People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research Université
M'hamed Bougara University - Boumerdès

Faculty of Sciences
Department of Computer Science

Domaine : Mathematics Computer Science

Sector : Computer Science

Specialty : Systèmes d'informatique

N° of the authorization decree for the specialty: decree No. 872 of 26/07/2016

Final thesis in view of obtaining the Academic Bachelor's Degree

Thème

ATS CV ADVISOR

Presented by:
AHMED NACER AYOUB
MENIA ABDELHAMID
SIROUKANE YAHIA

Defended on 03/06/ 2026 Before the jury composed of

Mr.Boulif Menouar : Examinator
Mr.Ahmed Nacer Masoud : Framer

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Acknowledgments

First and foremost, we express our profound gratitude to Allah Almighty for granting us the strength, patience, and grace to complete this work.

We extend our deepest and most heartfelt thanks to our beloved parents. Their unwavering moral support, endless encouragement, and sacrifices have been our greatest source of inspiration throughout this journey.

A special note of appreciation goes to our esteemed supervisor, Mr. Ahmed Nacer Masoud. We are deeply grateful for his trust, invaluable guidance, continuous availability, and the profound wisdom he shared with us throughout the development of this thesis.

Finally, we would like to express our sincere gratitude to everyone who, whether directly or indirectly, contributed to the success of this modest endeavor with their help and encouragement.

Table of Contents

General Introduction 5
CHAPTER 1: GENERAL CONTEXT AND SPECIFICATION 6
1.1 Introduction 6
1.2 Problem Statement & Critique of the Existing System 6
1.3 Comparative Analysis of Existing Solutions 6
1.4 Proposed Solution & Project Objectives 7
1.5 Conclusion 7
Chapter 2: Requirements Specification and State of the Art 8
2.1 Introduction 8
2.2 Functional Requirements 8
2.2.1 User Management and Security 8
2.2.2 Interactive Resume Builder 8
2.2.3 AI-Powered ATS Analysis 8
2.2.4 Dynamic Voice Interview Coach 8
2.2.5 Automated Cover Letter Generation 9
2.3 Non-Functional Requirements 9
2.3.1 Performance and Reliability 9
2.3.2 Security and Data Integrity 9
2.3.3 Usability and Localization 9
2.4 Conclusion 10
Chapter 3: Analysis and Design 11
3.1 Introduction 11
3.2 Use Case Diagram 11
3.3 Sequence Diagrams 12
3.3.1 User Authentication Sequence Diagrams 12
3.3.2 AI-Powered Resume Analysis Sequence Diagram 16
3.3.3 Voice Interview Execution Sequence Diagram 16
3.3.4 Cover Latter Generator Sequence Diagram 18
3.4 Classes Diagram 19
3.5 Conclusion 19
Chapter 4: Implementation and Experimental Results 20
4.1 Introduction 20
4.2 Development Environment 20
4.3 Technology Stack 20
4.3.1 Backend (Server-Side) 20
4.3.2 Frontend (Client-Side) 21

4.3.3 External APIs and Core Engines 21
4.4 Application Interfaces (Results) 21
4.4.1 Authentication and Access 21
4.4.2 The Main Analysis Workspace 22
4.4.3 AI Interactive Feedback Report 23
4.4.4 In-App Resume Builder (Iterative Sandbox) 24
4.4.5 Cover Latter Generator 25
4.4.6 Voice Interview Coach 26
4.5 Conclusion 27
General Conclusion 28

General Introduction

In the modern digital era, the recruitment landscape has undergone a massive transformation. To manage the influx of thousands of applications, organizations worldwide heavily rely on automated systems known as Applicant Tracking Systems (ATS). These systems filter, rank, and score resumes based on specific keywords, structural formatting, and algorithmic relevance before a human recruiter ever reviews them. While this solves a major administrative challenge for employers, it creates an invisible barrier for highly qualified job seekers who fail to optimize their resumes according to these hidden algorithmic rules.

This systemic friction highlights an urgent need for intelligent, accessible tools that empower candidates to evaluate and bridge the gap between their professional profiles and modern automated hiring standards.

To address these modern challenges, we propose the design and development of ATS-CV-Advisor, an end-to-end AI-powered career preparation platform. Our system centralizes the preparation process by offering a unified ecosystem featuring: an interactive resume builder, an AI-driven ATS analyzer (powered by Google Gemini), a tailored cover letter generator, and a dynamic voice-based mock interview coach (utilizing Qwen, Whisper, and Librosa).

To provide a clear and organized roadmap of this work, the thesis is structured into four core chapters as follows:

- Chapter 1: General Context and Specification, which establishes the foundational problem statement, the project's objectives, and the functional/non-functional system requirements.

- Chapter 2: State of the Art and Technology Stack, which explores the theoretical definitions of ATS and Large Language Models (LLMs), alongside a justified presentation of the development technologies.

- Chapter 3: System Analysis and Design, which translates the requirements into conceptual blueprints using Unified Modeling Language (UML) diagrams and database entity schemas.

- Chapter 4: Implementation and Results, which showcases the physical deployment environments, code integration, and screenshots of the finalized graphical user interfaces.

CHAPTER 1: GENERAL CONTEXT AND SPECIFICATION

1.1 Introduction

Every engineering project must begin with a comprehensive understanding of its operating context and a precise boundary definition. In this chapter, we will establish the foundational groundwork for the "ATS CV Advisor" project. We will analyze the current state of the market, identify its limitations through a comparative study, detail our proposed solution, and present the Agile methodology adopted to realize this work effectively.

1.2 Problem Statement & Critique of the Existing System

Currently, job seekers rely heavily on isolated, static, and often outdated tools to prepare for their job applications. Despite possessing the required qualifications, many are automatically rejected by Applicant Tracking Systems (ATS). Through our analysis, we identified several critical shortcomings that hinder a candidate's success:

1. **Formatting Issues:** Complex layouts, graphics, and tables created in external design tools (like Canva or Zety) often cause parsing errors. Text is frequently embedded as images, leading to "invisible" text for the ATS.
2. **Content and Keyword Mismatch:** Candidates often fail to align their resume phrasing with the exact keywords present in the job description. Existing builders are "blind" and do not offer real-time keyword gap analysis.
3. **Static Interview Preparation:** Practicing for interviews by reading generic answers does not simulate the psychological pressure of a real interview. There is a profound lack of accessible tools capable of analyzing a candidate's actual voice and spoken content.
4. **Fragmented Ecosystem:** A user currently has to use three or four different platforms to build a resume, check its ATS score, write a cover letter, and practice for an interview.

1.3 Comparative Analysis of Existing Solutions

In the current market, platforms like Jobscan focus purely on evaluation but lack in-app editing, while Novorésumé offers attractive templates that fail ATS parsing. To better situate our project, we conducted a comparative study:

Feature	Canva/Standard Builders	Jobscan	ATS-CV-ADVISOR (Proposed)
Document Generation	Highly Graphical / Static	None	Dynamic & ATS Optimized
Job Description Matching	No	Yes	Yes (Gemini AI API)
Voice Interview Coaching	No	No	Yes (Whisper, Qwen, Librosa)
Multilingual Interface	Manual	English mostly	Yes (EN, FR, AR, with RTL)

1.4 Proposed Solution & Project Objectives

To resolve the aforementioned inadequacies, we propose the design and development of ATS-CV-Advisor, an intelligent, decoupled web application. The system is built around four core intelligent modules:

- **Interactive Resume Builder:** A dynamic WYSIWYG editor that generates clean, ATS-compliant resumes directly from raw text inputs.
- **AI Resume Analyzer:** Leveraging the Google Gemini API, this tool evaluates the resume against a specific job description, providing a predicted match percentage and highlighting missing keywords.
- **Dynamic Voice Interview Coach:** An advanced mock interview system utilizing OpenAI's Whisper (transcription), Librosa (acoustic analysis), and the Qwen LLM to evaluate spoken content against the STAR method.
- **Cover Letter Generator:** An automated engine hosted via Google Colab to draft tailored, highly contextual cover letters based on the candidate's optimal resume.

1.5 Conclusion

Having defined "what" the system needs to do in terms of resolving the fragmented recruitment ecosystem, we can now pivot to the technical specifications. The next chapter will focus on translating these objectives into specific functional and non-functional requirements and describing the system's operational workflows.

Chapter 2: Requirements Specification and State of the Art

2.1 Introduction

The objective of this chapter is to clearly define the perimeter and technical scope of the "ATS-CV-Advisor" application. Before initiating the development phase, it is imperative to identify and document exactly what the system must do and the constraints under which it must operate. Therefore, this chapter is dedicated to detailing the functional requirements, which outline the system's core capabilities, and the non-functional requirements, which dictate the system's quality, security, and performance standards.

2.2 Functional Requirements

Functional requirements define the specific behaviors, responses, and features the system must execute to satisfy the users' needs. For our platform, these are categorized into the following core modules:

2.2.1 User Management and Security

- The system must allow users to register a new account securely using standard email credentials or via Single Sign-On (SSO) utilizing Google OAuth.
- The system must authenticate users and manage active sessions using local email/password credentials via Laravel Sanctum.
- The system must allow users to manage their profiles, update personal information, and securely reset forgotten passwords through email verification.

2.2.2 Interactive Resume Builder

- The system must provide a dynamic WYSIWYG (What You See Is What You Get) interface enabling users to build and format their resumes directly within the application.
- The system must allow users to save, update, and manage multiple text-focused resume templates tied to their individual accounts.
- The system must be capable of exporting the generated templates into clean, ATS-friendly formats.

2.2.3 AI-Powered ATS Analysis

- The system must accept resume data either via file uploads (PDF, PNG, JPEG) or by selecting a previously created in-app template.
- The system must accept raw text input representing the target job description.
- The system must process both the resume and the job description through the Gemini API to generate a detailed evaluation report. This report must include a match percentage, identify missing keywords, and highlight content strengths and weaknesses.

2.2.4 Dynamic Voice Interview Coach

- The system must be able to request browser microphone permissions and record user audio responses.

- The system must process the recorded audio payload to generate real-time textual transcriptions (using Whisper).
- The system must calculate acoustic and mathematical metrics from the audio, such as Words Per Minute (WPM) and pause counts (using Librosa).
- The system must evaluate the transcribed response against the STAR method (Situation, Task, Action, Result) and provide constructive feedback (using Qwen LLM).

2.2.5 Automated Cover Letter Generation

- The system must transmit the user's highest-scoring resume data and the target job description to a dedicated Colab endpoint.
- The system must generate and return a highly tailored, localized, and contextually relevant cover letter.

2.3 Non-Functional Requirements

Non-functional requirements specify the system's quality attributes, technological constraints, and user experience standards necessary to ensure a robust deployment:

2.3.1 Performance and Reliability

- **Decoupled Architecture:** The system must be built using a strict decoupled architecture, separating the React/Next.js frontend from the Laravel backend, to ensure rapid rendering and seamless API communication.
- **Timeout Management:** Given the computationally heavy nature of AI processing, the backend must implement strict timeout safety measures (e.g., a 120-second timeout configuration) to prevent silent gateway failures during LLM evaluation.
- **UI Responsiveness:** The user interface must display loading skeletons and progress indicators to reassure users during extended AI processing times.

2.3.2 Security and Data Integrity

- **API Protection:** All backend API endpoints must be strictly protected via token-based authentication (Laravel Sanctum).
- **Environment Variables:** Sensitive API keys (such as the Gemini API key) must be securely stored in backend environment variables and remain completely inaccessible to the client side.
- **Input Validation:** Uploaded files must undergo rigorous backend validation to check MIME types and enforce a maximum file size limit (e.g., 4MB) to prevent malicious injections or server overloads.
- **Data Privacy:** To optimize storage and ensure privacy, the system must not retain uploaded files after processing; only the extracted AI analysis results and user-generated text templates are stored in the database.

2.3.3 Usability and Localization

- **Multilingual Support:** The platform must be fully multilingual, providing its interface and AI analysis outputs in three languages: English, French, and Arabic.

- **RTL Compatibility:** To fully support the Arabic language, the UI must dynamically load language dictionaries and offer native support for Right-To-Left (RTL) CSS layouts.
- **Theming:** The interface must support seamless switching between Dark Mode and Light Mode to enhance user accessibility.

2.4 Conclusion

This chapter has thoroughly documented the technical specifications of the ATS-CV-Advisor platform. By clearly separating the system's functional capabilities—spanning AI analysis, voice coaching, and document generation—from its non-functional constraints regarding security, performance, and localization, we have established a definitive blueprint for the software. These clearly defined requirements will serve as the guiding metric for the subsequent design and implementation phases.

Chapter 3: Analysis and Design

3.1 Introduction

The analysis and design phase is a critical step in the software development life cycle. It bridges the gap between system requirements and implementation. In this chapter, we use the Unified Modeling Language (UML) to visualize, specify, and document the artifacts of the "ATS CV Advisor" system. We will present the Use Case diagrams to describe the system's functional behavior, the Sequence diagrams to illustrate the dynamic interactions, and the Class diagram to define the database structure.

3.2 Use Case Diagram

The Use Case diagram provides a high-level overview of the system's functionality from the user's perspective. It identifies the actors interacting with the system and the specific use cases (actions) they can perform.

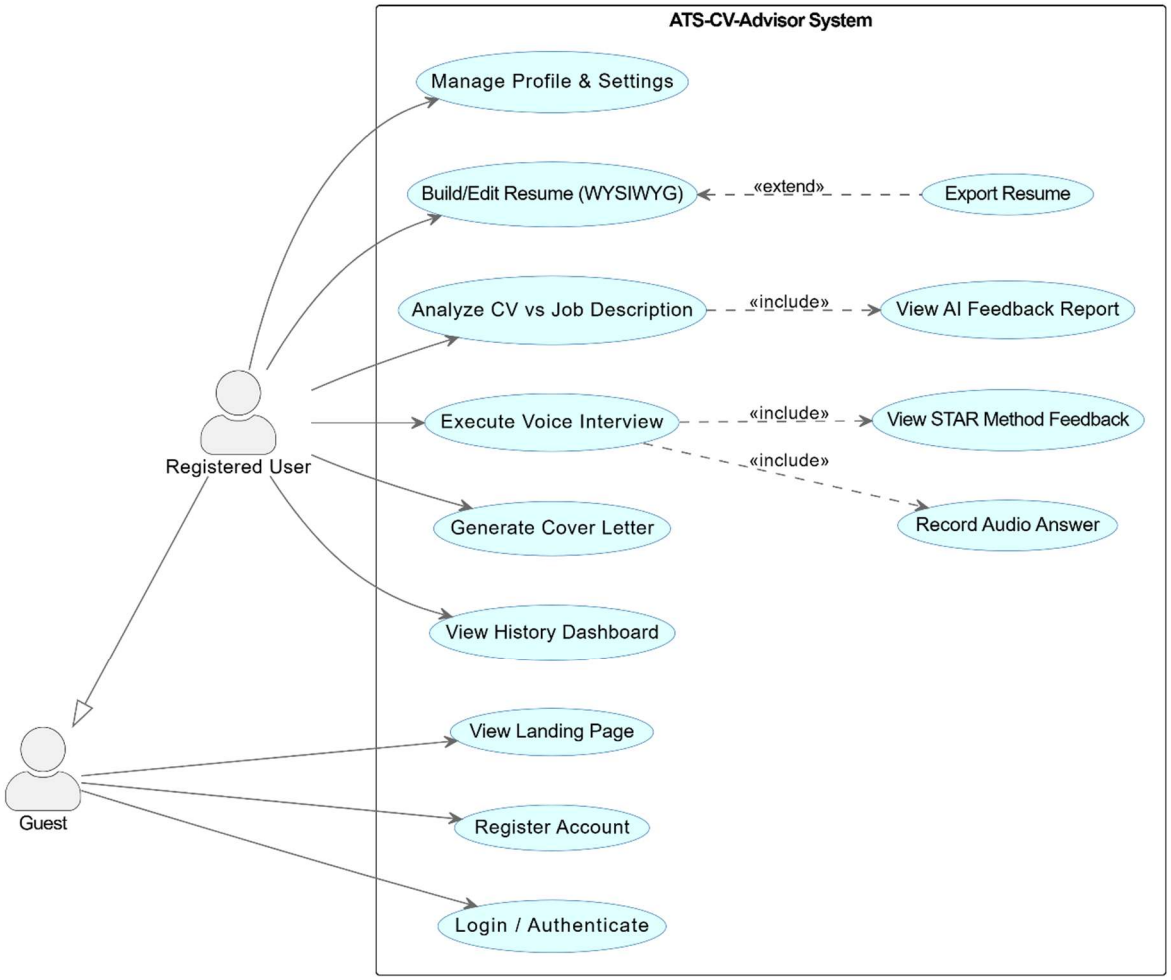


Figure 1 - Use Case Diagram

3.3 Sequence Diagrams

While the Use Case diagram shows "what" the system does, Sequence diagrams illustrate "how" the system components interact over time to execute a specific scenario. Given our decoupled architecture, these diagrams highlight the asynchronous HTTP API communication between the Next.js client, Laravel server, and external services.

3.3.1 User Authentication Sequence Diagrams

This section describes the chronological steps taken when a user attempts to authenticate or create an account.

3.3.1.1 Login via email and password Sequence Diagram

Description: This diagram illustrates the authentication process when a candidate attempts to log into the platform using their standard credentials (Email and Password). It details the communication between the UI, the backend API, and the database to securely issue an authentication token.

a. **Main Scenario (Success Path):**
    1. The Candidate enters their email and password into the frontend form and clicks "Login".
    2. The Frontend (Next.js) sends a secure POST request to the '/api/login' endpoint on the Backend (Laravel).
    3. The Backend queries the Database (MySQL) to verify if the user exists and validates the password hash.
    4. The Database confirms that the credentials match and are valid.
    5. The Backend generates a secure Sanctum Auth Token and returns a "200 OK" JSON response to the Frontend.
    6. The Frontend saves the token locally in the browser and redirects the candidate to the main Dashboard workspace.
b. **Alternative Scenarios (Error Paths):**
    - **Alternative A (Invalid Credentials):** If the email or password does not match any record, the backend returns a "401 Unauthorized" response. The frontend intercepts this and displays an error message to the user.
    - **Alternative B (Account Not Verified):** If the credentials are correct but the user's account status is unverified (is_verified = false), the backend returns a "403 Forbidden" response, and the frontend prompts the user to complete their email verification.

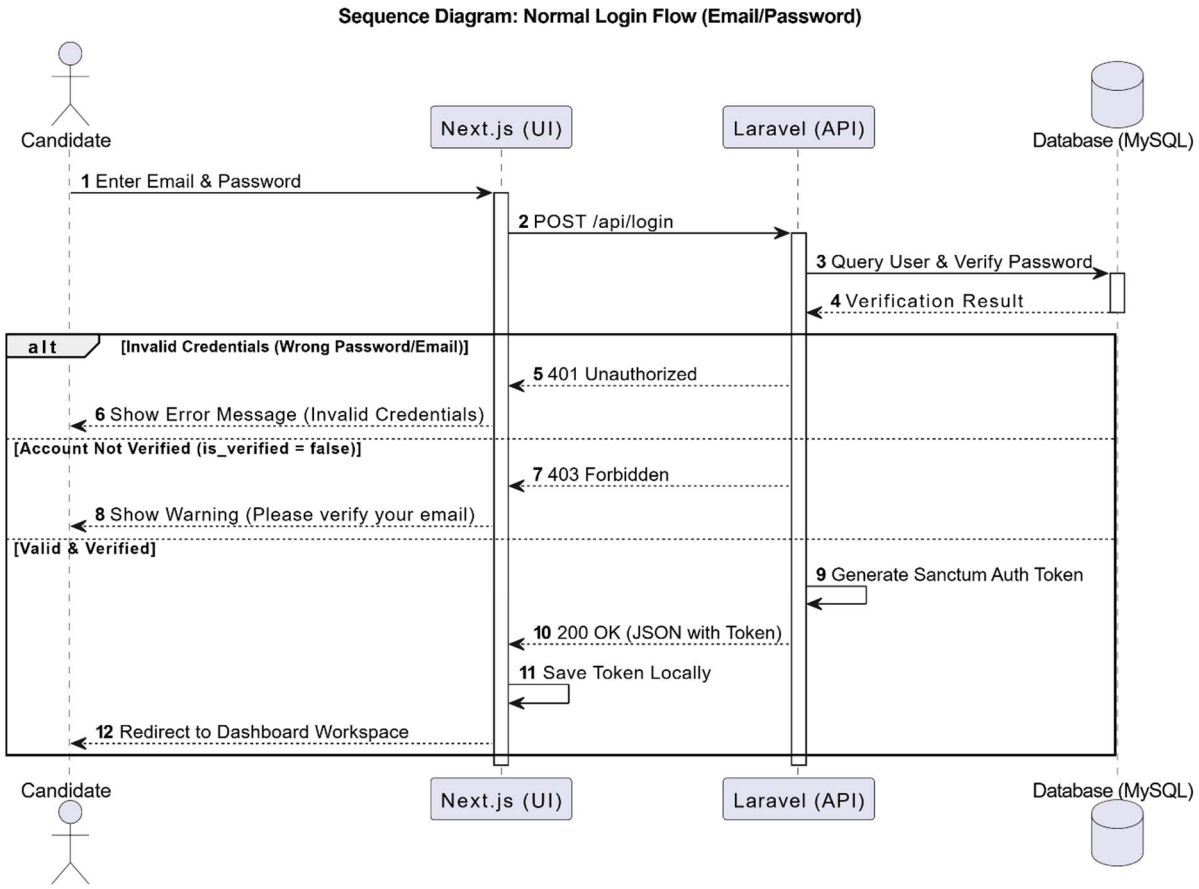


Figure 2 - Sequence Diagram" Login via email and password"

3.3.1.2 Registration Sequence Diagram

Description : This diagram shows the dual-stage lifecycle of creating a new user account and enforcing security via a secure One-Time Password (OTP) verification code sent to the candidate's email.

a. Main Scenario (Success Path):

1. The Candidate submits their registration details (Name, Email, Password) via the signup form.
2. The Frontend sends a POST request to '/api/register'.
3. The Backend verifies that the email is unique, creates a new user record with 'is_verified=false', and invokes the Mail Server.
4. The Mail Server delivers an OTP verification code to the candidate's email address.
5. The Frontend redirects the candidate to the verification code entry page.
6. The Candidate enters the received OTP code, and the frontend submits a POST request to '/api/verify-email'.
7. The Backend validates the code against the database, updates the status to 'is_verified=true', generates a Sanctum token, and redirects the user to the Dashboard.

b. Alternative Scenarios (Error Paths):

- Alternative A (Email Already Exists): If the email is already registered, the backend stops the process and returns a "422 Validation Error", prompting the frontend to display an "Email already taken" message.
- Alternative B (Invalid Code): If the entered OTP code is incorrect or expired, the backend returns a "400 Bad Request" response, allowing the candidate to retry entering the code.

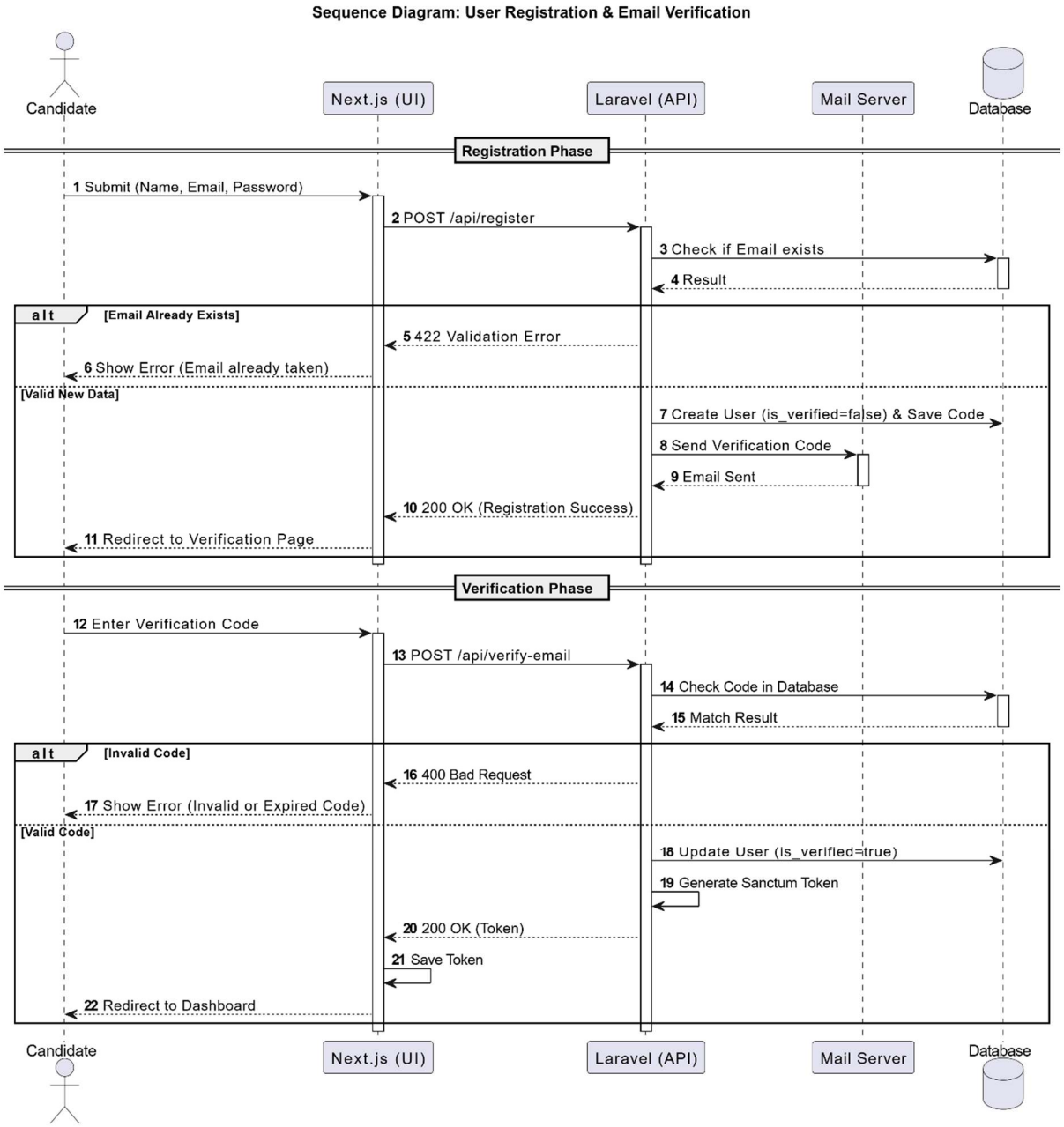


Figure 3 - Sequence Diagram" Registration"

3.3.1.3 Google OAuth Authentication Sequence Diagram

Description: This diagram details the integration with an external Identity Provider (IdP) using the Google OAuth 2.0 protocol, allowing users a seamless Single Sign-On (SSO) experience.

a. Main Scenario (Success Path):

1. The Candidate clicks the "Continue with Google" button on the login interface.
2. The Frontend requests the official authorization redirect URL from the Backend API ('/api/auth/google/redirect').
3. The Frontend redirects the candidate's browser to the secure Google Sign-In server.
4. The Candidate authenticates successfully with Google and grants permission to the app.
5. Google redirects the user back to the backend callback endpoint with an authorization code.
6. The Backend communicates directly with Google to exchange the code for the user's profile (Name, Email, Avatar).
7. The Backend verifies the email in the database; if it is a new user, it creates an account, marks it as verified, generates a Sanctum Token, and passes it back to the frontend.

b. Alternative Scenarios (Error Paths):

- Alternative A (User Denies Authorization): If the candidate cancels the Google login or denies permissions, Google redirects back with an "access_denied" flag. The backend routes the user back to the login page with a localized cancellation error.

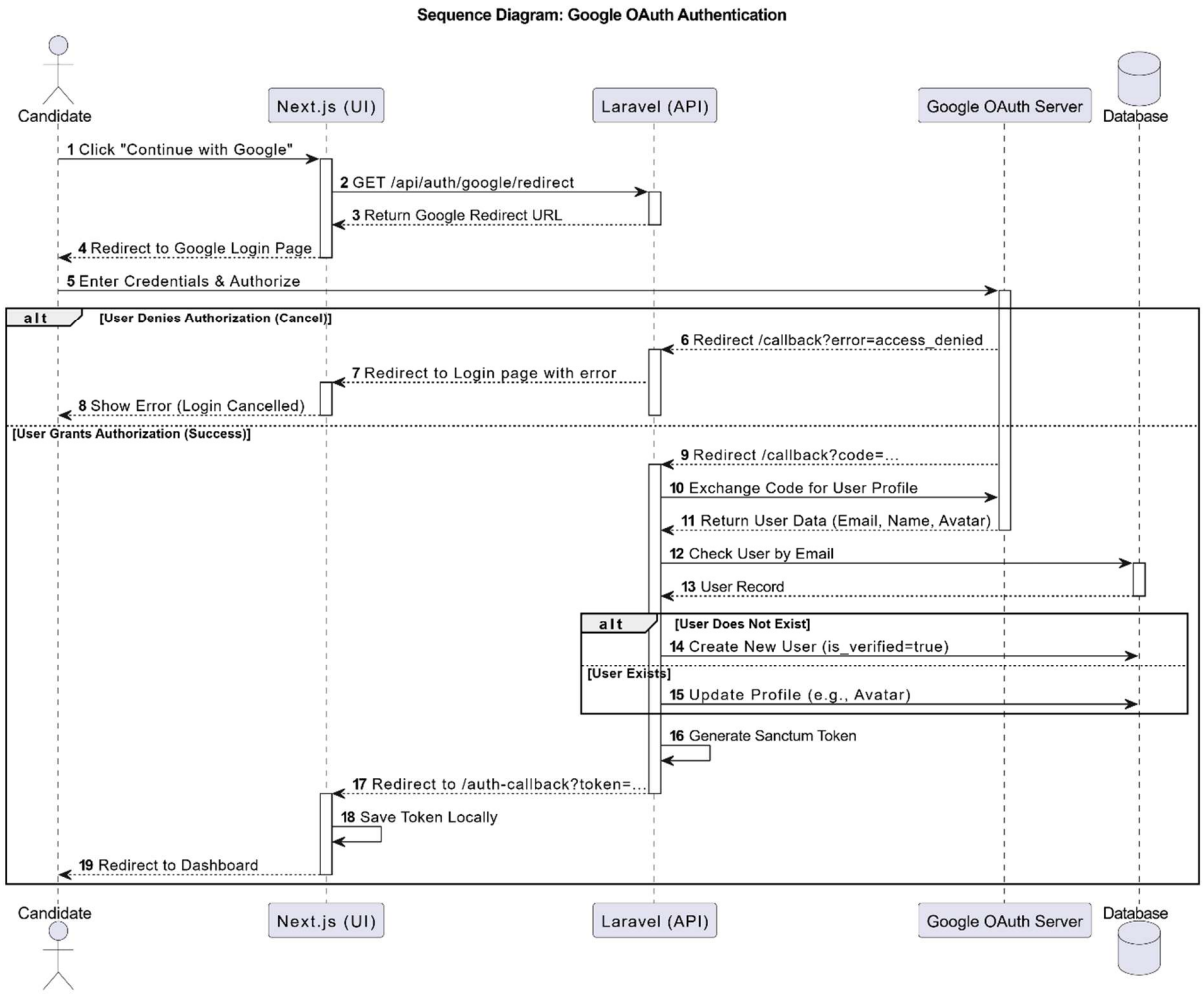


Figure 4 - Sequence Diagram" Google OAuth Authentication"

3.3.2 AI-Powered Resume Analysis Sequence Diagram

Description: This diagram outlines the core feature of the platform, showing how candidate resume data and job descriptions are synchronized, processed, and passed to the external Gemini AI engine to extract interactive feedback.

a. Main Scenario:

1. The Candidate selects their CV source (uploads a file or chooses a saved template), pastes the target Job Description, and clicks "Analyze".
2. The Frontend fetches the Sanctum Token and triggers a POST request to '/api/resume/analyze'.
3. The Backend Sanctum Middleware authenticates the request against the database to confirm user legitimacy.
4. The Backend handles the input data: either parsing the uploaded PDF file or executing the "buildResumeText()" method to turn a saved template into clean text.
5. The Backend structures an automated system prompt and sends an asynchronous HTTP POST request to the external Google Gemini AI API.
6. Gemini AI processes the natural language context and returns a strict, structured JSON payload containing scores, strengths, and missing skills.
7. The Backend stores this analysis result into the database and returns a successful response to the UI.
8. The Frontend parses the JSON data and renders the interactive optimization report.

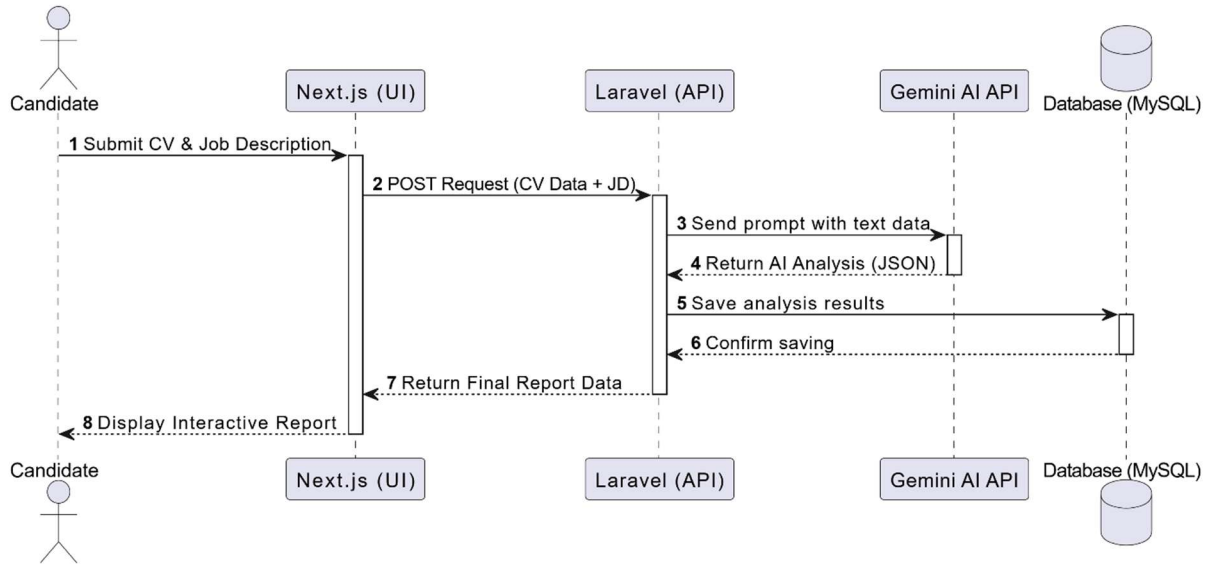


Figure 5 - Sequence Diagram" AI-Powered Resume Analysis"

3.3.3 Voice Interview Execution Sequence Diagram

Description: This Sequence Diagram illustrates the architecture and data flow of an AI-powered mock interview evaluation application. It outlines the journey of a user's audio recording starting from the web interface, passing through the backend server, and then being

analyzed by specialized AI models to evaluate vocal performance and answer quality. Finally, the results are saved in a database and interactively displayed to the end user.

a. Main Scenario:

1. Submit Audio: The User records their audio response and submits it via the frontend interface built with Next.js.
2. API Request: The Next.js application sends a POST request containing the audio data (Audio Payload) to the backend Laravel API via the /api/interview/analyze endpoint.
3. Forward Audio: The Laravel server forwards the audio file (Audio Blob) via HTTP to a processing microservice hosted on a Colab Microservice.
4. Initiate AI Processing: The Colab service initiates the audio processing and analysis pipeline using three distinct models/tools:
   - Whisper: Transcribes the audio recording into text.
   - Librosa: Performs acoustic math analysis, such as calculating Words Per Minute (WPM).
   - Qwen: An AI model that grades the response based on the STAR method.
5. Return Raw Data: Once the analysis is complete, these models return the raw results as a data dictionary (Raw analysis dictionary) back to the Colab microservice.
6. Format Response: The Colab service structures these results and sends them back as a Formatted JSON Response to the Laravel API.
7. Store Results: Laravel takes these results and saves them as an InterviewScore record in an SQLite Database for future reference.
8. Return Dashboard Data: The Laravel server sends the dashboard data in JSON format back to the Next.js frontend.
9. Render UI: Finally, the Next.js application receives the data and renders the dynamic scores and badges for the User on the screen.

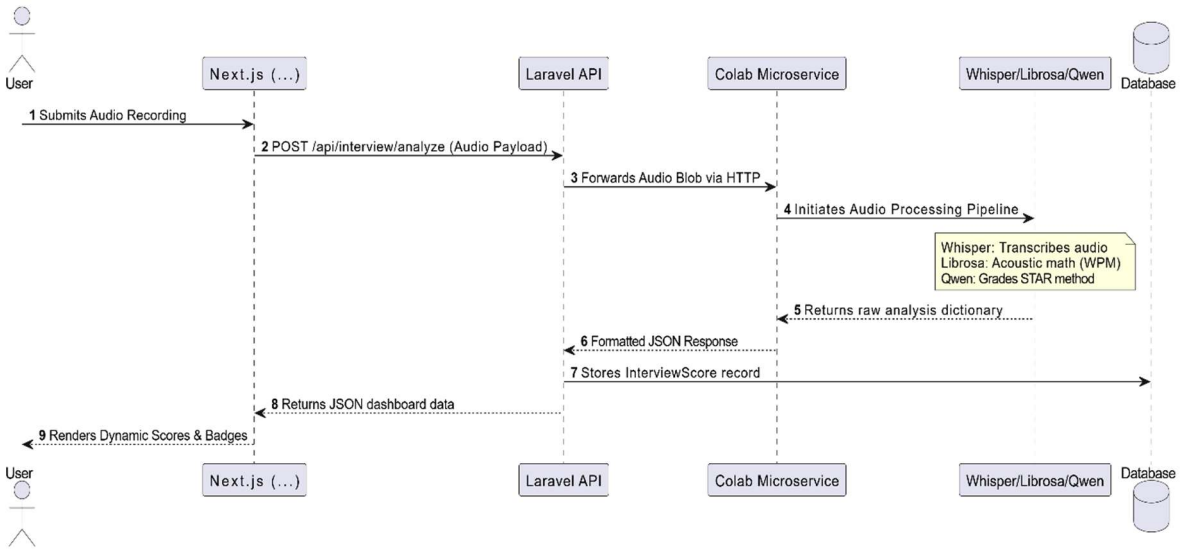


Figure 6 - Sequence Diagram" Voice Interview Execution"

3.3.4 Cover Latter Generator Sequence Diagram

Description: This feature allows the candidate to automatically generate a professional and highly tailored cover letter. The system merges the user's saved resume data with a target job description and transmits this context to an external AI API. The AI crafts a customized text which the user can then preview, edit, and save directly to their personal history dashboard.

a. Main Scenario:

1. The user navigates to the Cover Letter Generator interface.
2. The user selects a previously saved resume from the dropdown menu.
3. The user pastes the target Job Description and selects the desired tone and language.
4. The user clicks the "Generate" button.
5. The backend system aggregates the resume data and job description, transmitting them as a structured prompt to the external AI service (Cover Letter API).
6. The AI service returns the generated cover letter text, which the system immediately displays in the frontend editor for user preview.
7. The user clicks the "Save to History" button.
8. The system stores the cover letter record in the database (within the cover_letters table), links it to the user's account, and dynamically updates the sidebar to display the newly saved document.

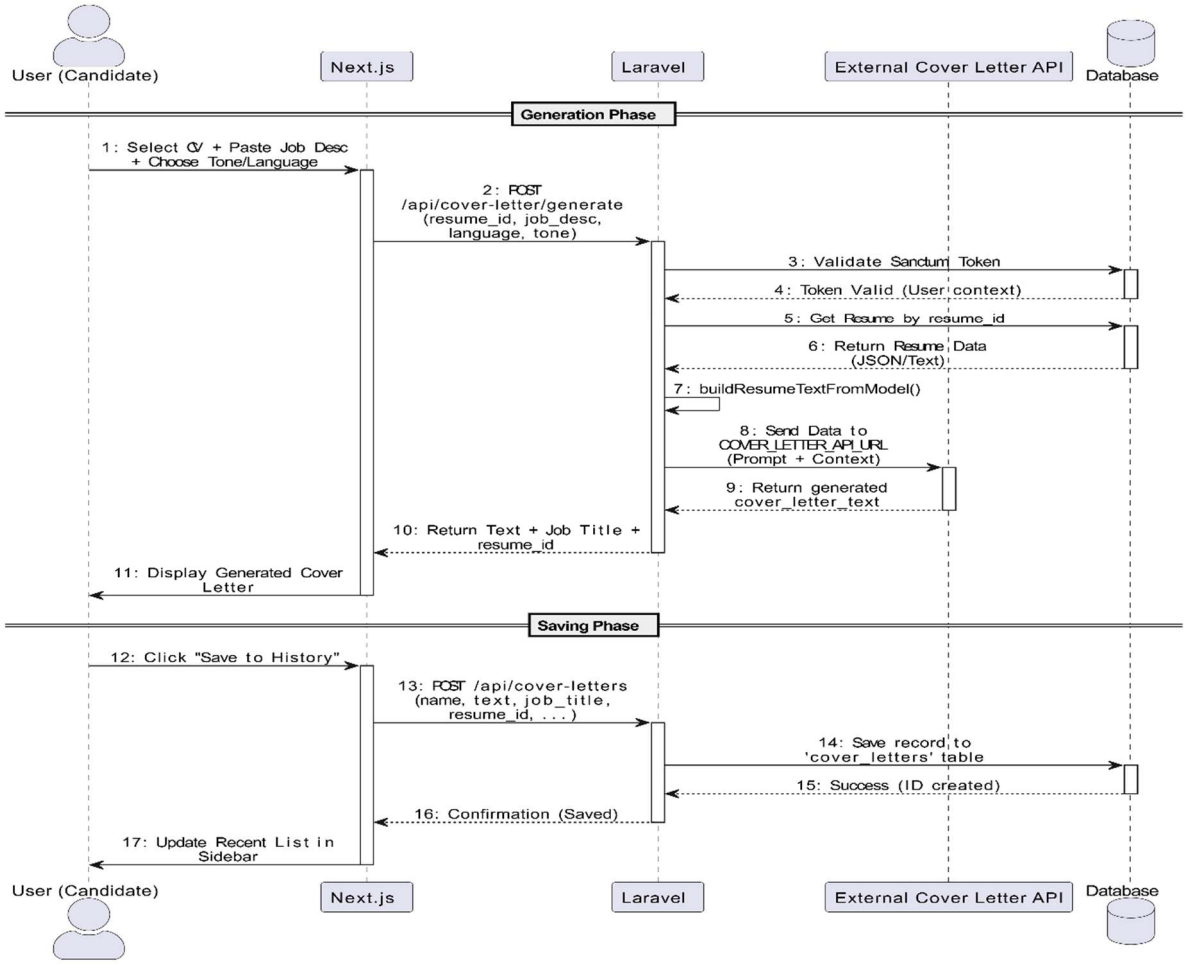


Figure 7 - Sequence Diagram" Cover Latter Generator"

3.4 Classes Diagram

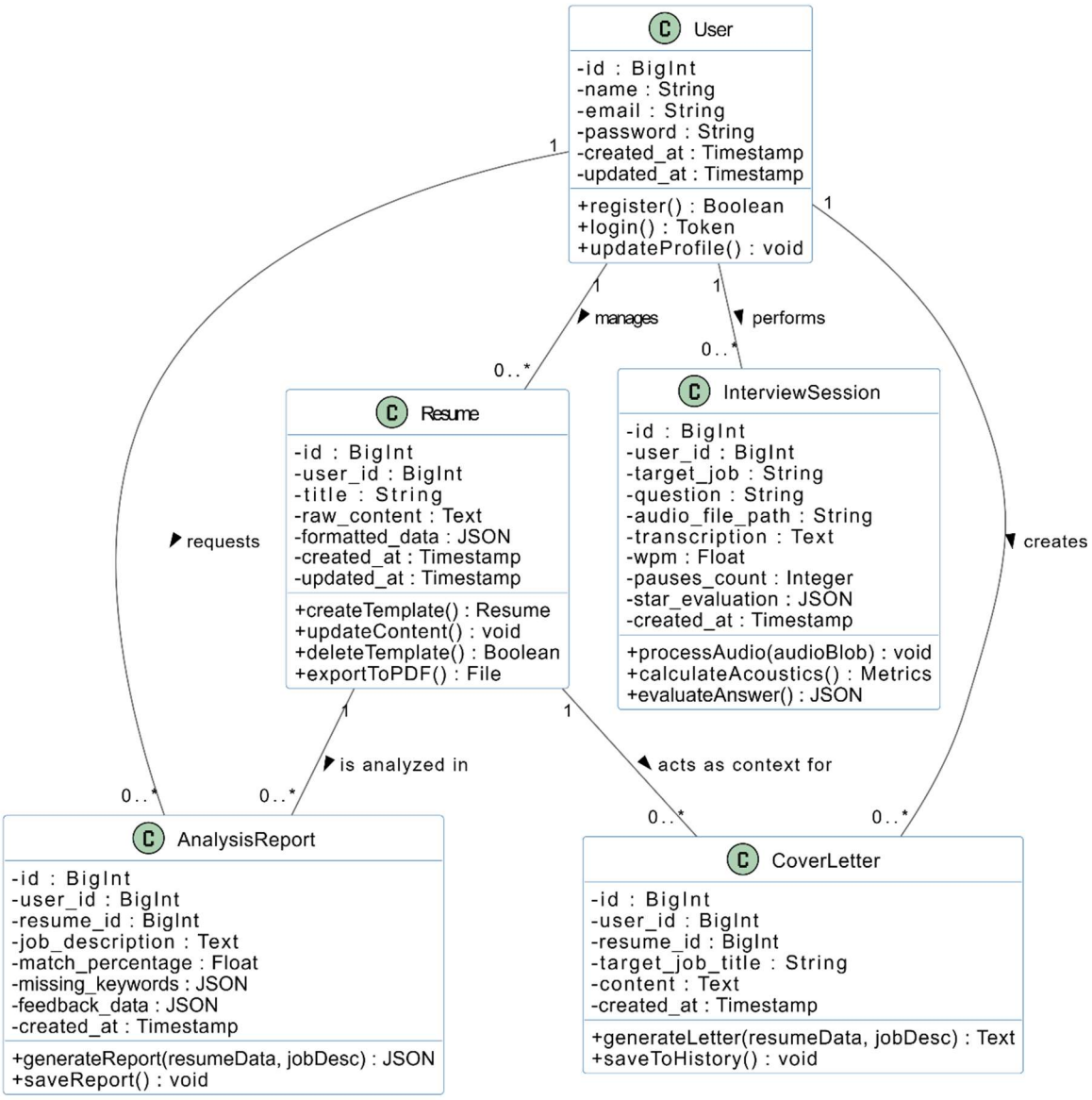


Figure 8 - Classes Diagram

3.5 Conclusion

Through comprehensive UML modeling, this chapter has successfully translated the theoretical requirements of the system into logical, actionable technical blueprints. Every critical workflow—from the decoupled authentication paths to the complex data pipelines driving the Gemini AI resume analysis and the multi-model voice interview microservice—has been structurally verified. By mapping these dynamic interactions alongside the underlying database schema, we have established a robust architectural foundation. With these blueprints finalized, we now possess the precise roadmap necessary to write the application code. The subsequent chapter will detail the realization of these designs into a fully functioning application.

Chapter 4: Implementation and Experimental Results

4.1 Introduction

After specifying the system requirements and establishing a robust architectural design through UML diagrams, the final phase of the software development life cycle is implementation. This chapter highlights the hardware and software environments used during the development process. Furthermore, it details the technology stack chosen to build the decoupled architecture of the "ATS CV Advisor" platform and presents screenshots of the final graphical user interfaces.

4.2 Development Environment

To ensure a smooth and efficient development workflow, the following tools and software were utilized:



- Code Editor: Visual Studio Code (VS Code), chosen for its extensive extension ecosystem and integrated terminal.



- Local Server: XAMPP, providing the Apache web server and MySQL database management system.





- Version Control: Git and GitHub for tracking code changes and project management.



- API Testing: Postman, used extensively to test and validate Laravel API endpoints before integrating them with the Next.js frontend.

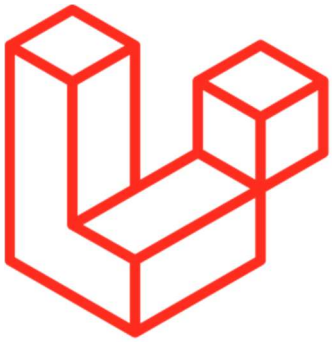


- Runtime Environment: Node.js (v20+), required to run the Next.js frontend application.

4.3 Technology Stack

The platform was built using a modern, decoupled architecture separating the client-side from the server-side to ensure scalability, security, and high performance.

4.3.1 Backend (Server-Side)



- Framework: Laravel 12 (PHP). Selected for its elegant syntax, built-in routing, and robust security features.
- Authentication: Laravel Sanctum was implemented
- to manage secure, token-based API authentication. Laravel Socialite was integrated to handle Google OAuth seamlessly.
- Database: MySQL, a reliable relational database management system used to store users, templates, and analysis history.

4.3.2 Frontend (Client-Side)



- Framework: Next.js 16 (React framework). Chosen for its fast rendering,
- App Router capabilities, and excellent developer experience.
- Styling: Tailwind CSS, a utility-first CSS framework that enabled the rapid creation of a highly responsive design with built-in Light/Dark mode support.

4.3.3 External APIs and Core Engines











- **Google Gemini 2.5 Flash API:** This Natural Language Processing (NLP) model powers the core logic of the application by evaluating resumes against job descriptions and generating strict JSON feedback.
- **OpenAI Whisper:** An advanced speech recognition (ASR) system used to accurately transcribe candidates' audio responses during mock interviews.
- **Librosa:** A specialized Python library that calculates acoustic metrics, such as Words Per Minute (WPM) and pause counts, to evaluate speaking fluency.
- **Qwen LLM:** A powerful language model utilized to evaluate the transcribed interview answers against the HR-standard STAR method.
- **Google Colab Endpoints:** A cloud computing environment used to host and execute heavy AI microservices (like audio processing and cover letter generation) via REST APIs.

4.4 Application Interfaces (Results)

In this section, we present the final graphical user interfaces of the platform, demonstrating the realization of the functional requirements.

4.4.1 Authentication and Access

The platform ensures a secure entry point. Users can sign in or register using a standard email and password (followed by OTP email verification) or utilize the Google Single Sign-On (SSO) feature for faster access.

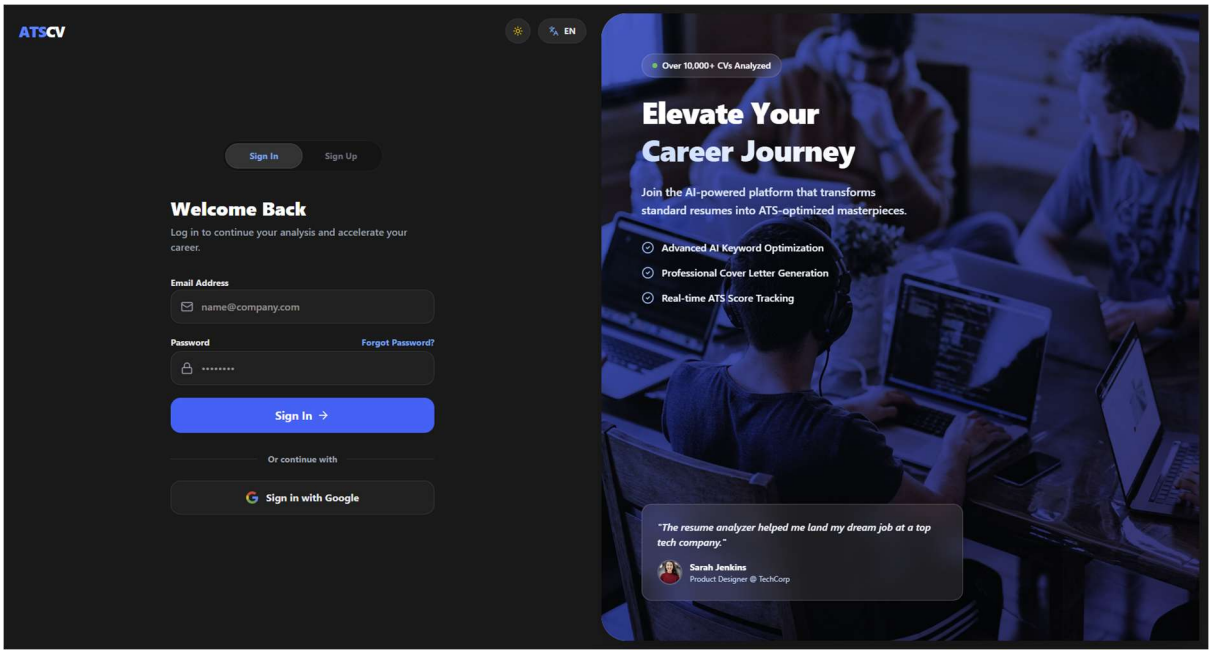


Figure 9 – Log in page

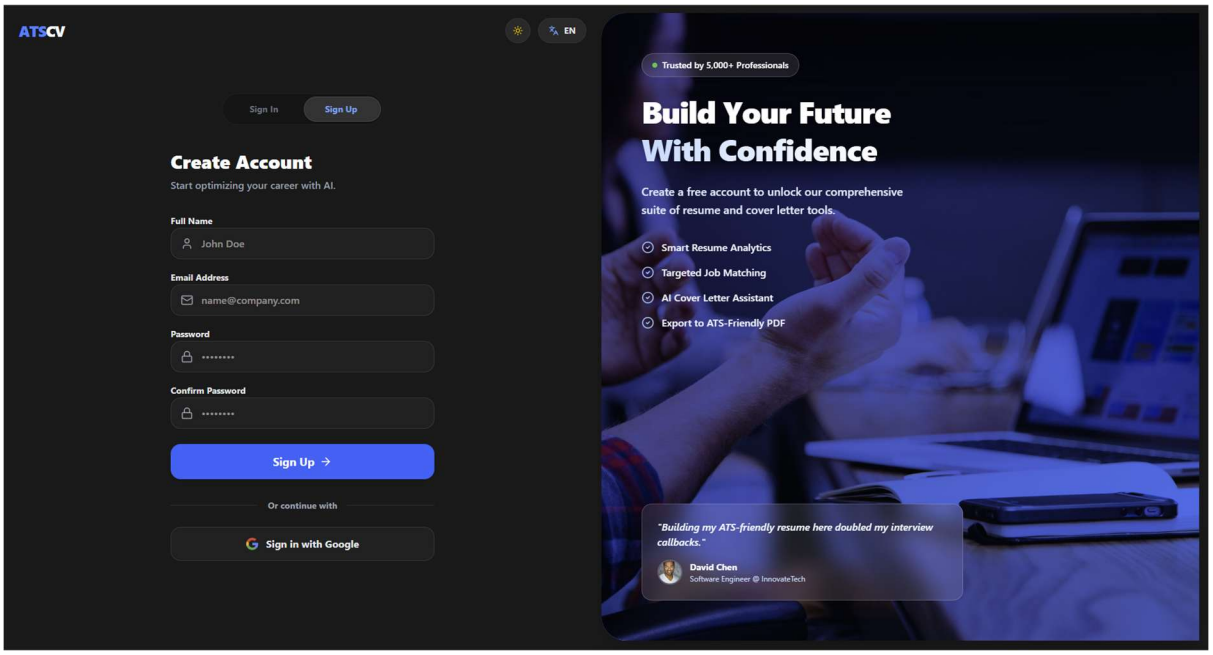


Figure 10 – Registration page

4.4.2 The Main Analysis Workspace

This is the core interface of the platform. The left sidebar maintains a history of past analyses. In the central area, the user can upload a PDF/Image or select a saved template, paste the target Job Description, and select the desired output language.

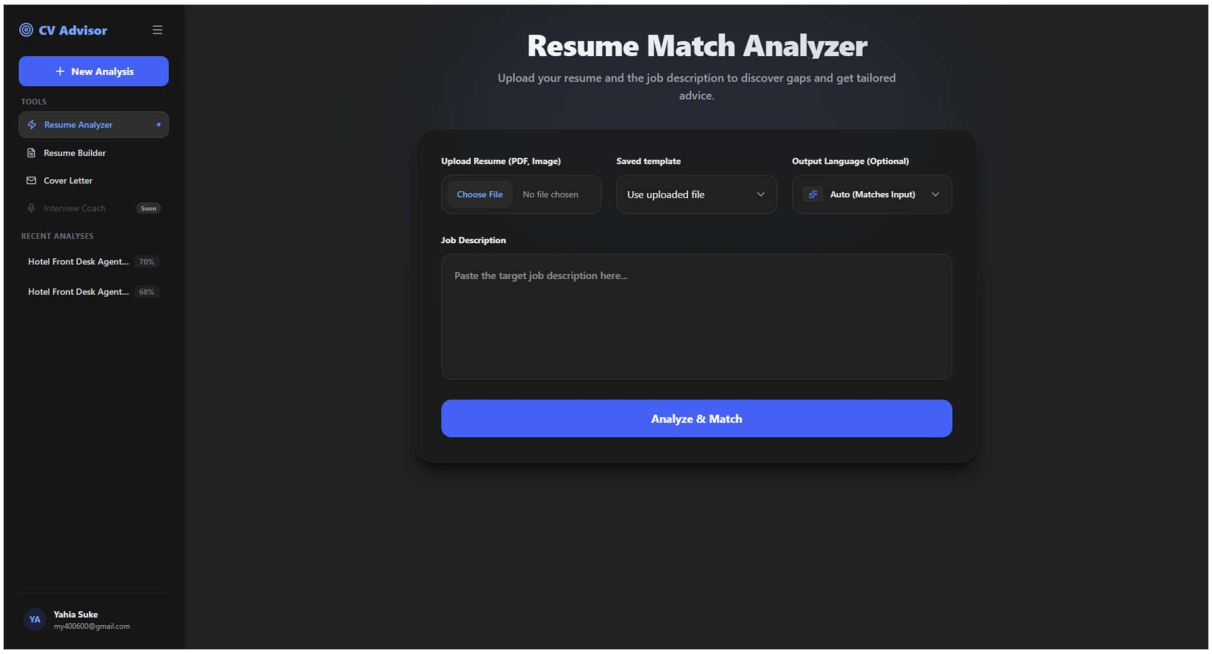


Figure 11 – Main dashboard and input form

4.4.3 AI Interactive Feedback Report

Upon processing the request, the system dynamically renders the Gemini AI feedback. This interface visually categorizes the results into an overall Match Score, Missing Skills, Strengths, and specific HR phrasing recommendations.

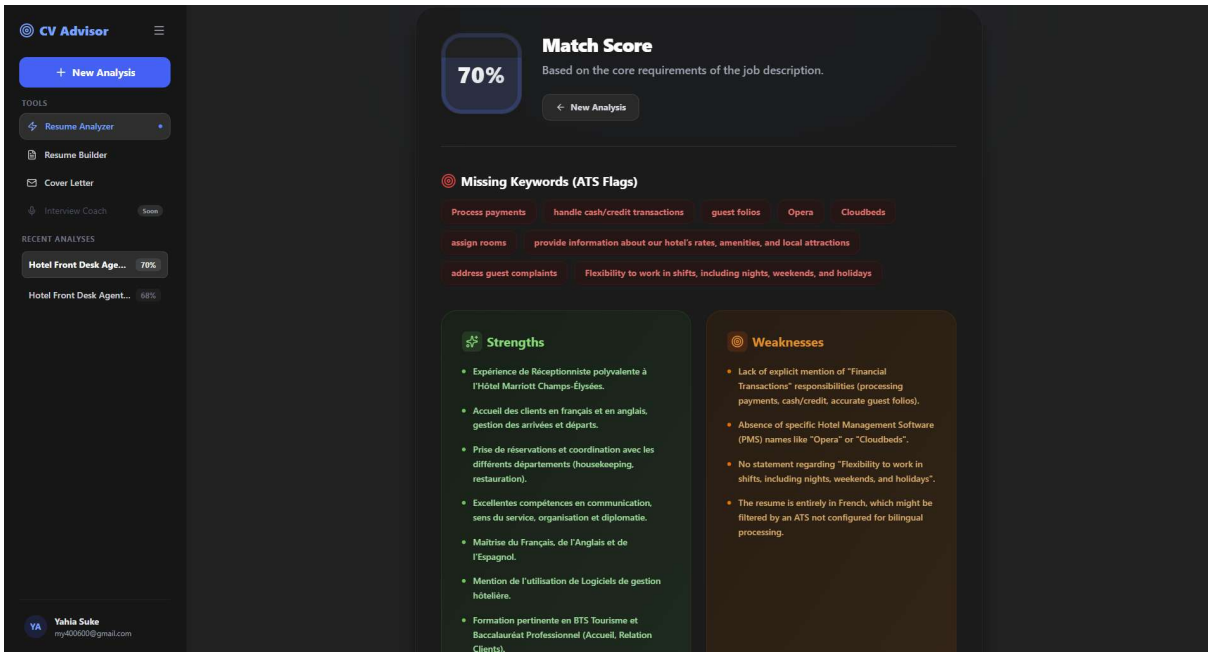




Figure 12 – AI-Generated evaluation and optimization report

4.4.4 In-App Resume Builder (Iterative Sandbox)

To solve the friction of external editing, this interface provides a structured form where candidates can manage their resume content. Users can apply the AI's suggestions in real-time, save the template to the database, and preview the generated raw-text structure.

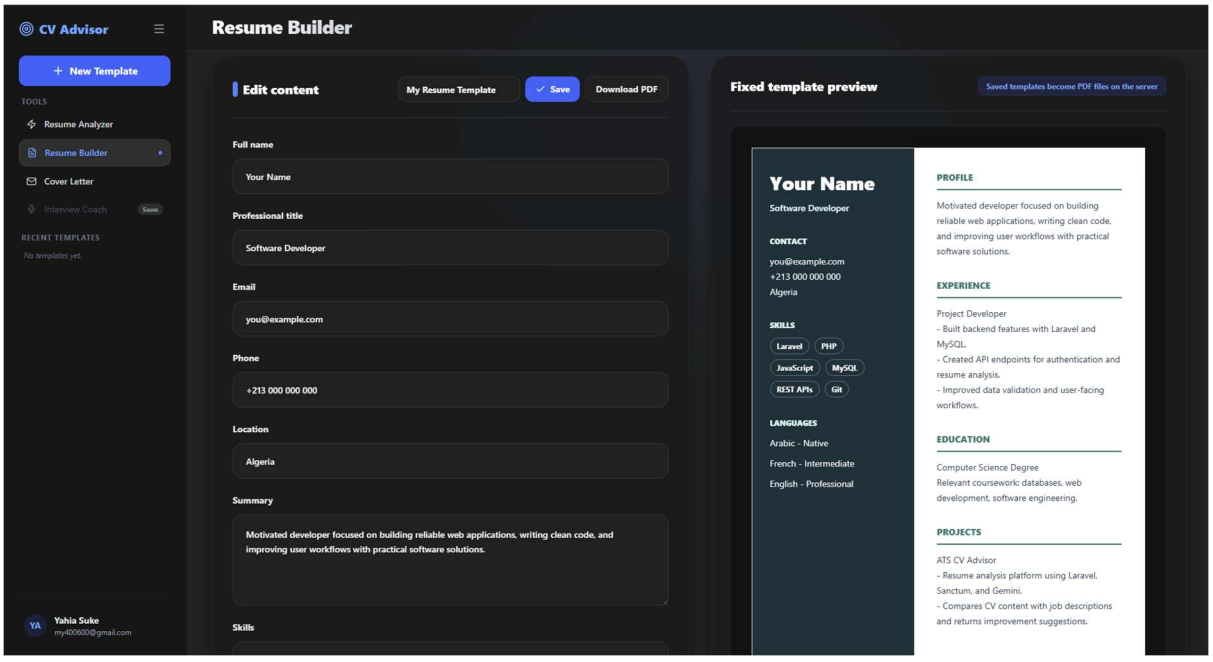


Figure 13 – In-App resume builder and template management

4.4.5 Cover Latter Generator

To eliminate the time-consuming process of writing tailored applications, this interface offers an automated generation tool where candidates merge their saved resumes with target job descriptions. Users can customize the tone and language, generate highly contextual cover letters via AI, and save the finalized document directly to their personal history.

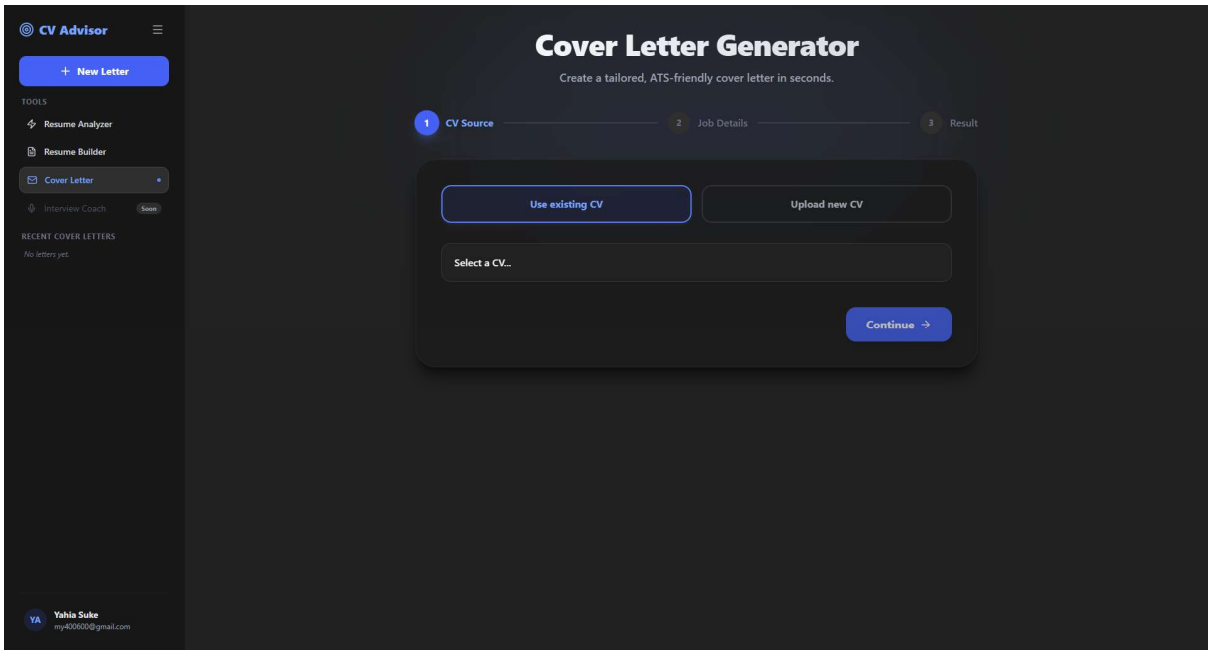


Figure 14 – Cover Latter “adding CV page”

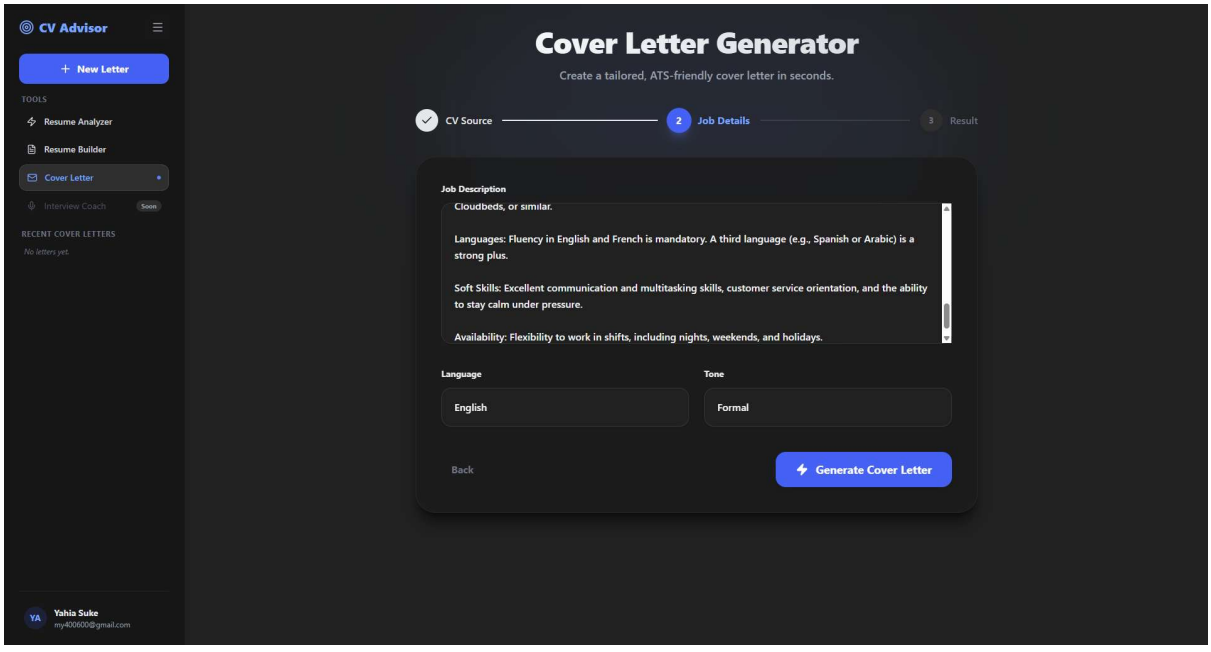


Figure 14 – Cover Latter “adding job descreptin page”

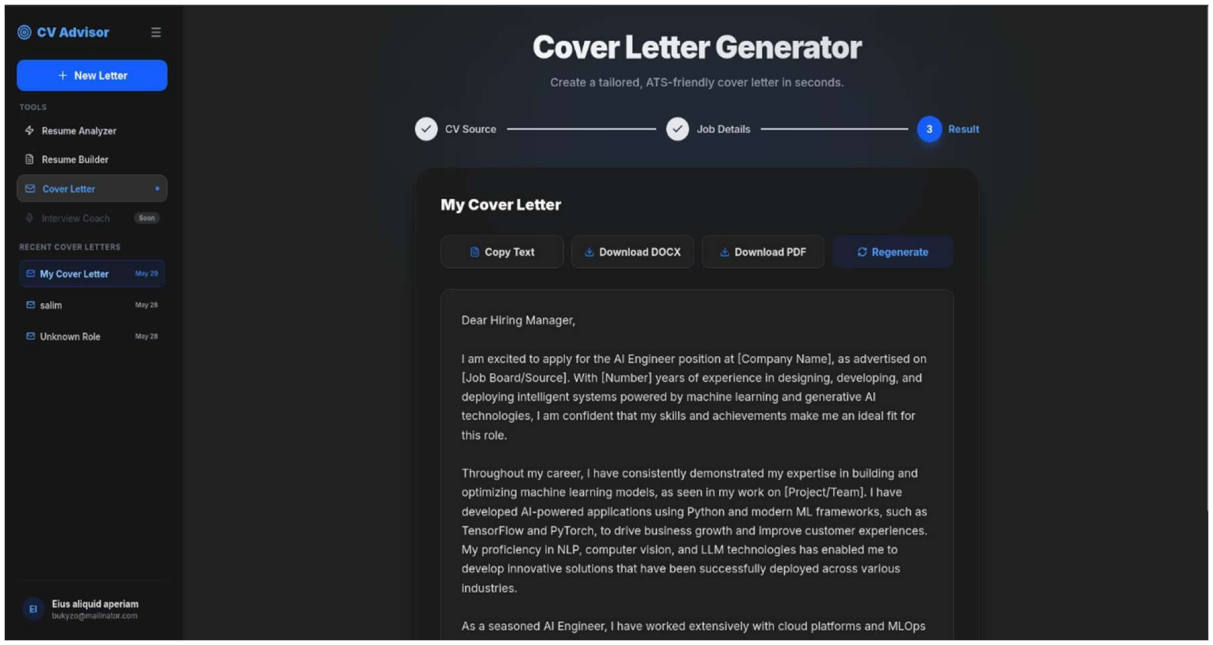


Figure 14 – Cover Latter “result page”

4.4.6 Voice Interview Coach

To overcome the limitations of static interview preparation, this interface provides a dynamic mock interview environment where candidates can record their verbal responses. Users can receive real-time acoustic metrics, review the AI-generated transcription, and explore structural feedback based on the STAR method to continuously improve their delivery.

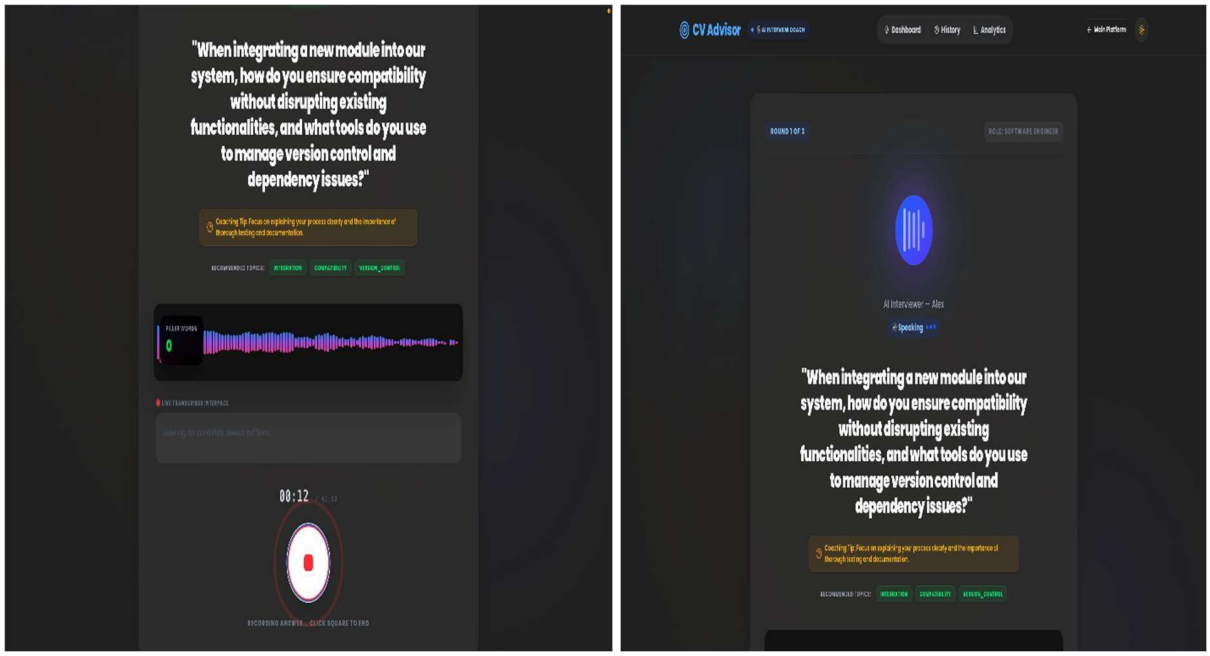


Figure 14 – Complete View of the Dynamic Voice Interview Dashboard

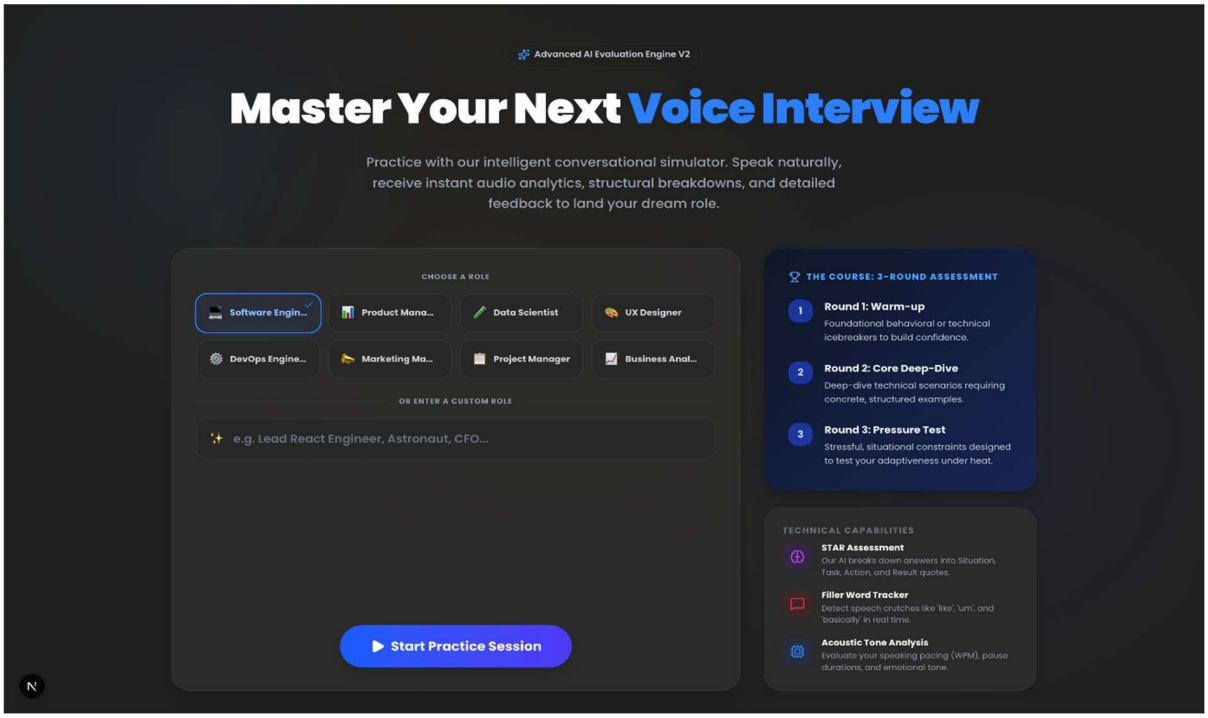


Figure 14 – Entry page for the voice interviewer AI

4.5 Conclusion

This final chapter has demonstrated the practical realization of the "ATS CV Advisor" platform, successfully transforming UML designs into highly interactive, fast, and secure web interfaces. By combining Laravel's secure API core with Next.js's rendering speed and Gemini's cognitive parsing, the functional requirements have been thoroughly validated.

General Conclusion

General Conclusion

The primary objective of this graduation project was to design and develop a comprehensive, intelligent web ecosystem named "ATS CV Advisor" to bridge the widening gap between job seekers and Applicant Tracking Systems (ATS). In today's highly competitive and automated recruitment landscape, candidates frequently struggle to optimize their professional profiles according to rigid HR algorithms, which often leads to premature rejections before reaching the interview stage. This project successfully addresses this systemic friction by democratizing access to enterprise-grade AI evaluation tools.

Throughout this work, we successfully delivered a strictly decoupled, full-stack software solution that harnesses the power of multiple Generative AI engines. By choosing a cutting-edge technology stack—utilizing Laravel 12 to build a secure, robust API backend and Next.js 16 to craft a fast, responsive user interface—the platform ensures exceptional scalability, data security, and seamless navigation. Furthermore, computationally heavy tasks were efficiently outsourced to Google Colab endpoints, ensuring the main application remains highly performant.

The core contribution of this project lies in its interconnected, automated career preparation cycle, which seamlessly integrates several advanced modules:

- **AI-Powered ATS Analysis & Resume Builder:** By integrating the Google Gemini 2.5 Flash API, the application goes beyond basic keyword matching. It performs deep contextual analysis, evaluates experience metrics against specific job descriptions, and renders real-time structural feedback. This is paired with an interactive sandbox builder that allows users to adapt and format their resumes dynamically.
- **Automated Cover Letter Generation:** The system intelligently merges the candidate's optimized resume data with targeted job requirements to automatically draft highly contextual and professional cover letters, drastically reducing application preparation time.
- **Dynamic Voice Interview Coach:** To prepare candidates beyond the initial screening phase, we engineered a state-of-the-art mock interview module. This feature orchestrates OpenAI Whisper for precise speech-to-text transcription, Librosa for extracting physical acoustic metrics (such as Words Per Minute and pause counts), and the Qwen LLM to evaluate the spoken content strictly against the HR-standard STAR method.

Additionally, the platform was developed with a strong emphasis on usability and inclusivity, featuring a fully multilingual interface that supports English, French, and Arabic (with native Right-To-Left CSS integration) alongside seamless Single Sign-On (SSO) authentication.

In conclusion, this project served as an invaluable experience in exploring modern Agile workflows, mastering complex decoupled full-stack architectures, orchestrating multiple external AI APIs, and applying rigorous software engineering principles from initial conceptualization to final deployment.