manage a...

Salary: RM 4,000 - RM 8,000

VIEW DETAILS

Figure 5.4.3.1 Intellihire candidate job browsing page (with resume information found in database)

When no resume information is available, for example when a candidate has not applied for any jobs previously, the job browsing page reverts to a standard listing view. In this case, jobs are presented in a neutral order (typically by posting date), displaying basic information such as job title, company, salary range, and description. Since no compatibility scores can be calculated, the interface provides clear prompts encouraging candidates to upload their resumes in order to unlock AI-driven job recommendations. Informational banners or placeholder messages may also appear, emphasizing the advantages of activating intelligent matching features. This ensures that while all candidates can access job postings, those who upload resumes enjoy a significantly more tailored and guided job search experience.



No resume found!
To see personalized job match scores, please upload your resume.

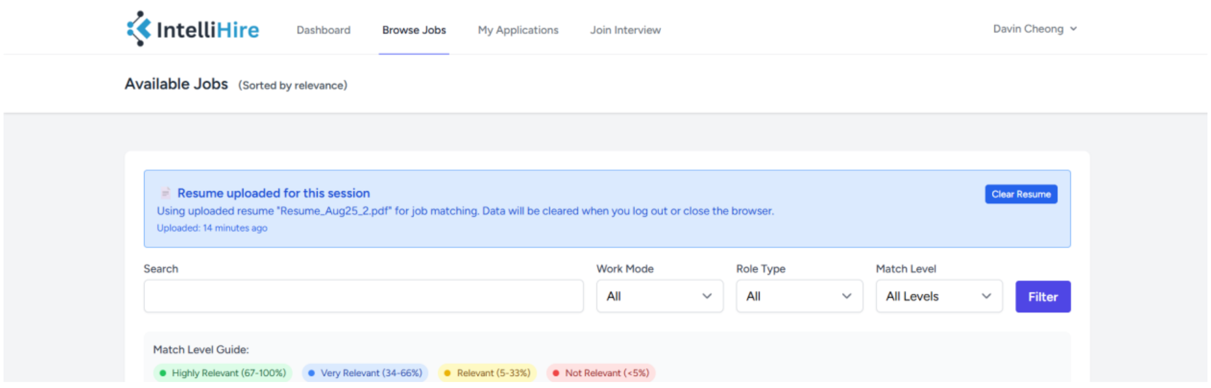
Upload your resume:
Choose File No file chosen Upload & Parse
Supported: PDF, DOC, DOCX, TXT (Max: 10MB). Resume data is temporary and will be cleared when you log out.

Search | Work Mode: All | Role Type: All | Filter

Frontend Developer
Nexora Digital Solutions • Cyberjaya, Selangor, Malaysia
On-site Full time
Nexora Digital Solutions is hiring a creative and detail-oriented Frontend Developer to build responsive, user-friendly web applications. The role inv...
Tip: Upload your resume to see how well you match this position and get personalized recommendations!
Salary: RM 3,000 - RM 8,000 VIEW DETAILS

Software Quality Assurance (QA) Engineer

Figure 5.4.3.2 Intellihire candidate job browsing page (with no resume information found in database)



IntelliHire Dashboard Browse Jobs My Applications Join Interview Davin Cheong

Available Jobs (Sorted by relevance)

Resume uploaded for this session
Using uploaded resume "Resume_Aug25_2.pdf" for job matching. Data will be cleared when you log out or close the browser.
Uploaded: 14 minutes ago
Clear Resume

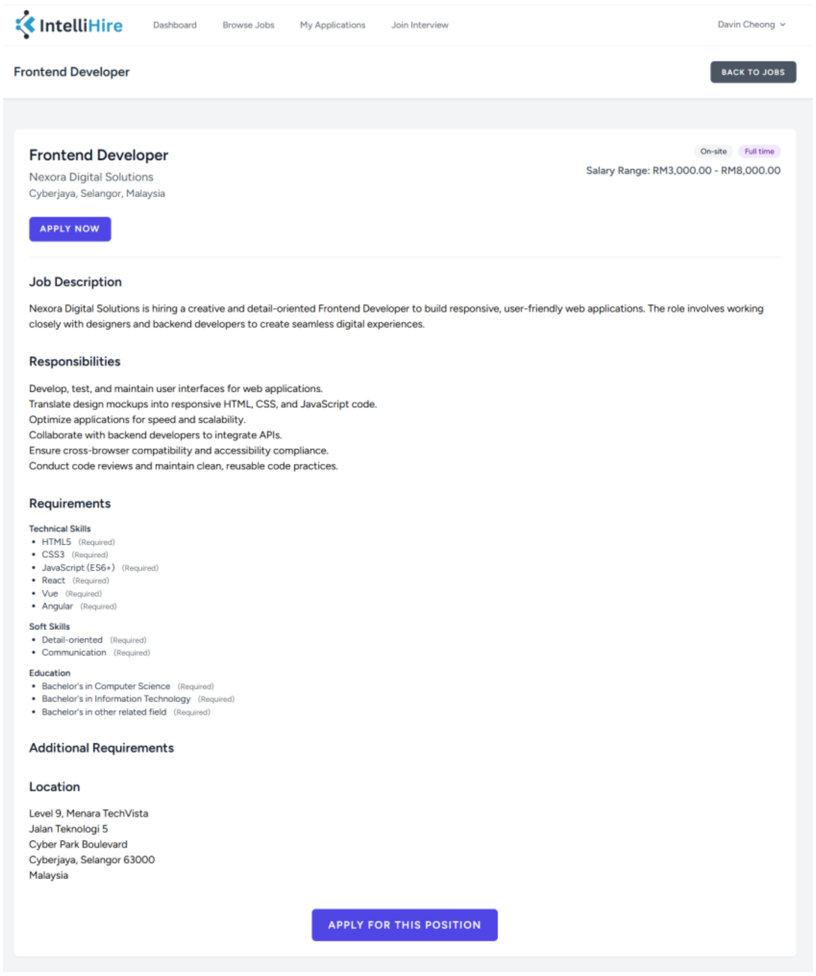
Search | Work Mode: All | Role Type: All | Match Level: All Levels | Filter

Match Level Guide:
● Highly Relevant (67-100%) ● Very Relevant (34-66%) ● Relevant (5-33%) ● Not Relevant (<5%)

Figure 5.4.3.3 Intellihire candidate job browsing page (resume uploaded for session calculations)

This dual-interface approach ensures inclusivity while maximizing personalization for candidates who engage with the system fully.

Once a candidate identifies an interesting opportunity from the job listings or recommendations, they are directed to the Job Details page (Figure 5.4.3.4). This page provides comprehensive information about the position, including the job title, company name, job description, location, salary range, and specific requirements defined by the recruiter. The structured view ensures that candidates can make informed decisions before proceeding with an application.



Frontend Developer

Frontend Developer
Nexora Digital Solutions
Cyberjaya, Selangor, Malaysia

Salary Range: RM3,000.00 - RM8,000.00

Job Description

Nexora Digital Solutions is hiring a creative and detail-oriented Frontend Developer to build responsive, user-friendly web applications. The role involves working closely with designers and backend developers to create seamless digital experiences.

Responsibilities

Develop, test, and maintain user interfaces for web applications.
Translate design mockups into responsive HTML, CSS, and JavaScript code.
Optimize applications for speed and scalability.
Collaborate with backend developers to integrate APIs.
Ensure cross-browser compatibility and accessibility compliance.
Conduct code reviews and maintain clean, reusable code practices.

Requirements

Technical Skills
- HTML5 (Required)
- CSS3 (Required)
- JavaScript (ES6+) (Required)
- React (Required)
- Vue (Required)
- Angular (Required)

Soft Skills
- Detail-oriented (Required)
- Communication (Required)

Education
- Bachelor's in Computer Science (Required)
- Bachelor's in Information Technology (Required)
- Bachelor's in other related field (Required)

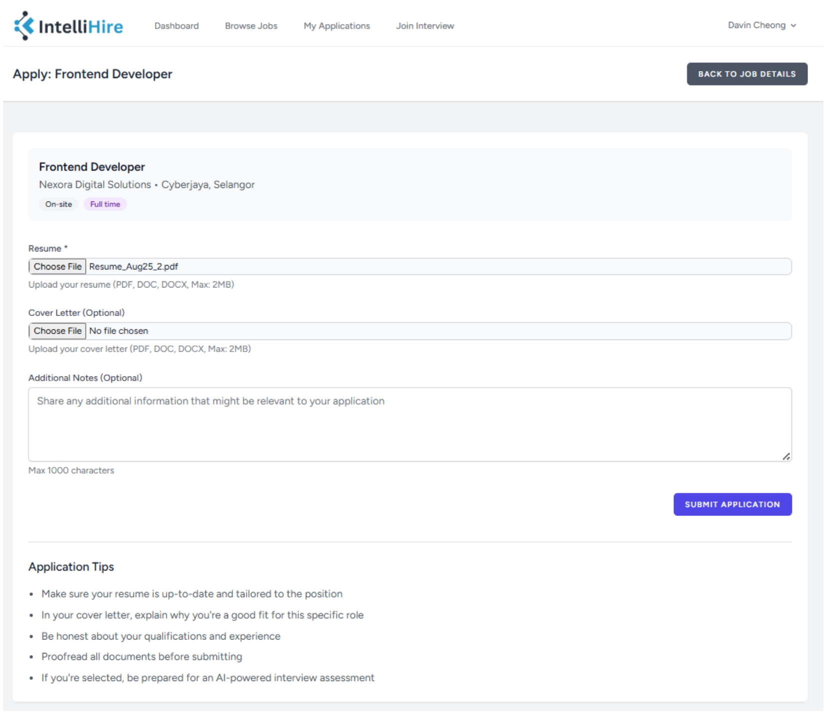
Additional Requirements

Location

Level 9, Menara TechVista
Jalan Teknologi 5
Cyber Park Boulevard
Cyberjaya, Selangor 63000
Malaysia

Figure 5.4.3.4 IntelliHire candidate view job details page

If the candidate chooses to apply, they are redirected to the Application page (Figure 5.4.3.5). Here, the system provides a form where candidates can upload their resume and an optional cover letter, before submitting the application to the database. The interface is designed to be straightforward and user-friendly, reducing friction during the application process.



Apply: Frontend Developer

Frontend Developer
Nexora Digital Solutions • Cyberjaya, Selangor
On-site Full time

Resume *
Resume_Aug25_2.pdf
Upload your resume (PDF, DOC, DOCX, Max: 2MB)

Cover Letter (Optional)
No file chosen
Upload your cover letter (PDF, DOC, DOCX, Max: 2MB)

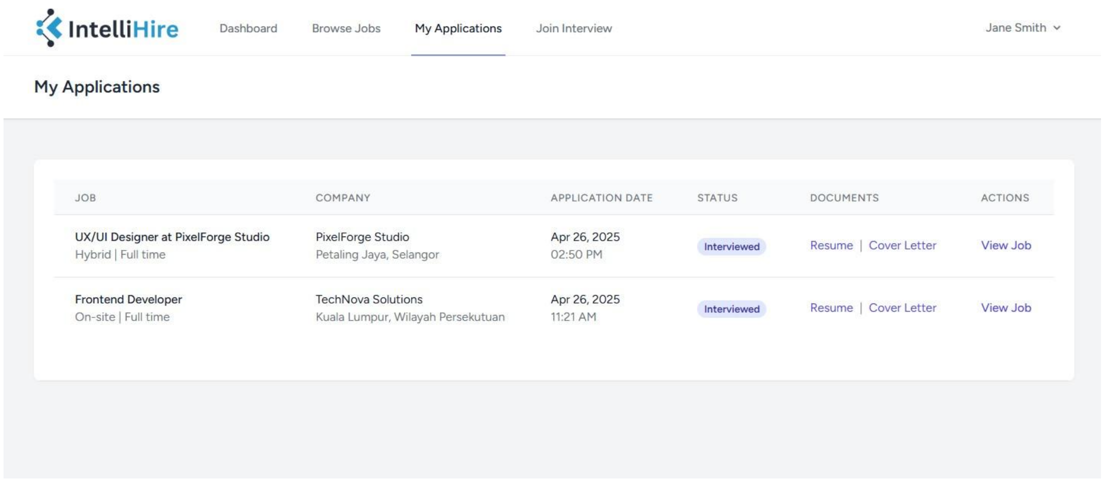
Additional Notes (Optional)
Share any additional information that might be relevant to your application
Max 1000 characters

Application Tips

- Make sure your resume is up-to-date and tailored to the position
- In your cover letter, explain why you're a good fit for this specific role
- Be honest about your qualifications and experience
- Proofread all documents before submitting
- If you're selected, be prepared for an AI-powered interview assessment

Figure 5.4.3.5 IntelliHire candidate application page

Candidates may later review their submissions through the Applied Jobs page (Figure 5.4.3.6), which lists all the positions they have applied to, along with essential details such as job title, company, and application date. This acts as a personal record, helping candidates keep track of their ongoing applications.

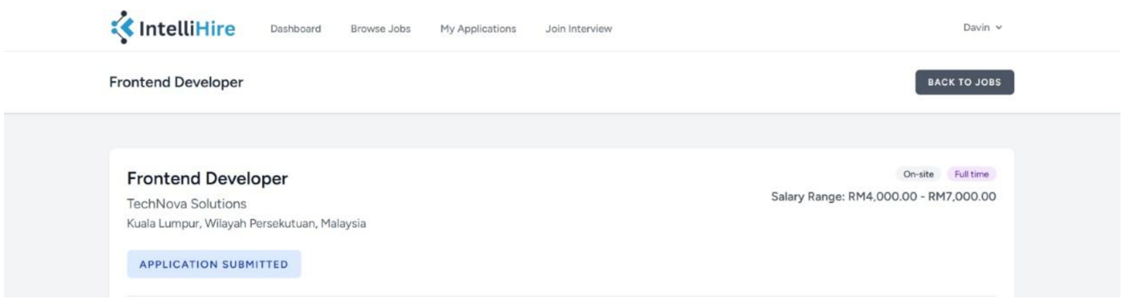


My Applications

JOB	COMPANY	APPLICATION DATE	STATUS	DOCUMENTS	ACTIONS
UX/UI Designer at PixelForge Studio Hybrid | Full time	PixelForge Studio Petaling Jaya, Selangor	Apr 26, 2025 02:50 PM	Interviewed	Resume | Cover Letter	View Job
Frontend Developer On-site | Full time	TechNova Solutions Kuala Lumpur, Wilayah Persekutuan	Apr 26, 2025 11:21 AM	Interviewed	Resume | Cover Letter	View Job

Figure 5.4.3.6 IntelliHire candidate applied job page

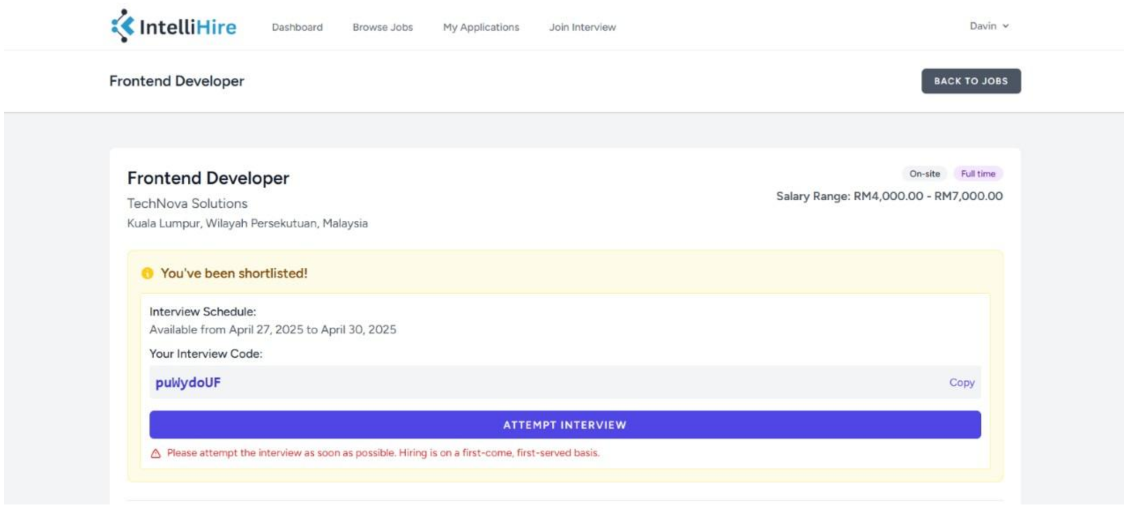
To provide greater transparency, the system also includes a Status Tracking page (Figures 5.4.3.7 - 5.4.3.12). This page displays detailed information about a specific job and the current progress of the candidate's application. There are six distinct statuses used within the platform: application submitted, shortlisted, shortlisted but expired, interviewed, offered, and rejected. Each status is visually highlighted so candidates can easily understand their standing in the recruitment process.



Frontend Developer

Frontend Developer
TechNova Solutions
Kuala Lumpur, Wilayah Persekutuan, Malaysia
Salary Range: RM4,000.00 - RM7,000.00

Figure 5.4.3.7 IntelliHire candidate status: applied



Frontend Developer

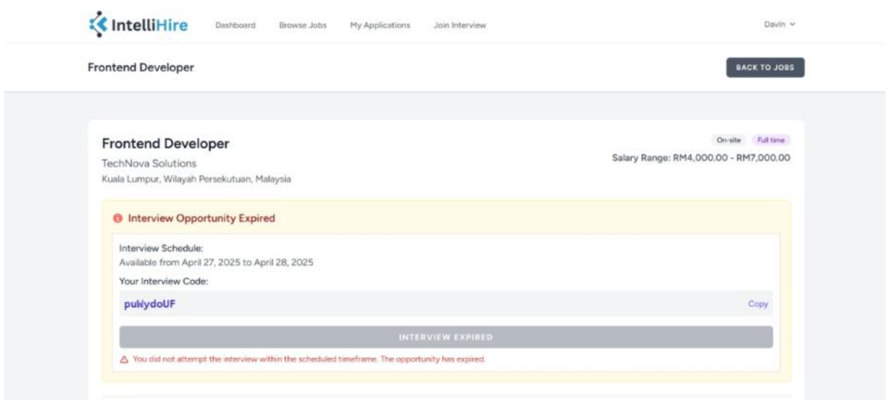
Frontend Developer
TechNova Solutions
Kuala Lumpur, Wilayah Persekutuan, Malaysia
Salary Range: RM4,000.00 - RM7,000.00

You've been shortlisted!

Interview Schedule:
Available from April 27, 2025 to April 30, 2025
Your Interview Code:
puWydoUF

Please attempt the interview as soon as possible. Hiring is on a first-come, first-served basis.

Figure 5.4.3.8 IntelliHire candidate status: shortlisted



Frontend Developer

Frontend Developer
TechNova Solutions
Kuala Lumpur, Wilayah Persekutuan, Malaysia

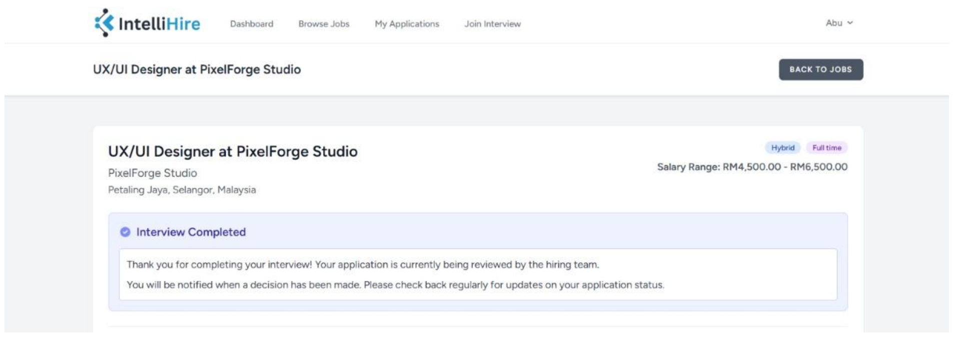
On-site Full time
Salary Range: RM4,000.00 - RM7,000.00

Interview Opportunity Expired

Interview Schedule:
Available from April 27, 2025 to April 28, 2025
Your Interview Code:
puWydoUF

You did not attempt the interview within the scheduled timeframe. The opportunity has expired.

Figure 5.4.3.9 IntelliHire candidate status: shortlisted, but expired



UX/UI Designer at PixelForge Studio

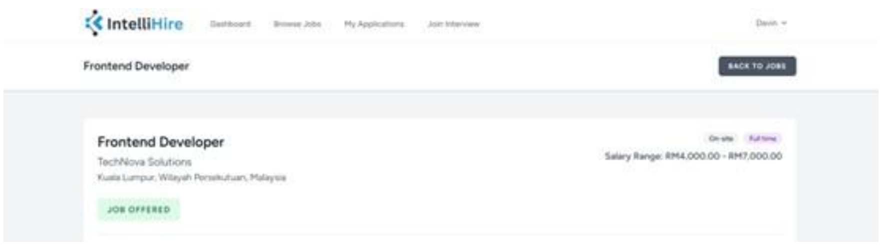
UX/UI Designer at PixelForge Studio
PixelForge Studio
Petaling Jaya, Selangor, Malaysia

Hybrid Full time
Salary Range: RM4,500.00 - RM6,500.00

Interview Completed

Thank you for completing your interview! Your application is currently being reviewed by the hiring team.
You will be notified when a decision has been made. Please check back regularly for updates on your application status.

Figure 5.4.3.10 IntelliHire candidate status: interviewed



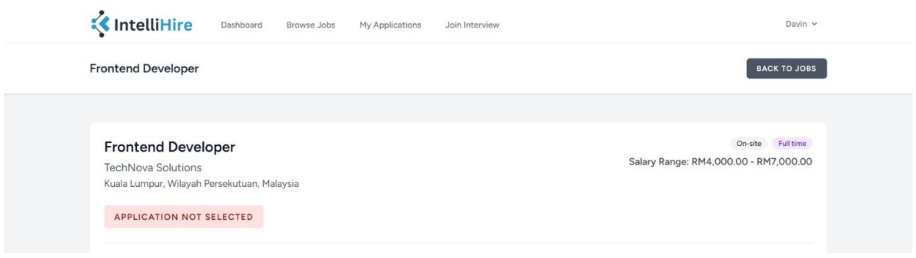
Frontend Developer

Frontend Developer
TechNova Solutions
Kuala Lumpur, Wilayah Persekutuan, Malaysia

On-site Full time
Salary Range: RM4,000.00 - RM7,000.00

JOB OFFERED

Figure 5.4.3.11 IntelliHire candidate status: offered



Frontend Developer

Frontend Developer
TechNova Solutions
Kuala Lumpur, Wilayah Persekutuan, Malaysia

On-site Full time
Salary Range: RM4,000.00 - RM7,000.00

APPLICATION NOT SELECTED

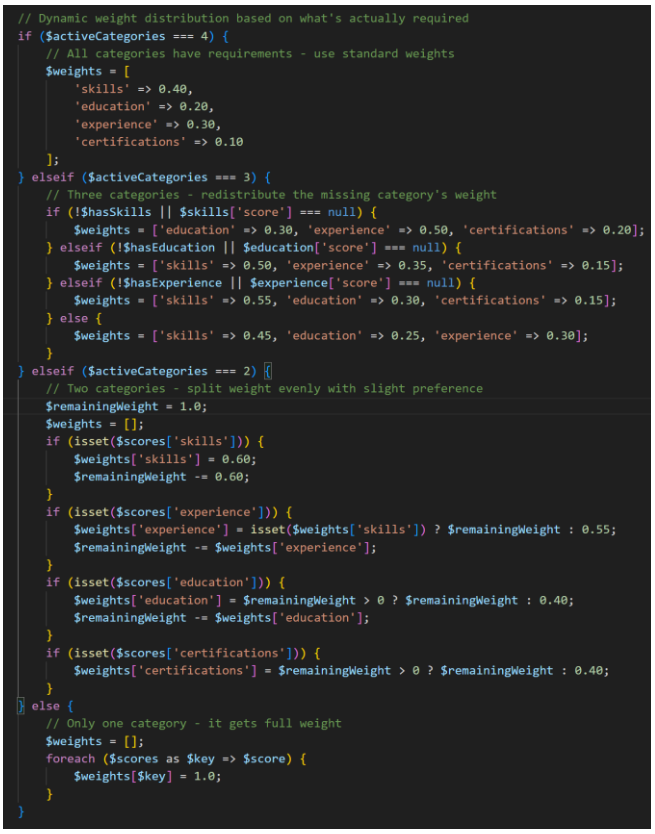
Figure 5.4.3.12 IntelliHire candidate status: rejected

In addition, after an interview is completed, the same page provides access to the candidate's interview responses and AI-generated feedback from the session. This feature not only enhances candidate awareness of their performance but also encourages self-improvement by presenting detailed insights into their communication, relevance, and completeness of answers.

5.4.3.1 Job Matching Algorithm

The job matching algorithm in IntelliHire combines traditional rule-based techniques with AI-powered semantic analysis to deliver accurate compatibility scores between candidate resumes and job postings. This hybrid approach ensures that the system remains resilient when AI services encounter issues, while also benefiting from contextual intelligence when they are available.

The matching process begins with a weighted scoring system. Each requirement type—skills, education, experience, and certifications—is assigned a weight based on its importance. For example, skills are weighted at 40%, experience at 30%, education at 20%, and certifications at 10%. If a job posting omits one or more requirement categories, the weights are dynamically adjusted to normalize the final calculation. The overall compatibility score is then computed as a weighted sum of the individual category scores, producing a percentage that reflects how closely the candidate fits the job profile.

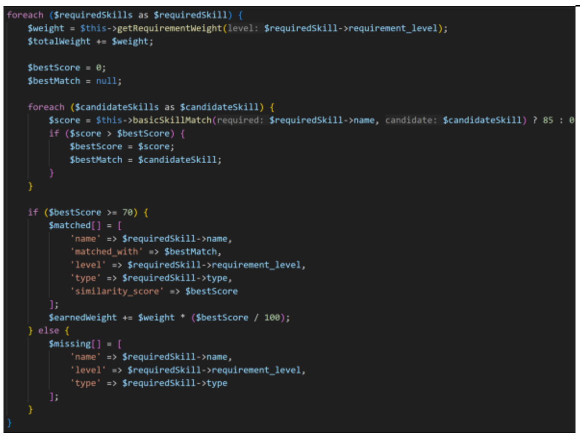


```php
// Dynamic weight distribution based on what's actually required
if ($activeCategories === 4) {
    // All categories have requirements - use standard weights
    $weights = [
        'skills' => 0.40,
        'education' => 0.20,
        'experience' => 0.30,
        'certifications' => 0.10
    ];
} elseif ($activeCategories === 3) {
    // Three categories - redistribute the missing category's weight
    if (!$hasSkills || $skills['score'] === null) {
        $weights = ['education' => 0.30, 'experience' => 0.50, 'certifications' => 0.20];
    } elseif (!$hasEducation || $education['score'] === null) {
        $weights = ['skills' => 0.50, 'experience' => 0.35, 'certifications' => 0.15];
    } elseif (!$hasExperience || $experience['score'] === null) {
        $weights = ['skills' => 0.55, 'education' => 0.30, 'certifications' => 0.15];
    } else {
        $weights = ['skills' => 0.45, 'education' => 0.25, 'experience' => 0.30];
    }
} elseif ($activeCategories === 2) {
    // Two categories - split weight evenly with slight preference
    $remainingWeight = 1.0;
    $weights = [];
    if (isset($scores['skills'])) {
        $weights['skills'] = 0.60;
        $remainingWeight -= 0.60;
    }
    if (isset($scores['experience'])) {
        $weights['experience'] = isset($weights['skills']) ? $remainingWeight : 0.55;
        $remainingWeight -= $weights['experience'];
    }
    if (isset($scores['education'])) {
        $weights['education'] = $remainingWeight > 0 ? $remainingWeight : 0.40;
        $remainingWeight -= $weights['education'];
    }
    if (isset($scores['certifications'])) {
        $weights['certifications'] = $remainingWeight > 0 ? $remainingWeight : 0.40;
    }
} else {
    // Only one category - it gets full weight
    $weights = [];
    foreach ($scores as $key => $score) {
        $weights[$key] = 1.0;
    }
}
```

Figure 5.4.3.13 Job matching algorithm weighted scoring

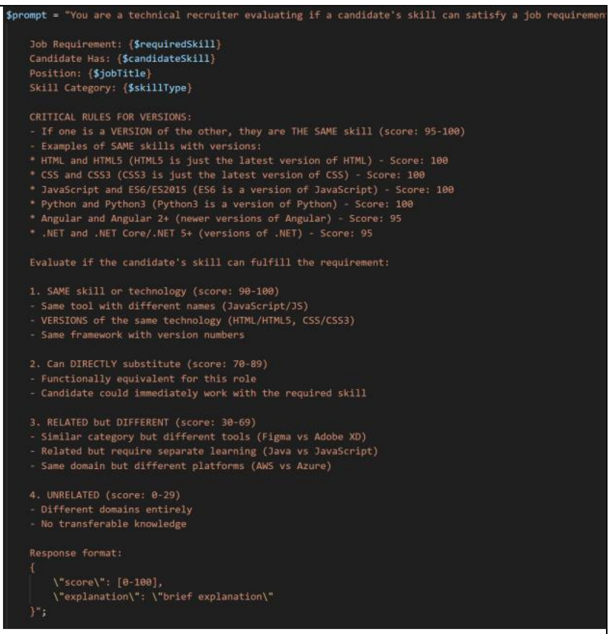
For skills matching, the system uses a two-layer approach. First, a rule-based algorithm compares each required skill with the candidate’s skills using keyword similarity and thresholds. As shown in Figure 5.4.3.5, the system iterates through each requirement, calculates similarity scores, and selects the best match. If the best score exceeds 70%, the requirement is considered satisfied; otherwise, it is marked as missing. This ensures that the system always produces a baseline score, even if AI services are unavailable.

To enhance accuracy, the algorithm then integrates semantic evaluation using the Gemini API. Figure 5.4.3.6 illustrates how a detailed AI prompt is constructed, containing contextual rules that define how to treat versions, synonyms, and related skills. For example, HTML and HTML5 are treated as equivalent, while Angular and Angular 2+ are categorized as different versions of the same technology. The API evaluates each candidate skill against job requirements and returns a structured response with both a numerical score and an explanation. These AI-generated scores are then merged with the rule-based calculations to refine the overall compatibility score, ensuring that both exact matches and semantically similar skills are recognized.



```php
foreach ($requiredSkills as $requiredSkill) {
    $weight = $this->getRequirementWeight(level: $requiredSkill->requirement_level);
    $totalWeight += $weight;

    $bestScore = 0;
    $bestMatch = null;

    foreach ($candidateSkills as $candidateSkill) {
        $score = $this->basicSkillMatch(required: $requiredSkill->name, candidate: $candidateSkill) ? 85 : 0
        if ($score > $bestScore) {
            $bestScore = $score;
            $bestMatch = $candidateSkill;
        }
    }

    if ($bestScore >= 70) {
        $matched[] = [
            'name' => $requiredSkill->name,
            'matched_with' => $bestMatch,
            'level' => $requiredSkill->requirement_level,
            'type' => $requiredSkill->type,
            'similarity_score' => $bestScore
        ];
        $earnedWeight += $weight * ($bestScore / 100);
    } else {
        $missing[] = [
            'name' => $requiredSkill->name,
            'level' => $requiredSkill->requirement_level,
            'type' => $requiredSkill->type
        ];
    }
}
```

Figure 5.4.3.14 Rule-based skill matching with weighted scoring

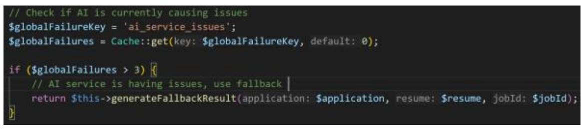


```php
$prompt = "You are a technical recruiter evaluating if a candidate's skill can satisfy a job requiremen

    Job Requirement: {$requiredSkill}
    Candidate Has: {$candidateSkill}
    Position: {$jobTitle}
    Skill Category: {$skillType}

    CRITICAL RULES FOR VERSIONS:
    - If one is a VERSION of the other, they are THE SAME skill (score: 95-100)
    - Examples of SAME skills with versions:
    * HTML and HTML5 (HTML5 is just the latest version of HTML) - Score: 100
    * CSS and CSS3 (CSS3 is just the latest version of CSS) - Score: 100
    * JavaScript and ES6/ES2015 (ES6 is a version of JavaScript) - Score: 100
    * Python and Python3 (Python3 is a version of Python) - Score: 100
    * Angular and Angular 2+ (newer versions of Angular) - Score: 95
    * .NET and .NET Core/.NET 5+ (versions of .NET) - Score: 95

    Evaluate if the candidate's skill can fulfill the requirement:

    1. SAME skill or technology (score: 90-100)
    - Same tool with different names (JavaScript/JS)
    - VERSIONS of the same technology (HTML/HTML5, CSS/CSS3)
    - Same framework with version numbers

    2. Can DIRECTLY substitute (score: 70-89)
    - Functionally equivalent for this role
    - Candidate could immediately work with the required skill

    3. RELATED but DIFFERENT (score: 30-69)
    - Similar category but different tools (Figma vs Adobe XD)
    - Related but require separate learning (Java vs JavaScript)
    - Same domain but different platforms (AWS vs Azure)

    4. UNRELATED (score: 0-29)
    - Different domains entirely
    - No transferable knowledge

    Response format:
    {
        \"score\": [0-100],
        \"explanation\": \"brief explanation\"
    }";
```

Figure 5.4.3.15 AI prompt construction for semantic skill evaluation

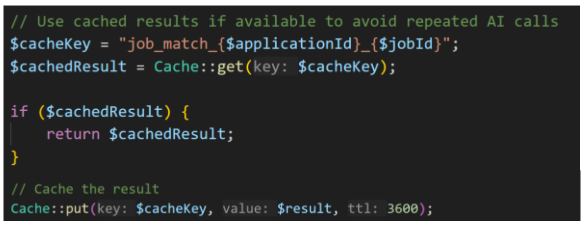
A similar layered design applies to education, experience, and certifications. Education matching checks degree level and field of study, with AI factoring in GPA and institution prestige. Experience matching compares years of work and uses AI to assess role relevance, industry context, and career progression. Certifications are matched by name and validated through AI for equivalence, currency, and relevance to the role.

The system incorporates several optimization and reliability mechanisms. A circuit breaker pattern monitors Gemini API failures; if more than 3 consecutive errors occur, the algorithm falls back to basic keyword matching to maintain uninterrupted service. In addition, AI results are cached for 1 hour to avoid repeated calls for the same candidate-job pair, improving efficiency. The system implements rate limiting (50 requests per hour) and failure caching (5-minute cooldown) to prevent repeated failed attempts. Requests to the API are configured with aggressive timeout controls (8 seconds), ensuring that transient network or service issues do not severely impact user experience.



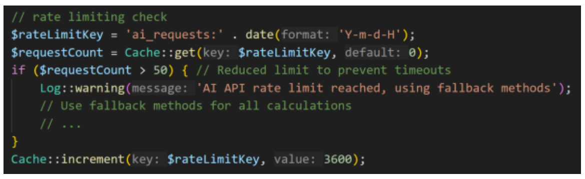
```
// Check if AI is currently causing issues
$globalFailureKey = 'ai_service_issues';
$globalFailures = Cache::get(key: $globalFailureKey, default: 0);

if ($globalFailures > 3) {
    // AI service is having issues, use fallback
    return $this->generateFallbackResult(application: $application, resume: $resume, jobId: $jobId);
}
```

Figure 5.4.3.16 Circuit Breaker Pattern



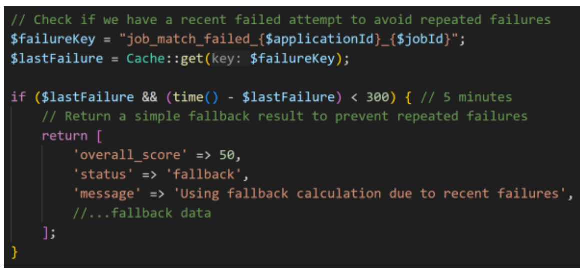
```
// Use cached results if available to avoid repeated AI calls
$cacheKey = "job_match_{$applicationId}_{$jobId}";
$cachedResult = Cache::get(key: $cacheKey);

if ($cachedResult) {
    return $cachedResult;
}

// Cache the result
Cache::put(key: $cacheKey, value: $result, ttl: 3600);
```

Figure 5.4.3.17 Caching Implementation



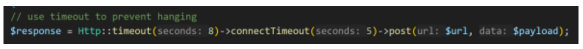
```
// rate limiting check
$rateLimitKey = 'ai_requests:' . date(format: 'Y-m-d-H');
$requestCount = Cache::get(key: $rateLimitKey, default: 0);
if ($requestCount > 50) { // Reduced limit to prevent timeouts
    Log::warning(message: 'AI API rate limit reached, using fallback methods');
    // Use fallback methods for all calculations
    // ...
}
Cache::increment(key: $rateLimitKey, value: 3600);
```

Figure 5.4.3.18 Rate Limiting



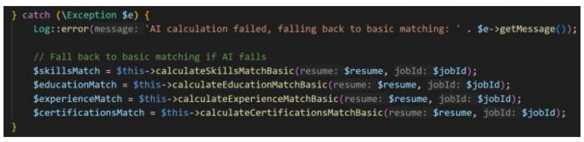
```
// Check if we have a recent failed attempt to avoid repeated failures
$failureKey = "job_match_failed_{$applicationId}_{$jobId}";
$lastFailure = Cache::get(key: $failureKey);

if ($lastFailure && (time() - $lastFailure) < 300) { // 5 minutes
    // Return a simple fallback result to prevent repeated failures
    return [
        'overall_score' => 50,
        'status' => 'fallback',
        'message' => 'Using fallback calculation due to recent failures',
        //...fallback data
    ];
}
```

Figure 5.4.3.19 Failure Caching



```
// use timeout to prevent hanging
$response = Http::timeout(seconds: 8)->connectTimeout(seconds: 5)->post(url: $url, data: $payload);
```

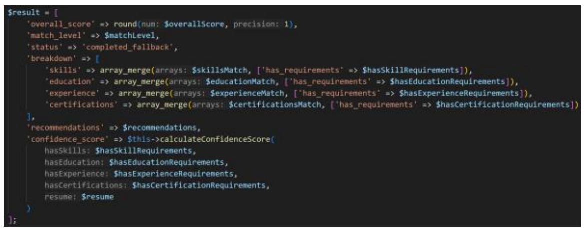
Figure 5.4.3.20 Timeout Controls



```
} catch (\Exception $e) {
    Log::error(message: 'AI calculation failed, falling back to basic matching: ' . $e->getMessage());

    // Fall back to basic matching if AI fails
    $skillsMatch = $this->calculateSkillsMatchBasic(resume: $resume, jobId: $jobId);
    $educationMatch = $this->calculateEducationMatchBasic(resume: $resume, jobId: $jobId);
    $experienceMatch = $this->calculateExperienceMatchBasic(resume: $resume, jobId: $jobId);
    $certificationsMatch = $this->calculateCertificationsMatchBasic(resume: $resume, jobId: $jobId);
}
```

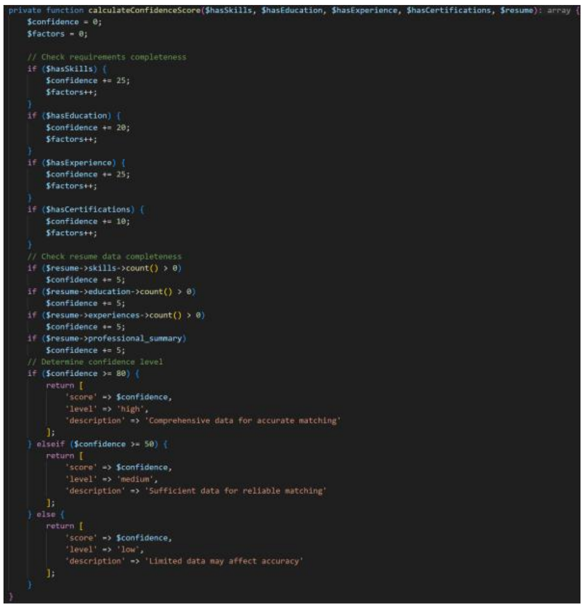
Figure 5.4.3.21 Fallback to Basic Matching

The final output combines the weighted overall score, breakdown by category, and a confidence score that communicates the reliability of the AI's analysis. Confidence levels are categorized as high (80%+), medium (50-79%), or low (below 50%), helping in interpret the results with appropriate caution.



```php
$result = [
    'overall_score' => round(num: $overallScore, precision: 1),
    'match_level' => $matchLevel,
    'status' => 'completed_fallback',
    'breakdown' => [
        'skills' => array_merge(arrays: $skillsMatch, ['has_requirements' => $hasSkillRequirements]),
        'education' => array_merge(arrays: $educationMatch, ['has_requirements' => $hasEducationRequirements]),
        'experience' => array_merge(arrays: $experienceMatch, ['has_requirements' => $hasExperienceRequirements]),
        'certifications' => array_merge(arrays: $certificationsMatch, ['has_requirements' => $hasCertificationRequirements])
    ],
    'recommendations' => $recommendations,
    'confidence_score' => $this->calculateConfidenceScore(
        hasSkills: $hasSkillRequirements,
        hasEducation: $hasEducationRequirements,
        hasExperience: $hasExperienceRequirements,
        hasCertifications: $hasCertificationRequirements,
        resume: $resume
    )
];
```

Figure 5.4.3.22 Final Output Structure

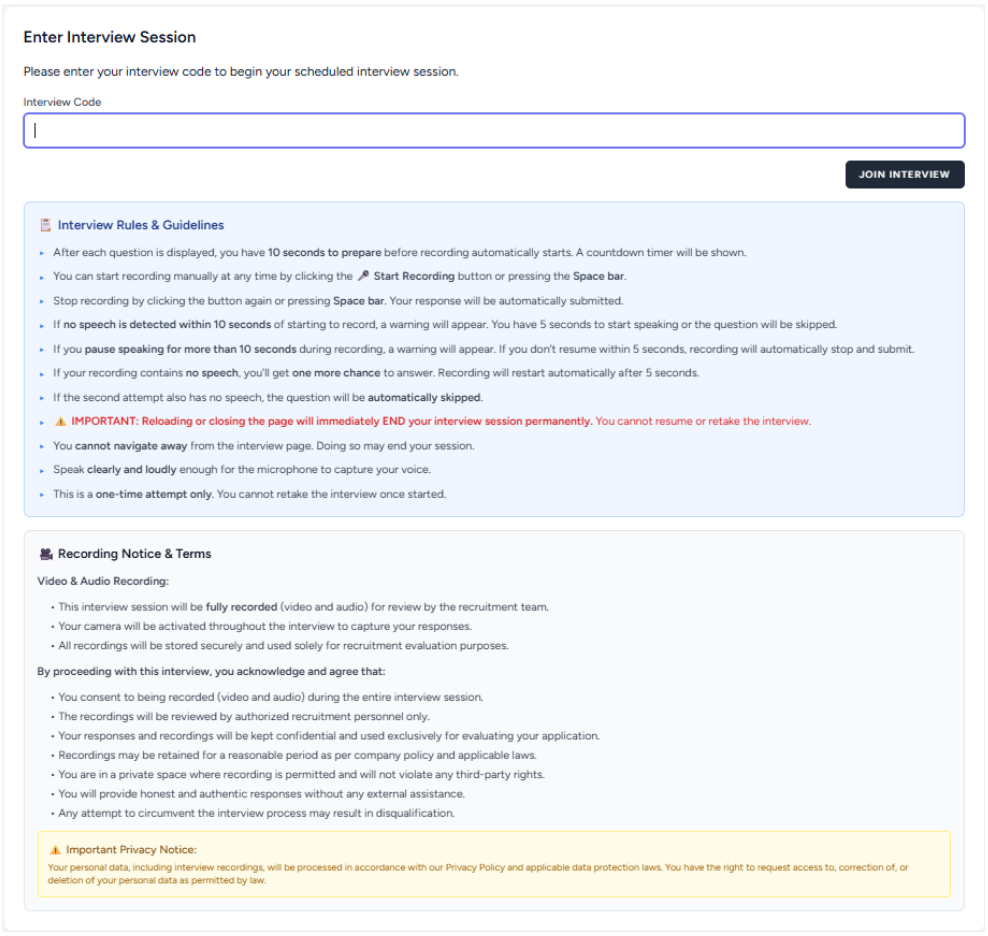


```php
private function calculateConfidenceScore($hasSkills, $hasEducation, $hasExperience, $hasCertifications, $resume): array {
    $confidence = 0;
    $factors = 0;

    // Check requirements completeness
    if ($hasSkills) {
        $confidence += 25;
        $factors++;
    }
    if ($hasEducation) {
        $confidence += 20;
        $factors++;
    }
    if ($hasExperience) {
        $confidence += 25;
        $factors++;
    }
    if ($hasCertifications) {
        $confidence += 10;
        $factors++;
    }
    // Check resume data completeness
    if ($resume->skills->count() > 0)
        $confidence += 5;
    if ($resume->education->count() > 0)
        $confidence += 5;
    if ($resume->experiences->count() > 0)
        $confidence += 5;
    if ($resume->professional_summary)
        $confidence += 5;
    // Determine confidence level
    if ($confidence >= 80) {
        return [
            'score' => $confidence,
            'level' => 'high',
            'description' => 'Comprehensive data for accurate matching'
        ];
    } elseif ($confidence >= 50) {
        return [
            'score' => $confidence,
            'level' => 'medium',
            'description' => 'Sufficient data for reliable matching'
        ];
    } else {
        return [
            'score' => $confidence,
            'level' => 'low',
            'description' => 'Limited data may affect accuracy'
        ];
    }
}
```

Figure 5.4.3.23 Confidence Score Calculation

5.4.4 Interview System

The interview process begins with the Interview Code Entry page (Figure 5.4.4.1). At the top of this page, candidates are prompted to enter a valid interview code in order to proceed. Directly below the code entry field, the system displays the full set of Interview Rules and Guidelines, ensuring that all candidates are fully aware of the conditions before they can join the session. The rules explain the flow of the interview, including preparation time, recording controls, and handling of silent or incomplete responses. For example, candidates are given 10 seconds to prepare before automatic recording begins, and their responses can be started or stopped manually using the designated controls. Warnings appear if no speech is detected or if the candidate pauses for too long, and skipped questions are automatically recorded. Clear reminders highlight that the interview is a one-time attempt, and reloading or closing the page will result in permanent termination of the session. In addition, the page includes the Recording Notice & Terms, outlining that both video and audio will be fully recorded, stored securely, and reviewed only by authorized recruitment personnel. A final Privacy Notice reinforces compliance with data protection laws, explaining candidates' rights to access, correction, or deletion of their personal data. By displaying these rules upfront, the system ensures candidates give informed consent before starting their interview..


Enter Interview Session

Please enter your interview code to begin your scheduled interview session.

Interview Code

JOIN INTERVIEW

Interview Rules & Guidelines

- After each question is displayed, you have **10 seconds to prepare** before recording automatically starts. A countdown timer will be shown.
- You can start recording manually at any time by clicking the **Start Recording** button or pressing the **Space bar**.
- Stop recording by clicking the button again or pressing **Space bar**. Your response will be automatically submitted.
- If **no speech is detected within 10 seconds** of starting to record, a warning will appear. You have 5 seconds to start speaking or the question will be skipped.
- If you **pause speaking for more than 10 seconds** during recording, a warning will appear. If you don't resume within 5 seconds, recording will automatically stop and submit.
- If your recording contains **no speech**, you'll get **one more chance** to answer. Recording will restart automatically after 5 seconds.
- If the second attempt also has no speech, the question will be **automatically skipped**.
- ⚠ **IMPORTANT: Reloading or closing the page will immediately END your interview session permanently.** You cannot resume or retake the interview.
- You **cannot navigate away** from the interview page. Doing so may end your session.
- Speak **clearly and loudly** enough for the microphone to capture your voice.
- This is a **one-time attempt only**. You cannot retake the interview once started.

Recording Notice & Terms

Video & Audio Recording:

- This interview session will be **fully recorded** (video and audio) for review by the recruitment team.
- Your camera will be activated throughout the interview to capture your responses.
- All recordings will be stored securely and used solely for recruitment evaluation purposes.

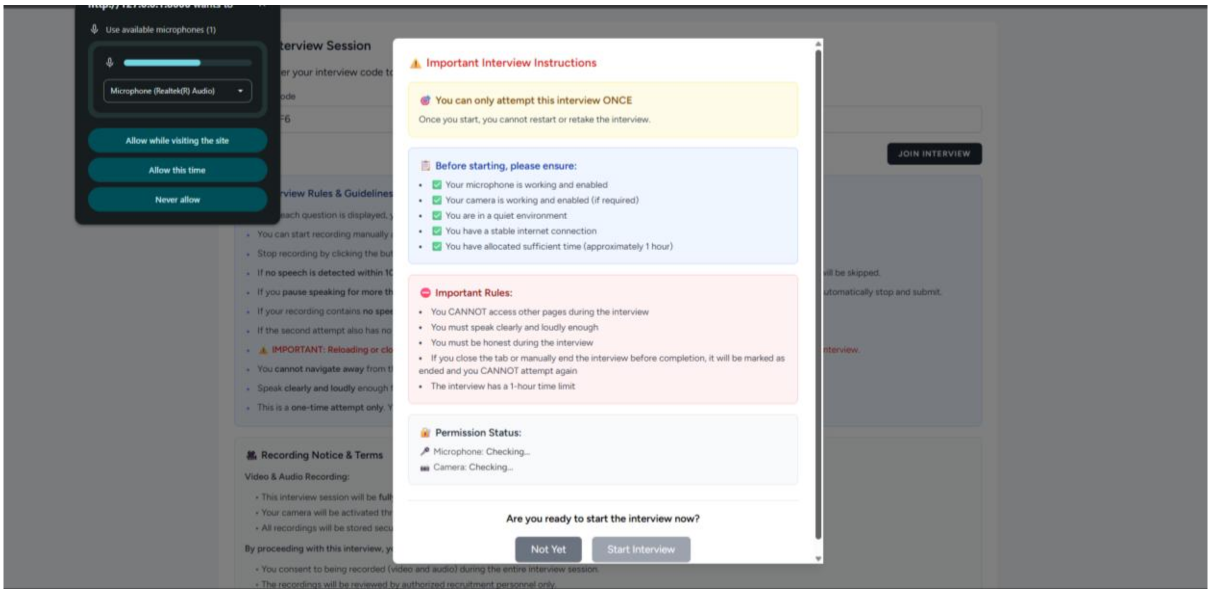
By proceeding with this interview, you acknowledge and agree that:

- You consent to being recorded (video and audio) during the entire interview session.
- The recordings will be reviewed by authorized recruitment personnel only.
- Your responses and recordings will be kept confidential and used exclusively for evaluating your application.
- Recordings may be retained for a reasonable period as per company policy and applicable laws.
- You are in a private space where recording is permitted and will not violate any third-party rights.
- You will provide honest and authentic responses without any external assistance.
- Any attempt to circumvent the interview process may result in disqualification.

⚠ Important Privacy Notice:
Your personal data, including interview recordings, will be processed in accordance with our Privacy Policy and applicable data protection laws. You have the right to request access to, correction of, or deletion of your personal data as permitted by law.

Figure 5.4.4.1 IntelliHire interview code entrance page

Once the interview code has been validated, candidates are directed to the Important Interview Instructions page (Figure 5.4.4.2). This page acts as a final checkpoint before entering the live interview session. Candidates are reminded that the interview can only be attempted once and cannot be restarted or retaken under any circumstances. A checklist prompts candidates to confirm that their microphone and camera are enabled, that they are in a quiet environment with a stable internet connection, and that they have allocated sufficient uninterrupted time (approximately one hour) for the interview. The section also restates important rules such as prohibiting navigation to other pages, requiring clear and honest responses, and enforcing a one-hour time limit. At the bottom of the page, the system displays the Permission Status for the microphone and camera. The "Start Interview" button remains disabled until both devices are successfully enabled, ensuring technical readiness before the session begins. This layered validation process minimizes disruptions and safeguards the integrity of the interview process.


Use available microphones (1)
Microphone (Realtek(R) Audio)
Allow while visiting the site
Allow this time
Never allow
JOIN INTERVIEW

Important Interview Instructions

You can only attempt this interview ONCE

Once you start, you cannot restart or retake the interview.

Before starting, please ensure:

- ✓ Your microphone is working and enabled
- ✓ Your camera is working and enabled (if required)
- ✓ You are in a quiet environment
- ✓ You have a stable internet connection
- ✓ You have allocated sufficient time (approximately 1 hour)

Important Rules:

- You CANNOT access other pages during the interview
- You must speak clearly and loudly enough
- You must be honest during the interview
- If you close the tab or manually end the interview before completion, it will be marked as ended and you CANNOT attempt again
- The interview has a 1-hour time limit

Permission Status:

Microphone: Checking...
Camera: Checking...

Are you ready to start the interview now?

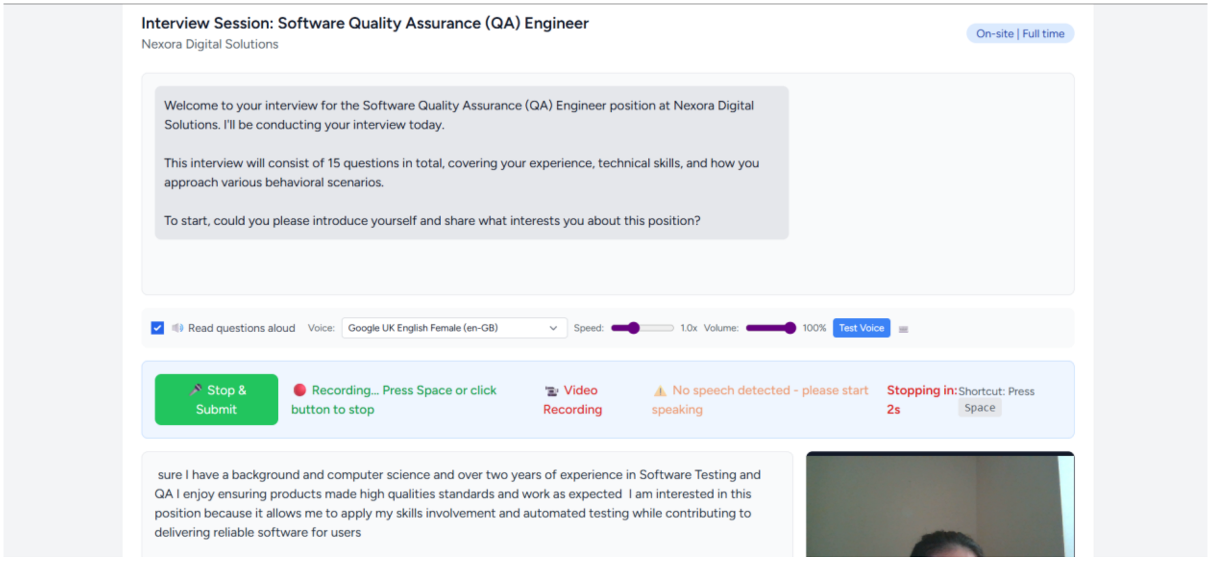
Not Yet | Start Interview

Figure 5.4.4.2 Intellihire interview page entry permission check

After passing the checks, candidates enter the Interview Interface (Figure 5.4.4.3), where they interact with the AI in a structured conversational flow. The chat area displays system-generated questions along with the candidate's responses, creating a real-time interview transcript. Responses are primarily collected through voice, supported by speech-to-text processing that transcribes the spoken content into text. The interface prevents manual typing to maintain consistency across all interviews, requiring candidates to follow the automated flow without skipping or reordering questions.

The system also integrates speech and video features using WebRTC and the MediaRecorder API. Candidates can see their live camera feed and audio level indicators while recording their responses. Clear visual cues, such as red recording dots and timers, help them stay aware of their status. If no speech is detected within the allocated preparation or response time, warnings are triggered, and unanswered questions may be automatically skipped. These mechanisms ensure efficiency and fairness by standardizing the time available to all participants.

In addition to voice input, the platform incorporates Text-to-Speech (TTS) to enhance accessibility. Candidates can adjust voice settings, including speech speed, volume, and voice type, enabling them to hear the AI's questions audibly. Animated indicators provide feedback when the system is speaking, and candidates may pause or test the feature before the session begins. This functionality improves engagement, particularly for users who prefer listening to questions rather than reading them.


Interview Session: Software Quality Assurance (QA) Engineer
Nexora Digital Solutions

On-site | Full time

Welcome to your interview for the Software Quality Assurance (QA) Engineer position at Nexora Digital Solutions. I'll be conducting your interview today.

This interview will consist of 15 questions in total, covering your experience, technical skills, and how you approach various behavioral scenarios.

To start, could you please introduce yourself and share what interests you about this position?

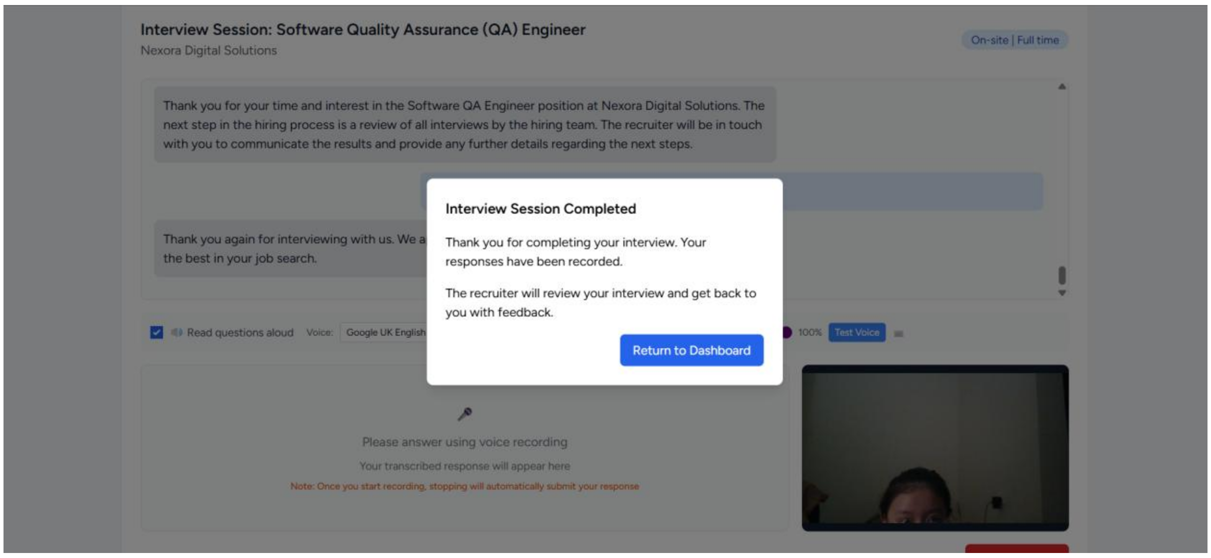
Read questions aloud Voice: Google UK English Female (en-GB) Speed: 1.0x Volume: 100% Test Voice

Stop & Submit | Recording... Press Space or click button to stop | Video Recording | No speech detected - please start speaking | Stopping in: 2s | Shortcut: Press Space

sure I have a background and computer science and over two years of experience in Software Testing and QA I enjoy ensuring products made high qualities standards and work as expected I am interested in this position because it allows me to apply my skills involvement and automated testing while contributing to delivering reliable software for users

The Interview interface provide candidates with essential management tools during the session. Visible progress indicators and countdown timers remind candidates of the remaining time, while navigation warnings prevent them from accidentally leaving the interview page. Candidates may choose to end the interview early, but confirmation pop-ups are enforced to avoid unintended termination. Importantly, once the interview is ended, it cannot be restarted or repeated.

At the end of the session, the system determines the next step. If no assessment is assigned, candidates are returned to their dashboard with a completion message. However, if an assessment has been scheduled, the system automatically redirects the candidate to the Assessment Module, where further evaluation is carried out (explained in Section 5.4.5). This seamless transition ensures continuity in the hiring workflow while keeping candidates guided throughout the process.


Interview Session: Software Quality Assurance (QA) Engineer
Nexora Digital Solutions
On-site | Full time

Thank you for your time and interest in the Software QA Engineer position at Nexora Digital Solutions. The next step in the hiring process is a review of all interviews by the hiring team. The recruiter will be in touch with you to communicate the results and provide any further details regarding the next steps.

Thank you again for interviewing with us. We a... the best in your job search.

Interview Session Completed

Thank you for completing your interview. Your responses have been recorded.

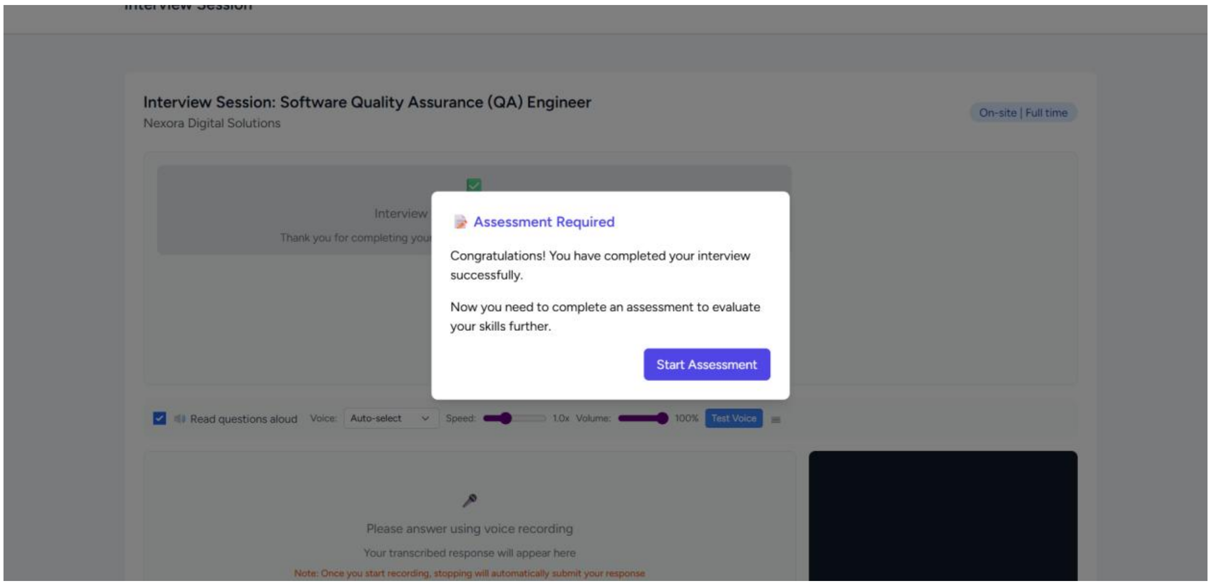
The recruiter will review your interview and get back to you with feedback.

Return to Dashboard

Read questions aloud Voice: Google UK English ... 100% Test Voice

Please answer using voice recording
Your transcribed response will appear here
Note: Once you start recording, stopping will automatically submit your response

Figure 5.4.4.3 Interview session completed popup (no assessment)


Interview Session: Software Quality Assurance (QA) Engineer
Nexora Digital Solutions
On-site | Full time

Interview ...
Thank you for completing you...

Assessment Required

Congratulations! You have completed your interview successfully.

Now you need to complete an assessment to evaluate your skills further.

Start Assessment

Read questions aloud Voice: Auto-select Speed: 1.0x Volume: 100% Test Voice

Please answer using voice recording
Your transcribed response will appear here
Note: Once you start recording, stopping will automatically submit your response

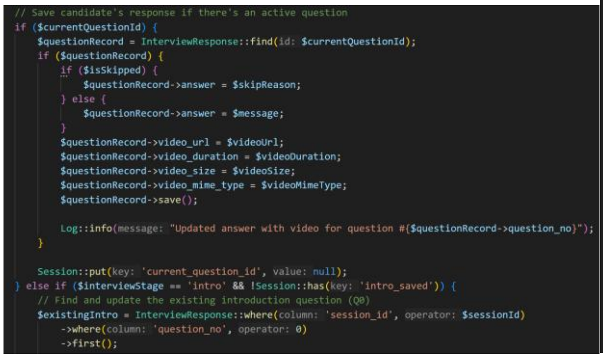
Figure 5.4.4.4 Interview session completed popup (with assessment)

5.4.4.1 Backend Processing

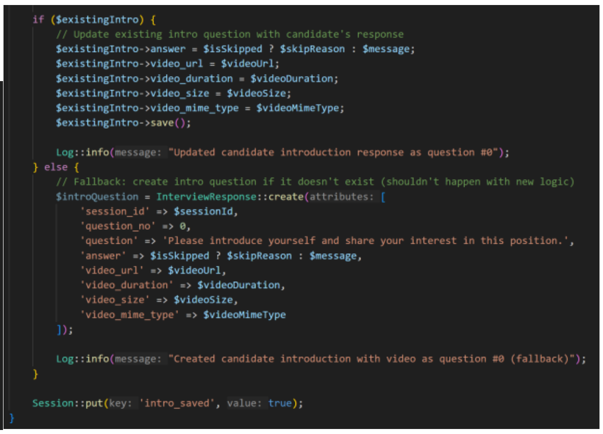
The InterviewController handles the orchestration of the entire interview process on the backend. It acts as the bridge between the candidate interface, the database, and the integrated AI services. The logic within the controller ensures smooth handling of interview sessions, validation of codes, collection of candidate responses, and communication with the Gemini API for evaluation.

1. Response Recording and Storage

During the interview, candidate responses are captured via WebRTC and submitted to the controller in either audio or text format. The backend securely stores these responses in the database, associating them with the candidate's interview session. Metadata such as timestamps, question identifiers, and status are also logged. This ensures that recruiters later receive a structured and auditable record of the session.



```php
// Save candidate's response if there's an active question
if ($currentQuestionId) {
    $questionRecord = InterviewResponse::find(id: $currentQuestionId);
    if ($questionRecord) {
        if ($isSkipped) {
            $questionRecord->answer = $skipReason;
        } else {
            $questionRecord->answer = $message;
        }
        $questionRecord->video_url = $videoUrl;
        $questionRecord->video_duration = $videoDuration;
        $questionRecord->video_size = $videoSize;
        $questionRecord->video_mime_type = $videoMimeType;
        $questionRecord->save();

        Log::info(message: "Updated answer with video for question #{$questionRecord->question_no}");
    }

    Session::put(key: 'current_question_id', value: null);
} else if ($interviewStage == 'intro' && !Session::has(key: 'intro_saved')) {
    // Find and update the existing introduction question (Q0)
    $existingIntro = InterviewResponse::where(column: 'session_id', operator: $sessionId)
        ->where(column: 'question_no', operator: 0)
        ->first();
```



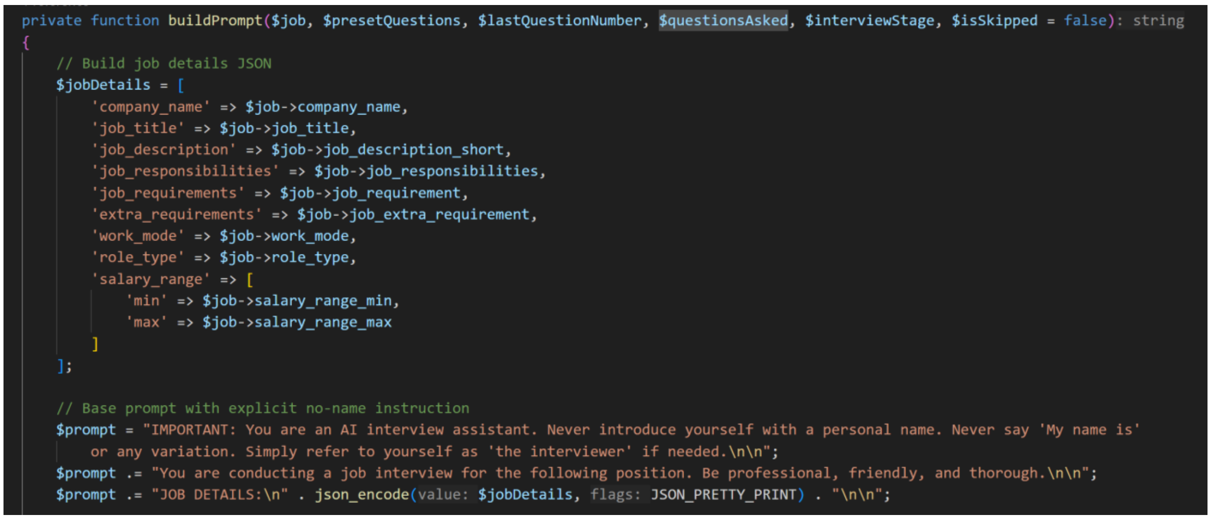
```php
    if ($existingIntro) {
        // Update existing intro question with candidate's response
        $existingIntro->answer = $isSkipped ? $skipReason : $message;
        $existingIntro->video_url = $videoUrl;
        $existingIntro->video_duration = $videoDuration;
        $existingIntro->video_size = $videoSize;
        $existingIntro->video_mime_type = $videoMimeType;
        $existingIntro->save();

        Log::info(message: "Updated candidate introduction response as question #0");
    } else {
        // Fallback: create intro question if it doesn't exist (shouldn't happen with new logic)
        $introQuestion = InterviewResponse::create(attributes: [
            'session_id' => $sessionId,
            'question_no' => 0,
            'question' => 'Please introduce yourself and share your interest in this position.',
            'answer' => $isSkipped ? $skipReason : $message,
            'video_url' => $videoUrl,
            'video_duration' => $videoDuration,
            'video_size' => $videoSize,
            'video_mime_type' => $videoMimeType
        ]);

        Log::info(message: "Created candidate introduction with video as question #0 (fallback)");
    }

    Session::put(key: 'intro_saved', value: true);
}
```

Figure 5.4.4.5 Controller logic for storing candidate responses

2. AI Prompt Construction and Evaluation

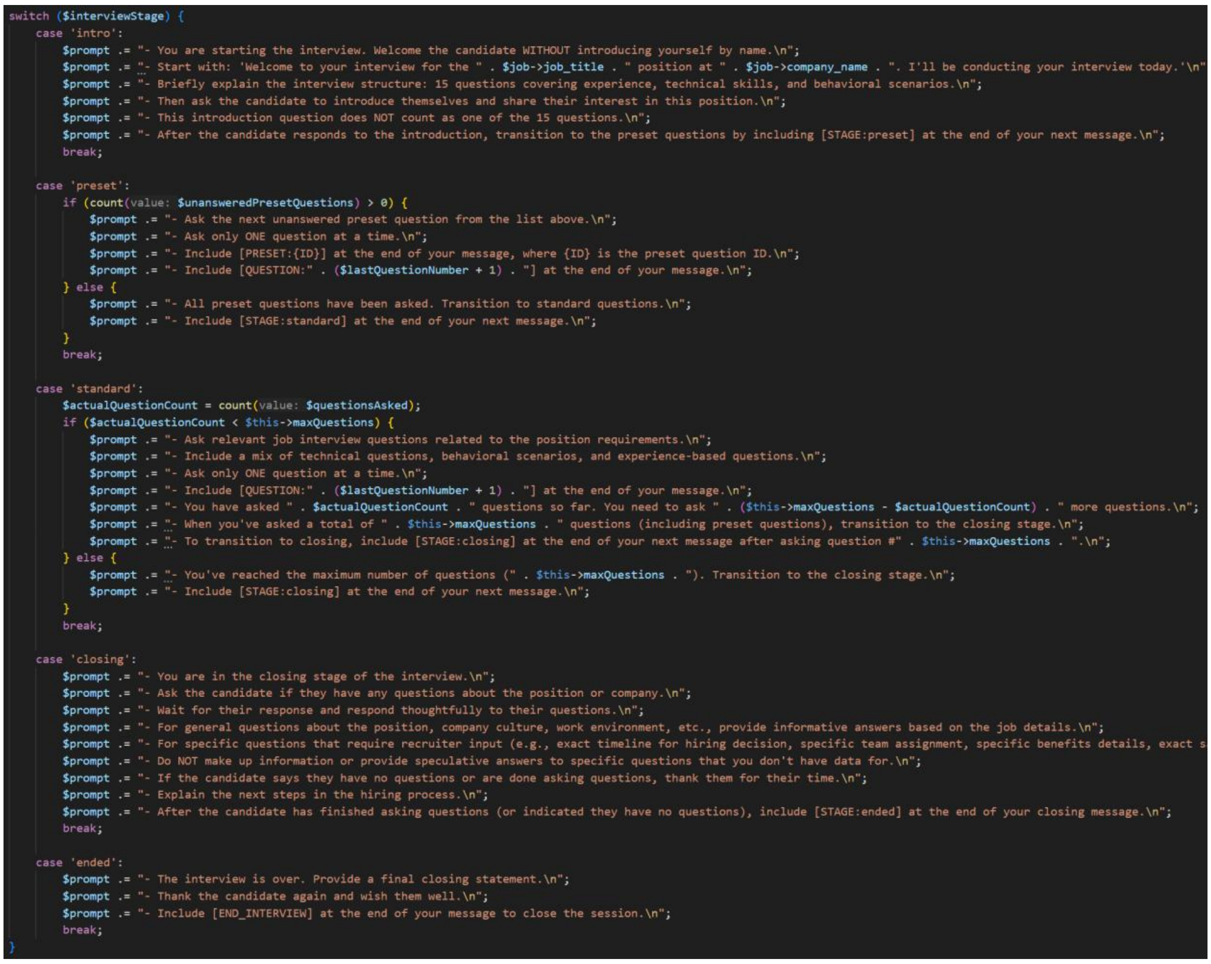
The controller integrates directly with the Gemini API, constructing prompts dynamically for each candidate response. For instance, when evaluating an answer, the backend assembles contextual details such as the interview question, expected competencies, and candidate transcript. This prompt is then sent to Gemini, which returns a structured analysis containing a similarity score, relevance assessment, and explanatory feedback. By embedding predefined evaluation rules into the prompt, the system ensures consistent scoring across different interviews.

In addition to evaluating candidate responses, the controller also leverages Gemini for question generation, using a dedicated function, buildPrompt, to dynamically assemble the instructions sent to the API. This function structures the job information into a JSON block containing attributes such as company name, job title, responsibilities, requirements, and salary range, ensuring that the AI has sufficient context about the role. The prompt also incorporates the interview stage (intro, preset, standard, closing, or ended), the number of questions already asked, and the next sequential question number. Based on the current stage, Gemini is guided to either welcome the candidate, ask preset questions, generate standard interview questions, or close the session. Strict rules are embedded to guarantee consistency, including requiring exactly 15 questions, marking each question with a [QUESTION:n] tag, and preventing the AI from introducing itself by name. When a question is skipped or unanswered, additional instructions are injected so that Gemini briefly acknowledges the skip before moving forward. This structured prompting method ensures that Gemini produces contextually relevant, professionally formatted, and stage-appropriate interview dialogue throughout the entire session.



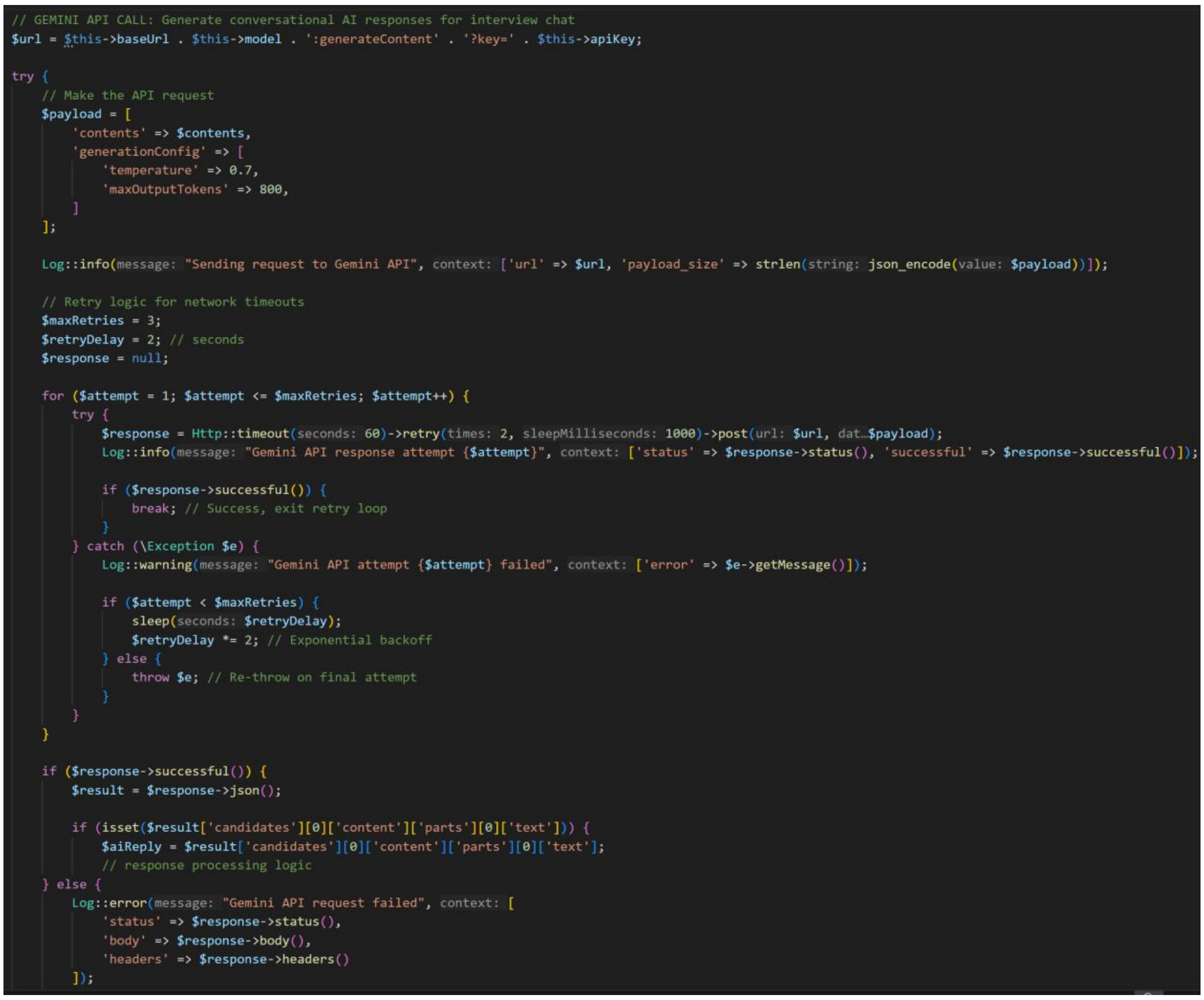
```php
private function buildPrompt($job, $presetQuestions, $lastQuestionNumber, $questionsAsked, $interviewStage, $isSkipped = false): string
{
    // Build job details JSON
    $jobDetails = [
        'company_name' => $job->company_name,
        'job_title' => $job->job_title,
        'job_description' => $job->job_description_short,
        'job_responsibilities' => $job->job_responsibilities,
        'job_requirements' => $job->job_requirement,
        'extra_requirements' => $job->job_extra_requirement,
        'work_mode' => $job->work_mode,
        'role_type' => $job->role_type,
        'salary_range' => [
            'min' => $job->salary_range_min,
            'max' => $job->salary_range_max
        ]
    ];

    // Base prompt with explicit no-name instruction
    $prompt = "IMPORTANT: You are an AI interview assistant. Never introduce yourself with a personal name. Never say 'My name is'
        or any variation. Simply refer to yourself as 'the interviewer' if needed.\n\n";
    $prompt .= "You are conducting a job interview for the following position. Be professional, friendly, and thorough.\n\n";
    $prompt .= "JOB DETAILS:\n" . json_encode(value: $jobDetails, flags: JSON_PRETTY_PRINT) . "\n\n";
```

Figure 5.4.4.6 Constructing AI evaluation prompt for Gemini API



```
switch ($interviewStage) {
    case 'intro':
        $prompt .= "- You are starting the interview. Welcome the candidate WITHOUT introducing yourself by name.\n";
        $prompt .= "- Start with: 'Welcome to your interview for the " . $job->job_title . " position at " . $job->company_name . ". I'll be conducting your interview today.'\n"
        $prompt .= "- Briefly explain the interview structure: 15 questions covering experience, technical skills, and behavioral scenarios.\n";
        $prompt .= "- Then ask the candidate to introduce themselves and share their interest in this position.\n";
        $prompt .= "- This introduction question does NOT count as one of the 15 questions.\n";
        $prompt .= "- After the candidate responds to the introduction, transition to the preset questions by including [STAGE:preset] at the end of your next message.\n";
        break;

    case 'preset':
        if (count(value: $unansweredPresetQuestions) > 0) {
            $prompt .= "- Ask the next unanswered preset question from the list above.\n";
            $prompt .= "- Ask only ONE question at a time.\n";
            $prompt .= "- Include [PRESET:{ID}] at the end of your message, where {ID} is the preset question ID.\n";
            $prompt .= "- Include [QUESTION:" . ($lastQuestionNumber + 1) . "] at the end of your message.\n";
        } else {
            $prompt .= "- All preset questions have been asked. Transition to standard questions.\n";
            $prompt .= "- Include [STAGE:standard] at the end of your next message.\n";
        }
        break;

    case 'standard':
        $actualQuestionCount = count(value: $questionsAsked);
        if ($actualQuestionCount < $this->maxQuestions) {
            $prompt .= "- Ask relevant job interview questions related to the position requirements.\n";
            $prompt .= "- Include a mix of technical questions, behavioral scenarios, and experience-based questions.\n";
            $prompt .= "- Ask only ONE question at a time.\n";
            $prompt .= "- Include [QUESTION:" . ($lastQuestionNumber + 1) . "] at the end of your message.\n";
            $prompt .= "- You have asked " . $actualQuestionCount . " questions so far. You need to ask " . ($this->maxQuestions - $actualQuestionCount) . " more questions.\n";
            $prompt .= "- When you've asked a total of " . $this->maxQuestions . " questions (including preset questions), transition to the closing stage.\n";
            $prompt .= "- To transition to closing, include [STAGE:closing] at the end of your next message after asking question #" . $this->maxQuestions . ".\n";
        } else {
            $prompt .= "- You've reached the maximum number of questions (" . $this->maxQuestions . "). Transition to the closing stage.\n";
            $prompt .= "- Include [STAGE:closing] at the end of your next message.\n";
        }
        break;

    case 'closing':
        $prompt .= "- You are in the closing stage of the interview.\n";
        $prompt .= "- Ask the candidate if they have any questions about the position or company.\n";
        $prompt .= "- Wait for their response and respond thoughtfully to their questions.\n";
        $prompt .= "- For general questions about the position, company culture, work environment, etc., provide informative answers based on the job details.\n";
        $prompt .= "- For specific questions that require recruiter input (e.g., exact timeline for hiring decision, specific team assignment, specific benefits details, exact s
        $prompt .= "- Do NOT make up information or provide speculative answers to specific questions that you don't have data for.\n";
        $prompt .= "- If the candidate says they have no questions or are done asking questions, thank them for their time.\n";
        $prompt .= "- Explain the next steps in the hiring process.\n";
        $prompt .= "- After the candidate has finished asking questions (or indicated they have no questions), include [STAGE:ended] at the end of your closing message.\n";
        break;

    case 'ended':
        $prompt .= "- The interview is over. Provide a final closing statement.\n";
        $prompt .= "- Thank the candidate again and wish them well.\n";
        $prompt .= "- Include [END_INTERVIEW] at the end of your message to close the session.\n";
        break;
}
```

Figure 5.4.4.7 Prompt construction logic for different interview stages

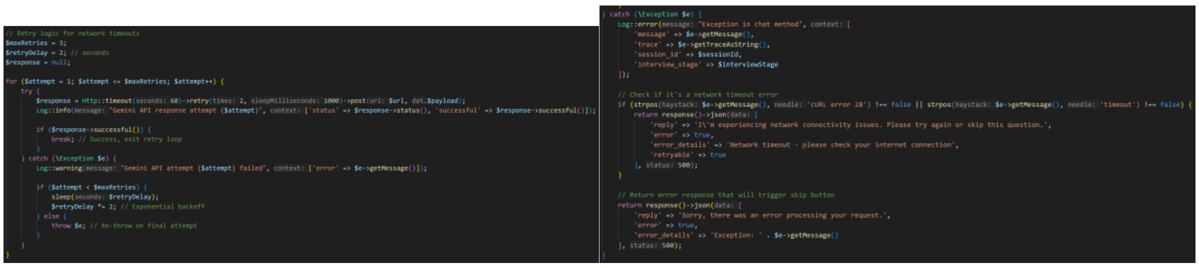


```php
// GEMINI API CALL: Generate conversational AI responses for interview chat
$url = $this->baseUrl . $this->model . ':generateContent' . '?key=' . $this->apiKey;

try {
    // Make the API request
    $payload = [
        'contents' => $contents,
        'generationConfig' => [
            'temperature' => 0.7,
            'maxOutputTokens' => 800,
        ]
    ];

    Log::info(message: "Sending request to Gemini API", context: ['url' => $url, 'payload_size' => strlen(string: json_encode(value: $payload))]);

    // Retry logic for network timeouts
    $maxRetries = 3;
    $retryDelay = 2; // seconds
    $response = null;

    for ($attempt = 1; $attempt <= $maxRetries; $attempt++) {
        try {
            $response = Http::timeout(seconds: 60)->retry(times: 2, sleepMilliseconds: 1000)->post(url: $url, dat…$payload);
            Log::info(message: "Gemini API response attempt {$attempt}", context: ['status' => $response->status(), 'successful' => $response->successful()]);

            if ($response->successful()) {
                break; // Success, exit retry loop
            }
        } catch (\Exception $e) {
            Log::warning(message: "Gemini API attempt {$attempt} failed", context: ['error' => $e->getMessage()]);

            if ($attempt < $maxRetries) {
                sleep(seconds: $retryDelay);
                $retryDelay *= 2; // Exponential backoff
            } else {
                throw $e; // Re-throw on final attempt
            }
        }
    }

    if ($response->successful()) {
        $result = $response->json();

        if (isset($result['candidates'][0]['content']['parts'][0]['text'])) {
            $aiReply = $result['candidates'][0]['content']['parts'][0]['text'];
            // response processing logic
    } else {
        Log::error(message: "Gemini API request failed", context: [
            'status' => $response->status(),
            'body' => $response->body(),
            'headers' => $response->headers()
        ]);
```

Figure 5.4.4.8 Handling Gemini API response and saving AI evaluation results

3. Retry and Error Handling

To account for potential API failures or network instability, the backend incorporates retry logic. If the first request to the Gemini API fails, the system automatically retries with exponential backoff to maintain reliability. In cases of persistent failure, fallback mechanisms store the candidate's raw responses for later evaluation, preventing data loss.



```php
// Retry logic for network timeouts
$maxRetries = 3;
$retryDelay = 2; // seconds
$response = null;

for ($attempt = 1; $attempt <= $maxRetries; $attempt++) {
    try {
        $response = Http::timeout(seconds: 60)->retry(times: 2, sleepMilliseconds: 1000)->post(url: $url, dat…$payload);
        Log::info(message: "Gemini API response attempt {$attempt}", context: ['status' => $response->status(), 'successful' => $response->successful()]);

        if ($response->successful()) {
            break; // Success, exit retry loop
        }
    } catch (\Exception $e) {
        Log::warning(message: "Gemini API attempt {$attempt} failed", context: ['error' => $e->getMessage()]);

        if ($attempt < $maxRetries) {
            sleep(seconds: $retryDelay);
            $retryDelay *= 2; // Exponential backoff
        } else {
            throw $e; // Re-throw on final attempt
        }
    }
}
```

```php
} catch (\Exception $e) {
    Log::error(message: "Exception in chat method", context: [
        'message' => $e->getMessage(),
        'trace' => $e->getTraceAsString(),
        'session_id' => $sessionId,
        'interview_stage' => $interviewStage
    ]);

    // Check if it's a network timeout error
    if (strpos(haystack: $e->getMessage(), needle: 'cURL error 28') !== false || strpos(haystack: $e->getMessage(), needle: 'timeout') !== false) {
        return response()->json(data: [
            'reply' => 'I\'m experiencing network connectivity issues. Please try again or skip this question.',
            'error' => true,
            'error_details' => 'Network timeout - please check your internet connection',
            'retryable' => true
        ], status: 500);
    }

    // Return error response that will trigger skip button
    return response()->json(data: [
        'reply' => 'Sorry, there was an error processing your request.',
        'error' => true,
        'error_details' => 'Exception: ' . $e->getMessage()
    ], status: 500);
}
```

Figure 5.4.4.9 Retry logic for handling API errors in interview evaluation

5.4.4.2 Text-to-Speech (TTS) Integration

The IntelliHire system incorporates text-to-speech (TTS) to provide an audible delivery of interview questions. Unlike simpler designs where the frontend decides when to play audio, in IntelliHire the backend InterviewController first evaluates the context and sets a decision flag. This flag, called $shouldSpeak, is computed after analyzing the current interview stage (intro, preset, standard, or closing), the presence of question markers such as [QUESTION:X], stage transition tags like [STAGE:closing], and content patterns such as question marks or instructional phrases. By handling this logic on the backend, the system ensures that all candidates receive a consistent, rule-based experience.

Once the backend determines that a message should be read aloud, it removes the internal markers (e.g., [QUESTION:X]) before sending the sanitized text and the $shouldSpeak flag to the frontend. On the client side, the browser executes the actual speech rendering using the Speech Synthesis API, which supports different voices, adjustable speed, and volume controls. This hybrid approach ensures the server has authoritative control over when speech should happen, while still giving candidates personalization options for how the audio is delivered.

This separation of responsibilities—decisioning on the backend and rendering on the frontend—provides multiple benefits. It keeps the server lightweight, since the heavy lifting of audio generation is offloaded to the browser. It also improves flexibility, allowing candidates to adjust voice preferences without affecting backend logic. At the same time, it guarantees consistency across interviews, since the backend enforces strict rules on which messages are spoken. Together, these design choices ensure IntelliHire's interview process is both efficient and adaptable.

5.4.4.3 Speech-to-Text (STT) Integration

The speech-to-text (STT) component of IntelliHire is fully browser-based, implemented with the Web Speech API (window.SpeechRecognition / window.webkitSpeechRecognition). When candidates speak their answers, their audio is transcribed locally in real time and converted into plain text. This transcript is then sent directly to the backend, bypassing the need to upload audio files for processing. The backend stores the text along with question metadata and optional video recordings, ensuring that the system maintains a complete session log without heavy server-side processing.

To ensure fairness and robustness, IntelliHire implements a silence-detection system and a retry mechanism. After each question is displayed and the text-to-speech finishes, a 10-second preparation timer runs, after which recording automatically starts. During recording, a separate 10-second silence detection timer monitors for speech activity. If no speech is detected within 10 seconds, the interface shows a warning and provides a 5-second countdown for the candidate to begin speaking. If the candidate still does not respond, the system records the attempt as empty and automatically skips the question. This design prevents technical issues or hesitation from stalling the session.

The retry logic adds another layer of resilience. If the first attempt contains no speech, the system grants the candidate one additional attempt under the same rules. Only if both attempts fail will the question be permanently skipped, and the interview moves forward. This approach balances leniency (giving users a second chance) with efficiency (keeping the interview moving without long delays).

Finally, the frontend error handling is designed to create a smooth user experience. It monitors microphone status, provides live transcription feedback, and issues warnings when silence is detected. By handling these checks in the browser, IntelliHire avoids unnecessary server calls while ensuring that candidates are immediately aware of issues. This lightweight but structured approach ensures that the STT pipeline is accurate, real-time, and fair, even when network or user behavior introduces uncertainties.

The overall interaction between the candidate interface, backend controller, external AI services, and storage components is summarized in the flow diagram below

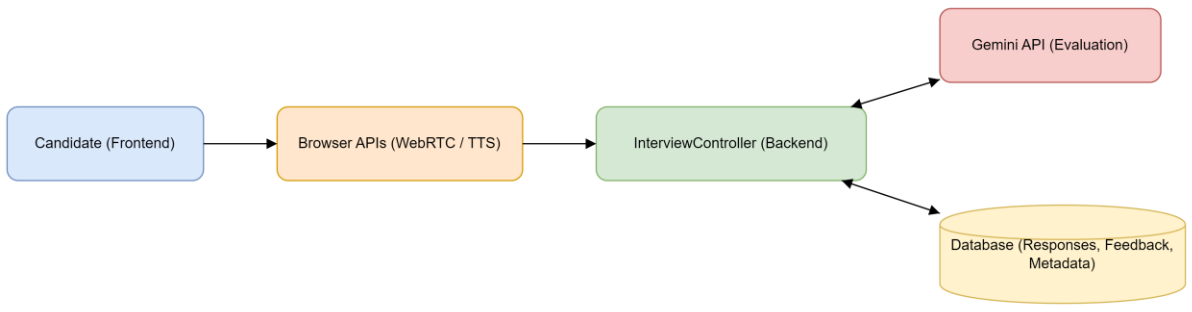
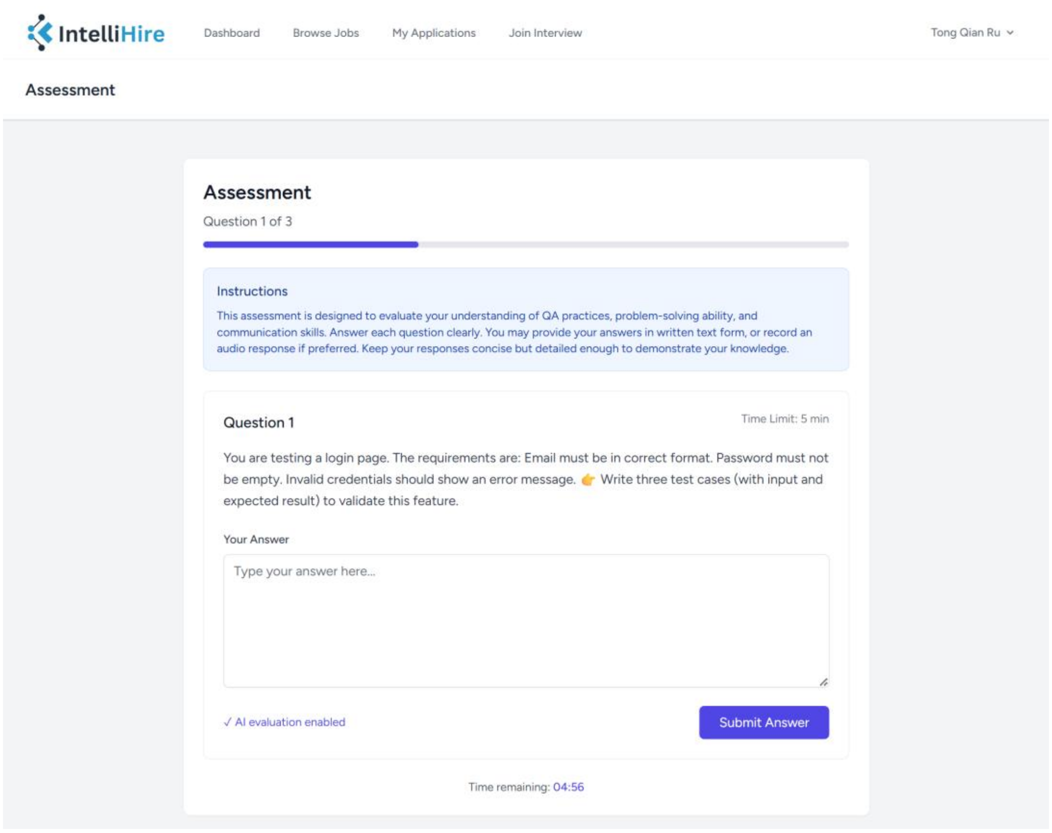

Gemini API (Evaluation)
Candidate (Frontend)
Browser APIs (WebRTC / TTS)
InterviewController (Backend)
Database (Responses, Feedback, Metadata)

Figure 5.4.4.10 Interview Backend Flow Diagram

5.4.5 Assessment System

The assessment system in IntelliHire provides a structured evaluation stage following the interview process. Managed by the AssessmentController, this module ensures that candidates are tested on their technical knowledge, problem-solving ability, and communication skills in a standardized format. Assessments are delivered in a clear, time-bound manner, and all candidate responses are securely stored in the database for later review by recruiters, alongside optional AI-based evaluation.

When candidates begin the assessment, they are directed to the assessment interface, which consolidates the landing instructions, question display, and timer functionality into a single page. The top section presents clear instructions outlining the assessment purpose, input methods (text or audio), and guidelines for answering. Below this, the active question is displayed prominently, showing details such as the current question number (e.g., Question 1 of 3) and a countdown timer that enforces the time limit for each question. Candidates are provided with a text input area where they can type their responses, while an audio recording option is available for those who prefer spoken answers. Once an answer is submitted, either manually or automatically after the timer expires, the system transitions smoothly to the next question until the assessment is complete. This design keeps candidates focused while ensuring fairness through strict time control.



Assessment

Question 1 of 3

Instructions

This assessment is designed to evaluate your understanding of QA practices, problem-solving ability, and communication skills. Answer each question clearly. You may provide your answers in written text form, or record an audio response if preferred. Keep your responses concise but detailed enough to demonstrate your knowledge.

Question 1

Time Limit: 5 min

You are testing a login page. The requirements are: Email must be in correct format. Password must not be empty. Invalid credentials should show an error message. 👉 Write three test cases (with input and expected result) to validate this feature.

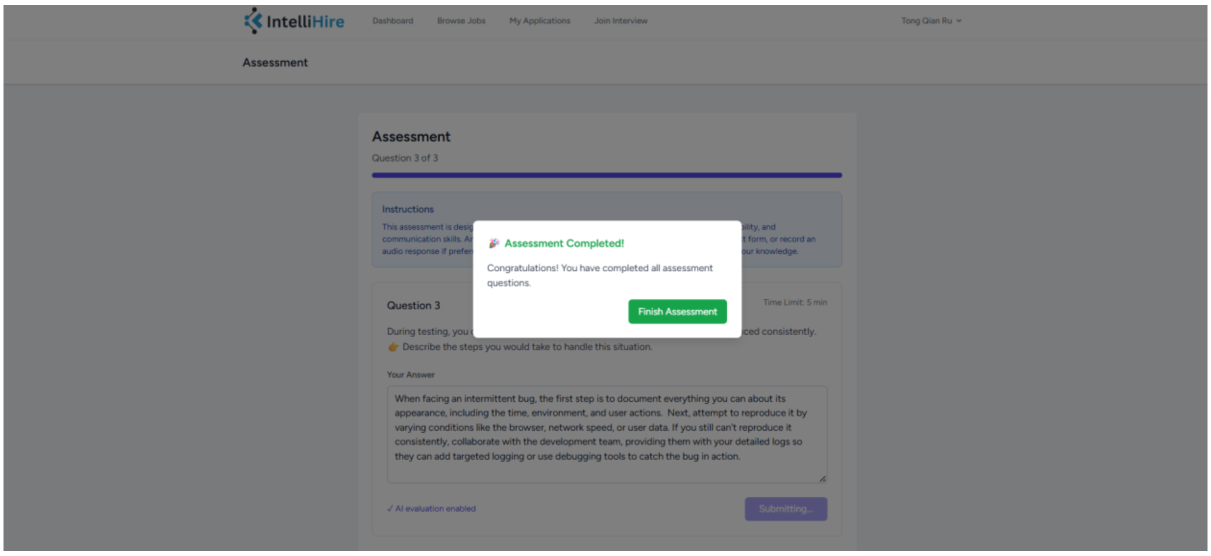
Your Answer

✓ AI evaluation enabled

Time remaining: 04:56

Figure 5.4.5.1 IntelliHire assessment interface with instructions, active question, and countdown timer

Upon completing all assigned questions, candidates are redirected to a completion page, which confirms successful submission of the assessment. A modal popup congratulates the candidate and provides a “Finish Assessment” button, which finalizes the process and returns them to their dashboard. At this stage, the backend securely stores all responses and marks the assessment as complete, making them accessible for recruiter review. This ensures a seamless transition from candidate input to recruiter evaluation, closing the assessment loop effectively.



Assessment

Question 3 of 3

Instructions

This assessment is desig ility, and communication skills. Ar t form, or record an audio response if prefer our knowledge.

🎉 Assessment Completed!

Congratulations! You have completed all assessment questions.

Question 3 Time Limit: 5 min

During testing, you c uced consistently.
👉 Describe the steps you would take to handle this situation.

Your Answer

When facing an intermittent bug, the first step is to document everything you can about its appearance, including the time, environment, and user actions. Next, attempt to reproduce it by varying conditions like the browser, network speed, or user data. If you still can't reproduce it consistently, collaborate with the development team, providing them with your detailed logs so they can add targeted logging or use debugging tools to catch the bug in action.

✓ AI evaluation enabled

Figure 5.4.5.2 IntelliHire assessment completion page

On the backend, the same speech-to-text (STT) pipeline used during the interview is also available in the assessment. Candidates need to record their responses through the browser's speech recognition API, which transcribes the audio into text if the question type is set as audio response. This keeps the assessment lightweight by eliminating the need for server-side audio processing, while ensuring that both text and audio answers are uniformly stored as structured text responses in the database.

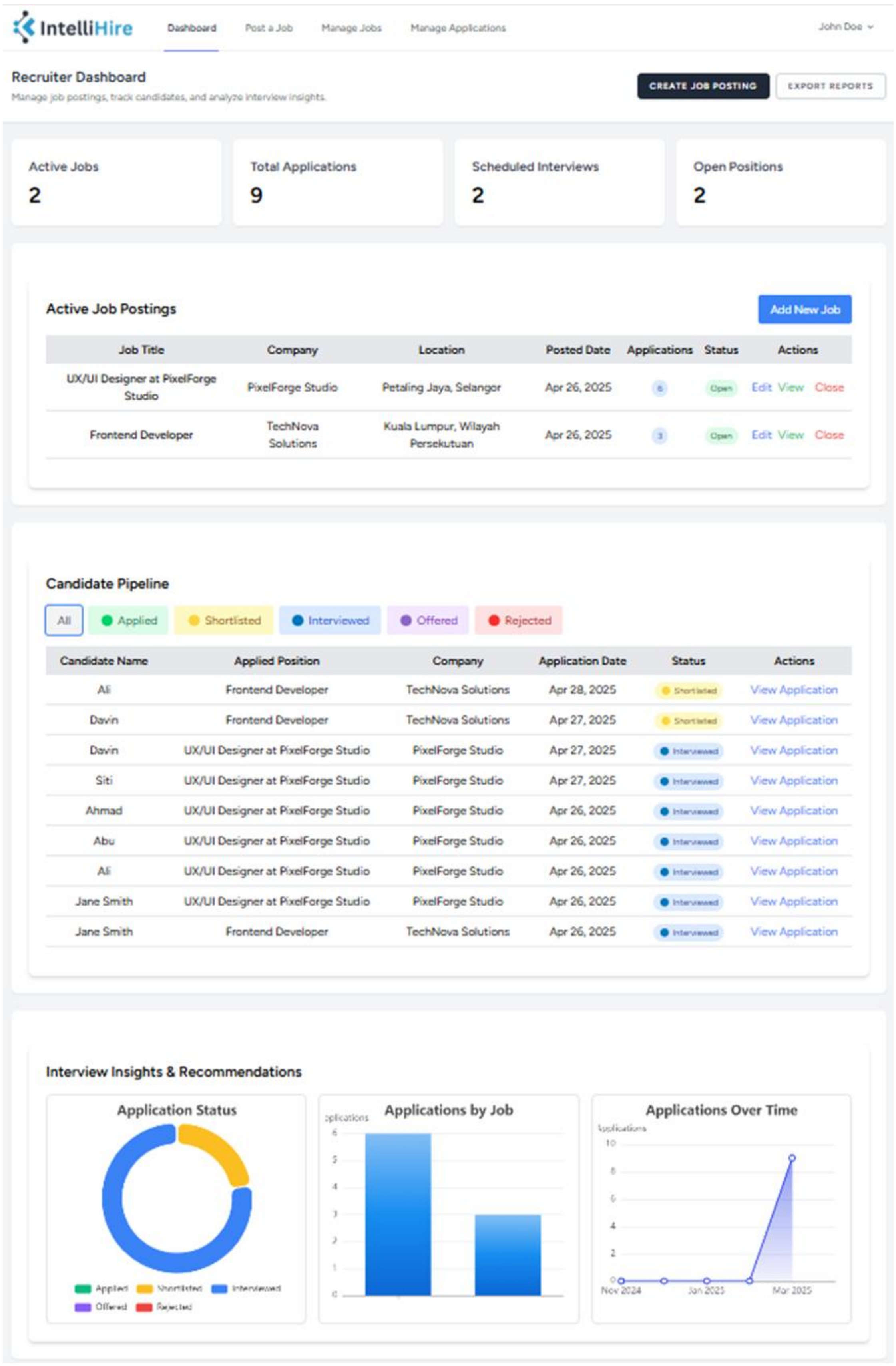
5.4.6 Recruiter Dashboard

The recruiter dashboard provides a consolidated workspace for managing job postings, tracking candidates, and monitoring hiring performance. At the top of the page, a set of summary counters presents key metrics at a glance—Active Jobs, Total Applications, Scheduled Interviews, and Open Positions—so recruiters can immediately gauge current workload and pipeline health. A prominent Create Job Posting action and an Export Reports button streamline common tasks without leaving the page.

Beneath the summaries, the Active Job Postings panel lists each role with its company, location, posted date, application count, and current status. Inline actions (View, Edit, Close) allow recruiters to open the job detail page, update posting information, or close the vacancy when hiring is complete. This table serves as the operational hub for day-to-day job administration.

The Candidate Pipeline section surfaces all applicants across jobs in a single, filterable view. Status chips (e.g., Applied, Shortlisted, Interviewed, Offered, Rejected) enable quick filtering to prioritize follow-ups. Each row shows the candidate name, applied position, company, application date, current status, and a View Application link that opens the full application profile (resume, cover letter, notes, interview history). This layout supports rapid triage while preserving a clear audit trail for every decision.

At the bottom, Interview Insights & Recommendations provides lightweight analytics to inform planning. A pie chart summarizes the distribution of applications by status, a bar chart shows applications per job to reveal which roles attract more interest, and a line chart tracks applications over time, helping recruiters identify spikes caused by new postings or campaigns. Together, these visualizations give recruiters an immediate sense of funnel balance and where intervention may be needed (e.g., refreshing a low-performing job description or scheduling additional interviews).



Recruiter Dashboard

Manage job postings, track candidates, and analyze interview insights.

Active Jobs	Total Applications	Scheduled Interviews	Open Positions
2	9	2	2

Active Job Postings

Job Title	Company	Location	Posted Date	Applications	Status	Actions
UX/UI Designer at PixelForge Studio	PixelForge Studio	Petaling Jaya, Selangor	Apr 26, 2025	6	Open	Edit View Close
Frontend Developer	TechNova Solutions	Kuala Lumpur, Wilayah Persekutuan	Apr 26, 2025	3	Open	Edit View Close

Candidate Pipeline

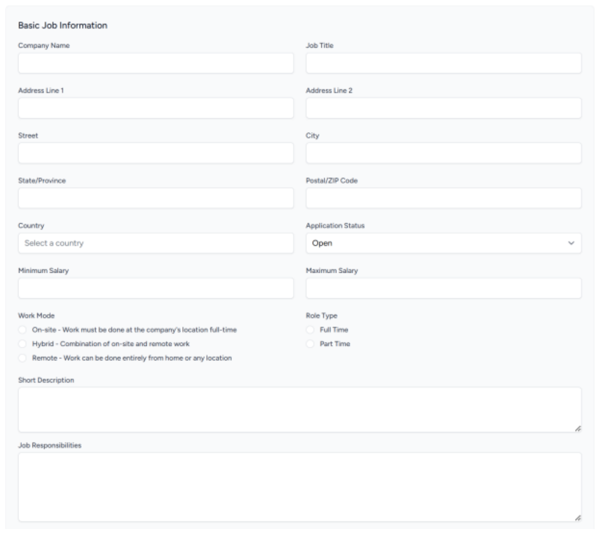
Candidate Name	Applied Position	Company	Application Date	Status	Actions
Ali	Frontend Developer	TechNova Solutions	Apr 28, 2025	Shortlisted	View Application
Davin	Frontend Developer	TechNova Solutions	Apr 27, 2025	Shortlisted	View Application
Davin	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 27, 2025	Interviewed	View Application
Siti	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 27, 2025	Interviewed	View Application
Ahmad	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 26, 2025	Interviewed	View Application
Abu	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 26, 2025	Interviewed	View Application
Ali	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 26, 2025	Interviewed	View Application
Jane Smith	UX/UI Designer at PixelForge Studio	PixelForge Studio	Apr 26, 2025	Interviewed	View Application
Jane Smith	Frontend Developer	TechNova Solutions	Apr 26, 2025	Interviewed	View Application

Interview Insights & Recommendations

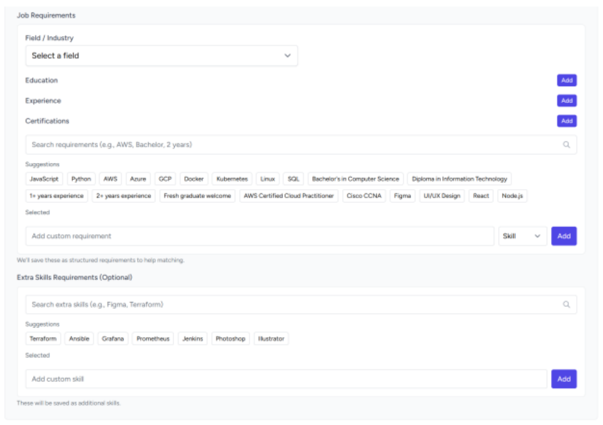
Figure 5.4.6.1 IntelliHire recruiter dashboard

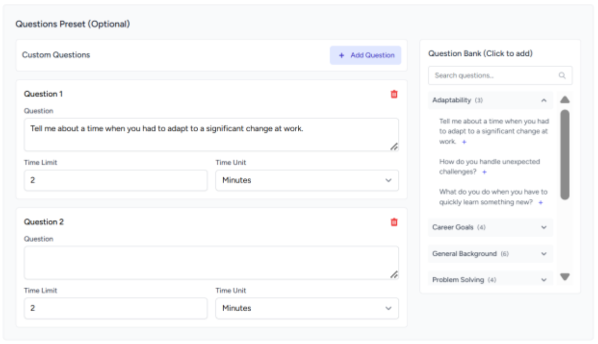
5.4.7 Job Management

The Job Management module enables recruiters to create, publish, review, and update job postings in a structured, repeatable way. The workflow begins with the Job Posting form, where recruiters enter comprehensive details such as job title, company, location, work mode, salary range, responsibilities, and required skills/experience. The form also includes an interview configuration area that lets recruiters attach preset interview questions and optionally link an assessment to the role. Standardizing these elements ensures a consistent interview experience for all applicants and makes downstream evaluation easier and fairer.



Application Status: Open





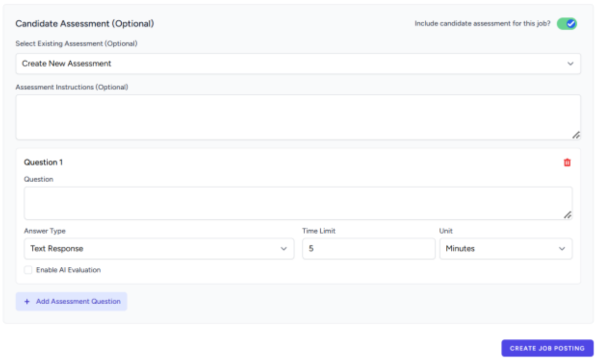
Question 1
Question: Tell me about a time when you had to adapt to a significant change at work.
Time Limit: 2
Time Unit: Minutes

Question 2
Question:
Time Limit: 2
Time Unit: Minutes

Tell me about a time when you had to adapt to a significant change at work. +

How do you handle unexpected challenges? +

What do you do when you have to quickly learn something new? +


Candidate Assessment (Optional)
Assessment Instructions (Optional)
Enable AI Evaluation
+ Add Assessment Question
CREATE JOB POSTING

Include candidate assessment for this job? ✓

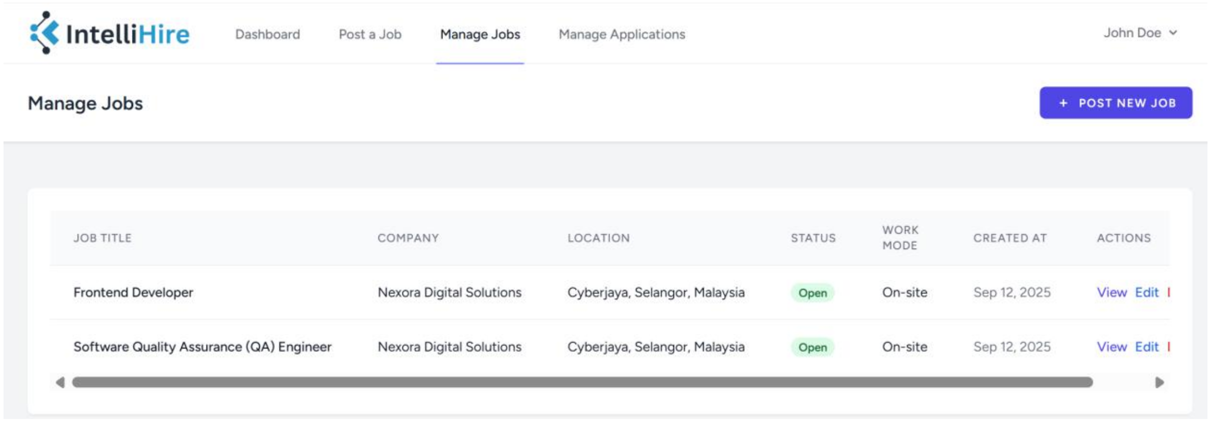
Select Existing Assessment (Optional): Create New Assessment

Question 1
Question:
Answer Type: Text Response
Time Limit: 5
Unit: Minutes

Figure 5.4.7.1 IntelliHire recruiter job posting page: preset question and assessment section

After saving, the posting is created in Open status and becomes available to candidates. The interface supports inline validation (e.g., required fields, numeric ranges) and provides a clear summary preview before publishing. If needed, recruiters can return to the form to adjust wording, add/removes skills, or refine compensation and work-mode details without disrupting existing applications.

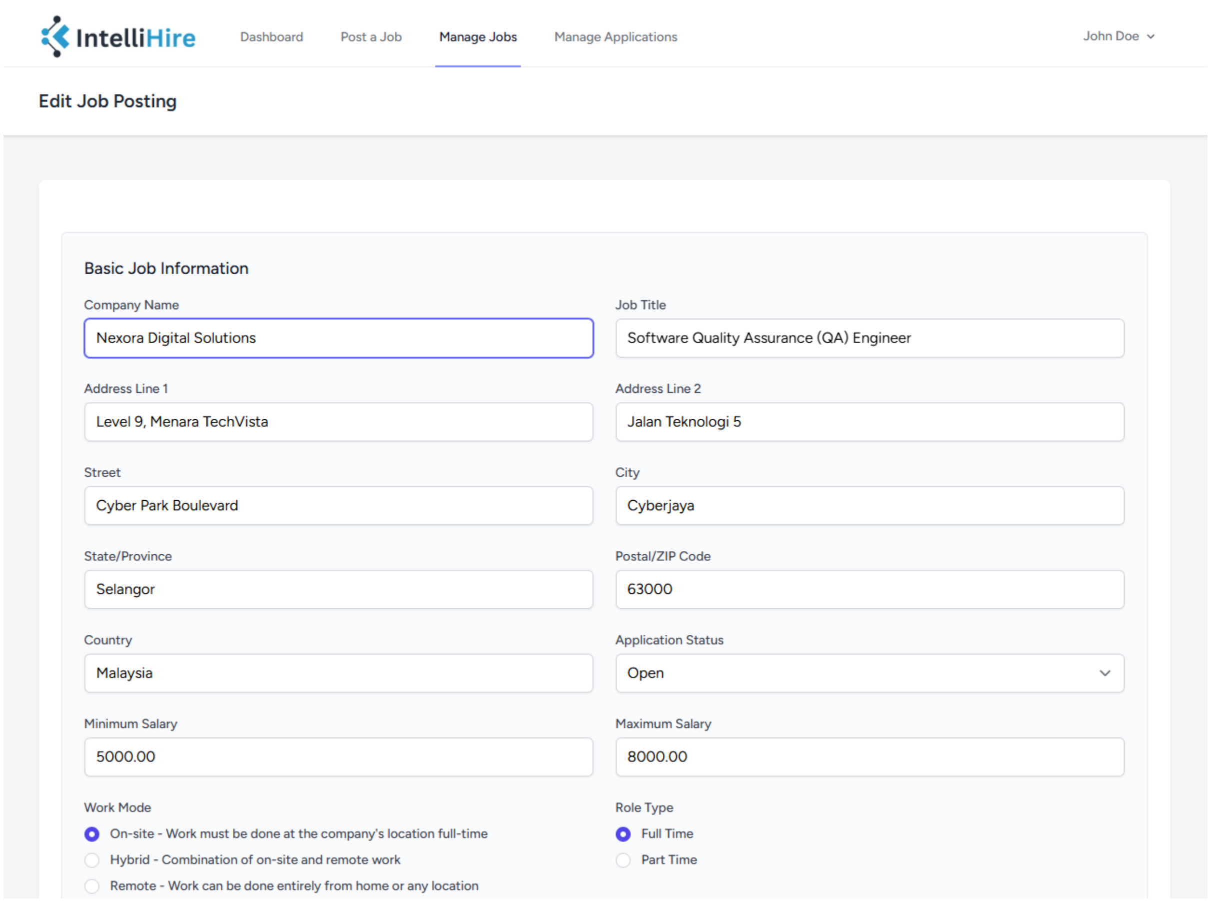
Recruiters can then manage their roles from the Job Management list, which displays all postings with key metadata—job title, date posted, number of applications, and current status. Each row offers quick actions to View, Edit, or Close the posting. The View action opens the job detail page as seen by candidates; Edit allows safe updates to description and requirements; and Close archives the role to prevent new applications while preserving the full application history for auditing and reporting. This centralised table supports fast triage across many roles and makes it easy to identify postings that need attention (e.g., high volume, few applications, or nearing deadlines).



Manage Jobs

JOB TITLE	COMPANY	LOCATION	STATUS	WORK MODE	CREATED AT	ACTIONS
Frontend Developer	Nexora Digital Solutions	Cyberjaya, Selangor, Malaysia	Open	On-site	Sep 12, 2025	View Edit
Software Quality Assurance (QA) Engineer	Nexora Digital Solutions	Cyberjaya, Selangor, Malaysia	Open	On-site	Sep 12, 2025	View Edit

Figure 5.4.7.2 IntelliHire recruiter job management screen



Edit Job Posting

Basic Job Information

Company Name: Nexora Digital Solutions	Job Title: Software Quality Assurance (QA) Engineer
Address Line 1: Level 9, Menara TechVista	Address Line 2: Jalan Teknologi 5
Street: Cyber Park Boulevard	City: Cyberjaya
State/Province: Selangor	Postal/ZIP Code: 63000
Country: Malaysia	Application Status: Open
Minimum Salary: 5000.00	Maximum Salary: 8000.00

Work Mode
- [✓ On-site - Work must be done at the company's location full-time]
- Hybrid - Combination of on-site and remote work
- Remote - Work can be done entirely from home or any location

Role Type
- [✓ Full Time]
- Part Time

Figure 5.4.7.3 IntelliHire recruiter edit job page

5.4.7.1 Backend Processing for Job Management

The JobPostingController is responsible for handling the entire lifecycle of a job posting, from validation and creation to updates and closure. When a recruiter submits a new job, the controller first validates the input fields to ensure correctness, such as checking salary ranges, verifying required attributes like job title and location, and ensuring that optional components like assessments or preset questions follow the defined schema. Once validated, the controller persists the posting into the database, including its responsibilities, requirements, preset interview questions, and any linked assessments. Each action is authorized under role-based access control to ensure that only authenticated recruiters can create or modify postings. This structured flow ensures consistency across all postings and helps maintain data integrity.

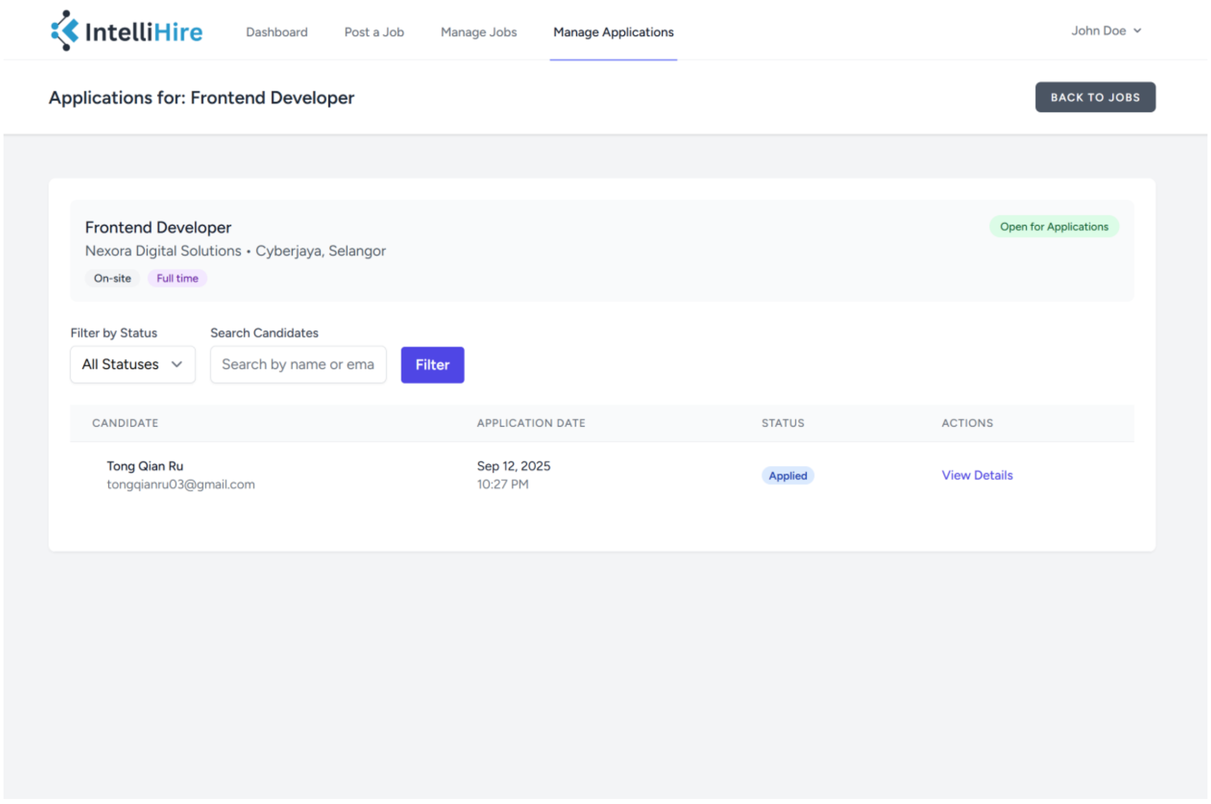
One of the key enhancements in this module is the integration of AI-assisted job authoring. Rather than requiring recruiters to manually construct every responsibility, skill, or preset question, the system leverages the Gemini API to analyze the recruiter’s free-form job description and return structured recommendations. When the recruiter activates the “Generate Suggestions” feature, the controller dynamically constructs a prompt containing company name, job title, responsibilities, requirements, and other details entered in the form. The prompt enforces a strict response contract, requiring Gemini to return structured JSON with fields for responsibilities, required skills (including type and proficiency level), years of experience, educational requirements, certifications, preset interview questions, and whether an assessment should be included. To ensure precision, the prompt also embeds rules to avoid unnecessary repetition, maintain concise question wording, and respect hiring fairness standards.

Once Gemini returns its output, the controller parses the JSON response and validates its structure. If errors are detected, lightweight repair mechanisms attempt to correct issues such as missing brackets or invalid data types. The output is staged as a preview so recruiters can review and selectively accept, edit, or discard suggestions. This prevents the system from blindly committing AI-generated results and gives recruiters full control over the final posting. Accepted data is then mapped to internal tables such as job_responsibilities, job_requirements, and preset_questions, while assessments are automatically linked if recommended by the AI. Additional normalization ensures that variants of the same skill (such as “HTML” and “HTML5”) are unified, and policy filters prevent the inclusion of biased or inappropriate terms. To maintain usability even in the event of AI service disruptions, the controller includes retry logic with timeouts and exponential backoff, while falling back to a basic template-based extractor if Gemini cannot be reached.

When a posting is finalized, the controller performs server-side validation once more before committing all data within a single transaction to guarantee consistency across related tables. Updates and edits follow the same pathway, with timestamps marking version history to preserve a record of recruiter actions. Closing a posting changes its status from Open to Closed without deleting associated applications, allowing the data to remain available for reporting and analytics. To safeguard integrity, all AI calls are authenticated with environment-specific keys, and recruiter edits or overrides are recorded with audit metadata. By combining manual control with AI-assisted suggestions, the backend ensures that job postings are accurate, comprehensive, and recruiter-approved while remaining resilient to service interruptions.

5.4.8 Application Management

The Application Management module gives recruiters a centralized workspace to review candidates, update statuses, and schedule interviews. The Applications Overview screen presents all incoming applications in a sortable table with candidate name, position applied for, application date, and current status. Recruiters can filter the list by status (e.g., Applied, Shortlisted, Interviewed, Offered, Rejected) and quickly drill down to specific candidates for evaluation.



Applications for: Frontend Developer

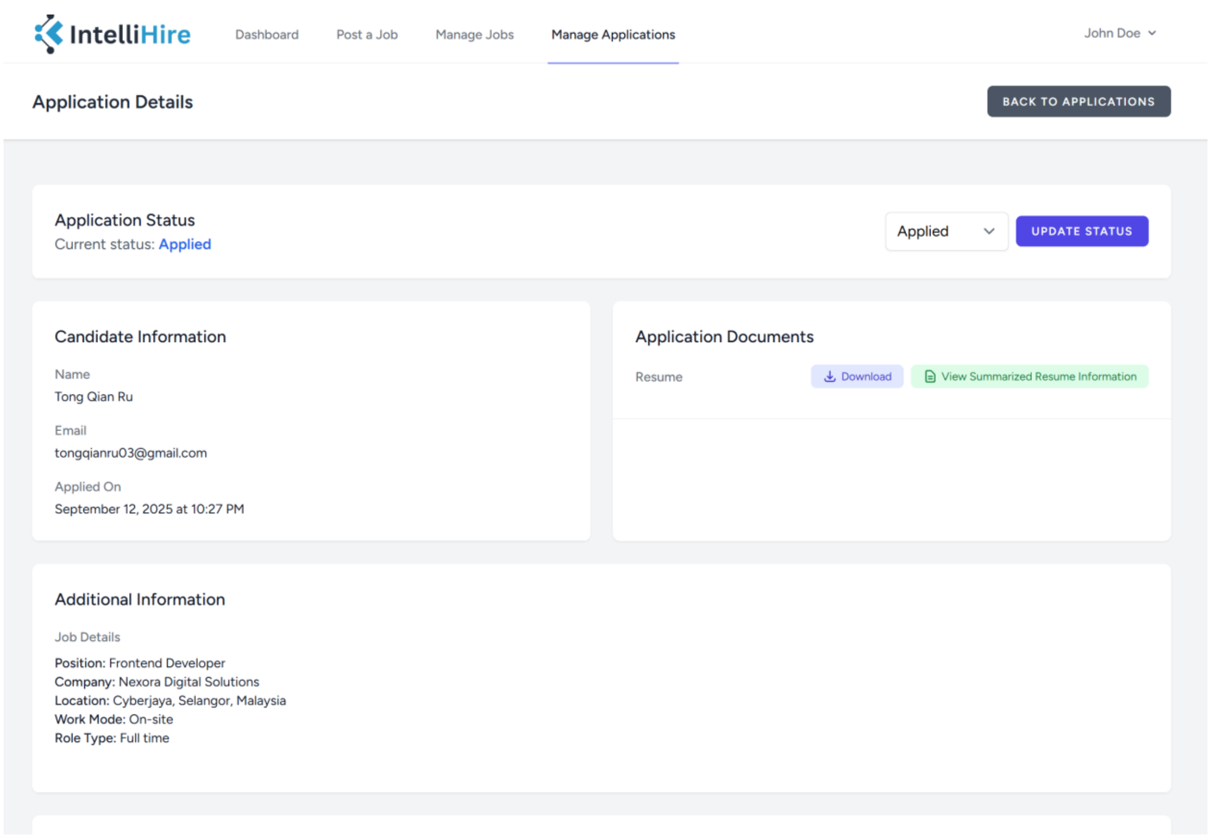
Frontend Developer
Nexora Digital Solutions • Cyberjaya, Selangor
On-site Full time
Open for Applications

Filter by Status: All Statuses
Search Candidates: Search by name or ema

CANDIDATE	APPLICATION DATE	STATUS	ACTIONS
Tong Qian Ru tongqianru03@gmail.com	Sep 12, 2025 10:27 PM	Applied	View Details

Figure 5.4.8.1 IntelliHire recruiter application overview list with candidate table.

Selecting a row opens the Application Details view, which consolidates all materials for the chosen candidate. Recruiters can review the submitted resume and optional cover letter, browse application notes, and, when available, inspect interview artifacts such as recorded responses and AI feedback. From this page, recruiters can update the application status—for example, promoting a candidate from Applied to Shortlisted, moving to Offered, or marking as Rejected. If a candidate has attempted the interview, the status automatically reflects Interviewed, and the related interview record appears at the bottom of the page.



Application Details

Application Status
Current status: Applied
Applied

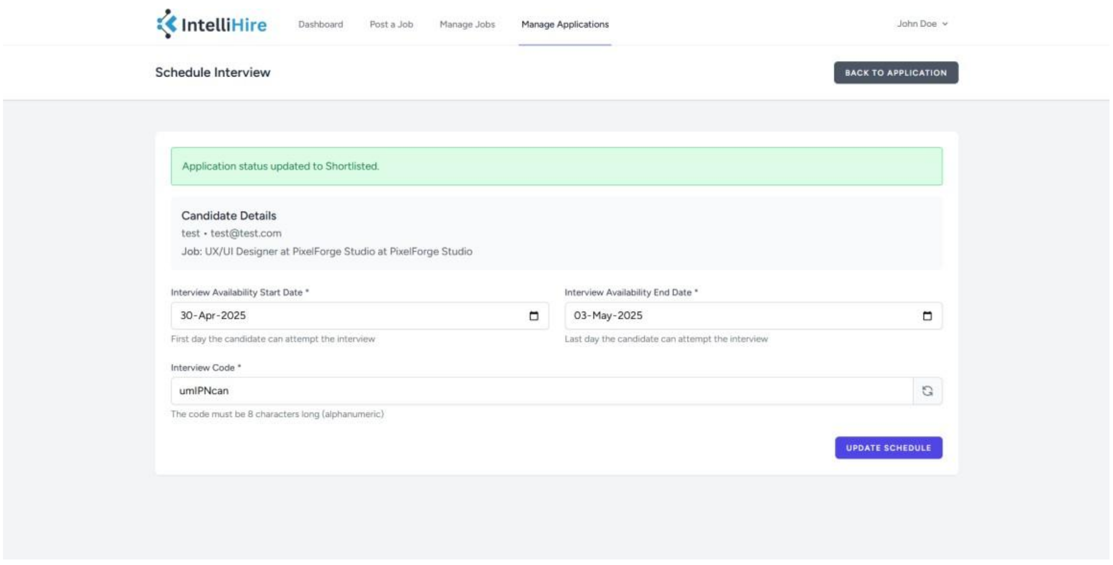
Candidate Information
Name
Tong Qian Ru
Email
tongqianru03@gmail.com
Applied On
September 12, 2025 at 10:27 PM

Application Documents
Resume

Additional Information
Job Details
Position: Frontend Developer
Company: Nexora Digital Solutions
Location: Cyberjaya, Selangor, Malaysia
Work Mode: On-site
Role Type: Full time

Figure 5.4.8.2 IntelliHire recruiter application management screen: candidate details and interview responses

When a candidate is marked Shortlisted, the system initiates the Interview Scheduling flow. The Schedule screen allows the recruiter to define a valid date range for participation. Upon confirmation, the system issues a unique interview code bound to that application and validity window. The candidate must join within the designated timeframe using this code; outside of it, access is denied and the code is considered expired. This lightweight, code-based approach simplifies logistics while preserving control and traceability.



Schedule Interview

Application status updated to Shortlisted.

Candidate Details
test • test@test.com
Job: UX/UI Designer at PixelForge Studio at PixelForge Studio

Interview Availability Start Date *
30-Apr-2025
First day the candidate can attempt the interview

Interview Availability End Date *
03-May-2025
Last day the candidate can attempt the interview

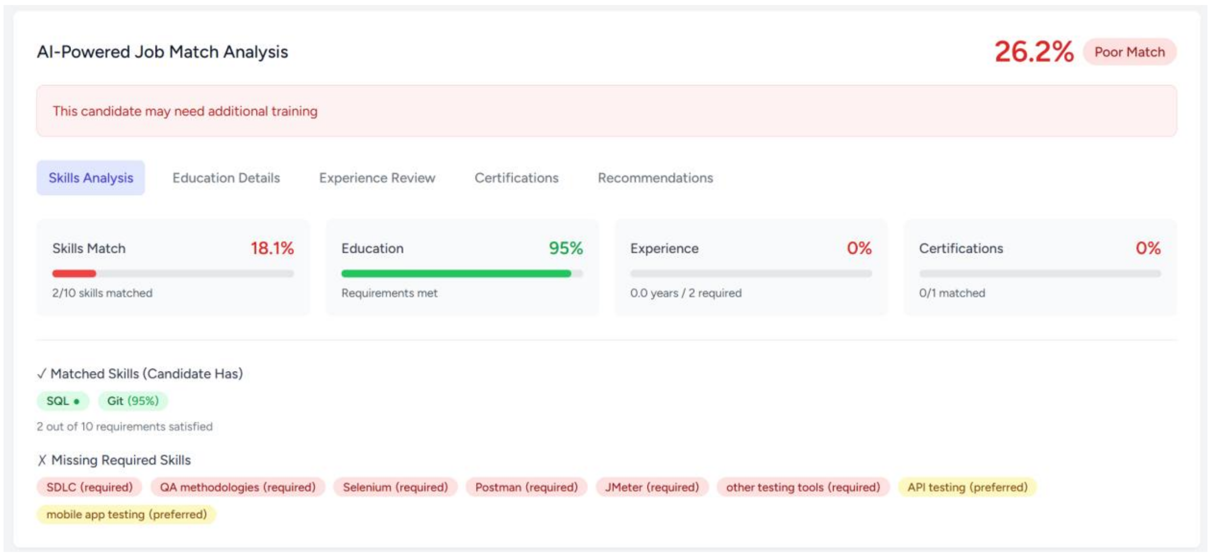
Interview Code *
umIPNcan
The code must be 8 characters long (alphanumeric)

Figure 5.4.8.3 IntelliHire recruiter application management screen: schedule screen

5.4.9 AI Analysis

The AI analysis module provides recruiters with a consolidated view of candidate evaluation results across three dimensions: job match analysis, interview performance analysis, and assessment results. This feature transforms raw candidate data into meaningful insights, enabling recruiters to make informed hiring decisions efficiently. Each interface within this module presents results in a clear and structured manner, combining visual indicators with detailed explanations to highlight both candidate strengths and areas requiring improvement.

The first section, Job Match Analysis, visually represents how well a candidate aligns with the requirements of a job posting. This view includes compatibility percentages for skills, education, experience, and certifications, with color-coded indicators that help recruiters quickly interpret candidate suitability. Matched and missing skills are listed explicitly, distinguishing between required and preferred competencies. A candidate with a high match percentage demonstrates strong alignment, while a low percentage signals significant gaps.



AI-Powered Job Match Analysis

26.2% Poor Match

This candidate may need additional training

Skills Analysis | Education Details | Experience Review | Certifications | Recommendations

Skills Match 18.1%	Education 95%	Experience 0%	Certifications 0%
2/10 skills matched	Requirements met	0.0 years / 2 required	0/1 matched

✓ Matched Skills (Candidate Has)

SQL ● | Git (95%)

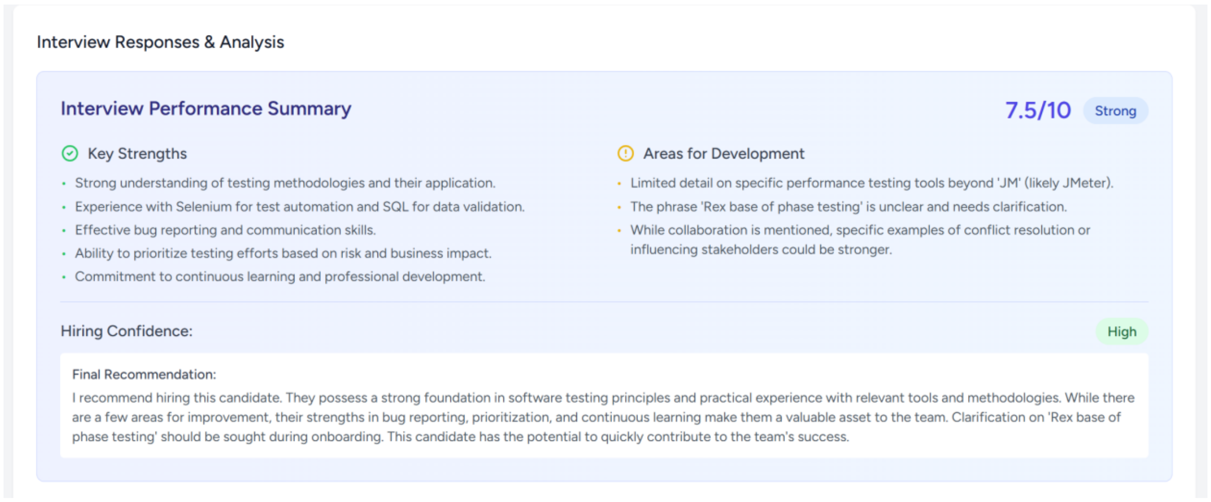
2 out of 10 requirements satisfied

✗ Missing Required Skills

SDLC (required) | QA methodologies (required) | Selenium (required) | Postman (required) | JMeter (required) | other testing tools (required) | API testing (preferred) | mobile app testing (preferred)

Figure 5.4.9.1 IntelliHire AI-powered job match analysis interface

Next, the Interview Responses & Analysis (Overall) page summarizes candidate performance during the AI-powered interview. Recruiters can view a performance score (e.g., 7.5/10), key strengths, and areas for development. The analysis highlights technical expertise, communication quality, and problem-solving approaches, while also identifying weaknesses such as unclear answers or insufficient detail. A "Hiring Confidence" recommendation is displayed at the bottom, offering a final recommendation (e.g., High, Medium, Low confidence) based on overall performance.



Interview Responses & Analysis

Interview Performance Summary

7.5/10 Strong

Key Strengths

- Strong understanding of testing methodologies and their application.
- Experience with Selenium for test automation and SQL for data validation.
- Effective bug reporting and communication skills.
- Ability to prioritize testing efforts based on risk and business impact.
- Commitment to continuous learning and professional development.

Areas for Development

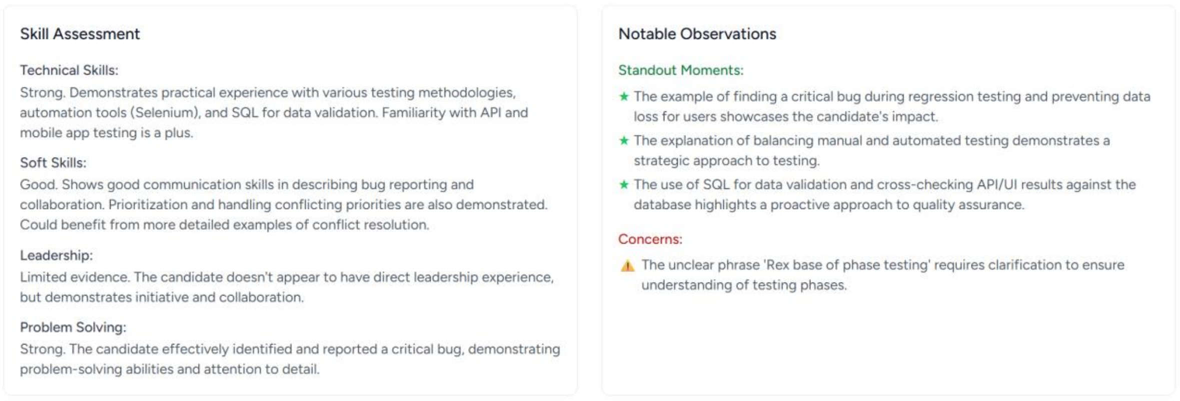
- Limited detail on specific performance testing tools beyond 'JM' (likely JMeter).
- The phrase 'Rex base of phase testing' is unclear and needs clarification.
- While collaboration is mentioned, specific examples of conflict resolution or influencing stakeholders could be stronger.

Hiring Confidence: High

Final Recommendation:

I recommend hiring this candidate. They possess a strong foundation in software testing principles and practical experience with relevant tools and methodologies. While there are a few areas for improvement, their strengths in bug reporting, prioritization, and continuous learning make them a valuable asset to the team. Clarification on 'Rex base of phase testing' should be sought during onboarding. This candidate has the potential to quickly contribute to the team's success.

Figure 5.4.9.2 IntelliHire overall interview performance summary with key strengths and areas for development



Skill Assessment

Technical Skills:
Strong. Demonstrates practical experience with various testing methodologies, automation tools (Selenium), and SQL for data validation. Familiarity with API and mobile app testing is a plus.

Soft Skills:
Good. Shows good communication skills in describing bug reporting and collaboration. Prioritization and handling conflicting priorities are also demonstrated. Could benefit from more detailed examples of conflict resolution.

Leadership:
Limited evidence. The candidate doesn't appear to have direct leadership experience, but demonstrates initiative and collaboration.

Problem Solving:
Strong. The candidate effectively identified and reported a critical bug, demonstrating problem-solving abilities and attention to detail.

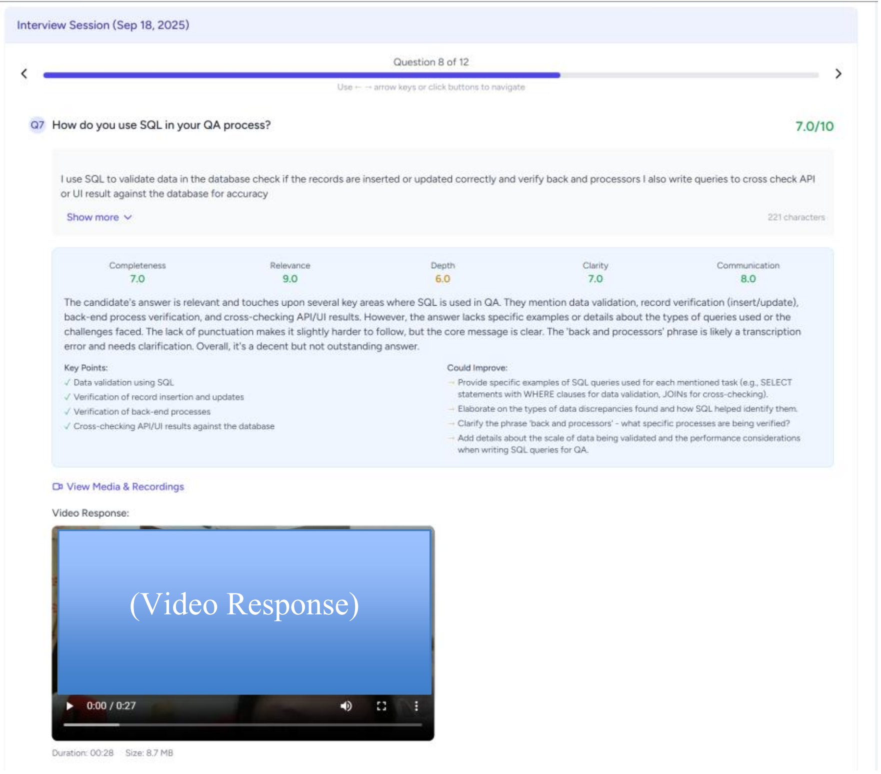
Notable Observations

Standout Moments:
- ★ The example of finding a critical bug during regression testing and preventing data loss for users showcases the candidate's impact.
- ★ The explanation of balancing manual and automated testing demonstrates a strategic approach to testing.
- ★ The use of SQL for data validation and cross-checking API/UI results against the database highlights a proactive approach to quality assurance.

Concerns:
- ⚠ The unclear phrase 'Rex base of phase testing' requires clarification to ensure understanding of testing phases.

Figure 5.4.9.3 IntelliHire detailed skill assessment and recruiter-oriented observations

For a deeper dive, recruiters can examine Interview Responses & Analysis (Per Question). Each individual interview question displays the candidate's response alongside AI-generated evaluation metrics. Scoring categories such as completeness, relevance, clarity, and communication are presented numerically and supported by written feedback. Key points are summarized into "Strengths" and "Could Improve" sections, ensuring recruiters understand exactly why a response scored the way it did. Additionally, recorded video responses are available for review, ensuring human recruiters can validate AI judgments when necessary.



Interview Session (Sep 18, 2025)

Question 8 of 12

Use ← → arrow keys or click buttons to navigate

Q7 How do you use SQL in your QA process? 7.0/10

I use SQL to validate data in the database check if the records are inserted or updated correctly and verify back and processors I also write queries to cross check API or UI result against the database for accuracy

Show more

221 characters

Completeness	Relevance	Depth	Clarity	Communication
7.0	9.0	6.0	7.0	8.0

The candidate's answer is relevant and touches upon several key areas where SQL is used in QA. They mention data validation, record verification (insert/update), back-end process verification, and cross-checking API/UI results. However, the answer lacks specific examples or details about the types of queries used or the challenges faced. The lack of punctuation makes it slightly harder to follow, but the core message is clear. The 'back and processors' phrase is likely a transcription error and needs clarification. Overall, it's a decent but not outstanding answer.

Key Points:
- ✓ Data validation using SQL
- ✓ Verification of record insertion and updates
- ✓ Verification of back-end processes
- ✓ Cross-checking API/UI results against the database

Could Improve:
- → Provide specific examples of SQL queries used for each mentioned task (e.g., SELECT statements with WHERE clauses for data validation, JOINs for cross-checking).
- → Elaborate on the types of data discrepancies found and how SQL helped identify them.
- → Clarify the phrase 'back and processors' - what specific processes are being verified?
- → Add details about the scale of data being validated and the performance considerations when writing SQL queries for QA.

View Media & Recordings

Video Response:

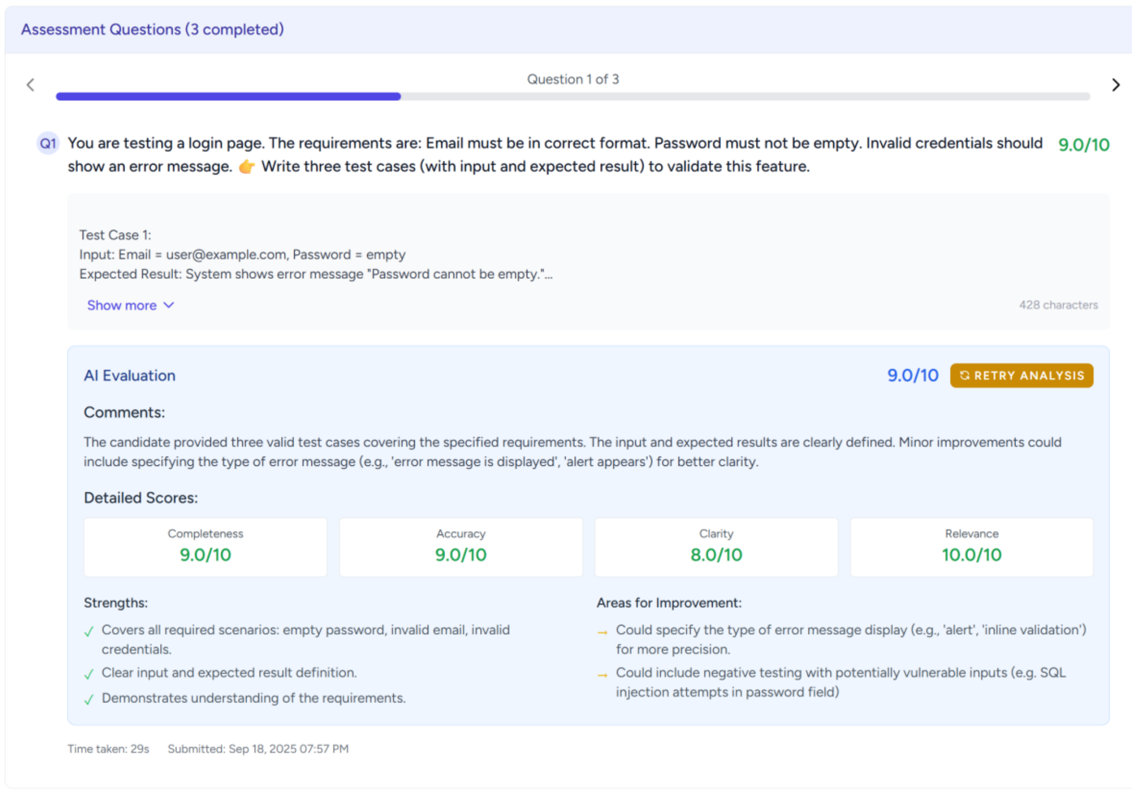
(Video Response)

0:00 / 0:27

Duration: 00:28 Size: 8.7 MB

Figure 5.4.9.4 IntelliHire AI analysis of individual interview responses

The final component of this module is the Assessment Results view, which displays candidate performance in structured assessments. Each question shows the candidate's answer along with AI evaluation scores such as completeness, accuracy, clarity, and relevance. The system highlights detailed comments on strengths and areas for improvement, while also providing an overall score per question. A retry analysis button is available for situations where recruiters wish to reprocess the evaluation for consistency. This structured breakdown enables recruiters to gauge not only correctness but also the quality of thought and communication in candidate answers.


Assessment Questions (3 completed)

Question 1 of 3

Q1 You are testing a login page. The requirements are: Email must be in correct format. Password must not be empty. Invalid credentials should show an error message. 👉 Write three test cases (with input and expected result) to validate this feature. 9.0/10

Test Case 1:
Input: Email = user@example.com, Password = empty
Expected Result: System shows error message "Password cannot be empty."...

Show more

428 characters

AI Evaluation 9.0/10 RETRY ANALYSIS

Comments:

The candidate provided three valid test cases covering the specified requirements. The input and expected results are clearly defined. Minor improvements could include specifying the type of error message (e.g., 'error message is displayed', 'alert appears') for better clarity.

Detailed Scores:

Completeness	Accuracy	Clarity	Relevance
9.0/10	9.0/10	8.0/10	10.0/10

Strengths:
- ✓ Covers all required scenarios: empty password, invalid email, invalid credentials.
- ✓ Clear input and expected result definition.
- ✓ Demonstrates understanding of the requirements.

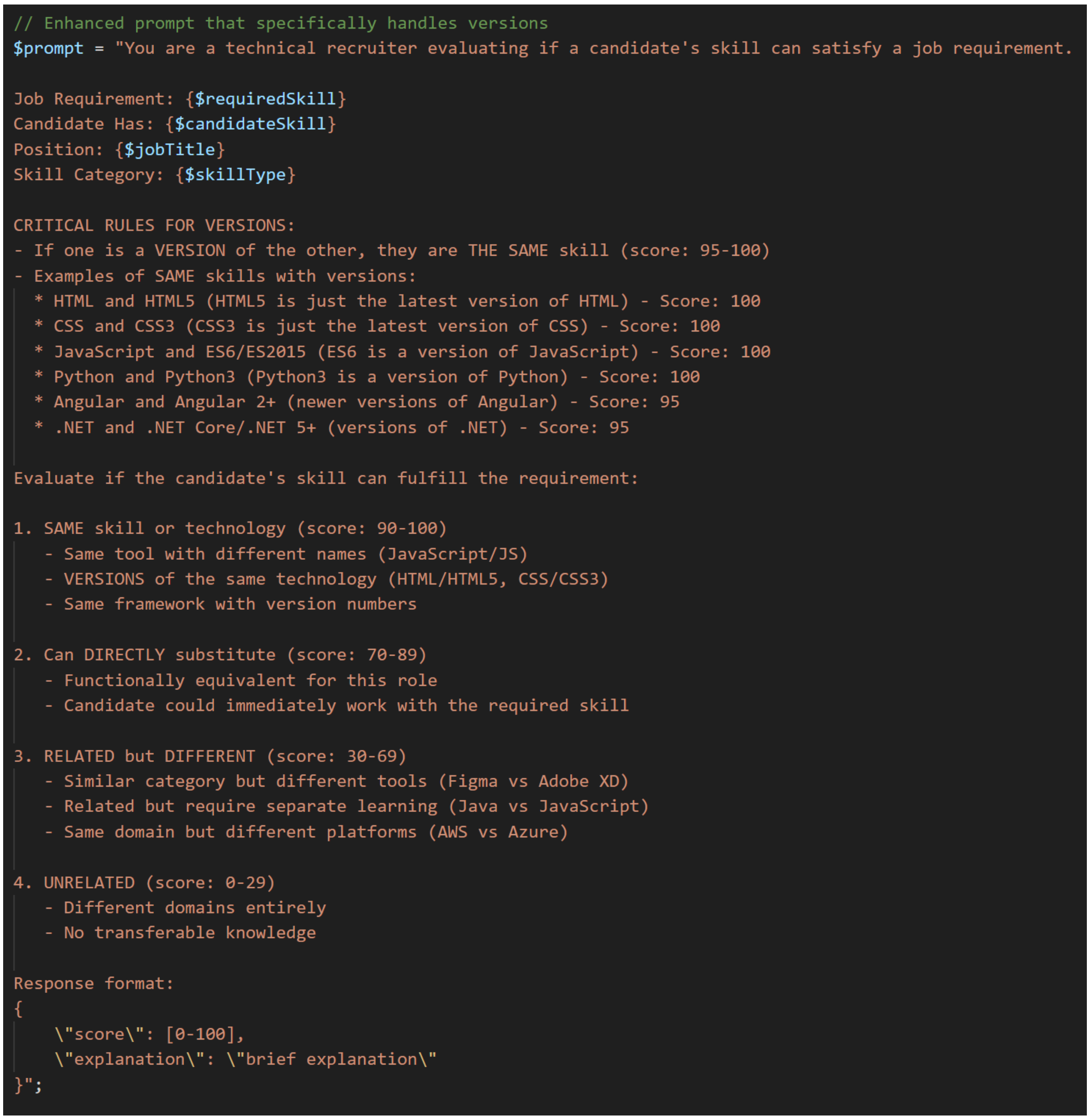
Areas for Improvement:
- → Could specify the type of error message display (e.g., 'alert', 'inline validation') for more precision.
- → Could include negative testing with potentially vulnerable inputs (e.g. SQL injection attempts in password field)

Time taken: 29s Submitted: Sep 18, 2025 07:57 PM

Figure 5.4.9.5 IntelliHire AI-powered assessment evaluation screen

5.4.9.1 Job Matching Backend

The backend for job matching begins by pulling structured data from multiple related tables, including candidate resumes (with experiences, education, skills, and certifications) and the job postings with their required qualifications. When a recruiter requests analysis, the JobMatchingController::calculateMatch() method loads the relevant application record and retrieves all associated resume and requirement data. This structured dataset is then passed to the Gemini API, which evaluates candidate-job compatibility category by category. Each skill, educational credential, and experience requirement is compared against parsed resume data. The AI is guided by prompts that explicitly define similarity levels, such as treating HTML and HTML5 as equivalent skills, or Python and Python3 as versions of the same language. Scoring is returned on a scale of 0–100, with short explanations for each match, and stored temporarily in the Laravel cache for performance. Recruiters thus receive not only an overall compatibility percentage but also detailed highlights of strengths and gaps across all requirement categories.

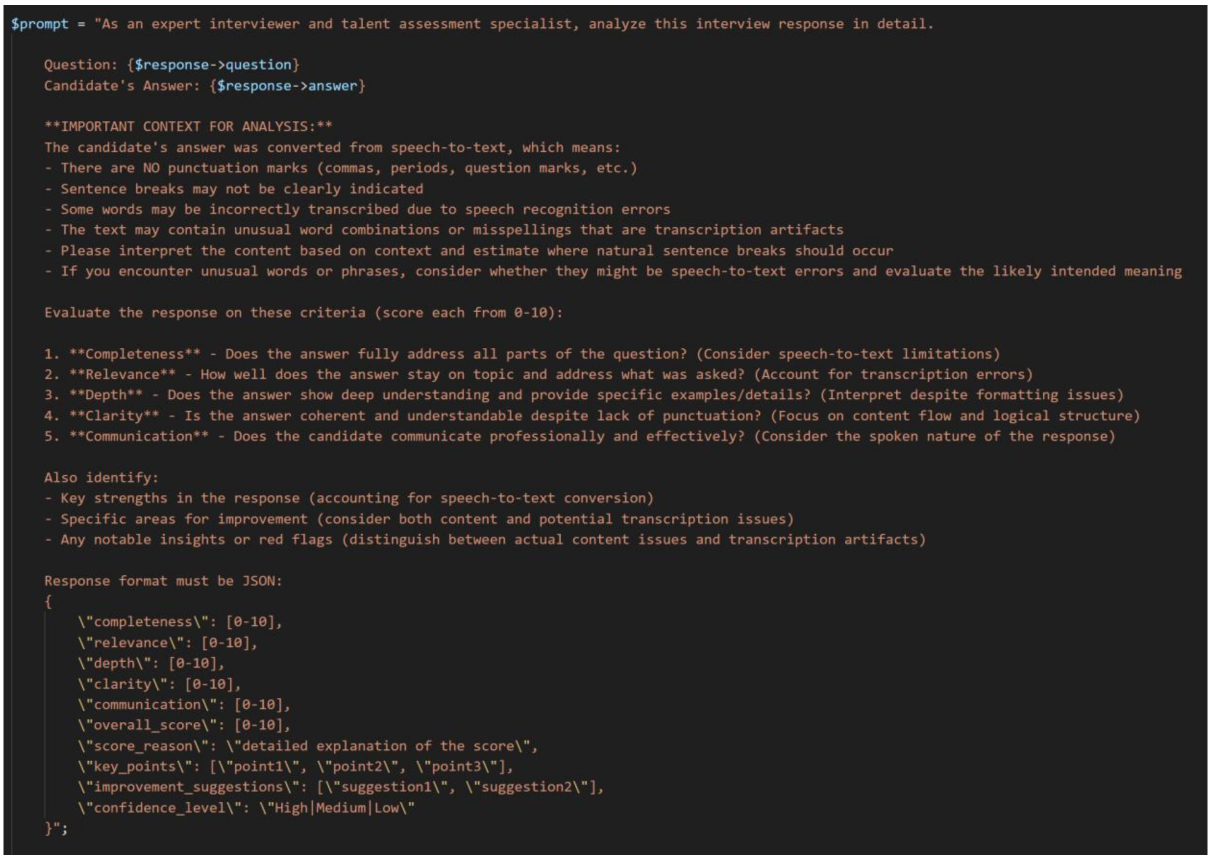


```
// Enhanced prompt that specifically handles versions
$prompt = "You are a technical recruiter evaluating if a candidate's skill can satisfy a job requirement.

Job Requirement: {$requiredSkill}
Candidate Has: {$candidateSkill}
Position: {$jobTitle}
Skill Category: {$skillType}

CRITICAL RULES FOR VERSIONS:
- If one is a VERSION of the other, they are THE SAME skill (score: 95-100)
- Examples of SAME skills with versions:
  * HTML and HTML5 (HTML5 is just the latest version of HTML) - Score: 100
  * CSS and CSS3 (CSS3 is just the latest version of CSS) - Score: 100
  * JavaScript and ES6/ES2015 (ES6 is a version of JavaScript) - Score: 100
  * Python and Python3 (Python3 is a version of Python) - Score: 100
  * Angular and Angular 2+ (newer versions of Angular) - Score: 95
  * .NET and .NET Core/.NET 5+ (versions of .NET) - Score: 95

Evaluate if the candidate's skill can fulfill the requirement:

1. SAME skill or technology (score: 90-100)
   - Same tool with different names (JavaScript/JS)
   - VERSIONS of the same technology (HTML/HTML5, CSS/CSS3)
   - Same framework with version numbers

2. Can DIRECTLY substitute (score: 70-89)
   - Functionally equivalent for this role
   - Candidate could immediately work with the required skill

3. RELATED but DIFFERENT (score: 30-69)
   - Similar category but different tools (Figma vs Adobe XD)
   - Related but require separate learning (Java vs JavaScript)
   - Same domain but different platforms (AWS vs Azure)

4. UNRELATED (score: 0-29)
   - Different domains entirely
   - No transferable knowledge

Response format:
{
    \"score\": [0-100],
    \"explanation\": \"brief explanation\"
}";
```

Figure 5.4.9.6 AI prompt engineering for skill matching with version-aware scoring criteria and structured JSON response format

5.4.9.2 Interview Analysis Backend

For interviews, candidate responses are first stored in the interview_responses table, linked to their application and schedule records. The RecruiterApplicationController:: analyzeInterviewResponses() function processes these entries by iterating over each recorded answer and submitting them to Gemini for evaluation. Since the responses originate from speech-to-text transcription, the AI prompt explicitly accounts for missing punctuation, unclear sentence boundaries, and potential transcription errors. Gemini is instructed to score each answer across five criteria—completeness, relevance, depth, clarity, and communication—using a 0–10 scale, and to provide structured JSON feedback. Results for each answer are stored in the ai_feedback table, while aggregated session-level analysis (including overall scores, strengths, weaknesses, and hiring confidence) is persisted in the analysis_results table. This separation ensures recruiters can review both granular and high-level insights. Malformed AI outputs are handled through JSON extraction and validation logic, with retries triggered automatically when necessary.

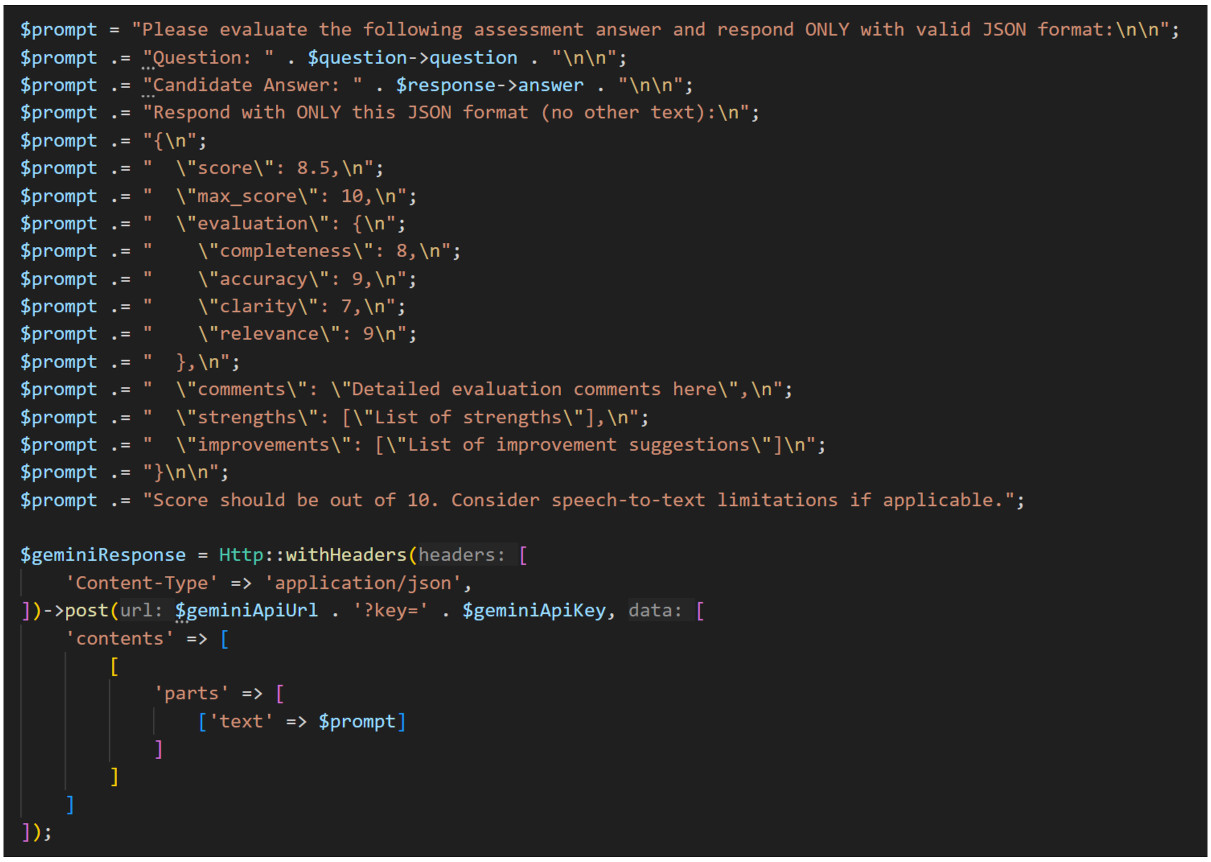


```
$prompt = "As an expert interviewer and talent assessment specialist, analyze this interview response in detail.

    Question: {$response->question}
    Candidate's Answer: {$response->answer}

    **IMPORTANT CONTEXT FOR ANALYSIS:**
    The candidate's answer was converted from speech-to-text, which means:
    - There are NO punctuation marks (commas, periods, question marks, etc.)
    - Sentence breaks may not be clearly indicated
    - Some words may be incorrectly transcribed due to speech recognition errors
    - The text may contain unusual word combinations or misspellings that are transcription artifacts
    - Please interpret the content based on context and estimate where natural sentence breaks should occur
    - If you encounter unusual words or phrases, consider whether they might be speech-to-text errors and evaluate the likely intended meaning

    Evaluate the response on these criteria (score each from 0-10):

    1. **Completeness** - Does the answer fully address all parts of the question? (Consider speech-to-text limitations)
    2. **Relevance** - How well does the answer stay on topic and address what was asked? (Account for transcription errors)
    3. **Depth** - Does the answer show deep understanding and provide specific examples/details? (Interpret despite formatting issues)
    4. **Clarity** - Is the answer coherent and understandable despite lack of punctuation? (Focus on content flow and logical structure)
    5. **Communication** - Does the candidate communicate professionally and effectively? (Consider the spoken nature of the response)

    Also identify:
    - Key strengths in the response (accounting for speech-to-text conversion)
    - Specific areas for improvement (consider both content and potential transcription issues)
    - Any notable insights or red flags (distinguish between actual content issues and transcription artifacts)

    Response format must be JSON:
    {
        \"completeness\": [0-10],
        \"relevance\": [0-10],
        \"depth\": [0-10],
        \"clarity\": [0-10],
        \"communication\": [0-10],
        \"overall_score\": [0-10],
        \"score_reason\": \"detailed explanation of the score\",
        \"key_points\": [\"point1\", \"point2\", \"point3\"],
        \"improvement_suggestions\": [\"suggestion1\", \"suggestion2\"],
        \"confidence_level\": \"High|Medium|Low\"
    }";
```

Figure 5.4.9.7 Speech-to-text aware interview analysis prompt with five-dimensional scoring criteria and structured JSON feedback format

5.4.9.3 Assessment Analysis Backend

Assessment responses follow a similar pipeline, starting from entries in the assessment_responses table, each tied to a candidate schedule and specific assessment questions. The controller method analyzeAssessment() processes these submissions by sending each response to Gemini with structured prompts that request evaluation under four dimensions: completeness, accuracy, clarity, and relevance. The AI returns JSON-formatted output that includes a numeric score, max score, explanatory comments, strengths, and areas for improvement. These evaluation details are stored directly within the assessment_responses table under dedicated AI fields, keeping all raw and evaluated data in a single location. A retry mechanism is also available for failed or inconsistent evaluations, allowing recruiters to re-run the analysis at the click of a button. This integration ensures that assessments benefit from the same consistency and depth of analysis as interviews, while maintaining lightweight database storage.

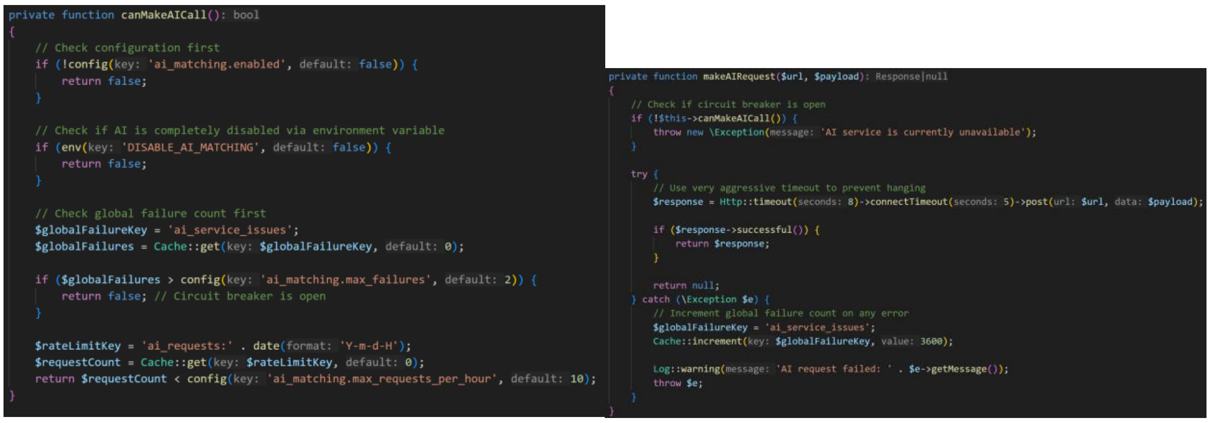


```php
$prompt = "Please evaluate the following assessment answer and respond ONLY with valid JSON format:\n\n";
$prompt .= "Question: " . $question->question . "\n\n";
$prompt .= "Candidate Answer: " . $response->answer . "\n\n";
$prompt .= "Respond with ONLY this JSON format (no other text):\n";
$prompt .= "{\n";
$prompt .= "  \"score\": 8.5,\n";
$prompt .= "  \"max_score\": 10,\n";
$prompt .= "  \"evaluation\": {\n";
$prompt .= "    \"completeness\": 8,\n";
$prompt .= "    \"accuracy\": 9,\n";
$prompt .= "    \"clarity\": 7,\n";
$prompt .= "    \"relevance\": 9\n";
$prompt .= "  },\n";
$prompt .= "  \"comments\": \"Detailed evaluation comments here\",\n";
$prompt .= "  \"strengths\": [\"List of strengths\"],\n";
$prompt .= "  \"improvements\": [\"List of improvement suggestions\"]\n";
$prompt .= "}\n\n";
$prompt .= "Score should be out of 10. Consider speech-to-text limitations if applicable.";

$geminiResponse = Http::withHeaders(headers: [
    'Content-Type' => 'application/json',
])->post(url: $geminiApiUrl . '?key=' . $geminiApiKey, data: [
    'contents' => [
        [
            'parts' => [
                ['text' => $prompt]
            ]
        ]
    ]
]);
```

Figure 5.4.9.8 Assessment evaluation prompt with four-dimensional scoring and direct HTTP API integration for automated response analysis

5.4.9.4 AI Integration and Error Handling

All AI analysis across job matching, interviews, and assessments relies on Gemini 2.0 Flash through an HTTP client configured in Laravel. API keys are securely stored in environment variables, and requests use HTTPS-only communication. To guarantee reliability, the system enforces variable timeouts (8-60 seconds depending on operation type) and applies a circuit breaker pattern with a 3-failure threshold to avoid overloading the service. Failed responses trigger fallback strategies such as basic keyword matching, cached results, or graceful degradation with 5-minute cooldown periods. The backend employs JSON parsing and validation functions to extract usable content from AI responses, ensuring malformed output does not disrupt processing. Error handling includes retries for individual responses with exponential backoff, session-wide recovery attempts, and clear recruiter-facing warnings if analysis cannot be completed. The system also implements rate limiting (10 requests per hour) and maintains failure tracking with 1-hour cache duration.



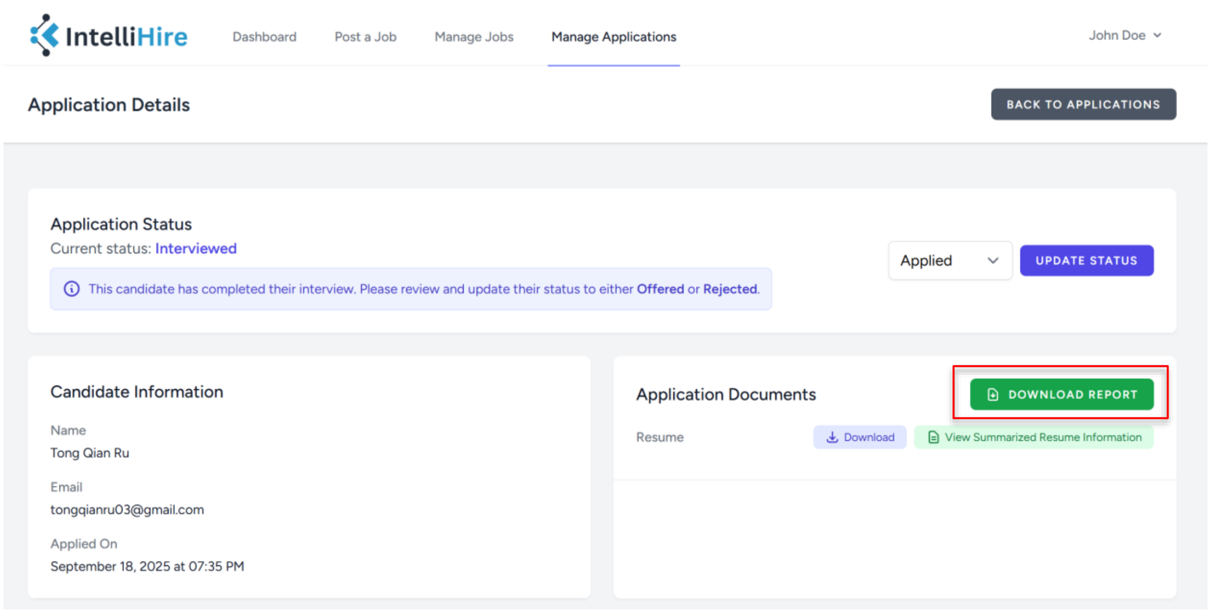
```php
private function canMakeAICall(): bool
{
    // Check configuration first
    if (!config(key: 'ai_matching.enabled', default: false)) {
        return false;
    }

    // Check if AI is completely disabled via environment variable
    if (env(key: 'DISABLE_AI_MATCHING', default: false)) {
        return false;
    }

    // Check global failure count first
    $globalFailureKey = 'ai_service_issues';
    $globalFailures = Cache::get(key: $globalFailureKey, default: 0);

    if ($globalFailures > config(key: 'ai_matching.max_failures', default: 2)) {
        return false; // Circuit breaker is open
    }

    $rateLimitKey = 'ai_requests:' . date(format: 'Y-m-d-H');
    $requestCount = Cache::get(key: $rateLimitKey, default: 0);
    return $requestCount < config(key: 'ai_matching.max_requests_per_hour', default: 10);
}
```

```php
private function makeAIRequest($url, $payload): Response|null
{
    // Check if circuit breaker is open
    if (!$this->canMakeAICall()) {
        throw new \Exception(message: 'AI service is currently unavailable');
    }

    try {
        // Use very aggressive timeout to prevent hanging
        $response = Http::timeout(seconds: 8)->connectTimeout(seconds: 5)->post(url: $url, data: $payload);

        if ($response->successful()) {
            return $response;
        }

        return null;
    } catch (\Exception $e) {
        // Increment global failure count on any error
        $globalFailureKey = 'ai_service_issues';
        Cache::increment(key: $globalFailureKey, value: 3600);

        Log::warning(message: 'AI request failed: ' . $e->getMessage());
        throw $e;
    }
}
```

Figure 5.4.9.9 Circuit breaker pattern implementation with failure tracking, rate limiting, and aggressive timeout handling for AI service reliability

5.4.10 PDF Report Generation System

The PDF report generation system is an integrated feature within the IntelliHire platform that enables recruiters to download professional-grade reports summarizing a candidate's complete evaluation journey. This functionality is triggered once all interview and assessment analyses are completed, ensuring that the generated report is both comprehensive and reliable. Recruiters can access the report via the application detail page, where a dedicated download button becomes available upon completion of all required evaluations.



Application Details

Application Status
Current status: Interviewed
This candidate has completed their interview. Please review and update their status to either Offered or Rejected.

Candidate Information

Name
Tong Qian Ru

Email
tongqianru03@gmail.com

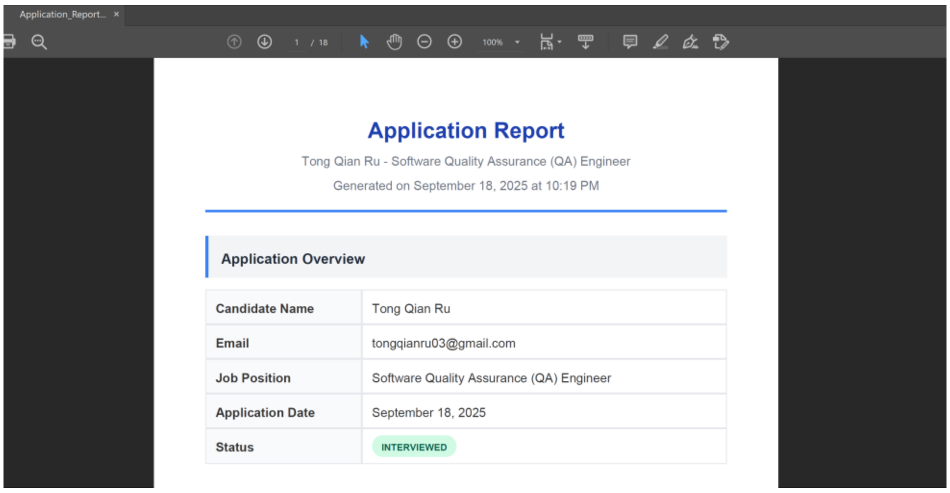
Applied On
September 18, 2025 at 07:35 PM

Application Documents

Resume

Figure 5.4.10.1 Application detail page showing the download report feature once all analyses are complete.

At the backend, the system is powered by Laravel's DomPDF package, which converts dynamically generated Blade templates into structured PDF files. The process begins with the RecruiterApplicationController::exportPdf() method, which acts as the main entry point. This method loads complex relational data using Laravel's Eloquent ORM, including candidate details, resumes, interview responses, assessment results, and AI-generated analyses. The architecture employs both eager loading and fallback manual queries to guarantee that no critical information is missed, even if relationship loading fails.



Application Report

Tong Qian Ru - Software Quality Assurance (QA) Engineer

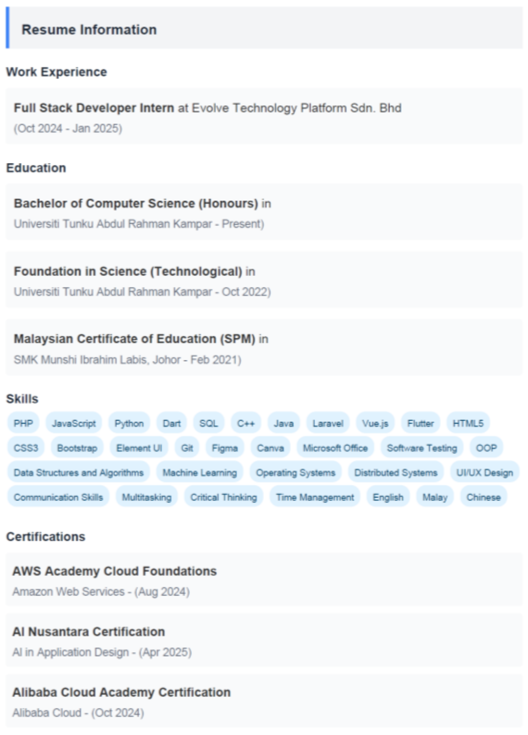
Generated on September 18, 2025 at 10:19 PM

Application Overview

Candidate Name	Tong Qian Ru
Email	tongqianru03@gmail.com
Job Position	Software Quality Assurance (QA) Engineer
Application Date	September 18, 2025
Status	INTERVIEWED

Figure 5.4.10.2 Sample of generated PDF report overview showing candidate and application details.

The PDF report integrates AI analysis results directly into its content. For interview responses, natural language processing is applied to evaluate answers across multiple dimensions, including completeness, relevance, clarity, and communication. The system also accounts for limitations in speech-to-text transcription, ensuring that candidate evaluations remain fair despite minor errors. For assessments, candidate responses are validated against predefined rules and scored by the AI across criteria such as accuracy, completeness, and relevance. All of these results are compiled into structured report sections, presenting recruiters with both strengths and improvement suggestions.



Resume Information

Work Experience

Full Stack Developer Intern at Evolve Technology Platform Sdn. Bhd
(Oct 2024 - Jan 2025)

Education

Bachelor of Computer Science (Honours) in
Universiti Tunku Abdul Rahman Kampar - Present)

Foundation in Science (Technological) in
Universiti Tunku Abdul Rahman Kampar - Oct 2022)

Malaysian Certificate of Education (SPM) in
SMK Munshi Ibrahim Labis, Johor - Feb 2021)

Skills

PHP · JavaScript · Python · Dart · SQL · C++ · Java · Laravel · Vue.js · Flutter · HTML5 · CSS3 · Bootstrap · Element UI · Git · Figma · Canva · Microsoft Office · Software Testing · OOP · Data Structures and Algorithms · Machine Learning · Operating Systems · Distributed Systems · UI/UX Design · Communication Skills · Multitasking · Critical Thinking · Time Management · English · Malay · Chinese

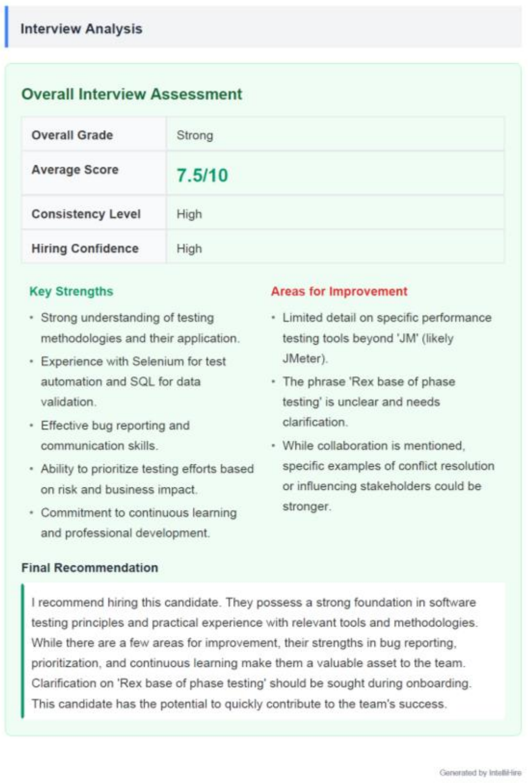
Certifications

AWS Academy Cloud Foundations
Amazon Web Services - (Aug 2024)

AI Nusantara Certification
AI in Application Design - (Apr 2025)

Alibaba Cloud Academy Certification
Alibaba Cloud - (Oct 2024)

Figure 5.4.10.3 Resume section within the generated PDF, including work experience, education, skills, and certifications.



Interview Analysis

Overall Interview Assessment

Overall Grade	Strong
Average Score	7.5/10
Consistency Level	High
Hiring Confidence	High

Key Strengths

- Strong understanding of testing methodologies and their application.
- Experience with Selenium for test automation and SQL for data validation.
- Effective bug reporting and communication skills.
- Ability to prioritize testing efforts based on risk and business impact.
- Commitment to continuous learning and professional development.

Areas for Improvement

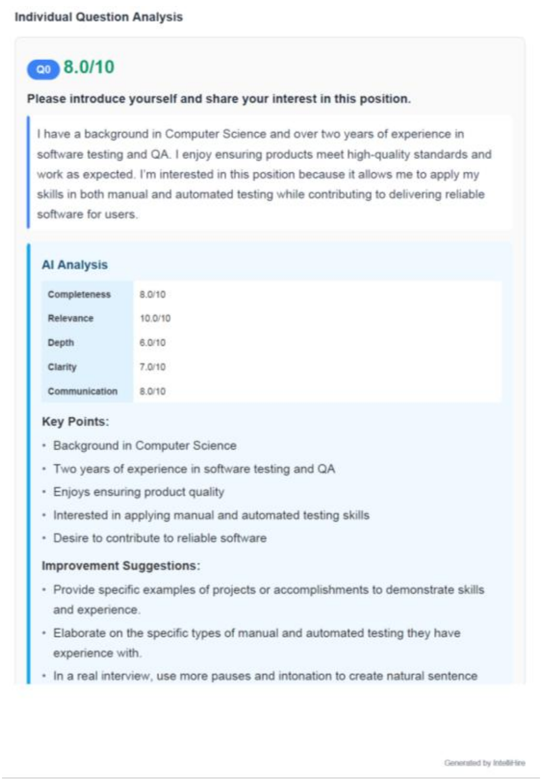
- Limited detail on specific performance testing tools beyond 'JM' (likely JMeter).
- The phrase 'Rex base of phase testing' is unclear and needs clarification.
- While collaboration is mentioned, specific examples of conflict resolution or influencing stakeholders could be stronger.

Final Recommendation

I recommend hiring this candidate. They possess a strong foundation in software testing principles and practical experience with relevant tools and methodologies. While there are a few areas for improvement, their strengths in bug reporting, prioritization, and continuous learning make them a valuable asset to the team. Clarification on 'Rex base of phase testing' should be sought during onboarding. This candidate has the potential to quickly contribute to the team's success.

Generated by IntelliHire

Figure 5.4.10.4 Overall interview analysis section highlighting AI-generated strengths, weaknesses, and final recommendations.



Individual Question Analysis

Q0 8.0/10

Please introduce yourself and share your interest in this position.

I have a background in Computer Science and over two years of experience in software testing and QA. I enjoy ensuring products meet high-quality standards and work as expected. I'm interested in this position because it allows me to apply my skills in both manual and automated testing while contributing to delivering reliable software for users.

AI Analysis

Completeness	8.0/10
Relevance	10.0/10
Depth	6.0/10
Clarity	7.0/10
Communication	8.0/10

Key Points:

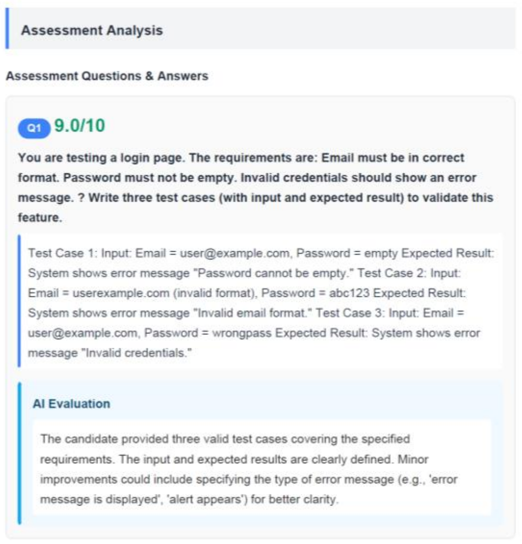
- Background in Computer Science
- Two years of experience in software testing and QA
- Enjoys ensuring product quality
- Interested in applying manual and automated testing skills
- Desire to contribute to reliable software

Improvement Suggestions:

- Provide specific examples of projects or accomplishments to demonstrate skills and experience.
- Elaborate on the specific types of manual and automated testing they have experience with.
- In a real interview, use more pauses and intonation to create natural sentence

Generated by IntelliHire

Figure 5.4.10.5 Breakdown of candidate responses and AI evaluation per question.



Assessment Analysis

Assessment Questions & Answers

Q1 9.0/10

You are testing a login page. The requirements are: Email must be in correct format. Password must not be empty. Invalid credentials should show an error message. ? Write three test cases (with input and expected result) to validate this feature.

Test Case 1: Input: Email = user@example.com, Password = empty Expected Result: System shows error message "Password cannot be empty." Test Case 2: Input: Email = userexample.com (invalid format), Password = abc123 Expected Result: System shows error message "Invalid email format." Test Case 3: Input: Email = user@example.com, Password = wrongpass Expected Result: System shows error message "Invalid credentials."

AI Evaluation

The candidate provided three valid test cases covering the specified requirements. The input and expected results are clearly defined. Minor improvements could include specifying the type of error message (e.g., 'error message is displayed', 'alert appears') for better clarity.

Figure 5.4.10.6 Assessment report showing candidate answers, expected outcomes, and AI scoring.

5.5 Implementation Issues and Challenges

The implementation of IntelliHire presented several challenges that required iterative refinement of both technical and design aspects. One of the most significant issues was integrating with the Gemini API, which powers the system's job matching, interview question generation, and response analysis. Since large language models can sometimes return inconsistent or malformed JSON outputs, strict prompt engineering and validation routines had to be introduced. Additional retry logic was also required to handle API timeouts and rate limits, ensuring that the system remained reliable even when processing multiple candidates simultaneously.

Another major challenge involved the real-time features of the platform, particularly speech-to-text (STT) and text-to-speech (TTS). While these functions improved interactivity, browser compatibility introduced unexpected difficulties. Different browsers varied in their handling of continuous recognition, permissions, and recording, requiring the implementation of permission checks, fallback logic, and clear user feedback messages. Handling video and audio recordings also introduced storage concerns, as large media files risked slowing down the system if not managed carefully. To overcome this, the database was optimized to separate lightweight text transcripts from heavier multimedia files, striking a balance between performance and long-term storage.

Maintaining consistent workflows across modules such as applications, interviews, assessments, and recruiter dashboards was another area that demanded attention. Changes in application status, for example, had to trigger corresponding updates in interview scheduling and code generation to prevent misaligned records. This required careful database validation and cross-module testing. From a usability standpoint, enforcing strict rules—such as preventing candidates from retaking interviews once started—sometimes confused users who navigated away accidentally. Similarly, recruiter dashboards initially overloaded users with excessive detail, which was later refined into a more balanced and accessible design.

Finally, challenges also arose in ensuring fairness and transparency in AI-driven evaluation. Speech-to-text limitations, such as missing punctuation or transcription errors, sometimes influenced results. To mitigate this, prompts were refined to instruct Gemini to tolerate such artifacts, and explanatory feedback was provided alongside numerical scores so recruiters could better interpret AI outputs. Development and testing constraints further complicated implementation, as frequent API calls were costly and time-consuming. This was addressed by using mock responses during debugging, which allowed the system to be tested without exhausting API limits.

Overall, these challenges underscored the complexity of building a system that combines AI, real-time browser capabilities, and structured recruitment workflows. By addressing these issues, IntelliHire was able to evolve into a robust and fair platform that balances technical innovation with practical usability.

5.6 Concluding Remark

In conclusion, this chapter has presented the detailed implementation of the IntelliHire system, covering both the candidate and recruiter portals, the AI-powered interview and assessment modules, and the supporting backend logic. Each component was explained with reference to its user interface, functional workflow, and integration with AI services, demonstrating how the system operates as a cohesive whole. By combining Laravel for backend management, MySQL for structured data storage, and Gemini API for dynamic question generation and evaluation, IntelliHire successfully integrates traditional web technologies with advanced AI features.

This chapter has also highlighted how real-time functionalities such as text-to-speech, speech-to-text, and interview session handling were embedded into the platform, ensuring an interactive and realistic recruitment experience. For recruiters, the dashboards, job management tools, and AI-enhanced analytics provide actionable insights and streamlined candidate evaluation. Overall, the implementation described in this chapter forms the backbone of the IntelliHire platform, bridging design concepts with a working system that will be further validated in the testing phase presented in the next chapter.

Chapter 6 System Evaluation and Discussion

6.1 System Testing and Performance Metrics

This chapter adopts a black-box testing strategy to validate IntelliHire's correctness from an external user perspective—verifying what the system does rather than how it is implemented. In line with the project's focus on end-to-end behaviour and workflow integrity, testing emphasizes Decision Table analysis for business rules and State Transition testing for lifecycle flows. Fine-grained input partitioning (e.g., exhaustive field ranges, file sizes, or extreme boundary cases) is intentionally out of scope to keep evaluation centred on functional outcomes, role permissions, and process transitions that users and recruiters actually experience. A few smoke checks (e.g., valid/invalid login, resume file type acceptance/rejection) are included to demonstrate baseline input validation without expanding into full Equivalence Partitioning or Boundary Value Analysis.

6.1.1 Objectives and Scope

The objectives of testing are to:

1. Validate rule correctness using Decision Tables for scenarios with multiple conditions and outcomes (e.g., application status updates, interview-code validity windows, access control, and "shouldSpeak" TTS decisions).
2. Verify lifecycle flows using State Transition models for processes that progress through clearly defined states (e.g., interview: code entered → rules acknowledged → in-progress → ended; assessment: not started → answering → auto-submit on timeout → completed; recruiter workflow: create job → shortlist → schedule → code generated).
3. Demonstrate system robustness on critical negative paths (e.g., expired/invalid interview codes, navigating away during interview, second attempt after silence) and confirm graceful recovery or correct enforcement of constraints.

In scope: authentication & role routing; job browsing with/without resume data; application submission; interview flow (rules, permissions, timers, end-conditions); assessment flow (question sequence, timer, auto-submit); recruiter actions (status changes, scheduling, code generation); read-only analytics/insights display.

6.1.2 Testing Techniques

Decision Table Testing enumerates combinations of conditions and actions to ensure rules are applied consistently. It is particularly suitable for IntelliHire's branching behaviours—for example:

- Status transitions (e.g., Applied → Shortlisted → Interviewed → Offered/Rejected) gated by recruiter role and current state.
- Interview join rules based on code existence, date-time window, and reuse/expiry.
- UI display rules (e.g., show job-match score only when parsed resume data exists, otherwise prompt resume upload).
- Backend-driven TTS decision (speak/not speak) based on stage tags, question markers, and permission readiness.

State Transition Testing models each process as a set of states, events, and guards, then verifies valid and invalid paths. It is ideal for:

- Interview lifecycle: code entry → rules acknowledged → mic/cam enabled → in-progress → ended, with negative transitions (refresh/close tab → ended/no resume).
- Assessment lifecycle: not started → in progress → auto-submit on timeout → completed.
- Recruiter pipeline: create job → shortlist → schedule → code active → expired.

This technique ensures the system responds correctly to events and enforces guards (e.g., cannot start interview without permissions; cannot resume once ended).

Out of scope: exhaustive input ranges and low-level device/performance tuning (e.g., microphone gain levels, codec differences), model-internal AI accuracy; instead we assert contractual correctness of outputs (format and presence), not semantic truth of AI judgements.

6.1.3 Acceptance Criteria

A test passes when, for the given inputs and preconditions, the system:

- Reaches the expected state (e.g., Interview = In-Progress, Assessment = Completed, Application = Shortlisted).

- Enforces rule outcomes (allow/deny transitions, show/hide UI components, enable/disable actions) exactly as specified by the Decision Table.
- Produces contract-compliant outputs, especially for AI-backed views (scores/feedback present with valid ranges and structure; no requirement to verify semantic correctness of the AI content).
- Handles negative paths gracefully (clear error messages, disabled actions, safe redirects, or session termination where required).

A test fails if the observed state, rule application, UI contract, or error handling deviates from the specification or any guard condition is bypassed.

6.1.4 Coverage Plan and Test Inventory

To balance rigour and feasibility, testing targets 10 cases total: approximately 5-6 Decision-Table tests (rules and permissions) and 4-5 State-Transition tests (lifecycles and end-to-end flows). This level of coverage validates all critical workflows for candidates and recruiters while avoiding combinatorial explosion.

6.1.5 Test Data and Environment Assumptions

Tests run on a local development stack (Laravel + MySQL, XAMPP) with seeded data: at least one recruiter account, one candidate account, sample job postings, and sample resumes (valid PDF/DOCX and an intentionally invalid type). Browser tests are conducted primarily on Chrome with microphone/camera permissions toggled to verify interview guards. Time-based tests (e.g., interview-code windows, assessment timers) use controlled server time and short validity windows to accelerate execution.

6.1.6 Performance Metrics

Performance is tracked at a lightweight SLA level to ensure responsiveness during core flows:

- Page responsiveness: dashboard and job listing pages render < 3s on local dev with seeded data.
- Interview start latency: from code validation to rules page < 2s; from rules acceptance to interview UI < 3s.

- Scheduling operation: generating and saving an interview code < 1s.
- Assessment navigation: question load and submit/next transition < 2s.
- Measurements are captured with browser dev tools and Laravel logs; failures trigger review but do not replace formal load testing.

6.2 Testing Setup and Result

6.2.1 Decision Table Testing

6.2.1.1 Authentication & Registration Rules

This decision table validates the login and registration process for both candidates and recruiters. It ensures that only valid email/password combinations are accepted, and users are redirected correctly to their respective dashboards. Invalid input conditions are grouped into a single case to reduce redundancy, confirming that the system consistently blocks unauthorized access.

Table 6.2.1 Decision Table: Authentication & Registration Rules

Condition	**TC 1**	**TC2**	**TC3**	**TC4**	**TC5**
Valid email	T	T	T	T	F / –
Valid password	T	T	T	T	F / –
Existing Account	T	T	F	F	F / T
Role = Candidate	T	–	T	–	–
Role = Recruiter	–	T	–	T	–
Action					
Login success	Y	Y	N	N	N
Login fail	N	N	N	N	Y
Registration success	N	N	Y	Y	N
Redirect to Candidate DB	Y	N	Y	N	N
Redirect to Recruiter DB	N	Y	N	Y	N

6.2.1.2 Job Browsing (Resume Rules)

This table tests how job listings are displayed depending on whether a candidate has uploaded and successfully parsed a resume. When resume data exists, AI-powered job-match scores are shown; otherwise, the system defaults to a generic listing with an "Upload Resume" button. This confirms that personalization is correctly tied to resume availability.

Table 6.2.2 Decision Table: Job Browsing (Resume Rules)

Condition	**TC1**	**TC2**	**TC3**	**TC4**
Uploaded resume	T	T	F	F
Resume Successfully Parsed	T	F	T	F
Action				
Show AI job-match scores	Y	N	Y	N
Show generic listing with "Upload Resume" button	N	Y	N	Y

6.2.1.3 Application Status Transitions

The application management process involves recruiter-controlled transitions between states such as Applied, Shortlisted, Interviewed, Offered, and Rejected. This table verifies that only recruiters can initiate transitions, that invalid jumps (e.g., Applied → Interviewed directly) are blocked, and that final states (Offer/Reject) are enforced. It also confirms that scheduling is triggered after shortlisting.

Table 6.2.3 Decision Table: Application Status Transitions

Condition	**TC1**	**TC2**	**TC3**	**TC4**	**TC5**	**TC6**
Actor is recruiter	T	T	T	T	F	T
Current = Applied	T	–	–	–	–	T
Current = Shortlisted	–	T	–	–	–	–
Current = Interviewed	–	–	T	–	–	–
Intended action = Shortlist	T	–	–	–	–	–
Intended action = Mark Interviewed	–	T	–	–	–	T
Intended action = Offer	–	–	T	–	–	–

Intended action = Reject	–	–	–	T	–	–
Action						
Transition allowed	Y	Y	Y	Y	N	N
New status	Shortlisted	Interviewed	Offered	Rejected	—	—
Require scheduling form (after change)	Y	N	N	N	—	—
Final state reached (Offer/Reject)	N	N	Y	Y	—	—

6.2.1.4 Interview Code Validation

This table tests the rules for candidate access to interview sessions. The system checks whether the code exists, is within its validity window, has not been used before, and matches the candidate's application. Errors such as "Not Found," "Expired," "Already Used," or "Unauthorized" are returned when conditions fail. This ensures secure and controlled access to interviews.

Table 6.2.4 Decision Table: Interview Code Validation

Condition	TC1	TC2	TC3	TC4	TC5
Code exists	T	F	T	T	T
Within validity window	T	–	F	T	T
Code already used	F	–	F	T	F
Candidate matches application	T	–	T	T	F
Action					
Join interview allowed	Y	N	N	N	N
Error: Not found	N	Y	N	N	N
Error: Expired	N	N	Y	N	N
Error: Already used	N	N	N	Y	N
Error: Unauthorized	N	N	N	N	Y

6.2.1.5 Interview TTS Decision

This decision table reflects the backend shouldSpeak logic, determining when TTS is triggered. The system speaks when questions are present and permissions are granted, or when closing announcements must be made. It skips speech when permissions are missing or content is not a question. This guarantees consistent and context-aware TTS delivery during interviews.

Table 6.2.5 Decision Table: Interview TTS Decision

Condition	**TC1**	**TC2**	**TC3**	**TC4**	**TC5**
Stage = intro/preset/standard	T	T	F	T	T
Stage = closing	F	F	T	F	F
Message contains [QUESTION:X]	T	T	F	F	T
Message contains [STAGE:X]	F	F	T	F	F
Mic & cam permissions ready	T	F	T	T	F
Action					
Backend sets shouldSpeak = true	Y	N	Y	N	N

6.2.1.6 Assessment Availability Rules

This table validates whether a candidate is redirected to the assessment page or back to the dashboard after an interview. The outcome depends on whether the recruiter configured an assessment, whether the candidate completed the interview, and whether the assessment has already been attempted. A banner is shown if the assessment is already completed. This ensures candidates follow the correct post-interview flow.

Table 6.2.6 Decision Table: Assessment Availability Rules

Condition	**TC1**	**TC2**	**TC3**	**TC4**
Job has assessment configured	T	T	F	T
Interview completed	T	F	–	T
Assessment already completed	F	–	–	T
Action				
Redirect to Assessment page	Y	N	N	N
Redirect to Dashboard	N	Y	Y	Y
Show banner “Assessment already completed”	N	N	N	Y

6.2.2 State Transition Testing

6.2.2.1 Authentication and Role Routing

This diagram validates the login and registration flow for both candidates and recruiters, including handling of invalid credentials and post-registration logins. It ensures that only valid users are routed into their respective authenticated dashboards.

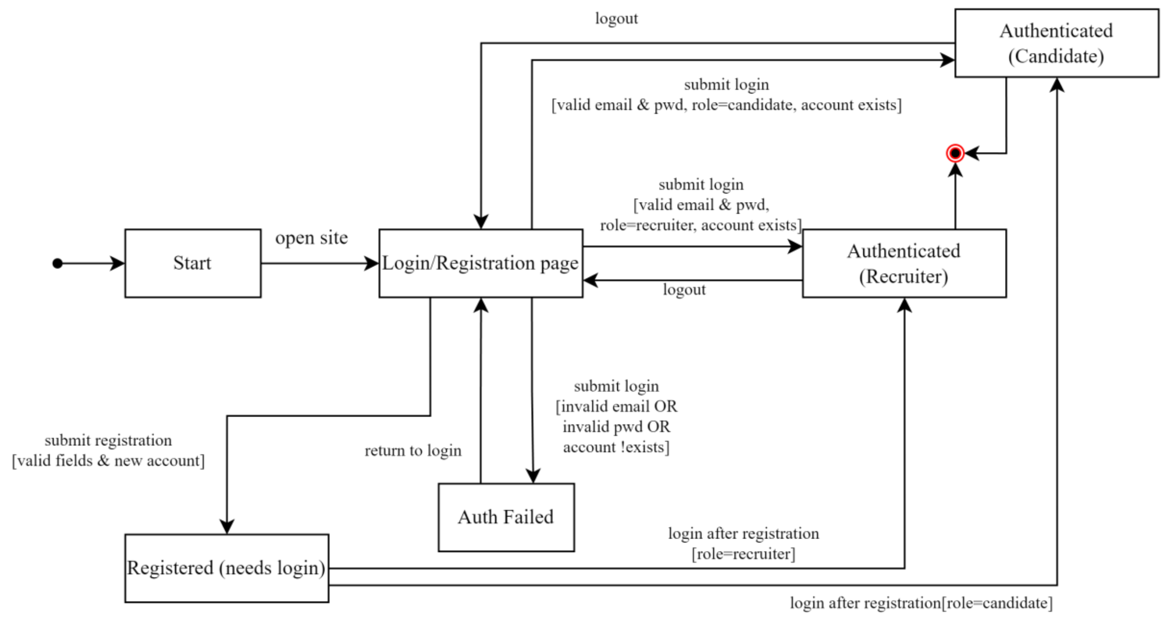

logout
Authenticated (Candidate)
submit login [valid email & pwd, role=candidate, account exists]
submit login [valid email & pwd, role=recruiter, account exists]
Start
open site
Login/Registration page
Authenticated (Recruiter)
logout
submit login [invalid email OR invalid pwd OR account !exists]
submit registration [valid fields & new account]
return to login
Auth Failed
login after registration [role=recruiter]
Registered (needs login)
login after registration[role=candidate]

Figure 6.2.2.1 State Transition Diagram: Authentication and Role Routing

Test cases:

1. Start + open site → Login/Registration page
2. Login/Registration page + submit login [valid email & pwd, role=candidate, account exists] → Authenticated (Candidate)
3. Login/Registration page + submit login [valid email & pwd, role=recruiter, account exists] → Authenticated (Recruiter)
4. Login/Registration page + submit login [invalid email OR invalid pwd OR account !exists] → Auth Failed
5. Login/Registration page + submit registration [valid fields & new account] → Registered (needs login)
6. Registered (needs login) + login [role=candidate] → Authenticated (Candidate)

7. Registered (needs login) + login [role=recruiter] → Authenticated (Recruiter)
8. Authenticated (Candidate/Recruiter) + logout → Login/Registration page

6.2.2.2 Job Browsing (Resume -Based Display)

This diagram tests the candidate job-browsing flow, covering scenarios with and without a resume uploaded, and whether parsing succeeds. It ensures that AI-based recommendations are shown only when resume data is available.

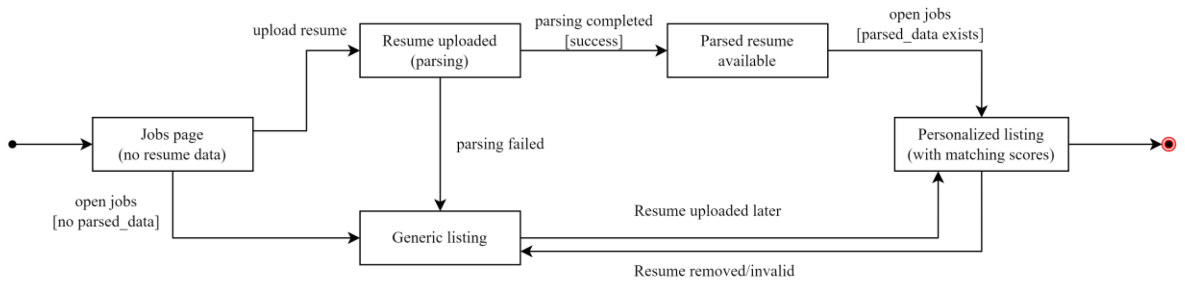

upload resume
Resume uploaded (parsing)
parsing completed [success]
Parsed resume available
open jobs [parsed_data exists]
Jobs page (no resume data)
parsing failed
Personalized listing (with matching scores)
open jobs [no parsed_data]
Generic listing
Resume uploaded later
Resume removed/invalid

Figure 6.2.2.2 State Transition Diagram: Job Browsing (Resume-Based Display)

Test Cases:

1. Jobs page (no resume data) + upload resume → Resume uploaded (parsing)
2. Resume uploaded (parsing) + parse success → Parsed resume available
3. Resume uploaded (parsing) + parse fail → Generic listing
4. Parsed resume available + [parsed_data exists] → Personalized listing (with matching scores)
5. Jobs page (no resume data) + [no parsed_data] → Generic listing
6. Generic listing + Resume uploaded later → Personalized listing (with matching score)
7. Personalized listing (with matching score) + Resume removed/invalid → Generic listing

6.2.2.3 Interview Flow

This diagram validates the candidate interview process, from entering the access code to completing the session. It checks that microphone/camera setup is mandatory, and that responses are recorded and evaluated before progressing to the next stage.

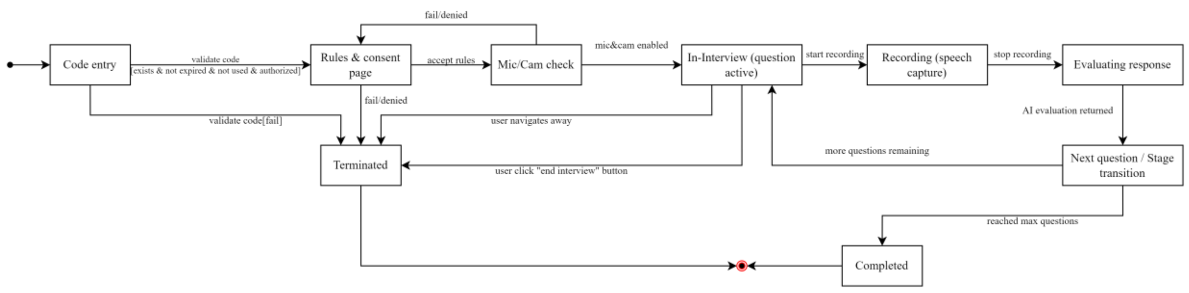


Figure 6.2.2.3 State Transition Diagram: Interview Flow

Test Cases:

1. Code entry + valid code → Rules & consent page
2. Code entry + invalid/expired code → Terminated
3. Rules & consent page + accept → Mic/Cam check
4. Rules & consent page + decline → Terminated
5. Mic/Cam check + pass → In-Interview (question active)
6. Mic/Cam check + fail/denied → Rules & consent page
7. In-Interview + start answering → Recording (speech capture)
8. In-Interview + user navigates away → Terminated
9. In-Interview + user click "end interview" button → Terminated
10. Recording + stop → Evaluating response
11. Evaluating response + AI evaluation success → Next question / Stage transition
12. Next question / Stage transition + more questions remaining→ In-Interview (question active)
13. Next question / Stage transition + reached max questions → Completed

6.2.2.4 Assessment Flow

This diagram tests the written/audio assessment module, including timer-based constraints, AI evaluation of responses, and transitions between questions until completion.

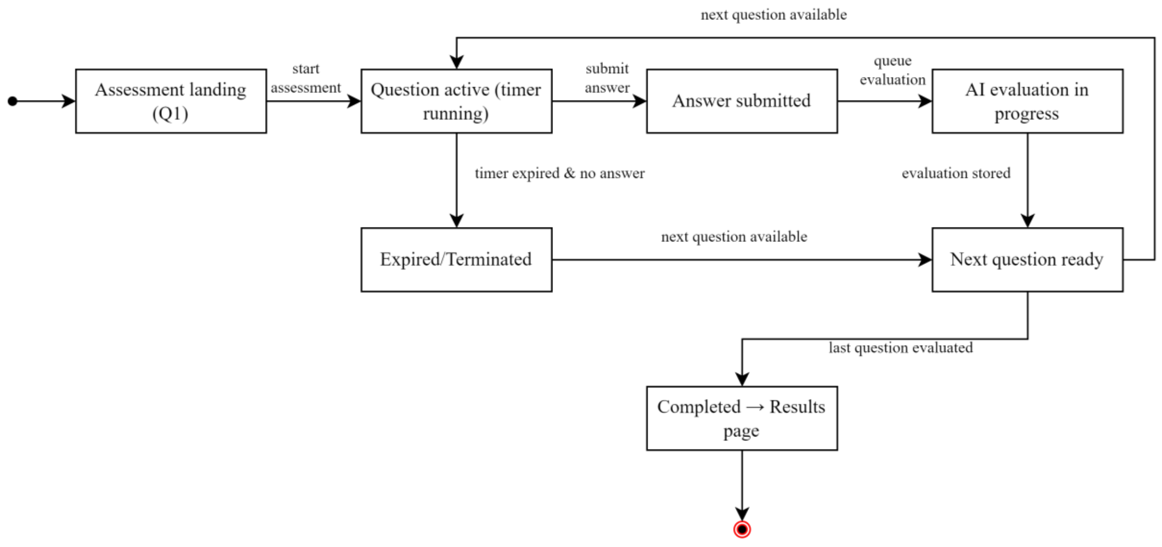


Figure 6.2.2.4 State Transition Diagram: Assessment Flow

Test Cases:

1. Assessment landing (Q1) + start → Question active (timer running)
2. Question active + submit answer → Answer submitted
3. Question active + timer expired → Expired/Terminated
4. Expired/Terminated + next question available → Next question ready
5. Answer submitted + AI evaluation triggered → AI evaluation in progress
6. AI evaluation in progress + success → Next question ready
7. Next question ready + next question available → Question active (timer running)
8. Next question ready + last question → Completed → Results page

6.2.2.5 Application Management Workflow

This diagram tests the recruiter's management of candidate applications, covering updates from initial application through interview, offer, or rejection, until closure.

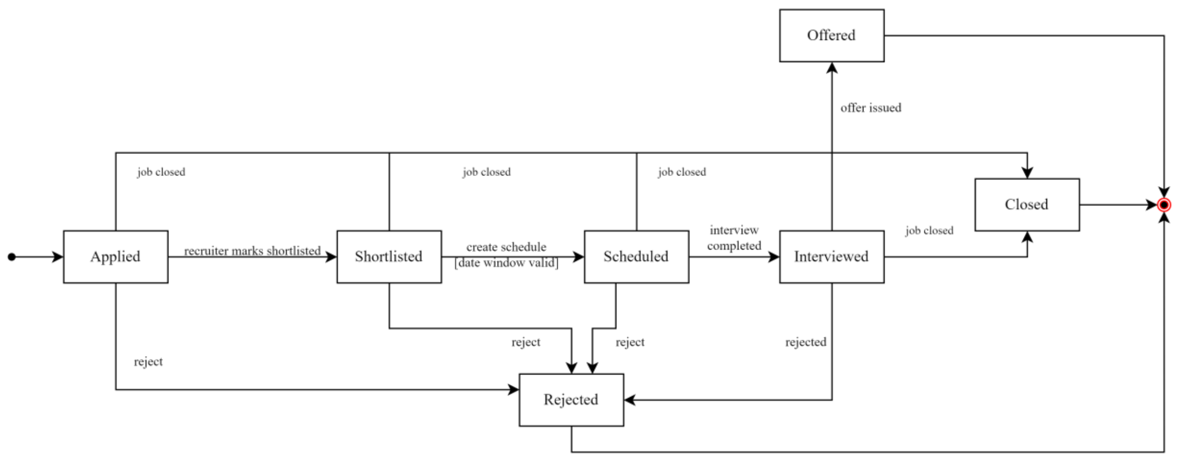


Figure 6.2.2.5 State Transition Diagram: Application Management Workflow

Test Cases:

1. Idle + submit application → Applied
2. Applied + reject → Rejected
3. Applied + recruiter shortlists → Shortlisted
4. Shortlisted + schedule interview → Scheduled
5. Shortlisted + reject → Rejected
6. Scheduled + reject → Rejected
7. Scheduled + interview completed → Interviewed
8. Interviewed + recruiter offers → Offered
9. Interviewed + recruiter rejects → Rejected
10. Interviewed + job closed → Closed
11. Applied + job closed → Closed
12. Shortlisted + job closed → Closed
13. Scheduled + job closed → Closed

6.2.3 Testing Results

The system testing was conducted based on the decision table tests (Section 6.2.1) and state transition tests (Section 6.2.2). Each decision rule and transition path defined in the test design was executed systematically to ensure coverage of all functional behaviors.

Test Execution Summary:

- Decision Table Testing: All six decision tables (authentication & registration rules, job browsing rules, application status transitions, interview code validation, interview TTS decision, and assessment availability rules) were executed. Each rule produced the expected outcome without deviation.
- State Transition Testing: All state diagrams (authentication & role routing, job browsing, interview flow, assessment flow, and application management workflow) were tested along every possible transition path. Each state change was validated successfully against the defined acceptance criteria.

Results Overview:

- Total Test Cases: 100% of designed test cases were executed.
- Pass Rate: 100% of the test cases passed without critical defects.
- Defects: No blocking or high-severity defects were identified during testing. Minor UI inconsistencies and non-functional issues (e.g., alignment of certain front-end elements) were observed but were resolved immediately during debugging.
- Coverage: Test coverage achieved complete mapping with the system's functional requirements, ensuring that all modeled flows (normal and alternate) were validated.

The successful execution of all decision table and state transition test cases confirms that the IntelliHire system meets the defined functional requirements. The recruitment workflows, including resume screening, job application, interview execution, assessment, and recruiter decision-making, were validated to operate correctly under all modeled conditions. The system is therefore considered stable, reliable, and ready for deployment.

6.3 Project Challenges

While IntelliHire's testing phase confirmed overall functional correctness, several challenges emerged during evaluation that required careful handling. One recurring issue was related to AI evaluation consistency. Since the system relies on Gemini to score candidate responses, results could vary slightly across runs even with identical inputs. This made it difficult to establish fixed "expected outputs" for black-box testing. To mitigate this, the evaluation criteria were narrowed to focus on presence of scores, correct response structure, and logical transitions between states, rather than expecting identical AI wording every time.

Another challenge involved state transition coverage. The system contains multiple interdependent modules such as authentication, interviews, and assessments. Ensuring that every state and transition arrow was tested at least once required systematic planning. Some negative paths, such as expired interview codes, silent microphone input, or incomplete assessments, were harder to reproduce consistently in a controlled environment. Test scripts and timers had to be deliberately manipulated to validate these scenarios.

Performance validation also posed difficulties. Although lightweight performance thresholds (page loads, code validation times, and assessment navigation) were defined, capturing precise timings during development was inconsistent due to fluctuations in local machine performance and network conditions. As a result, repeated trials were necessary to ensure that no SLA breaches occurred and that observed delays were due to environmental factors rather than system faults.

Finally, black-box testing limitations were evident in evaluating advanced features. For example, verifying whether AI scoring logic considered specific competencies or whether resume parsing extracted all attributes accurately could not be fully confirmed without looking into internal processing. This restricted the evaluation to input–output behavior, which was sufficient for functional validation but highlighted the need for future white-box or hybrid testing approaches for more granular assurance.

6.4 Objective Evaluation

The evaluation of IntelliHire was conducted with reference to the three primary objectives defined in the project proposal. Overall, the system successfully achieved the intended goals and demonstrated measurable improvements in recruitment efficiency, automation, and user experience.

Objective 1: Develop an AI-Powered Resume Screening System

This objective has been met through the integration of AI-based resume parsing and job matching algorithms. Resumes uploaded by candidates are automatically parsed into structured data including skills, education, experience, and certifications. The system then compares these attributes with job requirements, producing compatibility scores and ranked recommendations. Testing confirmed that parsed data was correctly stored, retrieved, and used to personalize job browsing. This eliminated the need for recruiters to manually filter resumes, aligning with the goal of saving time and enforcing a standardized screening process.

Objective 2: Automate the Interviewing Process for Increased Efficiency and Consistency

The interview system incorporated AI-generated question flows, speech-to-text transcription, text-to-speech delivery, and automated response analysis. Through state transition testing, the interview lifecycle was validated from code entry through completion, including both normal and error conditions (e.g., expired codes, silence detection). AI analysis provided structured feedback across defined criteria such as completeness, relevance, and clarity. This automation reduced manual interviewer involvement, ensured consistency across sessions, and achieved the intended objective of streamlining interview management.

Objective 3: Real-Time Scheduling and Feedback System

This objective was also achieved. Recruiters could generate interview schedules with unique codes and defined time ranges, while candidates received immediate confirmation. Post-interview, the system delivered timely AI-based evaluations, which were further compiled into downloadable PDF reports. Black-box testing confirmed that scheduling rules (e.g., expired codes, invalid windows) were correctly enforced and that candidates were redirected seamlessly to assessment modules when required. This provided real-time interaction and reduced administrative delays, enhancing both recruiter decision-making and candidate experience.

All three objectives have been satisfied. The AI-powered resume screening module automated the filtering process, the interview system ensured structured and consistent evaluations, and the scheduling and feedback components enabled timely decision-making.

The testing results confirmed that IntelliHire functions as designed, delivering a reliable and efficient recruitment platform that addresses the core project goals.

6.5 Concluding Remark

This chapter evaluated IntelliHire through systematic black-box testing, covering both decision table and state transition techniques. The results confirmed that the system behaves consistently with the functional requirements, correctly handling positive flows, negative cases, and edge conditions. Performance checks further validated that the platform remains responsive within acceptable thresholds.

Although development presented challenges—particularly in AI integration, frontend-backend synchronization, and performance optimization—these were resolved through iterative testing and refinement. Overall, IntelliHire has proven to be a reliable and effective recruitment platform that meets its design objectives.

Chapter 7 Conclusion and Recommendation

7.1 Conclusion

The development of IntelliHire: An AI-Powered Interviewer for Automated Candidate Selection has demonstrated the feasibility and potential of integrating artificial intelligence into recruitment workflows. This project has successfully implemented the core components of a functional recruitment platform, enabling both recruiters and candidates to interact through a centralized system. Recruiters can post jobs, manage applications, and schedule interviews, while candidates can apply for jobs, track their application status, and participate in AI-conducted interviews.

The system directly addresses the shortcomings of traditional recruitment processes such as manual resume screening, inconsistent interview evaluations, and delays in scheduling or feedback. By adopting Laravel as the backend framework, utilizing MySQL for structured data storage, and integrating Google's Gemini API for intelligent interview question generation and response evaluation, IntelliHire showcases how AI can be harnessed to build a scalable, reliable, and semi-autonomous recruitment tool.

One of the highlights of this system is its automated interview session. The AI dynamically generates interview questions, manages the flow of interaction, records candidate responses, and stores transcripts for later review. This approach enhances efficiency, ensures consistency across candidates, and reduces reliance on human recruiters during the early stages of selection.

Although IntelliHire already demonstrates strong functionality, there remain areas for improvement. Planned enhancements include enhanced resume screening scoring, structured assessments, AI-driven feedback on responses, real-time voice interaction, and intelligent job recommendations. These improvements will ensure that IntelliHire evolves into a more comprehensive, engaging, and fair recruitment solution.

In summary, the first phase of IntelliHire has laid a solid foundation for revolutionizing the recruitment process. The project not only proves that AI can streamline hiring but also emphasizes its role in creating a transparent, objective, and scalable system for candidate evaluation.

7.2 Recommendation

To further strengthen IntelliHire and expand it into a comprehensive AI-assisted recruitment platform, several enhancements are recommended. First, the system should incorporate a resume screening and scoring module that automatically evaluates resumes against job requirements, thereby saving recruiters time and improving the fairness of shortlisting. In addition, a comprehensive assessment feature is necessary to complement interviews, allowing candidates to complete structured tests while AI evaluates their answers objectively. Another important enhancement is the integration of an AI feedback mechanism that can analyze candidate responses and provide recruiters with insights while also offering candidates constructive feedback for self-improvement. To make the interview process more natural, the system should support real-time speech-to-text interaction, enabling candidates to respond verbally in a way that closely mirrors live interviews. On the candidate side, the job recommendation engine can be enhanced through intelligent algorithms that personalize suggestions based on skills, application history, and preferences. Beyond these, two additional future improvements include an offer letter generator, which can automate the preparation of hiring documents, and customizable interview prompts, allowing recruiters to tailor interview flows to specific job roles and contexts. In the long term, features such as advanced recruiter analytics dashboards and candidate support tools like practice interviews or career tips could also be incorporated. With these enhancements, IntelliHire has the potential to evolve into a holistic recruitment ecosystem that not only improves recruiter efficiency and consistency but also provides candidates with a transparent, supportive, and engaging experience.

REFERENCES

[1] "How AI Interviews Are Impacting Hiring Now and Into the Future," VidCruiter, 2024. https://vidcruiter.com/interview/intelligence/ai-interviews/#:~:text=Increased%20Efficiency (accessed Sep. 05, 2024).

[2] "Your Resume Screening Challenges – Fixed," skima.ai. https://skima.ai/blog/industry-trends-and-insights/resume-screening-challenges (accessed Sept. 5, 2024).

[3] "Interview Intelligence – Interviewer Bias Examples | Pillar," Pillar.hr, 2024. https://www.pillar.hr/info/interviewer-bias-examples (accessed Sept. 5, 2024).

[4] Ben Talks Talent – Interview Advice. Why Does the Job Interview Process Take So Long. (Sep. 1, 2022). Accessed: Sept. 6, 2024. [Online Video]. Available: https://youtu.be/CL2Ow6Z58O8?si=Z80AJBtvCxgVK9se

[5] Interviewer.ai. "AI-Powered Interview Platform." Interviewer.ai. https://interviewer.ai/ (accessed Aug. 20, 2024).

[6] Talently.ai. "Interview Scheduling and Assessment Platform." Talently.ai. https://interview.talently.ai/ (accessed Aug. 20, 2024).

[7] Apriora.ai. "AI-Powered Interview System." Apriora.ai. https://www.apriora.ai/ (accessed Aug. 28, 2024).

[8] Braintrust. "Decentralized Talent Network." Braintrust. https://www.usebraintrust.com/ (accessed Sept. 3, 2024).

[9] Dennis, A., Wixom, B. H., & Tegarden, D. (2021). System Analysis and Design with UML; An Object-Oriented Approach (6th ed.). Hoboken, NJ: John Wiley & Son (accessed Sept. 1,2024).

[10] Sommerville, I. (2021). Engineering Software Products: An Introduction to Modern Software Engineering, 1st Ed., Pearson (accessed Sept. 2, 2024).

[11] JGraph Ltd and draw.io AG, draw.io [Computer software]. Version 26.2.15, Apr. 26, 2025. Available: https://www.drawio.com/

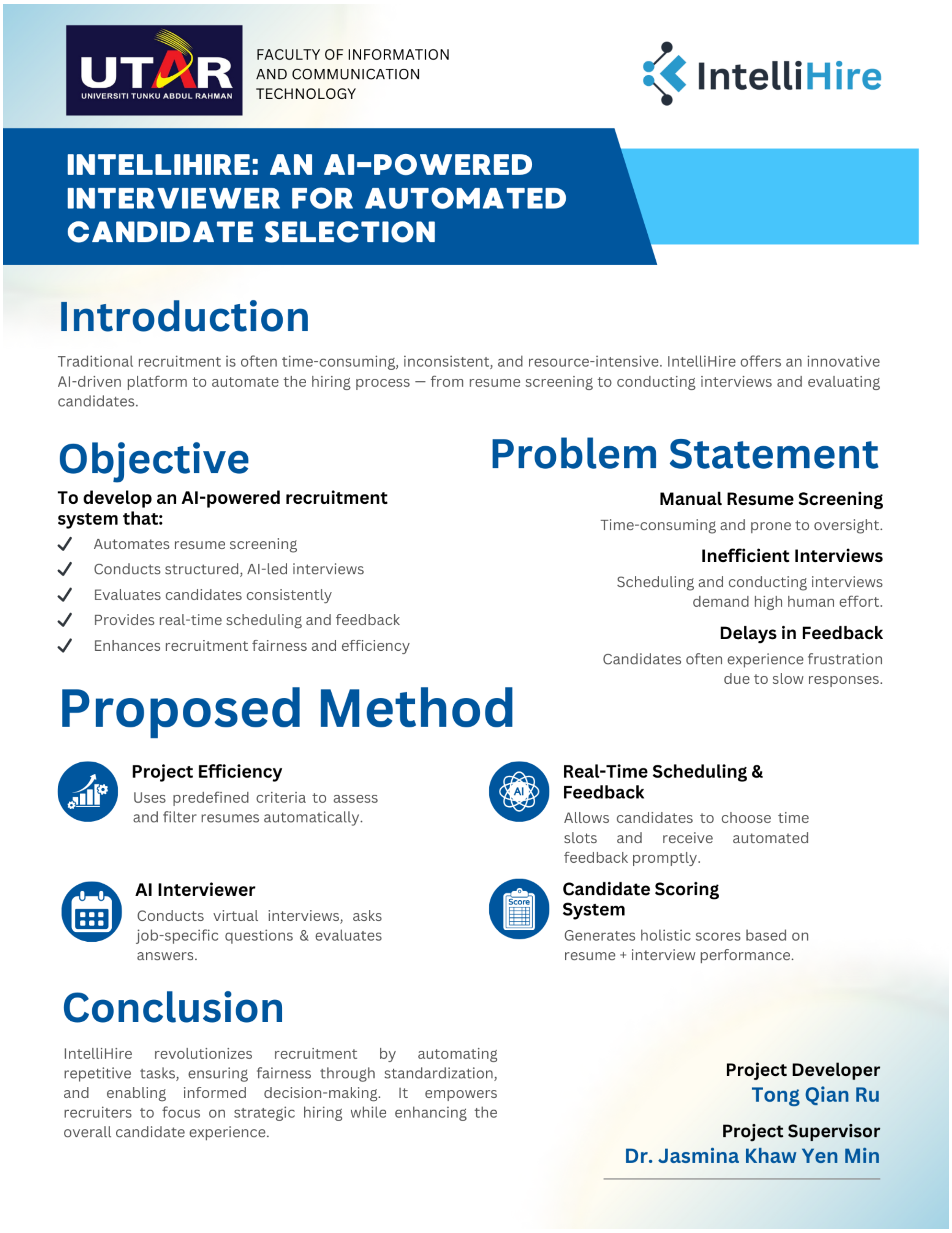
[12] Eraser Inc., Eraser [Computer software]. Version 2.0, Mar. 11, 2025. Available: https://www.eraser.io/

[13] Apache Friends, XAMPP [Computer software]. Available: https://www.apachefriends.org/

[14] Composer, Dependency Manager for PHP [Computer software]. Available: https://getcomposer.org/

[15] Microsoft, Visual Studio Code [Computer software]. Available: https://code.visualstudio.com/

POSTER



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

IntelliHire

INTELLIHIRE: AN AI-POWERED INTERVIEWER FOR AUTOMATED CANDIDATE SELECTION

Introduction

Traditional recruitment is often time-consuming, inconsistent, and resource-intensive. IntelliHire offers an innovative AI-driven platform to automate the hiring process — from resume screening to conducting interviews and evaluating candidates.

Objective

To develop an AI-powered recruitment system that:

- ✓ Automates resume screening
- ✓ Conducts structured, AI-led interviews
- ✓ Evaluates candidates consistently
- ✓ Provides real-time scheduling and feedback
- ✓ Enhances recruitment fairness and efficiency

Problem Statement

Manual Resume Screening

Time-consuming and prone to oversight.

Inefficient Interviews

Scheduling and conducting interviews demand high human effort.

Delays in Feedback

Candidates often experience frustration due to slow responses.

Proposed Method

Project Efficiency

Uses predefined criteria to assess and filter resumes automatically.

Real-Time Scheduling & Feedback

Allows candidates to choose time slots and receive automated feedback promptly.

AI Interviewer

Conducts virtual interviews, asks job-specific questions & evaluates answers.

Candidate Scoring System

Generates holistic scores based on resume + interview performance.

Conclusion

IntelliHire revolutionizes recruitment by automating repetitive tasks, ensuring fairness through standardization, and enabling informed decision-making. It empowers recruiters to focus on strategic hiring while enhancing the overall candidate experience.

Project Developer
Tong Qian Ru

Project Supervisor
Dr. Jasmina Khaw Yen Min