Senior Design Project

Detailed Design Report

Team Id: T2502

Project Name: RecruitAssistant

Group Members:

- Mehmet Anıl Yeşil 22102614
- Deniz Öztürk 22102126
- Yüksel Barkın Baydar 22201939
- Emir Ensar Sevil 22201926
- Cankutay Dündar 22103284

Supervisor: Prof. Özgür Ulusoy

Date: March 13, 2026

1. Introduction

1.1 Purpose of The System

Recruitment has become increasingly competitive for students and early-career professionals, and interview processes now require strong technical competence, clear communication, and role-specific preparation. However, most existing career preparation tools are fragmented: some focus only on resume building, others provide generic question banks, and many lack personalization and end-to-end support throughout the hiring journey. As a result, candidates often struggle to identify skill gaps, tailor their application materials to a target role, and practice interviews in a realistic setting with actionable feedback [9], [20]. RecruitAssistant is designed to address this gap by providing an AI-powered, end-to-end career assistance platform that supports users from CV preparation to final interview readiness.

RecruitAssistant is a web-based platform that acts as a personalized interview companion and career preparation system. It helps users generate job-specific CVs (and cover letter support when applicable), parse job descriptions to extract key role requirements, and conduct mock interviews in both technical and behavioral formats. During mock interviews, the system can capture user responses through voice or text interaction and convert spoken answers into text using speech-to-text technology, enabling consistent evaluation and feedback generation [10], [12]. RecruitAssistant evaluates user answers in terms of clarity, relevance, structure, and correctness, and then provides personalized improvement suggestions that guide users toward stronger interview performance [9], [11].

In addition to interview simulations, RecruitAssistant includes pre-interview preparation modules such as quizzes and targeted practice materials. These assessments are adapted to the user's weaknesses and the selected job role, allowing the platform to recommend focused topics for improvement and track progress over time. A dashboard and analytics view summarize performance trends and highlight strengths and weaknesses across repeated practice sessions, helping users systematically improve.

Overall, RecruitAssistant aims to deliver a unified, scalable, and user-centric career preparation experience by combining CV generation, job-specific preparation, mock interviews with transcription and evaluation, and personalized quizzes with progress tracking in a single platform [5], [6], [9].

1.2 Design Goals

Cost

Since the project is developed within limited academic and financial resources, RecruitAssistant aims to minimize development and operational costs by prioritizing open-source technologies and efficient cloud usage. The system is designed to utilize open-source machine learning models where possible and leverage cost-efficient cloud infrastructure.

Usability

The application's user interface will be intuitive and user-friendly so that users can understand and navigate the system without prior training. Users should be able to register, start a mock interview, generate a CV, or take a quiz with minimal steps.

Availability

The system should be available 24/7, allowing users to practice interviews, update their CVs, and review analytics at any time. Since job preparation often happens outside regular working hours, the platform must ensure continuous accessibility and stable backend availability during peak usage periods.

Performance

Users should receive feedback within an acceptable response time during quizzes and mock interviews. Short operations such as quiz grading and CV retrieval should return results immediately after submission. For interview evaluation and transcription tasks, the system should respond within defined performance thresholds suitable for interactive use.

Scalability

The system should be scalable to maintain optimal service quality when the number of users increases, particularly during peak job application seasons. The architecture should allow horizontal scaling of backend services and AI inference components when demand grows.

Security

Sensitive user data such as personal information, CV content, quiz results, and interview transcripts must be securely stored. Passwords will be hashed before being stored in the database, and all communication between the client and server will be encrypted via HTTPS.

Compatibility

RecruitAssistant will be implemented as a responsive web application to ensure compatibility across modern web browsers on desktop and mobile devices. The system should function correctly on commonly used browsers without requiring specialized hardware or software installations.

1.3 Definitions, acronyms, and abbreviations

React: React is an open-source JavaScript library used for building user interfaces, particularly single-page web applications [2]. In RecruitAssistant, React is used to implement the responsive web frontend, enabling dynamic interaction for modules such as CV Studio, mock interviews, quizzes, and analytics dashboards.

FastAPI: FastAPI is a modern, high-performance web framework for building APIs with Python [1]. It is used as the backend framework in RecruitAssistant to implement RESTful services, handle authentication, process user requests, and integrate AI/ML components efficiently.

PostgreSQL: PostgreSQL is an open-source relational database management system (RDBMS) [3]. In RecruitAssistant, it is used to store structured data such as user accounts, CV versions, quiz attempts, interview sessions, transcripts, and analytics information.

AWS (Amazon Web Services): AWS is a cloud computing platform that provides infrastructure services such as virtual servers, databases, and storage [14]. RecruitAssistant may use AWS services (e.g., EC2, S3, RDS) for hosting backend services, storing files, and managing databases.

NLP (Natural Language Processing): NLP is a branch of artificial intelligence that focuses on enabling computers to understand, process, and generate human language [9], [11]. In RecruitAssistant, NLP is used for CV generation, job description parsing, question generation, and evaluation of interview responses.

STT (Speech-to-Text): Speech-to-Text refers to the process of converting spoken language into written text [10], [12]. RecruitAssistant uses STT technology (e.g., Whisper) to transcribe spoken answers during mock interviews for automated evaluation.

Whisper: Whisper is an automatic speech recognition (ASR) model used to convert audio into text [10]. In RecruitAssistant, it enables real-time transcription of mock interview responses.

Machine Learning (ML): Machine Learning is a subset of artificial intelligence that allows systems to learn patterns from data and make predictions or decisions [20]. RecruitAssistant uses ML models to score interview responses, recommend quizzes, and generate personalized feedback.

API (Application Programming Interface): An API is a set of rules and protocols that allows different software components to communicate [5]. RecruitAssistant uses RESTful APIs to enable communication between the React frontend and the FastAPI backend.

REST (Representational State Transfer): REST is an architectural style for designing web services [6], [19]. RecruitAssistant's backend follows REST principles to structure endpoints for authentication, CV management, quizzes, interviews, and analytics.

HTTP/HTTPS: HTTP (Hypertext Transfer Protocol) is a protocol used for communication between web clients and servers, while HTTPS provides encrypted communication over the network [7], [8]. RecruitAssistant enforces HTTPS to protect user data during communication.

JWT (JSON Web Token): JWT is a token-based authentication mechanism that securely transmits information between parties [4]. It may be used in RecruitAssistant to manage authenticated user sessions and protect restricted endpoints.

GDPR (General Data Protection Regulation): GDPR is a European Union regulation governing data protection and privacy [18]. RecruitAssistant follows GDPR principles to ensure responsible processing and storage of user data.

KVKK (Kişisel Verilerin Korunması Kanunu): KVKK is Turkey's data protection law regulating the protection of personal data. RecruitAssistant aligns its data handling practices with KVKK to protect personal information such as CV content and interview transcripts.

CI/CD (Continuous Integration / Continuous Deployment): CI/CD refers to software development practices that automate building, testing, and deploying applications [18]. RecruitAssistant may adopt CI/CD workflows to improve code quality, reliability, and collaboration during development [13].

1.4 Overview

We aim to enhance the career preparation process by introducing the RecruitAssistant web application that targets students and early-career professionals preparing for job applications. The idea is to provide a more structured, efficient, and data-driven way of preparing for interviews and application processes. With the increasing competitiveness in the job market, preparing for interviews and tailoring CVs to specific roles has become a major challenge. Although various tools exist for resume or interview practice, these tools are often fragmented and lack personalization, making it difficult for candidates to systematically improve [9], [20].

RecruitAssistant aims to solve these problems by offering an integrated platform that supports users throughout the hiring preparation lifecycle. The application allows users to generate job-specific CVs, practice technical and behavioral mock interviews, and receive structured feedback on their performance. Instead of relying on generic question banks or static resume templates, users receive adaptive suggestions and evaluation based on their selected target role. This helps them identify weaknesses and focus on areas that require improvement [9], [11].

Additionally, users no longer have to rely solely on self-evaluation during interview practice. The system can capture user responses through text or voice input, transcribe spoken answers using speech-to-text technology, and evaluate them using predefined scoring criteria [10], [12]. The feedback highlights clarity, structure, relevance, and technical accuracy, providing actionable improvement suggestions. This transforms interview preparation into an interactive and measurable process rather than a one-time rehearsal.

Moreover, the platform recommends targeted quizzes and practice topics based on detected weaknesses. The system analyzes user performance trends and suggests the most relevant areas for improvement, ensuring that preparation time is used efficiently. Instead of practicing randomly, users follow a guided preparation path supported by analytics and progress tracking.

RecruitAssistant also benefits mentors or administrators by providing anonymized performance insights when applicable. By centralizing CV management, mock interviews, quizzes, and analytics into a single web-based platform, the system increases transparency and consistency in the preparation process [5], [6]. In conclusion, RecruitAssistant offers a comprehensive and scalable solution for structured career preparation. The platform provides a convenient, efficient, and personalized interview preparation experience while enabling continuous improvement through measurable feedback and progress tracking.

2. Current software architecture

Currently, there is no single application that fully integrates all of the functionalities offered by RecruitAssistant. However, several existing platforms address parts of the career preparation and interview readiness problem and are worth mentioning.

One such example is **Resume.io**, a popular online resume builder that helps users create professional CVs quickly and offers additional tools like AI-powered suggestions and career resources. However, Resume.io primarily focuses on resume creation and does not provide interview simulations or detailed performance analytics.

Another example is **InterviewPal**, a dedicated interview preparation platform that provides practice questions, instant AI feedback, and personalized response insights. InterviewPal helps candidates refine answers and understand strengths and weaknesses, but it lacks integration with role-specific CV tailoring or adaptive learning modules tied to a full preparation pipeline.

There are also AI-powered mock interview tools such as **FinalRound AI** and **XenHire**, which offer realistic mock interview experiences, question generation, and feedback for job seekers. FinalRound AI combines AI mock interviews with resume optimization and coaching tools, while XenHire focuses on adaptive interview scenarios and personalized feedback. Both assist candidates in interview practice, yet neither combines structured quizzes, analytics tracking, and CV generation in one unified system.

Platforms like **InterviewBee AI** and **Interviews by AI** further highlight AI-driven interview preparation tools that generate tailored questions based on job descriptions and offer feedback on responses to help users practice more effectively. These platforms simplify interview rehearsal with AI question generation and responsive evaluation, but they operate primarily as standalone mock interview or question generation tools.

The major difference between RecruitAssistant and these alternatives is the end-to-end integration of multiple preparation phases — CV generation tailored to specific job descriptions, adaptive quizzes based on performance trends, mock interview simulations (including speech-to-text transcription), structured evaluation, and progress tracking — all within a single web platform. RecruitAssistant’s unified workflow enables more comprehensive and efficient career preparation than the fragmented solutions currently available.

3. Proposed software architecture

3.1 Overview

The world today is moving towards intelligent web-based platforms and AI-assisted systems, and hence it is imperative to have a strong and robust software architecture for scalable web applications. This proposed software architecture leverages the power of **FastAPI** on the server side and **React** for rendering the user interface across modern web browsers. **PostgreSQL** is used as the database management system for efficient data storage and retrieval. The architecture ensures security by using token-based authentication (e.g., JWT) for user authentication and session management. Additionally, deployment, hosting, and continuous integration processes of the system can be managed using cloud infrastructure services (e.g., AWS) and version control platforms such as GitHub.

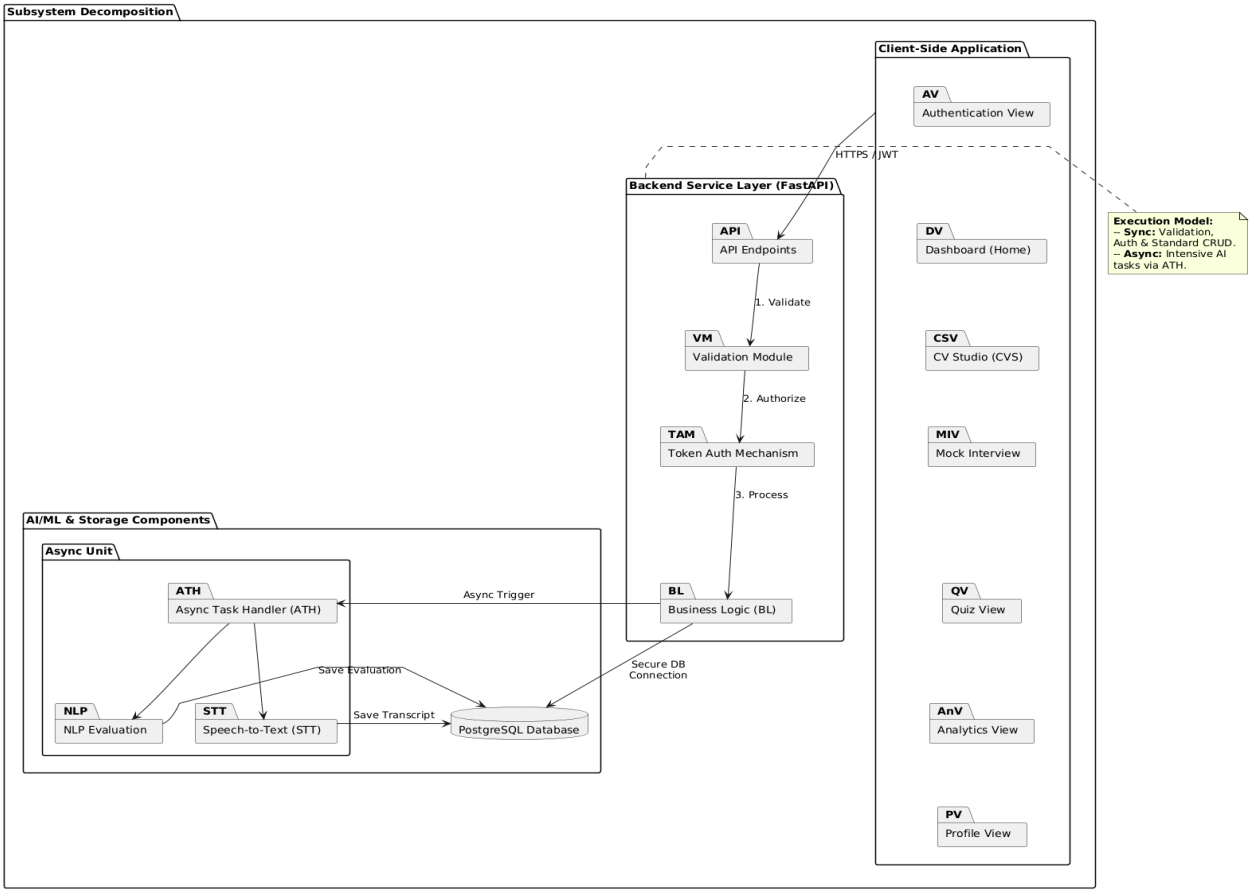
This proposed architecture will help in developing a scalable and maintainable web application that is efficient, secure, and performs optimally. The proposed software architecture also includes modular AI/ML components responsible for natural language processing and speech-to-text functionality. These components enable automated CV generation, interview question creation, response evaluation, and quiz adaptation. By structuring AI modules as independent services within the system, the architecture provides flexibility and scalability while maintaining clear separation of concerns.

By leveraging modular service design and REST-based communication between components, the proposed architecture provides a reliable and scalable system that ensures consistent data flow between the frontend, backend, and AI layers. This allows a distributed yet loosely coupled system to be built, where components communicate efficiently while remaining independently maintainable. Overall, the proposed software architecture is designed to facilitate effective communication between the different services, leading to a highly responsive and efficient web application.

All in all, we have divided our application into three parts: Client Side, Backend Service Layer, and AI/ML & Storage Layer. The client side communicates with the backend via REST endpoints, and the backend interacts with the database and AI modules internally. External cloud infrastructure services are used for hosting and scalability when needed. Some diagrams are given below to illustrate the hierarchy of our architecture.

3.2 Subsystem Decomposition

Our architecture can be decomposed into three parts: the client-side application, backend service layer, and AI/ML & storage components. The client part consists of authentication, dashboard (home), CV studio, mock interview, quiz, profile, and analytics views. Each view interacts with backend services to retrieve or submit data. The client sends HTTP requests to the backend API, where the business logic is executed and responses are returned accordingly. Incoming requests from the client are processed by the backend service layer built with FastAPI. Depending on the request type, operations may involve database queries, AI/ML processing, or orchestration between multiple internal services. Lightweight operations are handled synchronously, while computationally intensive AI tasks can be handled asynchronously to maintain responsiveness. If a request cannot be completed successfully due to validation errors or processing failures, an appropriate error response is returned to the client with clear feedback. Otherwise, the processed results—such as quiz scores, interview evaluations, or generated CV content—are sent back to the client interface. Persistent data management is handled through a PostgreSQL database connected directly to backend services via secure database connections. All access to user data requires authentication and authorization through token-based mechanisms. This layered architecture ensures separation of concerns, secure data handling, and subsystem communication.



3.3 Hardware/software mapping

The development of RecruitAssistant requires a clear mapping between the hardware infrastructure and the software components that constitute the system. On the client side, the hardware mainly consists of user devices such as laptops, desktop computers, tablets, and mobile devices that access the application through a web browser. These devices provide the interface through which users interact with the platform, allowing them to perform actions such as creating CVs, participating in mock interviews, solving quizzes, and reviewing analytics. Standard hardware components such as microphones may optionally be used during mock interview sessions for recording speech, which is later processed by speech-to-text services [10], [12].

On the server side, the system runs on cloud-based infrastructure that provides computational resources for the backend services and AI processing modules [14]. The backend service layer is implemented using FastAPI and operates on cloud-hosted servers that handle incoming HTTP requests from the client-side application [1], [7]. These backend services are responsible for performing request validation, user authentication, business logic execution, and coordination of AI-related processing tasks. The AI/ML components, including the natural language processing evaluation module and the speech-to-text service, operate either within the same computing environment or on dedicated processing instances depending on performance requirements [9], [11], [10]. These components process interview transcripts and generate evaluation results that are later stored in the database.

The overall software stack of RecruitAssistant consists of a React-based frontend application, a FastAPI backend service layer, AI processing modules responsible for evaluation and transcription, and a PostgreSQL database for persistent data storage [2], [1], [3]. Communication between the client and the backend occurs through secure HTTPS requests using RESTful APIs [5], [6], [8]. Internal communication between backend services and AI modules occurs within the protected cloud environment. This mapping between hardware and software components ensures that the system remains scalable, modular, and capable of handling both standard user operations and computationally intensive AI tasks [15].

3.4 Persistent data management

RecruitAssistant generates and manages various types of structured data during its operation, including user account information, CV versions, quiz attempts, interview session data, transcripts, evaluation results, and analytics summaries. To manage this data efficiently and reliably, the system uses PostgreSQL as its relational database management system [3]. PostgreSQL is selected due to its strong support for transactional consistency, scalability, and reliability in handling structured application data.

The database stores persistent information such as user profiles, authentication credentials, CV content and version history, interview transcripts generated by speech-to-text processing, evaluation outputs produced by the NLP module, and quiz results associated with user learning activities [10], [12], [9], [11]. Maintaining the integrity and consistency of this data is critical for the correct functioning of the platform. PostgreSQL provides ACID-compliant transactions, ensuring atomicity, consistency, isolation, and durability of database operations [17]. These guarantees help prevent data corruption and ensure that operations such as saving interview evaluations or updating CV content are executed reliably.

Access to the database is restricted to the backend service layer, preventing direct interaction between the client application and the persistent storage system. This design improves both security and data integrity by ensuring that all database operations pass through controlled backend logic. To maintain system reliability, database backups and monitoring mechanisms can be implemented within the cloud infrastructure [14]. Additionally, asynchronous task execution is used for AI-related operations to avoid blocking user requests, while standard CRUD operations remain synchronous to ensure immediate responsiveness [15].

3.5 Access control and security

RecruitAssistant implements a secure authentication and authorization mechanism to control access to system resources and protect user data. When a user logs into the system, the backend generates a secure authentication token, such as a JSON Web Token (JWT), which is returned to the client application [4]. This token contains limited identity information and is cryptographically signed by the backend service to prevent tampering. The client includes this token in subsequent requests to authenticate itself when accessing protected endpoints.

Each request received by the backend service is verified by validating the token’s signature and expiration time. If the token is valid, the request proceeds to the backend logic; otherwise, access is denied.

Communication between the client-side application and the backend services is protected using HTTPS with SSL/TLS encryption [8]. This ensures that data transmitted over the network cannot be intercepted or modified by unauthorized parties. Sensitive user information, such as passwords, is never stored in plain text; instead, secure hashing algorithms are used to store password representations safely. In addition to authentication and encryption, the system follows data privacy principles by limiting access to sensitive information and implementing appropriate safeguards for user-generated content such as interview transcripts and evaluation results [18]. Through these mechanisms, RecruitAssistant maintains a secure and reliable environment for its users while protecting the confidentiality and integrity of stored data.

4. Subsystem services

4.1 Client-Side Application

The first subsystem of RecruitAssistant is the Client-Side Application, which represents the user interface layer of the platform. This component is responsible for providing users with an interactive environment where they can access the system's main functionalities. The client communicates with the backend services through secure HTTPS requests that include JWT tokens for authentication and authorization [8], [4].

The client application contains several main views that support different user activities within the system. The Authentication View manages user login and registration processes and ensures that only authorized users can access system resources. The Dashboard (Home) View presents an overview of the user's activity and provides access to recently used features. The CV Studio View enables users to create, edit, and manage their CVs within the platform. The Mock Interview View allows users to participate in simulated interview sessions where their responses can later be analyzed by AI components [9], [11]. The Quiz View provides practice questions designed to help users prepare for technical or behavioral interview scenarios. The Analytics View displays insights related to the user's performance and progress over time. Finally, the Profile View manages user-specific information and account settings. Through these views, the client application acts as the primary interaction point between users and the system [2].

4.2 Backend Service Layer (FastAPI)

The second subsystem is the Backend Service Layer, which is implemented using FastAPI and is responsible for handling client requests and coordinating the core functionality of the system [1]. The backend receives requests from the client application through secure HTTPS connections that include JWT tokens for authentication [8], [4].

Each request first reaches the API Endpoints, which serve as the entry points to the backend system [5]. After receiving a request, the system performs a validation step using the Validation Module to ensure that the request data is properly structured and satisfies predefined constraints. Following validation, the request proceeds to the Token Authentication Mechanism, which verifies the authenticity and validity of the provided JWT token to ensure that the user is authorized to access the requested resource [4].

Once validation and authorization are successfully completed, the request is forwarded to the Business Logic component. This component implements the core functionality of the platform, including operations related to CV management, quiz processing, interview session handling, and coordination of AI-based evaluation tasks. When a request requires computationally intensive processing, such as analyzing interview responses, the Business Logic component triggers asynchronous processing through the AI/ML subsystem while continuing to handle standard operations synchronously [15].

4.3 External APIs

The third subsystem consists of the AI/ML and storage components responsible for processing interview data and managing persistent system information. This subsystem contains both the data storage layer and the asynchronous AI processing modules [20].

Persistent system data is stored in a PostgreSQL database, which maintains structured information such as user accounts, CV versions, interview transcripts, quiz results, and evaluation outputs [3]. The backend service layer interacts with the database through secure connections, ensuring that all persistent data operations remain protected and controlled.

Within this subsystem, computationally intensive AI tasks are handled by the Async Unit. The Async Task Handler (ATH) acts as a coordinator for asynchronous processing tasks triggered by the backend's Business Logic component. When an interview session requires analysis, the Async Task Handler activates the Speech-to-Text (STT) module to convert spoken interview responses into textual transcripts [10], [12]. These transcripts are then processed by the NLP

Evaluation module, which analyzes the responses and generates evaluation results [9], [11], [16]. The generated transcripts and evaluation outputs are subsequently stored in the PostgreSQL database for future access and analysis [17].

By separating AI processing tasks into an asynchronous subsystem, the architecture ensures that computationally intensive operations do not block standard user requests. This design improves system responsiveness while allowing the platform to perform advanced AI-based evaluation tasks efficiently [15].

5. Test Cases

5.1 Functional Tests

Test ID	T1
Test Type	Functional
Title/Objective	Verify successful Cover Letter generation using existing Workspace data.
Procedure	1. Navigate to a successfully created Workspace (which already contains the job description and the user's CV). 2. Click the "Generate Cover Letter" button.

Expected Outcome	The system should automatically fetch the Job Description and CV from the Workspace's database record, trigger the LLM service, and display a tailored cover letter without asking for additional input.
Severity	Critical

Test ID	T2
Test Type	Functional
Title/Objective	Verify the successful download of the generated job-specific Cover Letter.
Procedure	1. Generate a cover letter successfully within a Workspace. 2. Click the "Download Cover Letter" button. 3. Open the downloaded file.
Expected Outcome	The file should be downloaded securely (e.g., in PDF or DOCX format) and the content should perfectly match the generated text displayed on the screen.
Severity	Major

Test ID	T3
Test Type	Functional
Title/Objective	Verify that a generated Cover Letter is automatically saved and persists within the Workspace.
Procedure	1. Generate a cover letter successfully in a Workspace. 2. Navigate away from the Workspace (e.g., return to the main dashboard). 3. Re-enter the same Workspace. 4. Check the Cover Letter section.

Expected Outcome	The previously generated cover letter should be preserved and immediately displayed. The user should not have to click "Generate" again, preventing unnecessary LLM API calls and quota usage.
Severity	Critical

Test ID	T4
Test Type	Functional
Title/Objective	Verify that the Cover Letter generation dynamically adapts to updates made to the Job Description within the same Workspace.
Procedure	1. Generate an initial Cover Letter for a specific Job Description (e.g., requiring "Python"). 2. Edit the Job Description within the Workspace to demand a completely different skill (e.g., change "Python" to "Java"). 3. Click the "Generate Cover Letter" button again. 4. Review the newly generated text.
Expected Outcome	The system must functionally invalidate any previous cache and fetch the latest Workspace data. The newly generated Cover Letter must abandon the old context and explicitly highlight the new skill (Java) corresponding to the updated Job Description.
Severity	Critical

Test ID	T5
Test Type	Functional / Input Validation
Title/Objective	Verify the minimum length and word count validation for the Job Description before Cover Letter generation.

Procedure	1. Open a Workspace. 2. Edit the Job Description to be less than 50 characters or just a single continuous string (e.g., "asdasdasdasd"). 3. Click the "Generate Cover Letter" button.
Expected Outcome	The backend validation (via Pydantic schemas) should immediately reject the request. The system should display a warning modal (e.g., "Job description is too short or invalid") without triggering the external LLM API.
Severity	Major

Test ID	T6
Test Type	Functional
Title/Objective	Verify that job-specific quizzes are automatically available and accessible within a created Workspace.
Procedure	1. Open a successfully created Workspace. 2. Navigate to the "Quizzes" section within that Workspace. 3. Select a quiz from the list and click "Start".
Expected Outcome	The quiz list should be pre-populated based on the Workspace's job description. Clicking "Start" should immediately load the questions without triggering a new generation process or loading screen.
Severity	Critical

Test ID	T7
Test Type	Functional
Title/Objective	Check results on attempting to submit an incomplete job-specific quiz.
Procedure	1. Start a job-specific quiz inside a Workspace.

	2. Answer only 2 out of 5 questions. 3. Click the "Submit" or "Finish" button.
Expected Outcome	The system should detect the missing answers and display a warning modal (e.g., "You have unanswered questions"). It should ask for confirmation to submit anyway or return the user to the quiz.
Severity	Major

Test ID	T8
Test Type	Functional
Title/Objective	Verify correct score calculation and Workspace database update after quiz submission.
Procedure	1. Complete a 5-question job-specific quiz inside a Workspace. 2. Intentionally select 3 correct and 2 incorrect answers. 3. Submit the quiz. 4. Check the specific Workspace's results/history page.
Expected Outcome	The UI should instantly display a score of 60% (or 3/5). In the backend, the score column in the quizzes table must be accurately updated and properly linked to that specific workspace_id.
Severity	Critical

Test ID	T9
Test Type	Functional
Title/Objective	Verify that the system functionally displays the correct answers only after the quiz has been successfully submitted.
Procedure	1. Complete and submit a job-specific quiz inside a Workspace. 2. Navigate to the quiz results or review section.

	3. Check the displayed questions and options.
Expected Outcome	The UI should correctly display the user's selected answers alongside the actual correct answers. The correct answers must be functionally accessible and visible only in this post-submission state.
Severity	Major

Test ID	T10
Test Type	Functional
Title/Objective	Verify the system's behavior and UI state when a user accesses an already completed job-specific quiz within a Workspace.
Procedure	1. Complete and successfully submit a job-specific quiz in a Workspace. 2. Navigate back to the Quizzes list within that same Workspace. 3. Click on the previously completed quiz.
Expected Outcome	The system must recognize the quiz status as 'completed'. It should display the user's previous score and (optionally) their selected answers, but the "Submit" button must be hidden or strictly disabled to prevent multiple submissions and score manipulation.
Severity	Major

Test ID	T11
Test Type	Functional
Title/Objective	Verify that a new workspace can be created successfully via the workspace selector dropdown.
Procedure	1. Click the workspace selector in the top navigation. 2. Click "+ Create Workspace" at the bottom of the dropdown.

	3. Enter a valid workspace name (e.g. "Google Prep"). 4. Click the "Create" button.
Expected Outcome	A new workspace should be created, appear in the workspace list, and automatically become the active workspace. The header should display the new workspace name with an assigned emoji and color.
Severity	Critical

Test ID	T12
Test Type	Functional
Title/Objective	Verify that switching between workspaces updates the active workspace correctly.
Procedure	1. Click the workspace selector in the top navigation. 2. Observe the list of available workspaces. 3. Click on a different workspace (not the active one).
Expected Outcome	The selected workspace should become active. The header should update to show the newly selected workspace's name, emoji, and color. A checkmark should appear next to the active workspace in the dropdown.
Severity	Critical

Test ID	T13
Test Type	Functional
Title/Objective	Verify that a workspace can be renamed successfully.
Procedure	1. Click the workspace selector in the top navigation. 2. Hover over an existing workspace in the list.

	3. Click the pencil (edit) icon that appears on hover. 4. Enter a new name in the rename dialog. 5. Click "Save".
Expected Outcome	The workspace name should be updated in the dropdown list. If it was the active workspace, the header should also reflect the new name immediately.
Severity	Major

Test ID	T14
Test Type	Functional
Title/Objective	Verify that a workspace can be deleted successfully.
Procedure	1. Ensure there are at least 2 workspaces. 2. Click the workspace selector in the top navigation. 3. Hover over a workspace and click the trash icon. 4. Confirm deletion in the confirmation dialog.
Expected Outcome	The workspace should be removed from the list. If the deleted workspace was the active one, the system should automatically switch to another available workspace.
Severity	Critical

Test ID	T15
Test Type	Negative
Title/Objective	Verify the system's behavior and UI state when a user accesses an already completed job-specific quiz within a Workspace.

Procedure	1. Delete workspaces until only one remains. 2. Open the workspace selector dropdown. 3. Hover over the last workspace. 4. Observe whether a delete (trash) icon is visible.
Expected Outcome	The trash/delete icon should not appear for the last remaining workspace. The user must always have at least one workspace.
Severity	Major

Test ID	T16
Test Type	Functional
Title/Objective	Verify that creating a workspace with an empty name is not allowed.
Procedure	1. Click the workspace selector and then "+ Create Workspace". 2. Leave the name field empty or enter only spaces. 3. Attempt to click the "Create" button.
Expected Outcome	The "Create" button should remain disabled. No empty workspace should be created. The existing workspace list should remain unchanged.
Severity	Minor

Test ID	T17
Test Type	Functional (Persistence)
Title/Objective	Verify that newly created workspaces persist after a page refresh.

Procedure	1. Create a new workspace named "Interview Sprint". 2. Verify it appears in the workspace dropdown. 3. Refresh the browser page. 4. Open the workspace selector dropdown.
Expected Outcome	The "Interview Sprint" workspace should still be in the workspace list after page refresh, with the same emoji, color, and name. Total workspace count should remain the same.
Severity	Critical

Test ID	T18
Test Type	Functional (Persistence)
Title/Objective	Verify that the selected workspace persists after a page refresh.
Procedure	1. Select "Workspace 2" from the workspace dropdown. 2. Refresh the browser page (F5 or Ctrl+R). 3. Observe which workspace is displayed in the header.
Expected Outcome	After page refresh, "Workspace 2" should still be the active workspace. The workspace list and all workspace data should be restored from localStorage correctly.
Severity	Critical

Test ID	T19
Test Type	UI / Usability

Title/Objective	Verify that the workspace selector visually highlights the active workspace and shows hover actions.
Procedure	1. Click the workspace selector to open the dropdown. 2. Observe the active workspace row styling. 3. Hover over each workspace row. 4. Observe the edit (pencil) and delete (trash) icons.
Expected Outcome	The active workspace should have a highlighted/accent background and a checkmark icon. On hover, edit and delete action icons should appear for each workspace. The dropdown should have a clean, modern appearance consistent with the rest of the UI.
Severity	Minor

Test ID	T20
Test Type	Functional
Title/Objective	Verify that the interview starts successfully with the default setup values
Procedure	1. Open the mock interview page 2. Leave all settings at their defaults (HR/Behavioral, Junior, English, Mic Off, Time Limit on). 3. Click Start Interview

Expected Outcome	The page transitions to the active interview state. The first question (“Tell me about yourself…”) is displayed, the timer is visible.
Severity	Critical

Test ID	T21
Test Type	Functional
Title/Objective	Verify the microphone toggle switch works in the setup screen.
Procedure	1. Open the Mock Interview page 2. Toggle the microphone switch on. 3. Toggle the microphone switch off.
Expected Outcome	The switch visually changes state on each toggle. When On, it is in the “checked” position; when OFF it returns to “unchecked”.
Severity	Major

Test ID	T22
Test Type	Functional
Title/Objective	Verify the user can type an answer and submit it to move to the next question

Procedure	1. Open the Mock Interview page. 2. Start an interview (default settings). 3. Type an answer in the “Type your answer here…” text area. 4. Click on the Send button..
Expected Outcome	The page advances to the next question. The text area is cleared, and the new question is displayed.
Severity	Critical

Test ID	T23
Test Type	Functional
Title/Objective	Verify the Send button is disabled when the text area is empty.
Procedure	1. Open the Mock Interview page. 2. Start an interview. 3. Leave the answer text area empty. 4. Observe the Send button.
Expected Outcome	The Send button should be disabled (not clickable). The user cannot advance to the next question.
Severity	Critical

Test ID	T24
Test Type	Functional
Title/Objective	Verify the microphone can be toggled on and off during an active interview session.

Procedure	1. Open the Mock Interview page. 2. Start an interview with microphone OFF. 3. Click the Mic button in the interview area. 4. Observe UI feedback. 5. Click the button again to disable.
Expected Outcome	When toggled ON the icon changes to “Mic” and the text updates to ‘Listening…”. When toggled OFF the icon changes to “MicOff” and the text updates to “Microphone disabled.”.
Severity	Major

Test ID	T25
Test Type	Functional
Title/Objective	Verify the interview transitions to the completed/results state after all questions are answered.
Procedure	1. Open the Mock Interview page. 2. Start an interview. 3. Answer all questions. 4. Observe the interview state.
Expected Outcome	After all questions are answered , the page transitions to the completed state. An overall score is shown, a performance overview section appears.
Severity	Critical

Test ID	T26
Test Type	Functional

Title/Objective	Verify the user can restart a new interview from the results screen.
Procedure	1. Open the Mock Interview page. 2. Start an interview. 3. After completing that interview, click the “Start Another Interview” button.
Expected Outcome	The page returns to the setup state with the Interview Setup card displayed and all configuration options available.
Severity	Major

Test ID	T27
Test Type	Functional
Title/Objective	Verify the schedule page loads showing the current week with correct day labels.
Procedure	1. Open the Schedule page. 2. Observe the UI.
Expected Outcome	The weekly calendar is displayed. The header shows the correct week range (e.g., “Mar 2 - Mar 8, 2026”). Seven day columns are visible starting from Monday. Today’s date is highlighted.
Severity	Critical

Test ID	T28
Test Type	Functional

Title/Objective	Verify the left/right chevron navigates to the previous/next week.
Procedure	1. Open the Schedule page. 2. Note the current week range. 3. Click the left/right arrow button.
Expected Outcome	The week range updates to show the previous/next week (7 days earlier/later). Day labels and dates update accordingly.
Severity	Major

Test ID	T29
Test Type	Functional
Title/Objective	Verify clicking the on a day cell opens the Add Event dialog with the correct date.
Procedure	1. Open Schedule page. 2. Click on any day column (e.g. Wednesday's area). 3. Observe the UI.
Expected Outcome	The "Add Event" dialog opens. The dialog description displays the correct date for clicked day (e.g., "Schedule Event March 4").
Severity	Major

Test ID	T30
Test Type	Functional
Title/Objective	Verify a new event can be created with a title, type and time.
Procedure	1. Open the Schedule page. 2. Click on a day to open the Add Event dialog. 3. Enter “Data Structures Review” in the Activity Title field. 4. Select Practice Session from the Type dropdown. 5. Set the Time to “14:00” 6. Click Schedule Event.
Expected Outcome	The dialog closes. A new event card labelled “Data Structures Review” appears on the selected day at 14:00.
Severity	Critical

Test ID	T31
Test Type	Functional
Title/Objective	Verify behaviour when the user tries to schedule an event without providing a title.
Procedure	1. Open the Schedule page. 2. Open the Add Event dialog by clicking a day cell. 3. Leave the Activity Title field empty. 4. Select a type and time. 5. Click Schedule Event.
Expected Outcome	Either the event is not created and an error/validation message is displayed, or an event with an empty title created (documenting the current behaviour to identify a potential bug – missing validation)

Severity	Major

Test ID	T32
Test Type	Functional
Title/Objective	Verify the Add Event dialog can be dismissed without creating a
Procedure	1. Open the Schedule page. 2. Click a day to open the Add Event dialog. 3. Fill in some fields (e.g. title = “Test Event”). 4. Close the dialog by clicking outside it or pressing the X/Close or Cancel button. 5. Re-open the dialog on the same day.
Expected Outcome	No event is added to the calendar. When the dialog is re-opened, the form fields are reset to their defaults (empty title, type = Practice Session, time = 10:00) etc.
Severity	Major

Test ID	T33
Test Type	Functional
Title/Objective	Verify the Notes text area is functional during an active interview session
Procedure	1. Open the Mock Interview page 2. Start an interview. 3. Type text into the notes area (e.g., “Remember to mention leadership experience”). 4. Navigate to the next question.
Expected Outcome	The notes text persists across questions within the same interview session. The text area accepts input without errors.
Severity	Major

Test ID	T34
Test Type	Functional
Title/Objective	Verify that multiple events can be added to the same day and all are displayed..
Procedure	1. Open the Schedule page. 2. Click on a day and add an event. 3. Click on the same day again and add another event. 4. Observe the day cell.
Expected Outcome	Both event cards appear stacked in that day's column, each displaying their respective title and time. No event overwrites the other.
Severity	Major

Non-Functional Tests

Test ID	NFT-01
Test Type	Non-Functional Performance
Title/Objective	Check application load time under normal network conditions.
Procedure	1. Clear browser cache. 2. Navigate to the Workspace/Dashboard URL.
Expected Outcome	The application dashboard should fully load and become interactive within 3 seconds.
Severity	Critical

Test ID	NFT-02
Test Type	Non-Functional Performance

Title/Objective	Check system behavior under simultaneous multiple quiz submissions.
Procedure	1. Simulate 100 concurrent users accessing and submitting a quiz within the same workspace simultaneously.
Expected Outcome	The server should process all submissions without crashing, data loss, or significant delay (response time < 3s per submission).
Severity	Critical

Test ID	NFT-03
Test Type	Non-Functional Security
Title/Objective	Verify the session timeout mechanism triggered by user inactivity.
Procedure	1. Log in to the application. 2. Leave the application idle for 30 minutes. 3. Attempt to navigate to another page.
Expected Outcome	The system should automatically log the user out, redirect to the login page, and show 'Session expired'.
Severity	Major

Test ID	NFT-04
Test Type	Non-Functional Security
Title/Objective	Check protection against Cross-Site Scripting (XSS) in the interview text area.
Procedure	1. Start a mock interview. 2. Enter payload alert('XSS'). 3. Click Send.
Expected Outcome	The input should be sanitized and stored as plain text. The script must not execute.
Severity	**Critical**

Test ID	NFT-05
Test Type	Non-Functional Usability / Responsiveness
Title/Objective	Check responsiveness of the Mock Interview page on mobile devices.
Procedure	1. Open the Mock Interview page on a mobile device or emulator. 2. Verify camera feed, timer, and text area layout.
Expected Outcome	All UI elements scale correctly without horizontal scrolling and input remains accessible above the keyboard.
Severity	Major

Test ID	NFT-06
Test Type	Non-Functional Usability / Responsiveness
Title/Objective	Check layout stability when changing device orientation.
Procedure	1. Start a job-specific quiz on a mobile/tablet. 2. Rotate device from portrait to landscape.
Expected Outcome	Quiz elements adjust dynamically without overlapping or disappearing.
Severity	Major

Test ID	NFT-07
Test Type	Non-Functional Reliability
Title/Objective	Check behavior during temporary network disconnection.
Procedure	1. Start a quiz and select answers. 2. Disable the internet. 3. Attempt submission.
Expected Outcome	UI should show offline error and preserve selected answers until reconnection.

Severity	Major

Test ID	NFT-08
Test Type	Non-Functional Reliability
Title/Objective	Check hardware permission recovery during mock interview setup.
Procedure	1. Go to the interview setup. 2. Deny microphone permissions. 3. Try enabling a microphone in the UI.
Expected Outcome	The system should not crash and should show a clear message explaining permissions must be enabled.
Severity	Critical

Test ID	NFT-09
Test Type	Non-Functional Scalability
Title/Objective	Check database/UI performance with a large quiz history.
Procedure	1. Use a test workspace with 50,000 completed quiz records. 2. Open results/history page.
Expected Outcome	Page loads quickly (<2s) using pagination or lazy loading without freezing.
Severity	Major

Test ID	NFT-10
Test Type	Non-Functional Compatibility
Title/Objective	Check cross-browser compatibility of interview setup UI.
Procedure	1. Start a mock interview. 2. Repeat on Chrome, Firefox, Safari, Edge.

Expected Outcome	UI elements render and function consistently across browsers.
Severity	Major

Test ID	NFT-11
Test Type	Non-Functional Localization
Title/Objective	Check layout and text rendering when changing application language.
Procedure	1. Change language in settings. 2. Navigate through application pages.
Expected Outcome	All text updates instantly, layout remains stable, no placeholder keys appear.
Severity	Major

Test ID	NFT-12
Test Type	Non-Functional Security
Title/Objective	Check system protection against rapid automated requests (Rate Limiting).
Procedure	1. Use automated script. 2. Send 50 requests in 5 seconds to submit quiz/interview.
Expected Outcome	Server blocks excessive requests and returns HTTP 429 Too Many Requests.
Severity	Critical

6. Consideration of Various Factors in Engineering Design

During the development process of RecruitAssistant, there are various factors that can affect both the technical implementation and the overall system design. These factors are explained in detail below.

6.1 Public Health

RecruitAssistant is a career preparation platform and does not directly operate in a medical or health-related domain. Therefore, there is no direct or indirect impact on public health. However, the system should avoid generating misleading or harmful advice that could negatively affect users' psychological well-being. Feedback mechanisms should remain constructive and supportive rather than discouraging.

6.2 Public Safety

The system processes sensitive personal data such as usernames, passwords, email addresses, CV content, quiz results, and interview transcripts. This data must not be shared with third parties without explicit user consent. Therefore, while designing the application, reliable storage mechanisms, secure authentication, encrypted communication (HTTPS), and strict access control policies must be implemented to protect user data.

6.3 Public Welfare

RecruitAssistant aims to improve users' career readiness by providing structured preparation tools such as job-specific CV generation, mock interviews, quizzes, and analytics. By centralizing these tools into a single platform, the system reduces the need for multiple fragmented services and saves users time during their preparation process. Furthermore, by offering performance analytics and personalized recommendations, the system helps users allocate their preparation efforts more effectively and improve their employability prospects.

6.4 Global Factors

Although the MVP may initially target a specific group of students, the long-term vision of RecruitAssistant allows expansion to a broader international audience. Therefore, the system design should support adaptability in language, role categories, and regulatory compliance. Different data protection regulations (such as GDPR and KVKK) must be considered in the design to ensure compliance across jurisdictions.

6.5 Cultural Factors

Interview expectations, communication styles, and professional norms may vary across cultures. Therefore, the system should be designed in a way that allows flexible configuration of interview question styles, feedback tone, and role-specific expectations. While the core functionality remains universal, the evaluation criteria should avoid cultural bias and ensure fairness across different backgrounds.

6.6 Social Factors

RecruitAssistant may experience higher usage during peak job application periods, such as graduation seasons or major recruitment cycles. Social trends, economic conditions, and hiring waves can significantly affect system load. Therefore, the architecture must support scalable infrastructure to handle temporary spikes in user demand without performance degradation.

6.7 Environmental Factors

The environmental impact of RecruitAssistant is primarily related to computational resource usage in cloud infrastructure and AI model inference. Efficient model selection, optimized inference processes, and appropriate resource scaling strategies should be implemented to reduce unnecessary computational load and energy consumption.

6.8 Economic Factors

RecruitAssistant may operate under budget constraints during its development and early deployment phases. Infrastructure, storage, and AI inference costs must be carefully managed. Any potential monetization strategy (if implemented in the future) must be designed transparently and ethically, ensuring that users clearly understand any pricing policies before use. The exact business model, if applicable, will be defined prior to production deployment.

	Effect Level	Effect
Public Health	0	No direct effect
Public Safety	9	Protect sensitive user data
Public Welfare	10	Improve structured career preparation
Global Factors	8	Comply with international data regulations
Cultural Factors	5	Avoid bias in evaluation
Social Factors	8	Handle peak recruitment periods
Environmental Factors	4	Optimize computational resource usage
Economic Factors	7	Manage infrastructure and AI costs

7. Teamwork Details

Collaboration played a central role in the successful development of RecruitAssistant. From the early planning stages to implementation and testing, we emphasized structured coordination and clear task ownership. To maintain steady progress, we organized recurring internal meetings where we reviewed development milestones, addressed integration challenges, and aligned technical decisions. Version control and code collaboration were managed through GitHub, enabling transparent contribution tracking, branch-based development, and peer review before merging changes.

7.1 Contributing and functioning effectively on the team

To ensure productivity and accountability, the project was divided into functional domains rather than isolated tasks. Each domain had a primary responsible member while still allowing overlap and cross-support among teammates. The main domains included backend architecture, frontend development, AI/ML integration, database management, and system testing.

Backend system design and API coordination were primarily overseen by Mehmet Anıl Yeşil, who focused on structuring service layers and ensuring subsystem integration. Deniz Öztürk concentrated on frontend development, implementing user interface components and maintaining consistency in user experience. Emir Ensar Sevil handled database structure and data consistency concerns. Yüksel Barkın Baydar worked on AI-related components, including NLP-based processing and transcription integration. Cankutay Dündar focused on validation processes, test scenarios, and system robustness.

Despite these primary responsibilities, subsystem boundaries were not rigid. All team members reviewed architectural decisions and participated in integration discussions to ensure cohesion across the platform.

7.2 Helping creating a collaborative and inclusive environment

The development process was structured to encourage open discussion and collective reasoning. Because RecruitAssistant combines multiple technical disciplines—web development, backend services, AI processing, and data management—decisions often required evaluating trade-offs from different technical viewpoints. We deliberately avoided centralized decision-making and instead encouraged team-wide discussions for critical design choices.

By creating an environment where every member could question assumptions, propose improvements, and suggest alternatives, we reduced implementation risks and improved architectural consistency. This inclusive approach strengthened technical alignment and improved overall system quality.

7.3 Taking lead role and sharing leadership on the team

Leadership responsibilities were distributed according to subsystem expertise rather than assigned to a single coordinator. Each member took ownership of a specific technical area while remaining involved in system-wide discussions. This distributed leadership model allowed specialization without creating silos.

By sharing responsibility, the team maintained balanced workload distribution and improved responsiveness to challenges. When integration issues emerged, subsystem leads collaborated directly rather than escalating decisions hierarchically. This approach enabled faster resolution of technical problems and supported steady development progress throughout the project lifecycle.

8. Glossary

The terminology, abbreviations, and technical definitions used throughout this report are provided in Section 1.3 (Definitions, Acronyms, and Abbreviations). Readers may refer to that section for detailed explanations of key concepts referenced in this document.

9. References

[1] FastAPI. (n.d.). *FastAPI Documentation*. [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: 12-Mar-2026].

[2] React. (n.d.). *React – A JavaScript library for building user interfaces*. [Online]. Available: https://react.dev/. [Accessed: 12-Mar-2026].

[3] PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. [Online]. Available: https://www.postgresql.org/docs/. [Accessed: 12-Mar-2026].

[4] JSON Web Tokens. (n.d.). *Introduction to JSON Web Tokens*. [Online]. Available: https://jwt.io/introduction. [Accessed: 12-Mar-2026].

[5] Amazon Web Services. (n.d.). *What is an API?* [Online]. Available: https://aws.amazon.com/what-is/api/. [Accessed: 12-Mar-2026].

[6] Codecademy. (n.d.). *What is REST?* [Online]. Available: https://www.codecademy.com/article/what-is-rest. [Accessed: 12-Mar-2026].

[7] Cloudflare. (n.d.). *Hypertext Transfer Protocol (HTTP)*. [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/. [Accessed: 12-Mar-2026].

[8] Mozilla Developer Network. (n.d.). *HTTPS – HTTP over TLS*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/HTTPS. [Accessed: 12-Mar-2026].

[9] IBM. (n.d.). *What is Natural Language Processing (NLP)?* [Online]. Available: https://www.ibm.com/topics/natural-language-processing. [Accessed: 12-Mar-2026].

[10] IBM. (n.d.). *What is Speech to Text?* [Online]. Available: https://www.ibm.com/topics/speech-to-text. [Accessed: 12-Mar-2026].

[11] Stanford University. (n.d.). *Natural Language Processing Group*. [Online]. Available: https://nlp.stanford.edu/. [Accessed: 12-Mar-2026].

[12] Google Cloud. (n.d.). *Speech-to-Text Documentation*. [Online]. Available: https://cloud.google.com/speech-to-text/docs. [Accessed: 12-Mar-2026].

[13] Docker Inc. (n.d.). *What is Containerization?* [Online]. Available: https://www.docker.com/resources/what-container/. [Accessed: 12-Mar-2026].

[14] Microsoft Azure. (n.d.). *Cloud Computing Overview*. [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing/. [Accessed: 12-Mar-2026].

[15] Martin Fowler. (n.d.). *Microservices Architecture*. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed: 12-Mar-2026].

[16] OpenAI. (n.d.). *Natural Language Processing Applications*. [Online]. Available: https://platform.openai.com/docs. [Accessed: 12-Mar-2026].

[17] PostgreSQL Global Development Group. (n.d.). *PostgreSQL ACID Compliance*. [Online]. Available: https://www.postgresql.org/about/. [Accessed: 12-Mar-2026].

[18] OWASP Foundation. (n.d.). *OWASP Top 10 Web Application Security Risks*. [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed: 12-Mar-2026].

[19] Google Developers. (n.d.). *REST API Design Guide*. [Online]. Available: https://developers.google.com/discovery/v1/building-a-client-library. [Accessed: 12-Mar-2026].

[20] ACM Digital Library. (n.d.). *Artificial Intelligence in Interview Preparation Systems*. [Online]. Available: https://dl.acm.org/. [Accessed: 12-Mar-2026].