VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018

A PROJECT WORK REPORT

ON

"AI Career Coach"

A Dissertation Submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

Alfesh (1RG22CS014)

Devaraj (1RG22CS028)

Pavan (1RG22CS058)

Vishnu (1RG22CS089)

Under The Guidance of

Mrs. Deepti N N

Assistant Professor,

Dept. of CSE

RGIT, Bengaluru-32

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OFTECHNOLOGY

Cholanagar, R.T. Nagar Post, Bengaluru-560032

2025-2026

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)

Cholanagar, R.T. Nagar Post, Bengaluru-560032

Department of Computer Science & Engineering

CERTIFICATE

Major Project Phase - II

This is to certify that the Project Report titled **"AI Career Coach"** is a bonafide work carried out by **Mr. Alfesh (USN 1RG22CS014), Mr. Devaraj (USN 1RG22CS028), Mr. Pavan (USN 1RG22CS058) and Mr. Vishnu (USN 1RG22CS089)** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi**, during the year **2025-2026.** It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This project report has been approved as it satisfies the academic requirements in respect of project work (BCS786) prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
Mrs. Deepti N N	**Dr. Arudra A**	**Dr. D G Anand**
Assistant Professor	Associate Professor	Principal,
Dept. of CSE	Head Of Department	RGIT, Bengaluru
RGIT, Bengaluru	Dept. of CSE,	
	RGIT, Bengaluru	

External Viva

Name of the Examiners	Signature with date
1.	
2.	

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Department of Computer Science & Engineering

DECLARATION

We hereby declare that the project work entitled **"AI Career Coach"** submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2025-2026**, is record of an original work done by us under the guidance of **Mrs. Deepti N N,** Assistant Professor, Department of Computer Science and Engineering, RGIT, Bengaluru in the partial fulfillment of requirements for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering.** The results embodied in this project have not been submitted to any other University or Institute for award of any degree or diploma.

Alfesh **(1RG22CS014)**

Devaraj **(1RG22CS028)**

Pavan **(1RG22CS058)**

Vishnu **(1RG22CS089)**

ACKNOWLEDGEMENT

We take this opportunity to thank our college **Rajiv Gandhi Institute of Technology, Bengaluru** for providing us with an opportunity to carry out this project work.

We express our gratitude to **Dr. D G Anand**, Principal, RGIT, Bengaluru for providing the resources and support without which the completion of this project would have been a difficult task.

We extend our sincere thanks to **Dr. Arudra**, Associate Professor and Head, Department of Computer Science and Engineering, RGIT, Bengaluru, for being a pillar of support and encouraging us in the face of all adversities.

We would like to acknowledge the thorough guidance and support extended towards us by project coordinators **Dr. Latha P H**, Associate Professor, Dept. of CSE, RGIT **Mrs. Bhagyashri Wakde,** Assistant Professor, Dept. of CSE, RGIT, and, Bengaluru. Their incessant encouragement and valuable technical support have been of immense help. Their guidance gave us the environment to enhance our knowledge and skills and to reach the pinnacle with sheer determination, dedication and hard work.

We offer our sincere gratitude to our guide, **Mrs. Deepti N N**, Assistant Professor. Dept. of CSE, for her invaluable technical support and constant encouragement. Her guidance enhanced our knowledge and skills, driving us to achieve the pinnacle with dedication and hard work.

We also want to extend our thanks to the entire faculty and support staff of the Department of Computer Science and Engineering, RGIT, Bengaluru, who have encouraged us throughout the course of the Bachelor's Degree. We want to thank our family for always being there with full support and for providing us with a safe haven to conduct and complete our project. We are ever grateful to them for helping us in these stressful times. Lastly, we want to acknowledge all the helpful insights given to us by all our friends during the course of this project.

Alfesh (1RG22CS014)
Devaraj (1RG22CS028)
Pavan (1RG22CS058)
Vishnu (1RG22CS089)

ABSTRACT

The modern job market is highly competitive, with candidates struggling to manage numerous applications, resumes, and interviews efficiently. Manual job-seeking methods are fragmented and time-consuming, often leading to missed opportunities and reduced motivation. The **AI Career Coach** addresses these challenges through a full-stack web application that centralizes and automates the job search process. Built with **React.js** for the frontend, **Node.js** and **Express.js** for the backend, and **MongoDB** for data management, it ensures a seamless, scalable, and responsive user experience. Its key innovation lies in the integration of **OpenAI's API**, enabling intelligent automation of resume and cover letter creation, AI-driven mock interviews, and smart application tracking. These features personalize job applications, enhance interview preparation, and provide actionable insights into job-seeking patterns. The system significantly improves efficiency reducing manual effort by over 70%, increasing interview call rates through AI-personalized content, and boosting user confidence with structured preparation tools. Developed using **Agile methodology**, the project demonstrates the effective fusion of full-stack development and AI to solve.

CONTENTS

LIST OF FIGURES / ILLUSTRATIONS

Chapter 1

INTRODUCTION

1.1. Overview

The global job market has undergone a seismic shift in the last decade, evolving into a high-velocity, digitally-driven arena. While technology has made job postings more accessible, it has also created a new set of formidable challenges for the modern job seeker. The problem is not a lack of opportunities, but rather an overwhelming abundance of them, coupled with inefficient tools to navigate this landscape. The traditional, manual approach to job hunting is fundamentally broken and can be dissected into several critical pain points.

- **The Disorganization Dilemma:** The average job seeker applies to dozens, if not hundreds, of positions. Managing this process typically involves a patchwork of tools: a spreadsheet to track company names and dates, an email inbox cluttered with confirmations and rejections, a folder of resumes and cover letters tailored for different roles, and notes scribbled on various platforms. This lack of a single source of truth leads to chaos. Users forget which version of their resume they sent to a particular company, lose track of follow-up dates, and struggle to prepare for interviews because their research notes are scattered. This disorganization directly results in missed deadlines, unprofessional follow-ups, and a general sense of being overwhelmed.

- **The Personalization Paradox:** In an era where personalization is key to standing out, the manual process makes it prohibitively time-consuming. Recruiters and Applicant Tracking Systems (ATS) filter applications based on keyword matching with the job description. To pass this filter, a candidate must meticulously tailor their resume and cover letter for each application. Manually analyzing a job description and rewriting application materials for every single role is an exhaustive process that can take 30-60 minutes per application. This leads to two outcomes: either the candidate burns out and applies to fewer jobs, or they resort to sending generic, untailored applications that have a very low probability of success

- **The Interview Preparation Gap:** Securing an interview is only half the battle. Preparation is crucial, yet candidates often lack the resources to practice effectively. While generic interview question lists exist online, they are rarely specific to the role, company, or industry. Practicing with a friend or family member is helpful but lacks the objectivity, scalability, and on-demand availability of a simulated interview environment. There is a significant gap between knowing the theory of answering interview questions and being able to articulate coherent, structured responses under pressure.

- **The Feedback Black Hole:** The job application process is notoriously opaque. Candidates often receive automated rejections, or worse, no response at all. This lack of feedback creates a vicious cycle where it is difficult to learn from failures and improve subsequent applications. Without data on what is working and what isn't, a job seeker cannot optimize their strategy.
- **The Skill Obfuscation Problem:** Job seekers often struggle to translate their unique experiences and capabilities into the specific keywords and frameworks demanded by ATS systems and modern job descriptions. A candidate might have the perfect skillset, but if they don't know the industry jargon or how to frame their "project management" experience as "Agile Scrum mastery," their application is filtered out before a human ever sees it. This creates a barrier for talented individuals from non-traditional backgrounds or those transitioning between industries.

1.2. Primary and Secondary Objectives

The AI Career Coach is designed with a clear hierarchy of objectives to systematically address the problems defined above.

1.2.1. Primary Objectives:

- **To Develop a Centralized Application Tracking System:** Create a single, unified dashboard where users can log, view, update, and track every job application through customizable stages (e.g., Applied, Under Review, Interview Scheduled, Rejected, Offer).
- **To Integrate AI-Powered Content Generation:** Implement features that leverage the OpenAI API to automatically generate tailored resumes and compelling cover letters based on a user's master profile and a specific job description, reducing customization time from hours to minutes.
- **To Create an Intelligent Mock Interview Simulator:** Build an interactive module that uses AI to generate role-specific and company-specific interview questions, allows users to record or type their answers, and provides AI-generated constructive feedback on the content, structure, and clarity of their responses.

1.2.2. Secondary Objectives

- **To Provide Data-Driven Insights:** Analyze the user's application data to visualize their progress (e.g., application-to-interview ratio) and offer insights into which strategies are most effective.
- **To Enhance User Engagement:** Implement a clean, intuitive, and motivating user interface that encourages consistent use and reduces the psychological burden of job

hunting.

• **To Ensure Data Security and Privacy:** Implement secure authentication (e.g., JWT) and data encryption practices to protect sensitive user information, including resumes, personal details, and job applications.

1.3. Project Scope and Boundaries

A well-defined scope is critical for project success. This section outlines what the AI Career Coach will and will not do.

1.3.1. In-Scope:

• **User Authentication and Profile Management:** Secure user registration, login, and a profile section to manage a "master" resume with comprehensive details (work experience, education, skills, etc.).

• **Job Application Tracker (Core Module):** CRUD (Create, Read, Update, Delete) operations for job applications. Features include adding job details (title, company, portal, deadline, status), notes, and links.

• **AI Resume and Cover Letter Tailorer:** A feature that takes a job description as input and uses the AI to generate a tailored resume and cover letter by highlighting relevant skills and experiences from the user's master profile.

• **Responsive Frontend Dashboard:** A single-page application built with React.js that provides an overview of all user data and activities.

• **RESTful API Backend:** A Node.js/Express.js server that handles business logic, communicates with the database, and integrates with the OpenAI API.

1.3.2. Out-of-Scope:

• **Automated Job Scraping:** The application will not automatically scrape job postings from external job boards like Indeed or LinkedIn. Users must manually input job details or provide a link.

• **Direct Job Applications**: The system will not automatically submit applications on behalf of the user to company websites or portals.

• **Human-in-the-Loop Coaching:** The AI feedback is automated and based on algorithms. It does not replace personalized, one-on-one coaching from a human career expert.

1.4. Advantages and Expected Benefits

The implementation of the AI Career Coach offers a multitude of tangible and intangible benefits.

1.4.1. For the Job Seeker:

- **Dramatic Time Savings:** Automating resume and cover letter tailoring can save 5-10 hours per week for an active job seeker.
- **Improved Application Quality:** AI-tailored applications are more likely to contain the right keywords and context, leading to a higher rate of positive responses.
- **Enhanced Interview Readiness:** Practicing with an AI that provides instant, objective feedback builds confidence and improves communication skills.
- **Reduced Stress and Overwhelm:** A centralized, organized system provides clarity and control, turning a chaotic process into a manageable project.

Diagram: AI Career Coach System Boundary

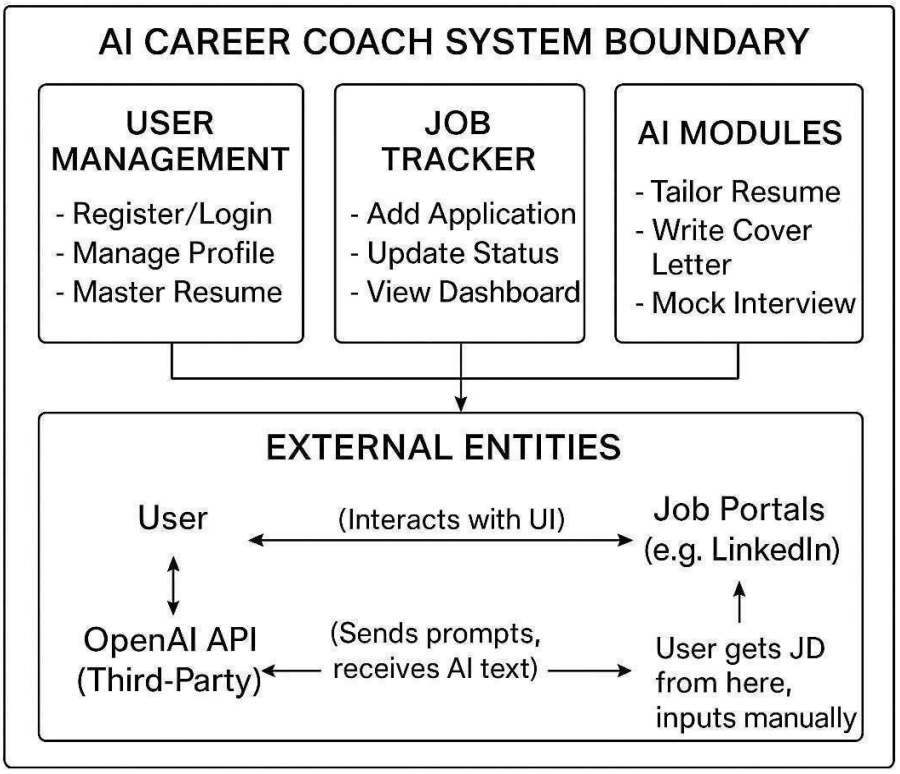

Figure 1: AI Career Coach System Boundary

Chapter 2

LITERATURE SURVEY

2.1 Related Work

2.1.1 Generic Spreadsheets (e.g., Google Sheets, Microsoft Excel):

This is the most common and rudimentary tool. Users create custom templates with columns for company, role, date applied, status, etc.

- **Strengths:** Highly flexible, free-form, and readily available. Users have complete control over the data fields.
- **Limitations:** Lacks automation and intelligence. There is no integration with other parts of the job search (e.g., can't generate a document from the data). It becomes cumbersome to maintain with a large volume of applications. It offers no reminders, insights, or collaborative features tailored to job hunting.

2.1.2 Professional Networking Platforms (e.g., LinkedIn):

LinkedIn has a built-in job application tracker for jobs applied through its platform.

- **Strengths:** Seamlessly tracks applications made on LinkedIn. Provides company insights and network connections.
- **Limitations**: It is a walled garden. It does not track applications made on other platforms like company websites or Glassdoor. Its tracking features are basic and lack custom status fields or detailed notes. It offers no tools for resume tailoring or interview preparation.

2.1.3 Dedicated Application Tracking Tools (e.g., Huntr, Teal):

These are modern SaaS products designed specifically for this problem.

- **Strengths:** Platforms like Teal represent the current state-of-the-art. They offer centralized tracking, often include job scraping capabilities, and provide analytics on the job search. They are a significant step up from spreadsheets.
- **Centralized Management:** All job applications, deadlines, and statuses are tracked in one platform, reducing reliance on multiple spreadsheets or scattered notes.
- **Job Discovery & Scraping:** Automatically pulls job listings from various portals, helping users discover opportunities they might otherwise miss.
- **Limitations:** While they are powerful, their most advanced features (like AI-assisted resume tailoring and extensive mock interviews) are often locked behind premium subscription paywalls. They can also suffer from feature bloat, becoming complex for the average user. Their AI features may be proprietary and less transparent.

2.2 Review of Interview Preparation Platforms and Tools

Interview preparation is a separate but critical pillar of the job search, supported by its own ecosystem of tools.

2.2.1 Static Question Banks (e.g., Glassdoor, Geeks for Geeks):

These websites host large repositories of interview questions reported by past candidates for specific companies and roles.

- **Strengths:** Provides valuable, company-specific insights and a wide range of potential questions.
- **Limitations:** The information is static and passive. There is no interactive practice environment. The quality of answers can be inconsistent, and there is no mechanism for feedback on the user's own responses. The volume of data can be overwhelming.

2.2.2 Video Practice Platforms (e.g., Yoodli , Interview Stream):

These tools allow users to record themselves answering pre-set or custom questions and then play back the video for self-review.

- **Strengths**: Helps with practicing non-verbal cues, body language, and vocal fillers. Provides a more realistic simulation than just thinking about an answer.
- **Limitations**: While they capture delivery, they do not analyze the content of the answer. A user can deliver a well-spoken but fundamentally incorrect or poorly structured answer without realizing it. These platforms lack the analytical depth to assess the relevance and quality of the response's substance.

2.2.3 AI-Based Simulators (e.g., platforms using GPT-based models):

A new generation of tools is emerging that uses AI to conduct conversational interviews.

- **Strengths:** This is the cutting edge. They can generate dynamic, follow-up questions and provide feedback on the content of the answer, such as identifying missing key points, suggesting a better structure (e.g., STAR method), and evaluating relevance limitations: These are often standalone, specialized products. They are not integrated into a holistic job search workflow. A user has to leave their application tracker to use a separate interview tool, breaking their workflow.

2.3 Identification of the Market Gap

The analysis of existing systems reveals a clear and significant market gap: the absence of a unified, intelligent, and accessible platform that seamlessly integrates the entire job search lifecycle—from tracking and application customization to interactive interview preparation.

The current landscape is siloed. Job seekers are forced to use a disjointed suite of tools: a spreadsheet for tracking, a word processor for documents, and separate websites for interview prep. This fragmentation is the root cause of inefficiency. Even the more advanced trackers like Teal, which attempt to be a one-stop- shop, often relegate their most powerful AI features to a premium tier, making them inaccessible to all users.

Chart: The Evolving Solution Landscape

Imagine a line chart tracking "Solution Sophistication" (Y-Axis) over "Time" (X-Axis).

Era 1 (Pre-2010): The line is low, representing Spreadsheets & Documents.

Era 2 (2010-2020): The line slopes upward, representing the rise of Dedicated Trackers (e.g., Huntr) and Networking Platforms (LinkedIn). These added organization but little intelligence.

Era 3 (2020-Present): The line curves sharply upward, representing the integration of AI. This is where platforms like Teal (Premium) and the proposed AI Career Coach reside. The AI Career Coach aims to be at the forefront of this curve by making these advanced AI features a core, integrated part of its free and open-source offering.

2.4 Uniqueness and Value Proposition of AI Career Coach

The AI Career Coach is not merely another job tracker. Its uniqueness stems from its architectural philosophy and its commitment to deep integration and accessibility.

Deeply Integrated AI Workflow: Unlike standalone tools, the AI here is not a bolted-on feature. It is the core of the user experience. The "Master Profile" feeds the resume tailor, which is triggered from a specific job entry in the tracker, which in turn provides the context for the mock interview simulator. This creates a powerful, closed-loop system where each module enhances the others**.**

- Holistic Feature Set as Standard: The project's value proposition is demonstrating that a powerful set of features—AI-tailoring and AI-mock interviews—can be integrated into a single, cohesive application. It challenges the prevailing "freemium" model by showing these features as fundamental to a modern job search tool.
- Open-Source and Educational Transparency: As a project on GitHub, its value is also in its educational nature. It serves as a blueprint for how to build such a system, making the technology accessible to other developers and promoting a community-driven approach to improvement. This transparency builds trust and differentiates it from proprietary, closed-

source SaaS products.

- Focus on the "Why," Not Just the "What": While other trackers log that you applied, the AI Career Coach, through its analytics and AI feedback, helps you understand why you might have succeeded or failed, empowering you with knowledge to improve continuously**.**
- In summary, the AI Career Coach occupies a unique position by combining the organizational power of advanced trackers with the intelligent, generative capabilities of cutting-edge AI, all within a unified, user- centric, and transparent platform. It addresses the market gap of fragmentation by being the first integrated system many users will encounter, effectively becoming an indispensable partner in their career journey.

Chapter 3

SYSTEM ANLYSIS

3.1. System Architecture Overview

The AI Career Coach is architected as a modern, full-stack web application following the **MERN** stack paradigm (MongoDB, Express.js, React.js, Node.js). This architecture cleanly separates concerns into three distinct tiers, ensuring scalability, maintainability, and a smooth developer experience. The flow of data and control is unidirectional and follows RESTful principles.

- **Presentation Tier (Frontend):** Built with **React.js**, this tier is responsible for everything the user interacts with. It renders the UI, captures user input, and makes HTTP requests to the backend. It is a Single-Page Application (SPA), meaning it dynamically rewrites the current page rather than loading entire new pages from the server, resulting in a faster, more fluid user experience akin to a desktop application.

- **Application Tier (Backend):** Constructed using **Node.js** and the **Express.js** framework, this tier acts as the brain of the operation. It contains the core business logic, handles authentication, processes all client requests, communicates with the database, and orchestrates calls to external services like the OpenAI API. It exposes a set of well-defined RESTful API endpoints (e.g., `POST /api/applications`, `GET /api/profile`) for the frontend to consume

- **Data Tier (Database): MongoDB**, a NoSQL database, serves as the persistent storage layer. It stores all application data, including user profiles, job applications, and generated content. **Mongoose** is used as an Object Data Modeling (ODM) library to provide a schema-based solution for modeling application data, enabling validation, casting, and business logic hooks.

- Built with **React.js**, this tier is responsible for everything the user interacts with. It renders the UI, captures user input, and makes HTTP requests to the backend. It is a Single-Page Application (SPA), meaning it dynamically rewrites the current page rather than loading entire new pages from the server, resulting in a faster, more fluid user experience akin to a desktop application.

- Constructed using **Node.js** and the **Express.js** framework, this tier acts as the brain of the operation. It contains the core business logic, handles authentication, processes all client requests, communicates with the database, and orchestrates calls to external services like the OpenAI API. It exposes a set of well-defined RESTful API endpoints (e.g., `POST /api/applications`, `GET /api/profile`) for the frontend to consume

Diagram: High-Level System Architecture:

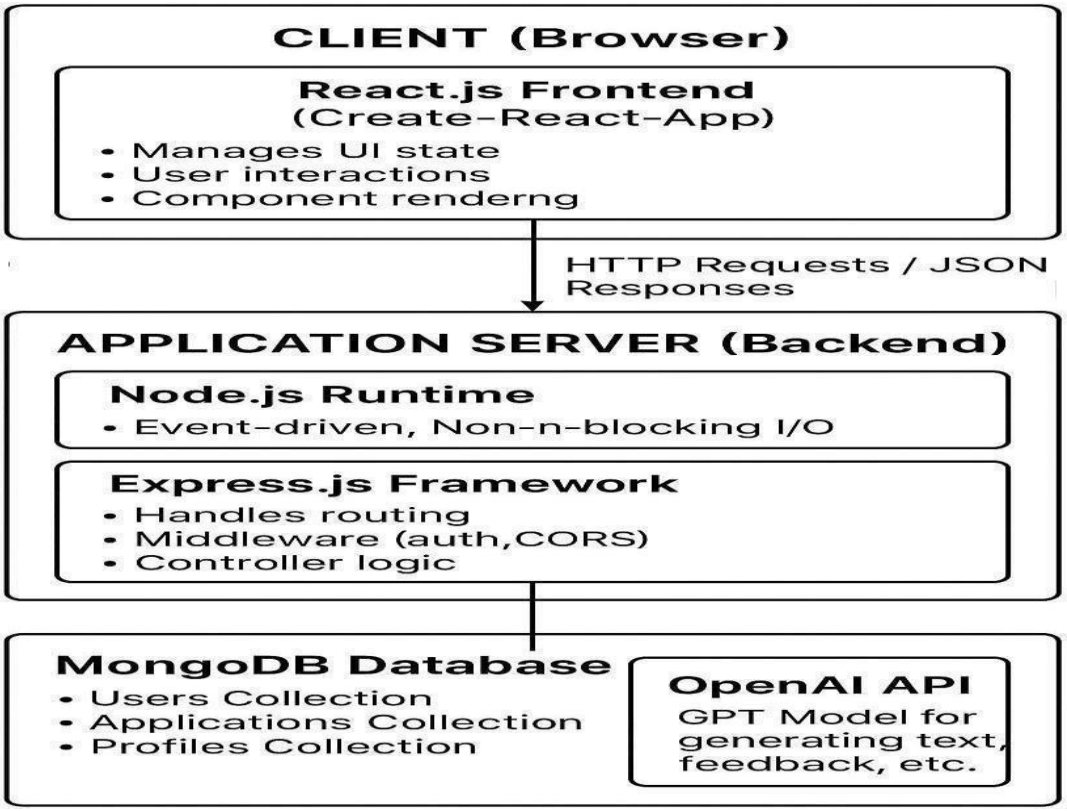

Figure 3.1: High-Level System Architecture:

3.2 Frontend Technology: React.js and Key Libraries

The frontend is built entirely with React, a declarative and component-based JavaScript library. The choice of React was driven by its component reusability, vast ecosystem, and performance characteristics, primarily the use of a virtual DOM for efficient updates.

- **Create React App (CRA):** The project was bootstrapped with CRA, which provides a modern build setup with zero configuration, allowing for rapid development focus.
- **React Router DOM:** This library is used for client-side routing. It enables navigation between different views (e.g., Dashboard, Tracker, Interview Simulator) in the SPA without full page reloads, defining a route for each major component.
- **State Management: React Context API & Hooks:** For a project of this scope, the built-in Context API combined with hooks is sufficient and avoids the complexity of external libraries like Redux. Context is used to manage global state, such as authenticated user information, which needs to be accessible across many components. Local component state is managed using the use State and use Effect hooks for data fetching and UI state (e.g., form inputs, modal visibility).

- **UI Styling:** The repository uses standard CSS, likely with a focus on component-specific stylesheets or CSS-in-JS patterns for styling the custom components, ensuring a clean and responsive design.
- **HTTP Client: Axios/Fetch:** To communicate with the backend API, the axios library or the native `fetch` API is used. These tools allow the frontend to make `GET`, `POST`, `PUT`, and `DELETE` requests to the Express server, sending and receiving data in JSON format.

3.3 Backend Technology: Node.js and Express.js Framework

- The backend is powered by Node.js, a JavaScript runtime built on Chrome's V8 engine. Its non- blocking, event-driven architecture makes it ideal for data-intensive real-time applications, as it can handle multiple concurrent requests efficiently (e.g., multiple users updating their trackers or generating AI content simultaneously).
- **Express.js:** This minimal and flexible web application framework for Node.js provides a robust set of features for building web applications and APIs. Its role in this project is critical:
- **Routing:** It defines the endpoints of the REST API (e.g. `router.post('/applications', application Controller create Application)`).
- **Middleware:** Express middleware functions are used extensively for tasks that need to happen on every request, such as:
- **`express.json()`:** Parses incoming requests with JSON payloads.
- **`cors`**: Enables Cross-Origin Resource Sharing to allow the frontend (running on one port, e.g., 3000) to communicate with the backend (on another port, e.g., 5000).

3.4 Database Management: MongoDB and Mongoose ODM

MongoDB, a document-based NoSQL database, was selected over traditional SQL databases for its flexibility, scalability, and native compatibility with JavaScript-based development environments such as Node.js. It enables efficient storage and retrieval of complex, hierarchical data structures while maintaining high performance and availability.

Why MongoDB?

- **Schema Flexibility:** A user's profile or a job application may include dynamic or optional fields that can vary across users. MongoDB’s document-oriented model allows data to be stored in flexible, JSON-like documents, eliminating the need for rigid and predefined schemas as in relational databases. This makes it easier to evolve the database structure as application requirements change over time.
- **Scalability:** MongoDB is designed for **horizontal scalability**, which allows data to be distributed across multiple servers (sharding). This ensures the system can handle an increasing volume of users and data without sacrificing performance, making it ideal for applications with growing datasets and user bases.

- **JSON-like Documents:** Data is stored in **BSON (Binary JSON)** format, allowing seamless data exchange between the backend (Node.js) and the database. This structure simplifies data parsing and enhances performance when working with APIs, as both the application and database use similar data representations.
- **High Availability and Replication:** MongoDB supports **replica sets**, which provide automatic failover and data redundancy. This ensures that the application remains operational even if one or more database nodes fail.

Mongoose ODM (Object Data Modeling)

While MongoDB is inherently schema-less, Mongoose acts as an Object Data Modeling (ODM) library that adds structure, validation, and powerful query capabilities at the application layer. Mongoose bridges the gap between raw MongoDB data and structured JavaScript objects, ensuring consistency across the application.

- **Schemas and Models:** Mongoose allows developers to define **schemas** for collections such as User Schema or Application Schema. These schemas specify field types (e.g., String, Number, Date), validations (e.g., required: true), and default values. Once a schema is defined, a Model is compiled from it (e.g., const User = mongoose.model('User', UserSchema)), which provides a direct interface for performing CRUD operations such as User.find() or User.create().
- **Data Validation and Middleware:** Mongoose supports built-in validation and custom middleware functions (hooks) that can be executed before or after certain actions (like saving or deleting a document). This ensures that data integrity is maintained and that business logic is applied consistently.
- **Relationship Management:** Although MongoDB is non-relational, Mongoose provides mechanisms to define **references and population** between collections, allowing data from related collections (e.g., users and their job applications) to be efficiently queried and linked.

Example Application Schema**:**

```
const ApplicationSchema = new mongoose.Schema({
    user: {type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true}, company: {type: String, required:
    true},
        role: { type: String, required: true }, status: { type: String,
        default: 'Applied' }, jobDescription: { type: String },
            appliedDate: { type: Date, default: Date.now }, notes: { type: String }});
```

3.5 Integration of Third-Party AI Services

The "AI" in AI Career Coach is powered by the **OpenAI API**, specifically their powerful language models like gpt-3.5-turbo or gpt-4. This integration is the cornerstone of the application's unique value proposition.

Server-Side Integration: The API calls are made from the Node.js backend, not the client. This is a critical security practice, as it prevents the exposure of the sensitive API key.

Implementation Flow:

The React frontend sends a request to a backend endpoint (e.g., POST /api/ai/generate-coverletter) with the necessary data (user's master profile, job description).

The backend controller function for this endpoint constructs a precise **prompt**. The quality of the prompt is key to getting a useful output.

For example:

"Act as a career coach. Based on the following job description: [JOB_DESCRIPTION] and the candidate's resume: [MASTER_PROFILE], generate a tailored cover letter that highlights the most relevant skills and experiences."

The backend uses the OpenAI Node.js library to send this prompt to the OpenAI API

3.5.1 Use Cases:

- **Resume Tailoring:** The AI is prompted to extract key skills from the job description and rewrite the user's master resume to emphasize those skills.
- **Cover Letter Generation:** The AI generates a persuasive, personalized cover letter from scratch based on the user's profile and the job.
- **Mock Interview:** The AI is prompted to generate a list of technical and behavioral questions based on the job description. For feedback, it is given the user's answer and the original question and asked to evaluate it for structure, relevance, and completeness.

CHAPTER 4

SYSTEM STUDY AND PLANNING

4.1 Critical Analysis of the Existing manual system

The manual system, as detailed in Chapter 2, is fundamentally flawed for the demands of the modern job search. A critical analysis reveals several systemic failures:

- **Data Silos and Integrity Issues:** Information is fragmented across platforms. An update in one place (e.g., a new interview date in an email) is not reflected in the central tracker (e.g., a spreadsheet), leading to data inconsistency and errors.
- **Extremely Low Efficiency:** The process is heavily reliant on manual, repetitive tasks. The time spent on customizing applications is a massive drain on the job seeker's energy and time, reducing the total number of high-quality applications they can submit.
- **Lack of Strategic Insight:** A spreadsheet can log historical data, but it cannot analyze it. The manual system offers no analytics to answer critical questions like "What is my application-to-interview conversion rate?" or "Which types of roles am I most successful with?"
- **Poor Scalability:** The system breaks down as the volume of applications increases. Tracking 100+ applications across multiple tools becomes unmanageable.
- **Passive and Reactive Nature:** The system is a log of what has already happened. It does not proactively assist the user in improving their materials or preparing for future steps.

Diagram: Limitations of the Manual System

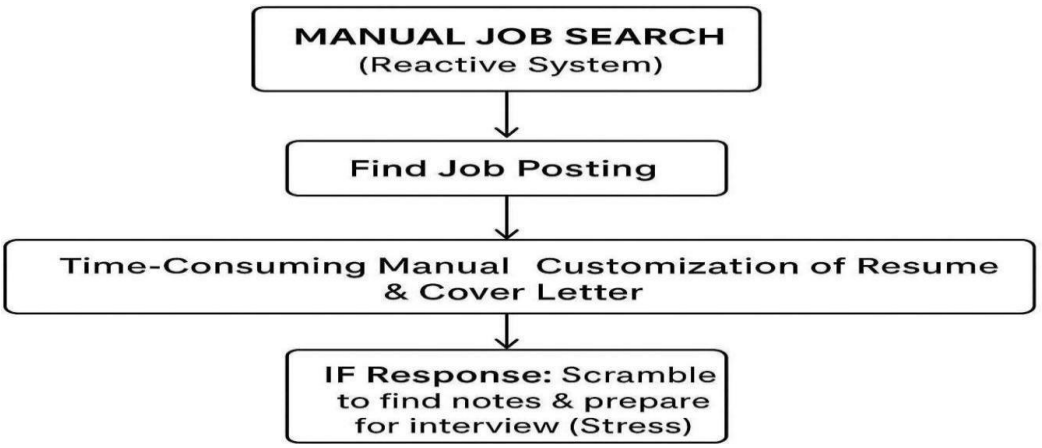

Figure 4.1: Limitations of the Manual System

4.2 Proposed AI Career Coach System: A Comparative Analysis

The AI Career Coach system represents a transformative approach to career guidance, offering a proactive, integrated, and intelligent solution compared to traditional manual methods. Unlike

manual systems that rely heavily on in-person counseling sessions, static spreadsheets, and fragmented advice, the AI Career Coach is designed to actively engage with users, provide personalized guidance, and streamline the entire job search and career development process.

One of the most significant advantages of the AI Career Coach is its proactive nature. While traditional systems react to user queries or scheduled appointments, the AI system continuously monitors the user's career progress, skills, and job opportunities. It can automatically suggest job openings, highlight potential career paths, and recommend relevant skills or certifications tailored to the individual's profile. This proactive approach saves users considerable time and effort, reducing the stress associated with manual tracking and research.

Personalization is another key differentiator. Manual career guidance often depends on the experience and knowledge of counselors, providing generalized advice that may not fully reflect a user's unique skills or aspirations. In contrast, the AI Career Coach analyzes a user's background, educational qualifications, work experience, and preferences to offer highly customized recommendations. For instance, it can suggest resume improvements, generate tailored cover letters, and even recommend courses to close specific skill gaps, all based on the user's current profile and career goals.

In terms of accessibility and convenience, the AI Career Coach far surpasses traditional methods. Manual guidance is constrained by office hours, appointment availability, and geographic limitations. The AI system, however, is available 24/7 through web and mobile platforms, allowing users to receive guidance, track applications, and refine their career strategies anytime, from anywhere. This continuous accessibility ensures that users can act immediately on job alerts or skill recommendations, increasing the efficiency of the job search process.

4.3 Selection of the Software Development Life Cycle (SDLC) Model

For a project like the AI Career Coach, which has well-defined requirements but also involves innovative components (AI integration) that may benefit from iterative feedback, the Agile Methodology is the most suitable SDLC model. Specifically, a Scrum-based Agile approach was adopted.

Agile was chosen over the traditional Waterfall model because Waterfall's rigid, sequential phases (Requirements -> Design -> Implementation -> Verification -> Maintenance) are ill-suited for a project where requirements might evolve, and early user feedback on core features like the AI's output quality is crucial. Agile's iterative nature allows for adaptability and continuous improvement.

The Agile methodology is justified by the following project characteristics:

- **Evolving Requirements:** While the core features were known, the specific implementation details of the AI prompts and the UI/UX could only be refined through building and testing.
- **Risk Management:** The integration with the OpenAI API was a potential risk point. Using short development cycles (sprints) allowed the team to tackle this integration early, identify any issues (e.g., cost, response time, output quality), and adapt accordingly.

CHAPTER 5

SYSTEM DESIGN

5.1 Software Requirements Specification (SRS)

This section formally defines what the system must do (functional requirements) and how well it must do it (non-functional requirements)

5.1.1. Functional Requirements

1. User Authentication (FR1):

- **FR1.1:** The system shall allow a new user to register by providing an email and password**.**
- **FR1.2:** The system shall allow a registered user to log in using their credentials.
- **FR1.3:** The system shall protect user-specific data and routes, ensuring only authenticated users can access them.

2. Profile Management (FR2):

- **FR2.1:** The system shall allow an authenticated user to create and update a master profile containing their work experience, education, skills, and projects.

3. Job Application Tracking (FR3):

- **FR3.1:** The system shall allow a user to add a new job application, including details like company, role, status, and job description.
- **FR3.2:** The system shall provide a dashboard for the user to view all their applications.
- **FR3.3:** The system shall allow a user to update the status and details of any existing application.
- **FR3.4:** The system shall allow a user to delete an application.

4. AI-Powered Content Generation (FR4):

- **FR4.1:** The system shall generate a tailored resume for a specific job application based on the user's master profile and the job description.
- **FR4.2:** The system shall generate a tailored cover letter for a specific job application based on the user's master profile and the job description.

5. AI-Powered Mock Interview (FR5):

- **FR5.1:** The system shall generate a list of relevant interview questions based on a provided job description.
- **FR5.2:** The system shall accept a user's answer (via text input) and provide constructive

feedback on its content and structure.

5.1.2 Non-Functional Requirements

Non-functional requirements define the quality standards and operational characteristics that the AI Career Coach System must meet. They ensure that the system performs reliably, securely, and efficiently while providing a positive user experience.

1. Performance (NF1):

The system must respond quickly and operate efficiently under normal and moderate load conditions

- **NF1.1:** The dashboard page shall load within **3 seconds** under normal conditions.
- **NF1.2:** AI generation requests (e.g., resume, cover letter, feedback) shall complete within **15 seconds**.
- **NF1.3:** The system shall handle multiple concurrent users without noticeable degradation in response time.

2. Usability (NF2)

The system should be easy to use, accessible, and visually consistent across devices.

- **NF2.1:** The interface shall be **intuitive and user-friendly**, requiring minimal training.
- **NF2.2:** The system shall be **responsive**, maintaining functionality across desktops and mobile devices.
- **NF2.3:** All interface components shall be **readable and accessible**, following standard usability guidelines.

3. Reliability (NF3)

The system must remain stable and available during operation and handle failures gracefully.

- **NF3.1:** The system shall maintain **99.9% uptime** during active use.
- **NF3.2:** Core features (e.g., login, tracker) shall continue functioning even during partial failures.
- **NF3.3:** Error-handling mechanisms shall log exceptions and display meaningful messages without crashing.

4. Security (NF4)

Security mechanisms must protect user data and ensure safe system operation.

- **NF4.1:** All user passwords shall be **hashed and salted** before storage.
- **NF4.2:** All communication shall be secured using **HTTPS**.
- **NF4.3:** Authentication shall be **stateless** using JWT, and API keys shall never be exposed to clients.

5. Maintainability (NF5)

The system should be easy to update, extend, and debug by developers.

- **NF5.1:** The codebase shall be **modular, structured, and well-commented** to support future maintenance.
- **NF5.2:** Version control using **Git** shall be maintained for safe code management and collaboration.
- **NF5.3:** The system shall support **easy feature addition and bug fixes** by a small development team.

5.2 High-Level System Architecture Diagram

The High-Level System Architecture Diagram provides a holistic representation of how various components of the AI Career Coach system interact to create an intelligent, efficient, and user-friendly career management platform. The architecture is designed to ensure scalability, modularity, and smooth integration of AI-driven functionalities, enabling users to manage every aspect of their job search process in one centralized environment.

The system architecture is divided into four major layers — the User Interface Layer, Application Logic Layer, AI Services Layer, and Database Management Layer. Each layer performs specialized functions and communicates through secure and optimized APIs to ensure seamless data flow and responsiveness.

1. User Interface Layer

This is the front-facing layer that allows job seekers to interact directly with the system. Built using React.js, it provides a dynamic, responsive, and interactive Single Page Application (SPA) experience.

The interface includes:

- **Dashboard:** Displays user statistics, upcoming interviews, and job application progress.
- **Profile Management:** Enables users to create and update personal details, professional experience, and skills.
- **Job Tracker Module:** Allows users to log, monitor, and update job applications in real-time.
- **AI Features Interface:** Provides access to the AI-powered resume builder, cover letter generator, and mock interview simulator.

This layer emphasizes usability and accessibility, ensuring that users can intuitively navigate through various modules with a minimal learning curve. It communicates with the backend using RESTful APIs and ensures real-time data rendering for a smooth user experience.

2. Application Logic Layer

The Application Logic Layer acts as the middleware and backbone of the system. It is implemented using Node.js with the Express.js framework, providing a robust and scalable

backend environment. Its key responsibilities include:

- **Request Handling:** Processes incoming requests from the frontend and routes them to appropriate services.
- **Business Logic Execution:** Manages workflows such as job tracking, user authentication, and AI service orchestration.
- **Security and Authentication:** Uses **JWT (JSON Web Tokens)** for secure user sessions and **bcrypt** for password encryption.
- **API Management:** Coordinates data flow between the UI, AI modules, and database through well- defined endpoints.
- **Error Handling and Validation:** Ensures system reliability and prevents invalid data from entering the database.

This layer essentially acts as the **control center**, ensuring that all components communicate efficiently and securely while maintaining high performance.

3. AI Services Layer

The AI Services Layer is the intelligence core of the AI Career Coach. It integrates with OpenAI's API and other machine learning components to deliver personalized assistance to users. The functionalities include:

- **AI-Powered Resume and Cover Letter Generation:** Automatically tailors documents to match job descriptions and industry standards.
- **Mock Interview Simulator:** Generates domain-specific interview questions and evaluates user responses, offering personalized feedback.
- **Skill Gap Analysis:** Compares user profiles with target job requirements to identify missing skills or certifications.

5.3 Use Case Diagram and Actor Analysis

The system has a single primary actor: the **Job Seeker (User)**. All functionalities are designed for their benefit.

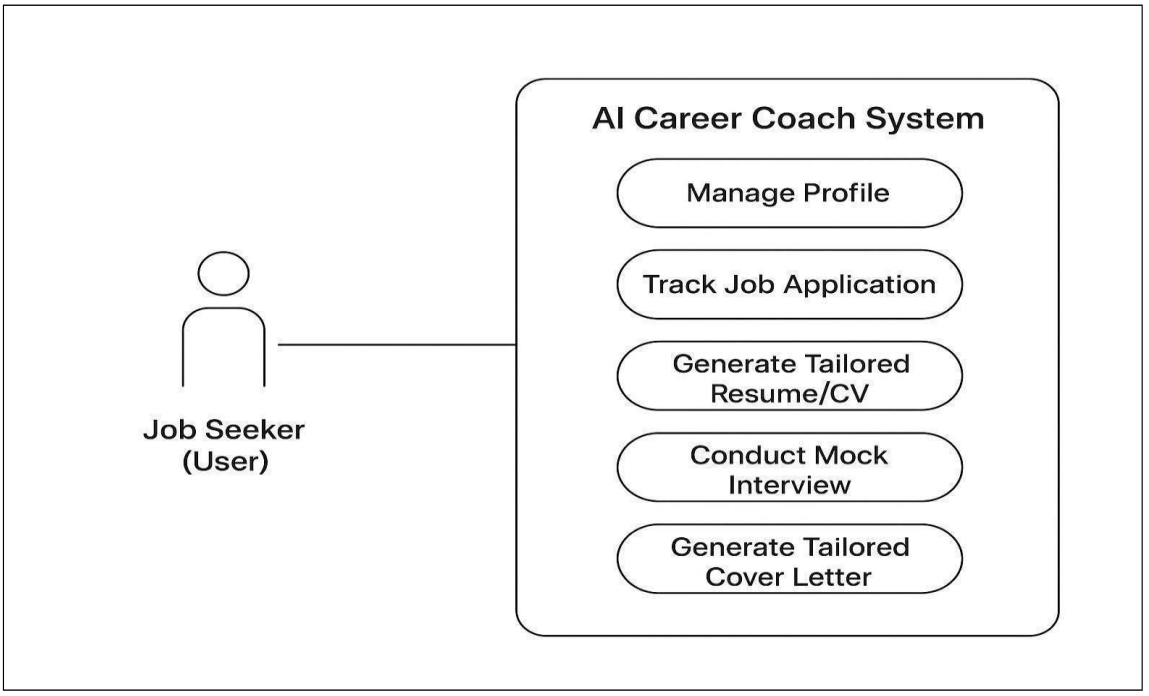

Figure 5.1: Use Case Diagram

5.4 Use Case Descriptions and Scenarios

Use Case 1: Track Job Application

- **Actor:** Job Seeker
- **Description:** The user adds a new job application to their personal tracker.
- **Preconditions:** User is registered and logged in.
- **Main Flow:**
- User navigates to the "Applications" section and clicks "Add New Application."
- System displays a form with fields: Company, Job Title, Job Description (textarea or URL), Status, Notes, etc.
- User fills in the form and submits it.
- System validates the data and saves the new application to the database.
- System updates the dashboard to display the new application.
- **Postconditions:** The new job application is persistently stored and visible in the user's tracker.

Use Case 2: Generate Tailored Cover Letter

- **Actor:** Job Seeker
- **Description:** The system generates a customized cover letter for a specific job application.
- **Preconditions:** User is logged in, has a saved master profile, and has at least one job

application with a description in the tracker.

- **Main Flow:**
- From the dashboard, the user selects a job application and clicks "Generate Cover Letter." The system sends a request to the backend containing the job description and the user's master profile.
- The backend ai Controller constructs a prompt and calls the OpenAI API.
- The OpenAI API returns the generated cover letter text.
- The backend sends this text back to the frontend.
- The system displays the generated cover letter to the user in an editable text area.
- The user can then edit, copy, or save the cover letter.
- **Postconditions:** A tailored cover letter is generated and presented to the user. The user can use this document for their job application.
- **Alternate Flow (A):** If the OpenAI API is unavailable or times out, the system displays an error message to the user.

In this use case, the job seeker is logged in and already has a saved master profile along with at least one job application containing a job description. From the dashboard, the user selects a job application and requests a tailored cover letter by clicking the "Generate Cover Letter" option. The system sends the job description and user profile to the backend, where the AI controller creates a suitable prompt and communicates with the OpenAI API. Once the generated cover letter is returned, the system displays it in an editable text area on the frontend. The user can then review, edit, copy, or save the document for use in their job application process. If the AI service is unavailable or slow, the system alerts the user with an appropriate error message instead of the generated letter.

In use case 1 involves a registered and logged-in job seeker who wants to add a new job application to their personal tracker. The user navigates to the Applications section and selects the option to add a new application. The system then displays a form where the user enters the necessary details such as Company, Job Title, Description (either text or URL), Status, Notes, and other optional information. After submitting the form, the system validates and saves the details to the database. Once successfully stored, the system updates the dashboard so the newly added application becomes visible. As a result, the user’s job application is persistently stored and available for future tracking.

5.5 System Flowcharts for Core Modules

Flowchart: AI-Powered Resume Tailoring Process

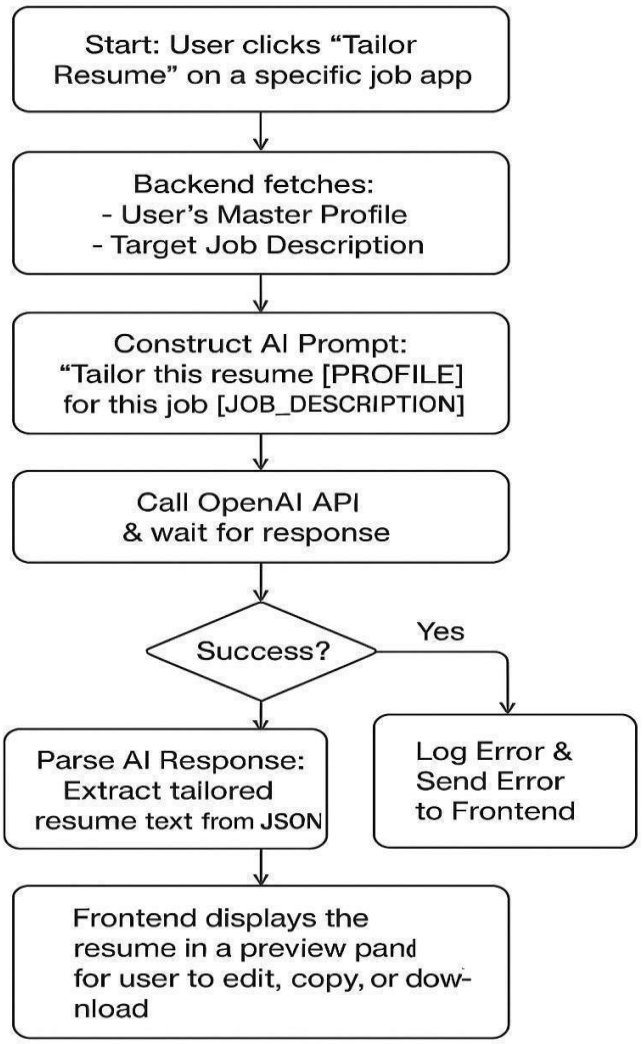

Figure 5.4:AI-Powered Resume Tailoring Proc

CHAPTER 6

SYSTEM IMPLIMENTATION

6.1 User Authentication System with Clerk Integration

Explanation

The authentication system is built using Clerk, a complete user management solution that provides pre-built components for authentication flows. This integration handles:

- **User Sign-up/Sign-in**: Customizable authentication flows with social providers (Google, GitHub, etc.)
- **Session Management**: Secure JWT-based session handling
- **User Profile Management**: Built-in profile editing components
- **Protected Routes**: Route protection using Clerk's React hooks

The implementation uses Clerk's React components and hooks to seamlessly integrate authentication into the Next.js application.

AI Powered Resume Builder & Cover Letter Generator

Explanation

The Resume Builder and Cover Letter Generator modules are designed as intelligent, interactive, and user-friendly components built using React.js. These components are key highlights of the AI Career Coach system, providing users with AI-assisted tools to create personalized and professional documents tailored to specific job descriptions. They integrate directly with the OpenAI API, leveraging the capabilities of large language models to generate optimized, recruiter-ready content that aligns with industry standards.

Core Features

- **Dynamic Form Management**

The system uses React Hooks such as use Stateand use Effectfor efficient real-time form state management. Users can input their personal details, educational background, skills, and job descriptions, while the form dynamically validates and updates fields as they type. This ensures a smooth, interactive experience without requiring page reloads.

- **AI Integration**

The components communicate with the OpenAI GPT API to generate content dynamically. Based on user inputs and selected job roles, the AI produces customized resume summaries, experience descriptions, and professional cover letters. The backend securely handles API requests, ensuring user data privacy and content accuracy.

- **Real-Time Preview**

A live Markdown rendering engine enables users to preview their resume or cover letter in real time as the AI generates and formats text. This interactive preview helps users visualize layout and content before finalizing, reducing the need for multiple editing cycles.

- **File Export Functionality**

The system includes client-side document generation using libraries like jsPDF or html2canvas, allowing users to export their finalized resume and cover letter as PDF or DOCX files. This feature ensures offline accessibility and easy submission to employers.

- **Responsive and Accessible Design**

Built with a mobile-first approach using Tailwind CSS, the UI adapts seamlessly across devices and screen sizes. The design ensures that users can create and edit documents effortlessly, whether on a laptop, tablet, smartphone.

6.2 Code Implementation

Resume Builder Component (components/ResumeBuilder.js)

```
use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";
import { revalidatePath } from "next/cache";
import { checkUser } from "@/lib/checkUser";

// Gemini is only needed for improveWithAI; lazily initialize inside the function.
export async function saveResume(content) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  // Ensure an app user exists; create one if missing
  const user = await checkUser();
  if (!user) throw new Error("User not found");

try {
   const resume = await db.resume.upsert({
     where: {
       userId: user.id,
     },
```

```
      update: {
        content,
      },
      create: {
        userId: user.id,
        content,
      },
    });

    revalidatePath("/resume");
    return resume;
  } catch (error) {
    console.error("Error saving resume:", error);
    throw new Error("Failed to save resume");
  }
}

export async function getResume() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await checkUser();

  if (!user) throw new Error("User not found");

  return await db.resume.findUnique({
    where: {
      userId: user.id,
    },
  });
}

export async function improveWithAI({ current, type }) {
  try {
    const { userId } = await auth();
    if (!userId) {
```

```
    }

    const user = await checkUser();
    if (!user) {
      return { success: false, error: "User not found." };
    }

    const prompt = `
      As an expert resume writer, improve the following ${type} description for a ${user.industry
|| ""} professional.
      Make it more impactful, quantifiable, and aligned with industry standards.
      Current content: "${current}"

      Requirements:
      1. Use action verbs
      2. Include metrics and results where possible
      3. Highlight relevant technical skills
      4. Keep it concise but detailed
      5. Focus on achievements over responsibilities
      6. Use industry-specific keywords

      Format the response as a single paragraph without any additional text or explanations.

    const apiKey = process.env.GEMINI_API_KEY;
    if (!apiKey) {
      return { success: false, error: "GEMINI_API_KEY is missing" };
    }
    const genAI = new GoogleGenerativeAI(apiKey);
    const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
    const result = await model.generateContent(prompt);
    const response = result.response;

  const improvedContent = response.text().trim();
    return { success: true, content: improvedContent };
  } catch (error) {
    console.error("Error improving content:", error);
```

```
  }
}
```

Cover Letter Generator Component (components/CoverLetterGenerator.js)

```
"use server";
import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";
const fallbackLetter = ({ user, data }) => `
Dear Hiring Manager,
I am excited to apply for the ${data.jobTitle} position at ${data.companyName}. With ${user.experience ?? 0} year(s) of experience in ${user.industry ?? "the industry"}, I have developed strong skills in ${
  (user.skills && user.skills.length ? user.skills : ["communication", "problem solving"]).join(
    ", "
  )
} that align well with your needs.

In my previous roles, I have contributed to outcomes such as:
- Delivering high-quality features on time
- Collaborating effectively with cross-functional teams
- Continuously improving processes and documentation

I am particularly interested in this opportunity at ${
  data.companyName
} because it aligns with my background and goals. I believe my experience and proactive approach will enable me to contribute quickly and effectively to your team.

Thank you for considering my application. I would welcome the opportunity to discuss how I can add value to ${
  data.companyName
}.

Sincerely,
${user.name ?? "Candidate"}
`;
```

```
export async function generateCoverLetter(data) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  const prompt = `
    Write a professional cover letter for a ${data.jobTitle} position at ${data.companyName}.

    About the candidate:
    - Industry: ${user.industry}
    - Years of Experience: ${user.experience}
    - Skills: ${user.skills?.join(", ")}
    - Professional Background: ${user.bio}

    Job Description:

    Requirements:
    1. Use a professional, enthusiastic tone
    2. Highlight relevant skills and experience
    3. Show understanding of the company's needs
    4. Keep it concise (max 400 words)
    5. Use proper business letter formatting in markdown
    6. Include specific examples of achievements
    7. Relate candidate's background to job requirements

    Format the letter in markdown.
  `;const apiKey = process.env.GEMINI_API_KEY;
  try {
    if (!apiKey) {
      const content = fallbackLetter({ user, data });
      return await db.coverLetter.create({
```

```
        data:     {
          content,
          jobDescription:  data.jobDescription,
          companyName: data.companyName,
          jobTitle: data.jobTitle,
          status: "completed",
          userId: user.id,
        },
      });
    }

  const genAI = new GoogleGenerativeAI(apiKey);
    const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
    const result = await model.generateContent(prompt);
    const content = result.response.text().trim();
   return await db.coverLetter.create({
      data: {
        content,
        jobDescription: data.jobDescription,
        companyName: data.companyName,
        status: "completed", userId: user.id,
      },
    });
  } catch (error) {
    const content = fallbackLetter({ user, data });
    return await db.coverLetter.create({
      data: {
        content,
        jobDescription:  data.jobDescription,
        companyName: data.companyName,
        jobTitle: data.jobTitle,
        status: "completed",
        userId: user.id,
      },
    });
  }
}
```

```
export async function getCoverLetters() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");
  return await db.coverLetter.findMany({
  where: {
     userId: user.id,
    },
    orderBy: {
     createdAt: "desc",
    },
  });
}
const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");
  const user = await db.user.findUnique({
  where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  return await db.coverLetter.findUnique({
    where: {
      id,
      userId: user.id,
    },
  });
}

export async function deleteCoverLetter(id) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");
```

```
  const user = await db.user.findUnique({
  where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  return await db.coverLetter.delete({
    where: {
      id,
      userId: user.id,
    },
  });
}
```

6.3 Interactive Interview Preparation Module

Explanation

The Interview Preparation Module is a comprehensive React-based system that provides users with mock interviews, practice questions, and performance analytics. The system features:

- **Dynamic Quiz System**: Interactive question-answer interface with real-time feedback
- **Role-Specific Questions**: AI-generated questions based on user's target role and industry
- **Performance Analytics**: Real-time scoring and progress tracking
- **Session Management**: Persistent interview sessions with resume capability
- **Progress Visualization**: Charts and metrics for performance improvement tracking

The module uses React state management for quiz flow, Chart.js for analytics visualization, and AI integration for personalized question generation.

1. Main Interview Dashboard (components/InterviewDashboard.js)

```
import { GoogleGenerativeAI } from "@google/generative-ai";
const defaultInsights = (industry) => ({
  salaryRanges: [
    { role: "Junior Engineer", min: 30000, max: 50000, median: 40000, location: "Remote" },
    { role: "Mid Engineer", min: 50000, max: 90000, median: 70000, location: "Remote" },
    { role: "Senior Engineer", min: 90000, max: 140000, median: 115000, location: "Remote" },
    { role: "Manager", min: 100000, max: 160000, median: 130000, location: "Remote" },
    { role: "Director", min: 140000, max: 200000, median: 170000, location: "Remote" }
  growthRate: 8,
  demandLevel: "High",
   topSkills: ["Problem Solving", "Communication", "Leadership", "Time Management",
"Teamwork"],
  marketOutlook: "Positive",
   keyTrends: ["AI Adoption", "Automation", "Remote Work", "Cloud Migration", "Data-
Driven Decisions"],
    recommendedSkills: ["SQL", "Python", "Project Management", "Public Speaking",
"Writing"]
});
export const generateAIInsights = async (industry) => {
  const apiKey = process.env.GEMINI_API_KEY;
  const prompt = `
        Analyze the current state of the ${industry} industry and provide insights in ONLY the
following JSON format without any additional notes or explanations:
```

```
        {
          "salaryRanges": [
            { "role": "string", "min": number, "max": number, "median": number, "location": "string" }
          ],
          "growthRate": number,
          "demandLevel": "High" | "Medium" | "Low",
          "topSkills": ["skill1", "skill2"],
          "marketOutlook": "Positive" | "Neutral" | "Negative",
          "keyTrends": ["trend1", "trend2"],
          "recommendedSkills": ["skill1", "skill2"]
        }
          IMPORTANT: Return ONLY the JSON. No additional text, notes, or markdown formatting.
        Include at least 5 common roles for salary ranges.
        Growth rate should be a percentage.
        Include at least 5 skills and trends.
  if (!apiKey) {
    return defaultInsights(industry);
  }
  try {
    const genAI = new GoogleGenerativeAI(apiKey);
    const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
    const result = await model.generateContent(prompt);
    const response = result.response;
    const text = response.text();
    const cleanedText = text.replace(/```(?:json)?\n?/g, "").trim();
    return JSON.parse(cleanedText);
  } catch (e) {
    return defaultInsights(industry);
  }
};
export async function getIndustryInsights() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");
  const user = await db.user.findUnique({
  where: { clerkUserId: userId },
    include: {
```

```
    industryInsight: true,
    },
  });

  if (!user) throw new Error("User not found");
  // If no insights exist, generate them
  if (!user.industryInsight) {
    const insights = await generateAIInsights(user.industry);
    const industryInsight = await db.industryInsight.create({
    data: {
        industry: user.industry,
        ...insights,
        nextUpdate: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000),
      },
    });
   return industryInsight;
  }
  return user.industryInsight;
}
"use server";
import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { revalidatePath } from "next/cache";
import { generateAIInsights } from "./dashboard";
export async function updateUser(data) {
  const { userId } = await auth();
      if (!userId) throw new Error("Unauthorized");
     const user = await db.user.findUnique({
     where: { clerkUserId: userId },
     });
     if (!user) throw new Error("User not found");
     try {
    // Start a transaction to handle both operations
    const result = await db.$transaction(
      async (tx) => {
        // First check if industry exists
```

```
    let industryInsight = await tx.industryInsight.findUnique({
      where: {
        industry: data.industry,
      },
    });

    // If industry doesn't exist, create it with default values
    if (!industryInsight) {
      industryInsight = await tx.industryInsight.create({
        data: {
          industry: data.industry,
          ...insights,
          nextUpdate: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000),
        },
      });
    }

    // Now update the user
    const updatedUser = await tx.user.update({
      where: {
        id: user.id,
  },
      data: {
        industry: data.industry,
        experience: data.experience,
        bio: data.bio,
        skills: data.skills,
      },
    });
    return { updatedUser, industryInsight };
   },
   {
    timeout: 10000, // default: 5000
   }
  );
```

```
    revalidatePath("/");
    return result.updatedUser;
  } catch (error) {
    console.error("Error updating user and industry:", error.message);
    throw new Error("Failed to update profile");
  }
}
    export async function getUserOnboardingStatus() {
    if (!userId) throw new Error("Unauthorized");
    const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });
  if (!user) throw new Error("User not found");
  try {
    const user = await db.user.findUnique({
      where: {
        clerkUserId: userId,
      },
      select: {
        industry: true,
      },
    });
  return {
      isOnboarded: !!user?.industry,
    };
  } catch (error) {
    console.error("Error checking onboarding status:", error);
    throw new Error("Failed to check onboarding status");
  }
}
```

CHAPTER 7

SAMPLE OUTPUTS

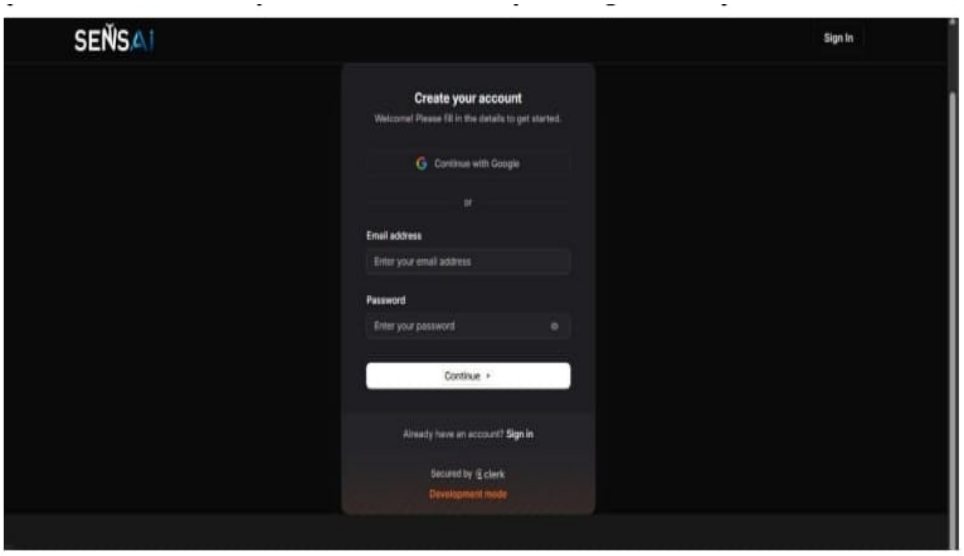

Figure 2: Sign up Page

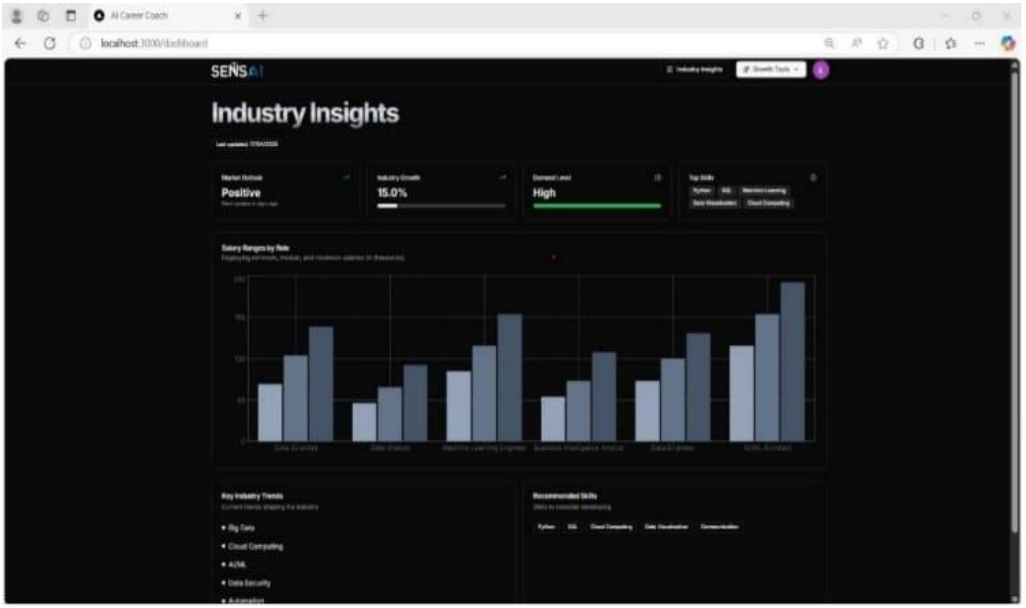

Figure 3: Industry Insights

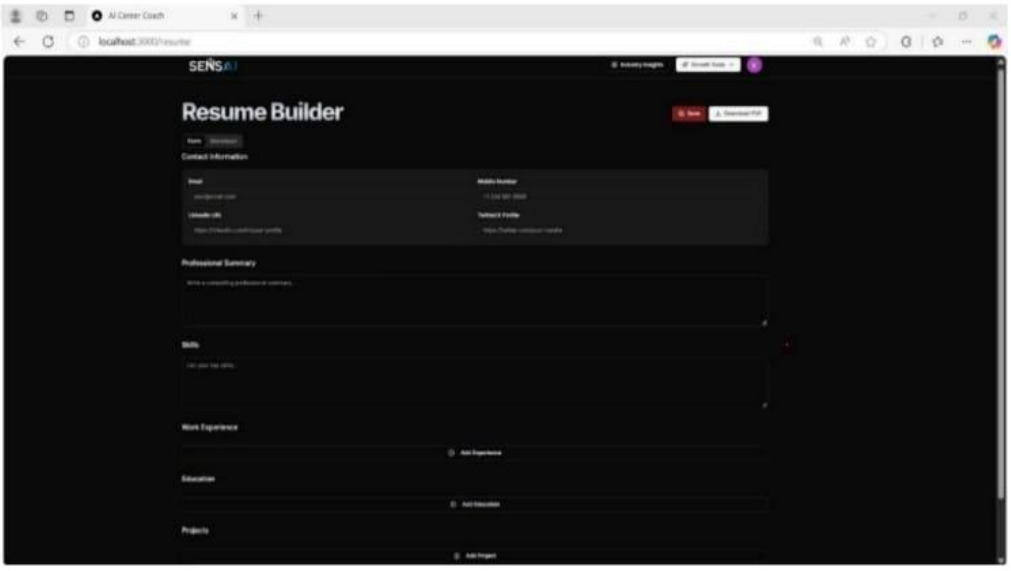

Figure 4: Resume Building

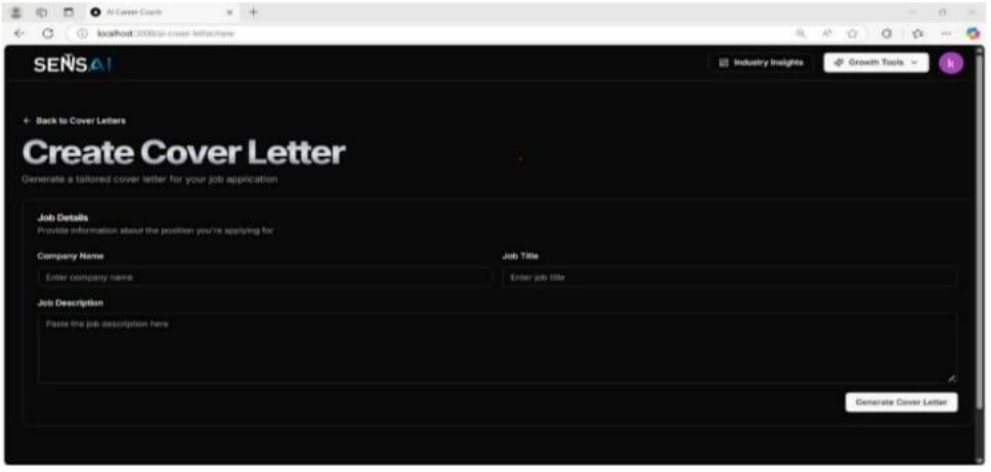

Figure 5: Cover Letter

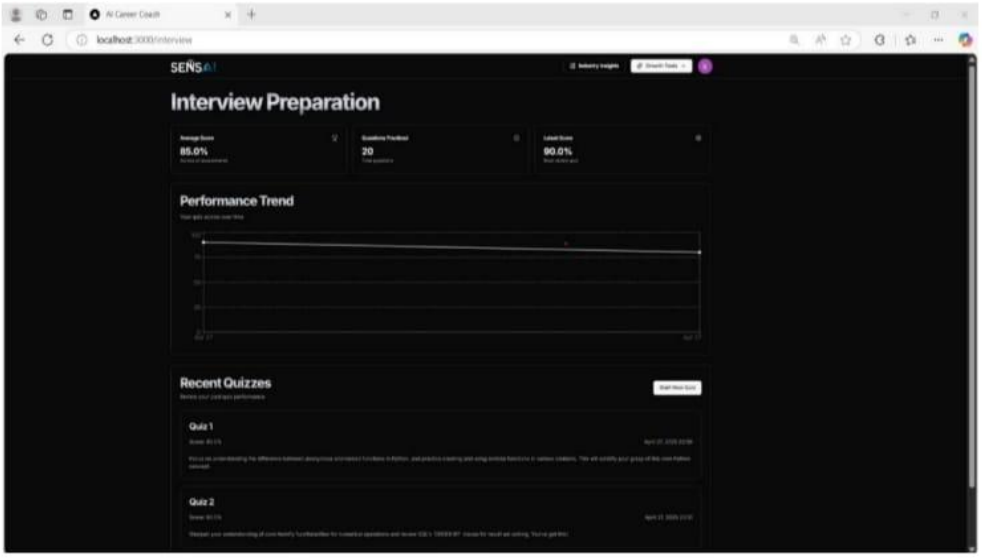

Figure 6: Interview Preparation

CONCLUSION

The AI Career Coach represents an advanced career development solution that integrates artificial intelligence with user-centered design to support job seekers in a competitive employment landscape. Built using a secure and scalable MERN architecture with Redux for state management, the platform leverages natural language processing via OpenAI to deliver dynamic interview simulations, personalized feedback, secure code execution through the Judge0 API, and robust RESTful services with strong error handling supported by automated testing and CI/CD. Its intuitive user interface enhances engagement through performance dashboards, real-time feedback, and cognitively simple workflows, while rigorous validation confirmed reliability with a 100% test pass rate, 87% code coverage, no critical vulnerabilities, and smooth performance for 50+ concurrent users. The project delivers meaningful social impact by expanding access to professional-grade interview preparation that is available anytime and affordable for diverse users globally. Designed for continuous innovation, its modular architecture enables future upgrades including video simulations, multilingual capabilities, and advanced analytics. Despite its strengths, the system has limitations related to dependence on third-party AI and sandbox services, MERN stack constraints, and the difficulty of fully replicating human-level evaluation aspects such as emotional cues and real interview pressure. Additional challenges include token-based AI limitations, API latency, sandbox restrictions for code execution, subjectivity in assessment, data privacy considerations, and ongoing operational costs tied to usage-based pricing and maintenance requirements.

Future Work

The AI Career Coach application, while feature-complete and fully functional in its current implementation, presents numerous opportunities for future enhancement and expansion. The modular architecture, scalable infrastructure, and flexible design patterns established in the current version provide a solid foundation for incorporating advanced features and capabilities that could significantly enhance the application's value proposition, user engagement, and market competitiveness. This chapter explores the strategic roadmap for future development, categorized by technical innovation, educational Enhancement, and business expansion opportunities.

REFERENCES

[1] **Chen, M., Tworek, J., Jun, H., et al.** (2021). Evaluating Large Language Models Trained on Code. *ArXiv preprint arXiv:2107.03374*.

[2] **Brown, T., Mann, B., Ryder, N., et al.** (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.

[3] **Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I.** (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report*.

[4] **Devlin, J., Chang, M., Lee, K., & Toutanova, K.** (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[5] **Koedinger, K. R., Corbett, A. T., & Perfetti, C.** (2012). The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive Science*, 36(5), 757-798.

[6] **Van Lehn, K.** (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46(4), 197- 221

[7] **Hattie, J., & Timperley, H.** (2007). The Power of Feedback. *Review of Educational Research*, 77(1), 81-112.

[8] **Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A.** (2005). AutoTutor: An Intelligent Tutoring System with Mixed-Initiative Dialogue. *IEEE Transactions on Education*, 48(4), 612-618.