Bilkent University

Department of Computer Engineering

Senior Design Project

T2406

Recruit4Me

Final Report

22101007 Kanan Zeynalov kanan.zeynalov@ug.bilkent.edu.tr

22003108 Musa Yiğit Yayla yigit.yayla@ug.bilkent.edu.tr

22103449 Burak Efe Öğüt efe.ogut@ug.bilkent.edu.tr

22003416 Mehmet Ege Kayaselçuk mehmet.kayaselcuk@ug.bilkent.edu.tr 22003672

Bahadır Gunenc bahadir.gunenc@ug.bilkent.edu.tr

Fazlı Can, Mert Bıçakçı, Atakan Erdem

TABLE OF CONTENTS

1.Introduction

Recruit4Me aims to provide a web based software platform for hiring IT & tech professionals while hosting numerous features enhanced by technologies including AI & ML. Project proposes high innovation through two main stages, automated candidate elimination and pay as you go interviewer expert hiring done by companies. This way companies with limited HR resources and large applicant pools reduce costs by AI automation, followed by HR & technical expert hiring for conducting candidate elimination. Thus, with pay as you go hiring, further employment opportunities are introduced for interviewer experts, who also can be employed in some full time job, proposing extra revenue. First stage of Recruit4Me consists of automation tools, which companies can use, including CV parsing ML model labeling candidates' technical and soft skills, AI generated technical questions relevant to the job position, and speech to text models. This first phase reduces HR labor cost by performing automated and accurate preliminary candidate elimination. Furthermore, companies are able to customize these models with respect to their professional and ethical principles. Subsequently, the second phase, interview expert hiring, involves companies employing HR and technical experts for managing their recruitment. Given permission, interview experts can also manage candidate pools and introduce new potential candidates. We also include searching and filtering candidates' profiles by their skills, job listings by required expertise and skills, enabling further interaction.

1.1 Purpose of the system

The system's main purpose is to integrate the CV processing and testing question based candidate elimination in an automated manner, where companies are not required to handle different platforms to accept candidates.

1.1.1 Recruit4Me's Two-Stage Innovation

Recruit4Me is designed as a two-stage hiring platform that combines automated AI-driven screening with human insights in a sequential process. This approach tackles the recruitment funnel from both ends: the first stage rapidly filters and evaluates candidates using machine learning, and the second stage injects expert human judgment on a pay-per-use basis. By structuring the workflow into these stages, We aim to significantly reduce the burden on in-house HR teams while still ensuring that final hiring decisions benefit from human oversight and domain expertise.

1.1.2. Stage 1: Automated Candidate Elimination

In Stage 1, we employ a suite of AI and ML tools to conduct automated candidate screening. The process begins as candidates submit their CVs to the platform. At this point, a custom ML model parses each CV in detail, extracting key information such as the candidate's education, technical skills, work experience, project history, and even language proficiency. Using natural language processing (NLP) – including Named Entity Recognition (NER) to identify relevant entities like degrees, job titles, or skill keywords – the system builds a structured profile of each applicant from the unstructured resume text. It then evaluates the candidate across multiple categories (on the order of four or five major criteria) to determine their suitability. For example, the ML model rates or scores the candidate's qualifications in areas such as:

- **Educational Background** – e.g. the level and relevance of degrees, and even the ranking or prestige of institutions attended.
- **Technical Skills** – the depth and breadth of the candidate's skill set, programming languages or tools known, and how well these match the job requirements
- **Work Experience and Projects** – the quality, size, and impact of past projects or roles, indicating practical experience in the field.

Each category yields a component score, and together these form an aggregate candidate profile score that quantitatively represents the candidate's overall fit. Candidates are effectively ranked based on these scores. This data-driven elimination step allows the system to confidently short-list the top candidates for further evaluation, filtering out those who do not meet the role's baseline criteria. Importantly, the Stage-1 model is not a one-size-fits-all static filter – it can be customized to a company's specific requirements and values. For instance, an employer can adjust the weightings or thresholds in the model to emphasize certain skills or to align with the company's professional and ethical principles. The outcome of Stage 1 is a decision on whether each candidate is "qualified" for Stage 2. Only those who meet or exceed the predefined benchmark (e.g. top N% or above a score cutoff) are advanced to the next stage, while others are politely eliminated from the process (with the possibility of feedback). By automating this first-pass filtering, our Stage 1 dramatically reduces HR workload and cost: it performs an "accurate preliminary candidate elimination" without any human intervention, allowing companies – especially those with limited HR staff – to handle large applicant pools efficiently.

After a candidate submits their CV, the platform performs NLP-driven data processing (extracting entities like skills and experience and grading the resume on key criteria such as education, skill-match, projects, etc.). It then proceeds to question generation, presenting the candidate with structured interview questions. The candidate's spoken responses undergo response handling – they are transcribed via speech-to-text and analyzed with AI (OpenAI GPT-based models) – and a final AI evaluation is produced. A feedback loop continuously fine-tunes the ML models using accumulated data, improving accuracy over time.

Illustration of our Stage-1 automation pipeline.

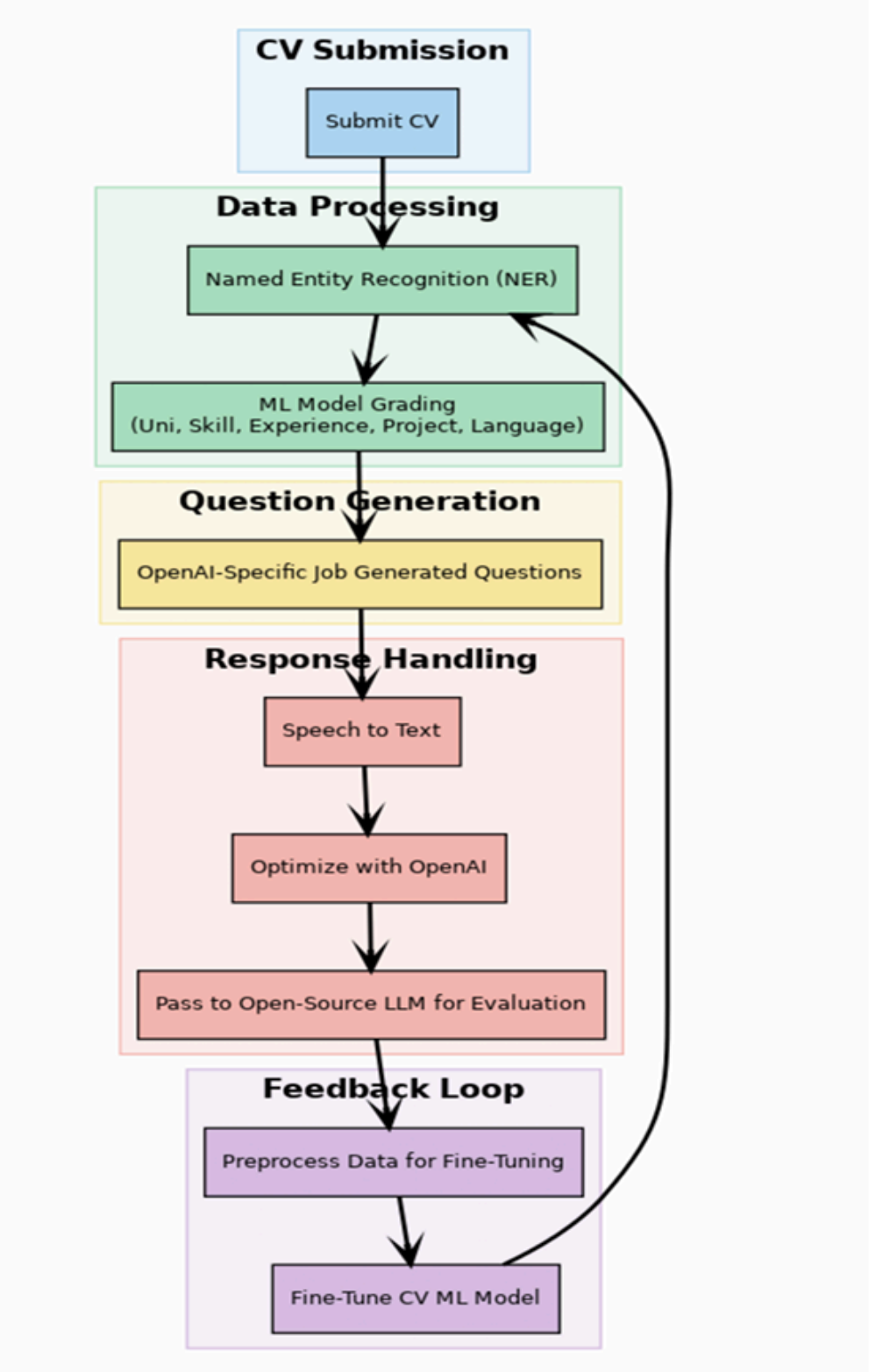


Figure 2. Real-time usage of system

Beyond static resume data, our Stage 1 incorporates an automated technical interview phase to further vet candidates. Drawing on the candidate's profile and the target job description, the platform uses an AI question-generation module to pose a series of structured questions to the candidate. These questions are designed to evaluate both technical expertise and problem-solving abilities, and are dynamically tailored to the role's requirements. In practice, the system can generate a set of questions of varying difficulty – for example, a few basic questions to verify fundamental knowledge, followed by more challenging scenario-based or algorithmic questions to probe the candidate's depth of understanding. This structured easy–medium–hard question set ensures a comprehensive assessment across proficiency levels.

We adopt a similar philosophy but use the powerful language understanding of GPT-4 to perform a nuanced evaluation of what the candidate said. The end result of Stage 1 is a rich, multidimensional profile for each candidate that includes their resume-derived scores, their performance on auto-generated interview questions, and a recommendation on whether to proceed. Only the candidates who pass this AI-driven screening (roughly analogous to making a "shortlist") move on to Stage 2.

1.1.3. Stage 2: Pay-as-You-Go Expert Hiring

Candidates who clear the automated Stage 1 are then forwarded into Stage 2, which introduces human expertise into the hiring process. Stage 2 is characterized by "pay-as-you-go" expert involvement. In practical terms, this means that companies using Recruit4Me can hire external HR and technical interviewers on-demand to conduct in-depth evaluations of the shortlisted candidates. Rather than relying solely on their internal HR personnel (which some startups or small firms might lack, or which might be overwhelmed during mass hiring), employers can tap into a marketplace of vetted interview experts through the platform. These experts – who are mostly experienced HR professionals or senior engineers in relevant tech domains – are engaged as needed (per interview or per batch of candidates), hence "pay-as-you-go." This model is beneficial for companies with limited recruiting resources because it converts a fixed overhead (maintaining a large full-time recruitment team) into a variable cost tied to actual hiring activity. It also creates a new kind of gig economy opportunity for interviewers: many of these experts could be professionals with full-time jobs who conduct interviews for extra income, allowing their specialized skills to be shared across organizations.

In Stage 2, the focus is on human judgment and interaction to complement the AI's Stage-1 assessments. The hired experts review the detailed candidate profiles and performance data coming out of Stage 1. They can conduct live interviews (e.g. a technical deep-dive or a soft-skill/cultural fit interview) with each candidate to validate and expand upon the AI's findings. The platform facilitates these interactions and provides the experts with tools to manage candidates. Experts can add qualitative evaluations, double-check the technical abilities (perhaps with coding exercises or system design questions in a live setting), and assess intangible qualities like teamwork, creativity, or attitude that automated tools might not fully capture. Additionally, we enable experts to manage candidate pools on the platform. For example, an expert can maintain a pool of "promising data scientists" identified through the process, which the company can revisit for future openings. Experts, with the company's permission, can also introduce new candidates into the pipeline – this could occur if, say, an expert knows a strong candidate in their network and wants to add them to the pool, thereby continuously enriching the talent database beyond those who originally applied. Throughout Stage 2, we provide tools for scheduling, feedback collection, and candidate tracking so that the collaboration between the company and the hired experts is seamless. The end result of Stage 2 is that the employer receives a curated set of final candidates, each vetted both by AI metrics and human interviews, ready for final hiring decisions or on-site interviews. By integrating this expert hiring stage, we ensure that the ultimate hiring decisions are informed by human expertise and contextual judgment – aspects like cultural fit or nuanced technical insight – which algorithms alone might miss, all while substantially reducing the burden on the company's own HR staff.

1.2. Design goals

- **Usability**

Recruit4Me prioritizes a user-friendly and intuitive interface to ensure smooth navigation for all users, including candidates, HR personnel, and experts. The system feature responsive design, accessible functionality compliant with WCAG standards, and real-time feedback through form validation and tooltips. Multi-language support and a built-in FAQ system are expected to enhance user experience and accessibility, making the platform approachable for a diverse audience.

- **Reliability**

The application is implemented to have a high reliability, which should be fault-tolerant. Robust error handling and proper data management using ACID-compliant databases assist

these to make an advancement for reliability. Load balancers, failover servers and automated backups with efficient fallback mechanisms are planned to be implemented to maintain an uninterrupted service experience for the users.

- **Performance**

Performance optimization is a core part of Recruit4Me, with fast response times for key features like login, CV parsing and getting candidate results. Asynchronous operations and caching reduce server load, while database optimization ensures efficiency even with large datasets. Stress testing guarantees the system can handle significant user traffic and heavy workloads without degrading performance.

- **Supportability**

Recruit4Me is built with modularity and maintainability in mind, ensuring easy debugging, updates, and enhancements. Comprehensive logging, monitoring, and documentation support developers, while CI/CD pipelines and extensive test coverage enable safe and seamless deployment of updates without disrupting user operations.

- **Scalability**

The application is designed to handle increased workload during peak times and user-load through proper cloud integration (AWS, Google, Azure) and database optimization methods such as sharding and replication. It supports horizontal scaling and microservices, to ensure the system can manage growth of workload without affecting performance.

2. Requirements Details

Functional Requirements

Recruit4Me must support all steps of a two-stage AI-driven hiring process. Companies (HR and hiring managers) need to create and manage job postings with required skills, experience levels, and screening criteria. Candidates must be able to register, create profiles, upload resumes, and apply to jobs. In Stage 1 (Automated Screening), the system automatically parses each resume (using a resume parser or AI model) to extract skills and experience and compares them against the job requirements. It then generates a customized screening test: structured quiz questions and coding problems relevant to the job. For coding questions, candidate code submissions are sent to the Judge0 code execution service for compilation and automated evaluation. The system scores each candidate (combining resume-job match, quiz answers, and coding results) and filters out those who fail the minimum threshold, producing a shortlist. In Stage 2 (Expert Interview), shortlisted candidates are made available for human experts. The platform should allow companies or contracted experts to schedule live technical interviews (e.g. via an integrated Jitsi video session) with candidates. Interviewers can review the candidate's Stage 1 profile and then record qualitative assessments (e.g. teamwork,

communication) in the system. Throughout, the system must expose RESTful APIs for all front-end workflows (user signup/login, job management, application status, test-taking interface, interview scheduling, etc.) and enforce role-based actions (e.g. only HR/company users can see all applicant scores, only candidates can take tests). System functionality also includes notification (e.g. email alerts for application updates or interview invites) and reporting (companies can view ranked candidate lists and test results).

Non-Functional Requirements

The system is designed to be responsive and usable: the web UI must be responsive and intuitive, with clear workflows for the various user types (candidates, company users, HR administrators) and form validation or tooltips in real-time. Accessibility (e.g. WCAG compliance) and multi-language support must be added in order to accommodate multiple users. In terms of performance, Recruit4Me must be able to process long-running tasks without locking out users. Long-running operations (resume parsing, LLM scoring, code checking) have to be done asynchronously or in the background so that API calls remain quick (e.g. page loads and submit answers must be less than a few seconds). With Azure Cosmos DB, it is possible to achieve low-latency data access (in single-digit milliseconds), which itself provides responsive UIs and quick query results even during load. The system should also be reliable and fault-tolerant: a breakdown in one component shouldn't take down the whole platform. This involves robust error handling, retries, and data redundancy. Automatic backups and database replication (Cosmos DB geo-redundant storage) ensure that data is not lost, and failover health checks (e.g. redundant Kubernetes pods) ensure that services don't go down. Security and access controls are top priority: everything must be communicated over HTTPS, and we have strict authentication. We implement stateless JWT tokens with embedded user roles so that each API request includes an encrypted token authenticating the user and authorizing them. Role-Based Access Control (RBAC) enforces, for instance, users in HR can only create job listings, company representatives can see that company's candidate pool and not others, and candidates should not be allowed to view internal scoring data. All sensitive operations (e.g. admin fixes, hiring needs modifications) must be logged for audit and must have rate-limits or alerts for unusual usage. Scalability and maintainability are also key. The architecture must accommodate user and data growth; for example, with microservices and container orchestration (AKS), we can scale out the web/API servers horizontally during peak loads. We design the data tier (Cosmos DB with MongoDB API, Azure Blob Storage for resume files) to scale out as required. This allows for peak recruitment campaigns or concurrent code-test submissions in volume without getting bogged down. We also prioritize clean, modular code (with heavy logging and test automation) so that new features or future AI models will be easily added. In short, the non-functional goals of the platform—usability, performance, reliability, security, and scalability—directly benefit the two-stage recruiting environment, delivering a smooth user experience and robust, scalable performance under real hiring loads.

2.1 Functional Requirements - Flow

2.1.1 User Management & Authentication

- **User Registration & Profiles**
    - Candidates register with email, password, and basic profile (name, contact, CV upload).
    - Company users register under an organization, assign roles (HR manager, expert).
    - Multi-factor authentication (email verification code; optional 2FA).
- **Role-Based Access Control (RBAC)**
    - Four primary roles: Candidate, HR Admin, Recruitment Expert, Company.
    - Permissions matrix defines which API endpoints and UI views each role can access.
    - JWT tokens with embedded role claims, auto-expiring, refreshed via secure endpoints.
- **Profile & Organization Management**
    - Company users create & update Organization records (name, domain, logo).
    - HR Admins can invite colleagues, assign or revoke roles.
    - Candidates can link/unlink social profiles (e.g. GitHub, LinkedIn) for profile enrichment.

2.1.2 Job & Application Lifecycle

- **Job Posting Creation**
    - HR Admins define job title, description, required skills, preferred skills, experience level, location, and application deadline.
    - Ability to set custom screening thresholds and test configurations per posting.
- **Job Listing & Discovery**
    - Public and private listing modes.
    - Search by keyword, location, skill tags, salary range.
    - Pagination and relevance sorting (weighted by match score).
- **Candidate Application Workflow**
    - One-click apply: candidate's profile and CV auto-attached.
    - Email notification to candidate and HR team on submission.
    - In the UI, candidates see "Submitted," "Screening," "Shortlisted," "Interview," "Offer," "Rejected" statuses.

2.1.3 Stage 1: Automated AI-Driven Screening

- **Resume Parsing & Scoring**

    - **NLP Extraction**: Education, skills, certifications, past employers, project descriptions.
    - **Feature Engineering**: University ranking lookup, skill frequency counts, project domain matching.
    - **Scoring Model**: Ensemble of rule-based filters (must-have skills) + ML ranker (logistic regression or fine-tuned language model) outputs a **fit score** (0–100).

- **Structured Assessment Generation**
    - **Question Bank**: Curated repository of technical & behavioral questions, classified by topic and difficulty.
    - **Adaptive Selection**: Based on fit-score, system selects 5–8 questions:
        - 2 Easy (verify basics)
        - 3 Medium (apply concepts)
        - 2 Hard (design/algorithmic).
- **Coding Challenge Evaluation**
    - **Judge0 Integration**: For each coding problem, candidate source is sent to Judge0.
    - **Test Suite**: 10–15 unit tests per problem (correctness, edge cases, performance).
    - **Scoring**: Pass rate (percentage of tests passed) + runtime efficiency (time to pass).
- **Text-Answer Evaluation**
    - **LLM Scoring**: Questions requiring explanations are scored via Azure OpenAI (GPT-4) prompts.
    - Returns structured JSON: { correctness: 0–10, depth: 0–10, clarity: 0–10 }.
- **Aggregate Screening Decision**
    - Weighted sum: 40% resume, 40% coding, 20% text answers → **overall score**.
    - Threshold (e.g. ≥ 65/100) → advances to Stage 2; else, automated “Thank you” and optional feedback.

2.1.4 Stage 2: Pay-As-You-Go Expert Interviews

- **Expert Marketplace**
    - Registered technical interviewers and HR consultants list their expertise areas, hourly rates.
    - Companies browse, invite, and schedule sessions.
- **Interview Scheduling & Management**
    - Integrated calendar (Google/Outlook sync) and time-slot booking.
    - Automated email/SMS reminders to candidate and expert.
- **Live Interview Tools**
    - Embedded Jitsi video session with screen-share and whiteboard for coding/design questions.
    - Recording & transcription (for audit, internal review).
- **Expert Feedback Collection**
    - Standardized evaluation form: technical depth, cultural fit, communication, problem-solving.
    - Free-text remarks.
    - Scores auto-aggregated with Stage 1 results for final shortlist.

2.1.5 Reporting & Analytics

- **Recruiter Dashboard**
    - Real-time charts: number of applicants per job, pass rates, average scores, funnel conversion.
    - Drill-down by time period, location, skill tag, or demographic.
- **Candidate Portal**
    - History of applications, screening results, interview feedback.
    - Downloadable certificates of completion.
- **Data Export & API**
    - CSV/Excel export of candidate lists, test results.
    - REST API endpoints for external BI tools (Power BI, Tableau).

2.2 Non-Functional Requirements

2.2.1 Usability & Accessibility

- **Responsive Design**
    - Mobile-first layout; automatically adapts to screen widths 320px–1920px.
    - Touch-friendly controls.
- **Intuitive Workflows**
    - “Wizard” steps for setting up new jobs, conducting assessments, and managing interviews.
    - Progress indicators, contextual help (tooltips, inline documentation).
- **Accessibility Compliance**
    - Meet WCAG 2.1 AA standards: keyboard navigation, ARIA labels, color-contrast ratio ≥ 4.5:1.

2.2.2 Performance & Scalability

- **Latency Targets**
    - < 200 ms for authenticated page loads (via CDN caching).
    - < 50 ms for AJAX calls to fetch candidate scores.
    - < 5 s end-to-end for Judge0 code evaluation (parallel test execution).
- **Throughput**
    - Handle up to 5,000 concurrent users.
    - 1,000 simultaneous code submissions per minute.
- **Elastic Scaling**
    - Azure Kubernetes Service (AKS) auto-scales pods by CPU/memory usage.
    - Cosmos DB automatically partitions data to maintain single-digit millisecond reads/writes.

2.2.3 Reliability & Availability

- **Uptime SLA** ≥ 99.5%
    - Azure availability zones for Kubernetes nodes and Cosmos DB replicas.
    - Health probes and automatic pod restarts on failure.
- **Disaster Recovery**
    - Daily backups of Cosmos DB and Blob Storage to secondary region.
    - “Playbook” scripts for full infrastructure redeployment via ARM/Terraform.

2.2.4 Security & Compliance

- **Data Encryption**
    - At-rest: Azure Storage Service Encryption and Cosmos DB TDE.
    - In-transit: TLS 1.2+ for all endpoints.
- **Authentication & Authorization**
    - JWT with RS256 signatures; short-lived access tokens (< 15 min), refresh tokens (< 7 days).
    - Fine-grained RBAC enforced by API gateway policies.
- **Secret Management**
    - Azure Key Vault for API keys (Judge0, OpenAI), database credentials, JWT private keys.
- **Audit Logging**
    - Immutable audit trails of key actions (job creation, user role changes, score overrides) stored in Azure Monitor logs and forwarded to a SIEM.

2.2.5 Maintainability & Observability

- **Modular Codebase**
    - Microservice separation: each bounded context (User, Job, Assessment, Interview, Reporting) in its own repo.
- **CI/CD Pipelines**
    - GitHub Actions builds, tests, containerizes, and deploys to AKS on every merge to main.
    - Canary and blue/green deployment strategies.
- **Monitoring & Alerting**
    - Application insights for performance telemetry.
    - Prometheus + Grafana dashboards for container metrics.
    - PagerDuty alerts for error rate spikes or resource exhaustion.

3. Final Architecture and Design Details

Recruit4Me employs a layered, cloud-native architecture. The UI layer is a web-based single-page application (implemented in Angular) that provides different dashboards for each role (candidate, recruiter, expert). It uses dynamic component rendering so that, for example, candidates only see application and test-taking views, while HR users see job management and

candidate analytics panels. The UI communicates with the backend solely over REST APIs (secured by JWT).

The Authentication layer is integrated into the API gateway: upon login, the backend issues a signed JWT that encodes the user's role (HR, company admin, candidate, etc.) and user ID. Every subsequent request must present this token, and the service validates it and enforces RBAC rules (e.g. "only users with Company role can create job postings", "candidates can only submit answers for their own applications"). This JWT-based auth avoids server-side sessions and ensures stateless, scalable session management.

The Business Logic (Service) layer is implemented with MongoDB REST Framework. This layer contains modular services for each functional domain: user/account management, job and application management, assessment generation/evaluation, and interview scheduling. Each service has its own set of REST endpoints. For example, the Assessment Service handles Stage 1 workflows: it accepts a candidate's resume and job ID, invokes the resume-parsing component (which uses NLP or Azure Cognitive Services), and then generates and stores a set of test questions. When the candidate submits answers, this service routes them appropriately—text answers go to the LLM evaluator (Azure OpenAI), and code submissions go to the Judge0 service.

The Judge0 integration is encapsulated as an external evaluation service. Judge0 is an open-source, scalable online code execution system often used in coding-assessment platforms. We host a Judge0 instance (e.g. on an Azure VM) and expose an API endpoint to it. When a candidate submits code, the backend sends the source code and test cases to Judge0 via a REST call. Judge0 compiles and runs the code in a secure sandbox and returns test results, which our service then records. By using Judge0's proven infrastructure, we ensure that running untrusted candidate code is isolated from our core system and can scale independently.

For AI-driven evaluation, we integrate Azure OpenAI (GPT-4). After coding tests, or for any free-text answer, the backend sends the candidate's answer and the question context to the Azure GPT endpoint. The model (prompted or fine-tuned for our domain) returns a structured score and feedback. For instance, GPT-4 can check if the candidate's answer covers all required points and return separate sub-scores (e.g. correctness, completeness, clarity). This LLM-based scoring component plugs into the business logic layer and feeds results back into the candidate's overall evaluation.

The Database layer uses Azure Cosmos DB (with the MongoDB API) as the primary datastore. Cosmos DB stores all structured candidate information: user accounts, job postings, candidate profiles, application records, and test results. Its distributed, multi-region nature provides automatic high availability and very low-latency reads/writes. (For example, a read of a candidate's scores or a job's requirements typically complete in single-digit milliseconds.) Large binary or semi-structured data—such as uploaded resumes and video interview recordings—are kept in Azure Blob Storage (with hot/cool tiering and optional CDN) for cost-effective, scalable file storage. We also use Azure SQL Database for certain relational aspects (e.g. if transactional consistency is needed), and Azure Cognitive Search to index candidate skills and profiles so that recruiters can quickly find top matches; these services complement Cosmos.

Infrastructure and Subsystems: All these components are containerized with Docker. We run them in an Azure Kubernetes Service (AKS) cluster, which provides automated deployment, scaling, and management. For example, each Django-based microservice is a separate pod, and an Angular frontend pod serves the UI. We chose Standard D8s_v5 VM nodes (8 vCPUs, 32 GB RAM) to ensure ample compute for ML inference and web services. AKS lets us easily add more pods when user traffic spikes (horizontal scaling) or roll out updates with zero downtime (rolling upgrades). Sensitive data (API keys, JWT secrets) is kept in Azure Key Vault and injected securely into pods. Communication between services within the cluster uses an internal network, and ingress is secured via HTTPS/TLS. Role-based restrictions (e.g. only the API gateway can talk to the Judge0 service) are enforced by network policies and by validating JWT scopes in each service. Logging and monitoring (via Azure Monitor) are in place to detect anomalies.

3.1. ML Based CV Evaluation

At the center of Stage 1 is the ML model that parses and evaluates CVs. This model operates as a pipeline with multiple steps. First, when a resume is uploaded (in PDF or Word format), the system extracts the raw text and applies Natural Language Processing to interpret the content. Using techniques like entity recognition, the system identifies important entities and sections in the resume (education, skills, companies, job titles, dates, etc.). It employs domain-specific vocabularies and possibly pre-trained embeddings (e.g. language models fine-tuned on resume data) to accurately recognize technical skills and credentials even if they are presented in varied formats. Once key features are extracted, they are fed into a machine learning evaluation engine. This engine consists of a trained classifier model that outputs a "job

fit score" for the candidate, broken down by category. The model has been trained on historical hiring data or simulated data: for supervised learning, the team prepared input-output pairs where inputs are candidate features (skills, experience, etc.) and outputs could be an expert-defined rating of candidate quality or a hire/no-hire label. Modern approaches could involve an ensemble of algorithms – for example, a combination of a rule-based matching (to ensure essential requirements are met) and an ML-based ranking (to sort the remaining candidates by predicted suitability). We are giving scores based on factors like university ranking, depth of skills, project quality, and so on, indicating a scoring methodology that quantifies each dimension of the CV. All these factor scores are aggregated into an overall profile score. Candidates are categorized (or tagged) based on this profile – for instance, categorized as "Highly Qualified" vs "Moderate match" vs "Unsuitable," or simply sorted in descending order of their score. Such categorization helps in managing candidate pools: the system could automatically group candidates by skill match percentage, or highlight which specialized role(s) they best fit, given that Recruit4Me is initially focused on IT roles like Data Scientist, ML Engineer, etc.

A notable aspect is our project's ability to match candidates to companies' specific requirements. When a company creates a job listing, they input required and preferred skills, experience level, and other criteria. The ML evaluation engine uses these as parameters to adjust scoring. For example, a candidate skilled in Python might score very high for a Data Scientist role requiring Python, but lower if the job is for a Java Developer. This dynamic matching ensures the scoring is context-aware. The output of this ML model is not just a binary keep/discard, but a nuanced profile that can be reviewed. The system provides a job suitability score and ranking for each candidate, which the platform can present on a dashboard to the employers. By translating a complex resume into a set of standardized metrics, the tool enables quick comparisons across candidates on an "apples-to-apples" basis.

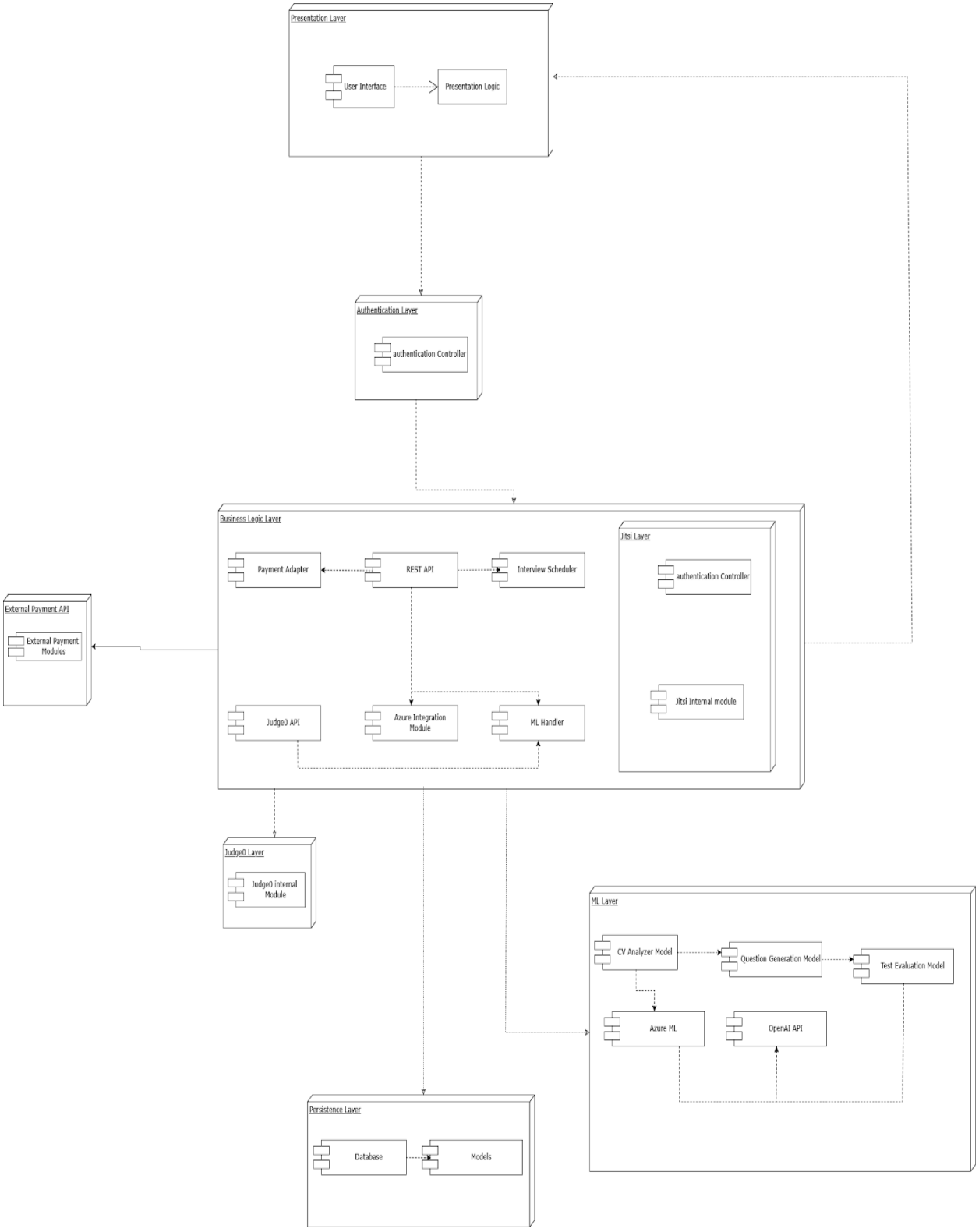
From an architectural perspective, this ML component is built in Python, utilizing frameworks like PyTorch and TensorFlow for model training. We are using a pre-trained model (LLaMA 70B) as a starting point and fine-tuning it on their recruitment data and additionally fine-tuning gpt4o on Azure Services to get best results. The training pipeline involves data collection and labeling, followed by fine-tuning on a pre-trained model and even reinforcement learning to continuously improve the model. There is also a feedback loop: as more candidate data flows through the system and as human experts in Stage 2 provide feedback on candidates, those outcomes are fed back into updating the ML model. We are using a

continuous fine-tuning approach provided by the Azure GPT Architecture and keep putting on the performance of the model by this new data.This continuous learning approach means the resume screening model should get more accurate over time, adapting to what we can consider a “successful hire.”

3.2. Structured Question Selection and Adaptive Testing

A key innovation of our platform is the structured, company-driven question selection system, which ensures a customizable yet standardized approach to candidate evaluation. Rather than relying on AI to generate interview questions dynamically, we provide companies with a pre-defined, high-quality dataset of technical questions. Companies using our platform can customize the assessment process by selecting specific question topics (e.g., "Greedy Algorithms," "Binary Search," "Dynamic Programming") and difficulty levels (Easy, Medium, Hard) tailored to their hiring needs. This allows employers to create a flexible, role-specific technical assessment while maintaining consistency across candidate evaluations.

When a candidate progresses past the CV screening stage, they are prompted with a structured set of questions selected based on the hiring company's criteria. The company has full control over which topics are assessed, ensuring that the questions align with the job description and the skills required for the role. Unlike AI-generated question banks that might produce inconsistent or overly generic prompts, our curated dataset provides a balanced mix of theoretical, problem-solving, and applied coding questions, making the evaluation process both precise and relevant to real-world engineering tasks. In edge cases we have a question dataset in Azure VM to handle these.


Presentation Layer
User Interface
Presentation Logic
Authentication Layer
authentication Controller
Business Logic Layer
Payment Adapter
REST API
Interview Scheduler
Judge0 API
Azure Integration Module
ML Handler
Jitsi Layer
authentication Controller
Jitsi Internal module
External Payment API
External Payment Modules
Judge0 Layer
Judge0 internal Module
ML Layer
CV Analyzer Model
Question Generation Model
Test Evaluation Model
Azure ML
OpenAI API
Persistence Layer
Database
Models

Since our platform relies entirely on Microsoft Azure, we designed our deployment architecture to be cloud-native, ensuring high availability, scalability, and secure data management. Our solution requires a balance between compute power for ML inference, database performance for handling user data, and storage capacity for resumes and interview transcripts.

The backend and frontend is deployed on Azure Kubernetes Service (AKS), allowing for efficient containerized deployment and scalable management of web services.

Instance Type: Standard D8s v5

- 8 vCPUs, 32 GB RAM
- 512 GB SSD (Premium Storage) for fast disk access
- Supports containerized deployment with Docker & Kubernetes
- Cost: ~$0.64 per hour *(Region: EAST US2)*

For database management, we use Azure Cosmos DB (MongoDB API) to store structured candidate profiles, job listings, and recruitment history. Cosmos DB provides low-latency access to data, allowing fast retrieval of candidate information for AI-driven evaluations.

Azure Cosmos DB (MongoDB API)

- 1 TB storage, scalable to demand
- Geo-redundant backups for reliability
- Cost: ~$0.025 per GB per month

For our Machine Learning model(GPT 4o and LLAMA 70B), we are using Azure ML Compute Instance.

Instance Type: Standard NC6s v3 (GPU-enabled)

- 6 vCPUs, 112 GB RAM, 1 NVIDIA Tesla V100 GPU (16GB VRAM)
- 1 TB SSD for storing model artifacts and candidate data
- Cost: ~$1.50 per hour *(Region: EAST US2)*

This instance is dynamically allocated only when active candidate evaluations are running. When not in use, it scales down automatically, reducing costs. Considering other costs such as Azure Image Extraction and Cognitive services, we are assuming that we will have a 500$ estimated cost each month. Deploying and using open source LLM models for evaluating speech would help us decrease our costs by %15, which are in future plans to be completed.

3.3. Persistent data management

Managing a significant amount of both structured and unstructured data, such as test results, video interview transcripts, candidate resumes, and AI-generated assessments, is part of our senior project. We use Microsoft Azure for cloud-based data management, guaranteeing

scalability, security, and high availability, in order to effectively store, process, and retrieve this data.

3.4. Database and Storage Solutions

Unstructured data, including test submissions, video interview recordings, and candidate resumes (in PDF and DOCX formats), are stored using Azure Blob Storage. Hot, cool, and archive tiers are among the tiered storage options offered by this storage solution, which maximizes cost effectiveness while guaranteeing quick access when required. Additionally, Azure Blob Storage integrates with Azure Content Delivery Network (CDN) to improve performance for users and speed up data retrieval.

We use Azure SQL Database and VM to store structured data, including test results, interview assessments, candidate profiles, and recruitment progress. By guaranteeing ACID compliance, this relational database offers data transactions consistency and integrity. High availability and auto-scaling capabilities guarantee that the database can manage varying workloads during periods of high recruitment.

Semi-structured data, such as activity logs, dynamically generated AI replies, and metadata from the hiring process, are also stored in Azure Cosmos DB. Flexible schema management and multi-model database formats, including document and key-value storage, are supported by the NoSQL architecture. Low-latency access across several regions is made possible by its worldwide distribution capabilities.

To enhance search and retrieval performance through full-text search, semantic ranking, and AI-powered candidate profile indexing is our goal by using Azure Cognitive Search. Therefore, it becomes quite easy for HR to search and find out the candidates who are best suited for their job openings mainly based on their matching qualifications, experience, and talents. Additionally, to make the search process even more relevant, and to rank the results better, we can add machine learning models to the search service.

3.4.1. Data Processing and AI Model Deployment

Our AI model for HR automation is OpenAI models on Azure and VMs. Once the model is trained and deployed in this manner, machine learning workloads are running with optimized performance. Additionally, the Azure Machine Learning Service is used for model versioning, automated hyperparameter tuning, and MLOps integration.

Azure Functions is used to implement automation for various data processing tasks such as document parsing, test result evaluation, AI-driven resume screening, etc. This serverless compute solution which empowers event-driven execution minimizes continuous infrastructure management. Running of Functions is done when a candidate submits a new application, runs AI-based evaluations, and updates the recruitment status change.

3.4.2. Security, Compliance, and Data Integrity

All the stored data is protected from hacking, phishing, and other cyber threats by using Azure Key Vault in the encryption process. On the other hand, Role-Based Access Control (RBAC) is the foundation of all security to guarantee the employer's trustworthiness in dealing with candidate’s data. To make it even more secure, Azure Active Directory (Azure AD) provides

access to acceptable and secure authentication and access along with multi-factor authentication (MFA) for added security.

Our approach to data management is furthermore aligned with the GDPR and privacy protection acts and thus, ensure that candidate data is only used in privacy and law abidance ways. Analysis and monitoring by Azure Monitor and Azure Security Center act as sources of information about data traffic and the systems' functionality as well as the performance of the relevant system. Using the infrastructure running in Microsoft Azure, we ensure a smooth transfer of candidate data, employ AI driven workflows to the offing, and boost safety in handling HR sensitive data.

3.5. Access Control and Security

Accessing only the authorized resources through the implementation of an effective access control system is one of the vital goals of the HR automation system. Access control is based on the role (RBAC) and token-based authentication which check if someone is a real user or not, respectively. As the case is, for complete security protocols, the system employs a JSON Web Token (JWT) authentication process the function of which is to test every code containing the possible security breaches before providing access to the API.

Authentication and Role-Based Access Control

The system assigns a role to each user at the time of registration or system entry with a specific set of privileges. When the login is successful, they are provided with an encrypted JWT that contains the role followed by this information which is used as a check on every request. The platform sorts out HR personnel, company representatives, and candidates to make sure each one interacts only with the data and features associated with their respective roles. HR personnel are endowed with the following capabilities: They are allowed to assess employment licenses, manage intakes, and set the job intake date. In any case, only by working within the boundaries of candidate management, they are able to do these tasks. Company representatives have admin rights where they can create or rearrange job advertisements, parametrically define exam going criteria for a standpoint, conduct the human resources pick up role, and not adjust project appraisal by HR but they are not authorized to modify candidate ratings registered by HR. Candidates mainly can only take tests, make applications, and attend interviews without allowable access such as company configurations or internal score reports by the company.

Security Measures and Restricted Access

To ensure the integrity and prevent unauthorized alterations to the system, important operations are enforced with multi-layered security techniques. Those sensitive activities, e.g., hiring criteria upgrades, test results overrides, company's confidential data access, are entered in logs and actively monitored. Unauthorized attempts provoke security alerts, and failure to enter the right credentials result in the temporary suspension of access.

The system also protects the intercommunication with other services such as Judge0 online compiler and ML-based test evaluation model. Only the platform's direct requests with the services have the right to interact with them, such that the common data submissions and

evaluations remain under the full command and are not tampered with. API endpoints executing tests and candidate evaluation, using access tokens, are hardened against unauthorized use, preventing unauthorized running of test cases or data tampering.

Data Protection and API Security

Moreover, to strengthen the security of data transfer, all the interactions between the client, the server, and the external services are carried through HTTPS and shielded APIs. Confidential user data, i.e., candidate scores, interview recordings, and CV data, is defended with a strict database access control policy, which guarantees that only authorized personnel are able to handle or change the stored information. Furthermore, such policies of token expirations enable the system to automatically invalidate a session after a certain time of inactivity, which in turn makes it impossible for unauthorized users to access the system due to token leakage. Integration of RBAC, JWT authentication, API security and monitoring mechanisms in the system make for the data of hiring's protection and confidentiality and resistance to unauthorized access and manipulation. Thus the hiring process is kept on the right track.

3.6 Subsystem Services

3.6.1 UI Layer

Recruit4Me is aimed to be delivered with an elegant and useful user interface. The UI layer aims to increase ease of use, by favoring interactive components and coupling relevant panels into the same pages. Recruit4Me aims to avoid complex UI which hinders comprehension of the application. Conditional rendering technique is used to enable different types of users, namely company, interviewer and candidate to perform their respective use cases.

3.6.2 Auth Layer

Recruit4Me aims to protect the confidential data of users, i.e. email, password and personal information. This layer aims to prevent injections, penetrations etc. security attacks that can happen with Azure Key Vault and Azure Networking Services.

3.6.3 Business Logic(Service) Layer

Naturally, RESTful endpoints are exposed to interact with backend services and the database instance(s) of our application. While developing these endpoints, we followed crucial software design patterns such as obeying law of demeter and meaningful coupling to adapt quickly to future changes and nicely persist our codebase.

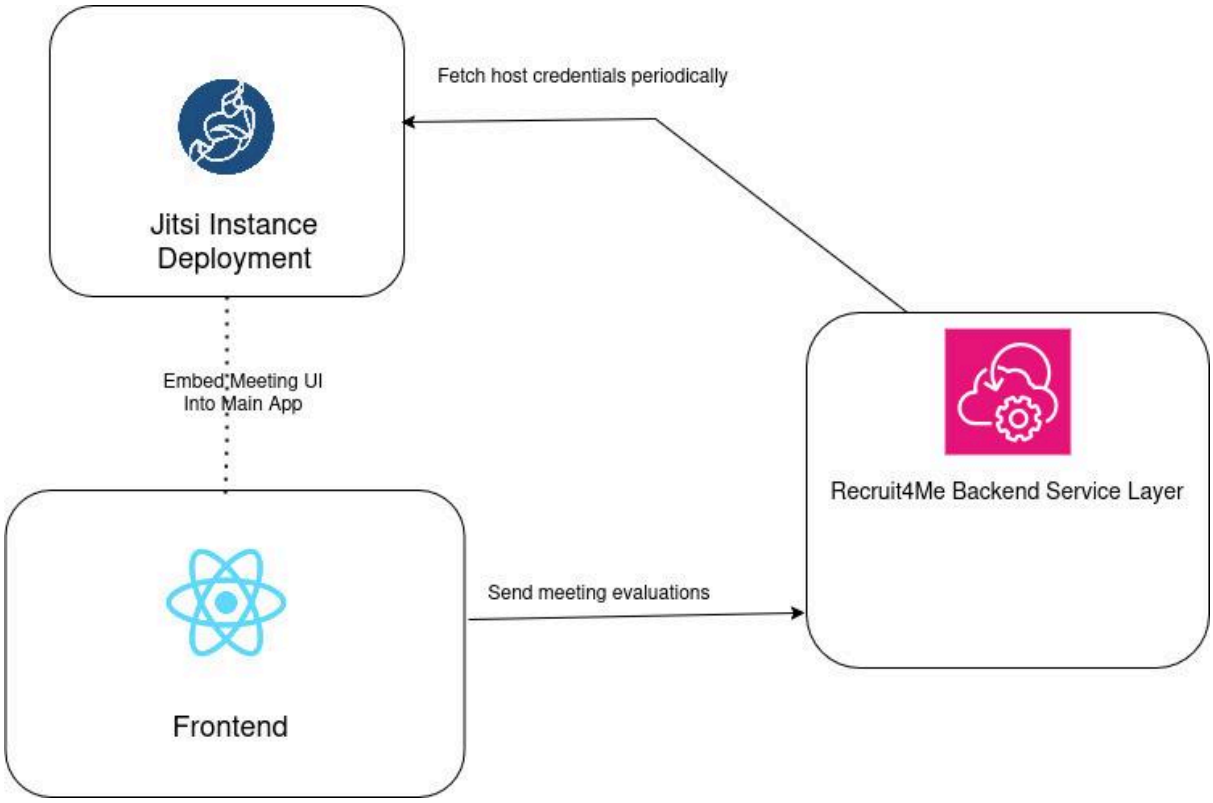
Different service layers exist to couple functionality regarding roles, and other entities such as interviews and job listings.

3.6.4 Database Layer

Database persistence is a must for Recruit4Me. Since the initial design of our ER diagram and the database schema, common database development principles have been applied. Moreover, various triggers are used to enhance the database.

3.6.5 Jitsi Instance Layer (Hosting Interviews)

Recruit4Me utilizes Jitsi, which is an open source platform enabling conducting online video meetings. Jitsi allows self hosting in Ubuntu servers. Recruit4Me hosts a Jitsi instance to enable interviewer experts to conduct meetings with candidates. The website for Jitsi instance is accessible through all ip addresses by default, but when creating and joining meetings users are required to authenticate through Jitsi's user interface with credentials given by Recruit4Me.



(Figure 4.5.1)

The Jitsi deployment for Recruit4Me is periodically invoking an endpoint exposed solely for this instance in the backend service layer, to fetch host credentials. Users use these credentials to participate in meetings. Jitsi UI is embedded in the main application, allowing users to remain at the original website (Figure 4.5.1).

Upon meetings are conducted, each participant is asked to evaluate their experience. Upon getting acceptable response from participants, meetings are marked as terminated and interview experts are awaiting payment. In case an issue arises, participants can state this during meeting evaluation, and if necessary logs in the Jitsi instance regarding meeting details (e.g join, leave, create times) can be manually checked. Later this process can be automated as well (Figure 4.5.1).

4. Development and Implementation Details

This section explains how we implemented the system components.

Frontend - We chose React JS for component-based UI, using Material-UI for a responsive design. The app communicates with backend APIs over HTTPS and handles user session state (storing the JWT in browser storage securely). Key pages include: Login/Register, Employer Dashboard (job and candidate list), Candidate Profile page, Expert Dashboard (assigned interviews, question manager), and an Admin panel. Forms include client-side validation (e.g. email format, required fields). We also implemented client-side pagination for long lists (e.g. showing 20 candidates per page), and lazy loading of images. The frontend consumes Azure AD's login widget (MSAL for single sign-on) to authenticate. During development, we used React Router for navigation and Redux for state management. We integrated rich text editors for job descriptions and question entries. CSS is structured with BEM naming to maintain clarity. Accessibility features (tab order, alt text, aria-labels) are implemented in accordance with WCAG guidance. We also implemented error boundaries so the UI can gracefully show error messages if an API call fails. The source is organized into modules (Auth, Employer, Candidate, Expert, Common). Build and deployment use a CI/CD pipeline: on each commit, a GitHub Action runs tests and then deploys the build artifact to Azure Static Web Apps, which serves the frontend under HTTPS.

Backend - The backend is built with Node.js, containerized with Docker. We structured it as multiple microservices (as in the design). Each service has its own code repository and Dockerfile, allowing independent development. Key aspects:

- **API Design:** All services expose RESTful endpoints (e.g. /candidates, /jobs, /interviews). The endpoints follow standard HTTP status codes. We implemented Swagger/OpenAPI documentation for clarity. The API gateway (or a reverse proxy) routes requests to the correct service based on URL patterns, and injects the authentication token for verification.
- **Authentication Service:** We integrated with Azure AD B2C. The service uses the passport-azure-ad library to validate JWT tokens. On login, we map AD groups/roles to our internal RBAC (e.g. "Employer" group → employer-role). JWTs have short expiration times for security; the client uses refresh tokens to renew.
- **Business Services:** Each microservice uses an internal library of data-access objects (DAOs) to talk to the database. We used Sequelize ORM for Azure SQL and the Azure SDK for Blob Storage. Business logic checks enforce all requirements: for instance, the

Interview Service checks that only the assigned expert can submit an interview result. We used Mocha/Chai for unit tests and Postman for integration tests.

- **LLM Integration:** For our fined model, we utilized Azure OpenAI Service (GPT-4o) and Meta LLaMA 3. We fine-tuned LLaMA 70B on a corpus of anonymized resumes and job descriptions. The fine-tuning pipeline used PyTorch on the AzureML compute. Once trained, the model was deployed as a managed endpoint. The Candidate Service sends an HTTP request to this endpoint with the resume text; the model returns parsed fields. We also implemented logic to handle prompts for the model (e.g. “List the degrees and skills mentioned”), ensuring the model operates in a predictable manner. The system respects token limits by trimming excess text. In testing, we validated that the LLM parsing achieved high accuracy on benchmark CVs. In addition, we explored using an LLM to generate interview questions. Using GPT-4o via API, the Question Service can propose new questions based on a job’s keywords. This feature is optional and was implemented as an asynchronous task, not part of the mandatory pipeline.
- **Data Storage and Access:** The application connects to Azure SQL using a secure connection string (with Azure Key Vault). Database migrations are scripted via a tool (e.g. Liquibase) for schema changes. Blob Storage is accessed via the Azure SDK – for example, when a resume is uploaded, the backend streams the file directly to a Blob container, then stores a reference URL in SQL. We enabled soft-delete on blobs in case of accidental deletion.
- **Logging and Monitoring:** We implemented centralized logging using Azure Monitor and Application Insights. Each service logs errors and key events (e.g. “Interview scheduled: ID=123”). Health probes (HTTP endpoints) are defined for each microservice so that Kubernetes (AKS) or App Service can auto-restart failed pods.

LLM Integration Details Large Language Models are at the core of our system’s intelligence. We fine-tuned a version of LLaMA 3 (70B) on recruitment data, and we also configured access to GPT-4o for on-demand tasks. For resume parsing, the model input is the plain text of the candidate’s resume; the output is a JSON with fields (e.g. “education”: [“B.Sc. Computer Science”], “skills”: [“Python”, “Angular”], “years_experience”: 3, etc.). It is done by the document intelligence service of Azure. To train the model, we used Azure GPU instances and datasets of labeled resumes. We took care to handle privacy (removing PII in training data) and to evaluate the model’s fairness. For question generation, we send a prompt like “Generate a medium-difficulty coding interview question for a Java developer.” The model returns a new question string. Experts can review and edit any AI-generated question before use. We also integrated Azure Cognitive Services for minor tasks: e.g. Language understanding (to detect English vs other languages) and OCR for scanned resumes. However, the main parsing is done by the LLMs. Finally, for scalability, we set up Azure Kubernetes Service (AKS) clusters with horizontal pod autoscaling for the LLM endpoints. This ensures that during peak resume-upload periods (e.g. after a job posting), the model can handle bursts.

5. Test Cases

Test-ID	T-1	**Category**	**CV Processing**	**Severity**	**High**
Objectives	Verify that the system correctly extracts relevant information (name, experience, skills, education) from various PDF's. Edge Cases: CVs with images, scanned documents, missing sections.				
Steps	Upload differently organized CV PDFs. The system extracts and structures the data. Compare extracted data with the actual CV content.				
Expected	Extracted data should match at least 95% of the actual CV information.				
Date-Result	28.02.2025 - Information from various formats of CVs have been extracted successfully.				

Test-ID	T-2	**Category**	**Model Evaluation**	**Severity**	**High**
Objectives	Ensure that the model correctly assigns different scores for junior, mid, and senior candidates. Edge Cases: CVs with similar experience but different education levels or vice versa.				
Steps	Submit three CVs for the same role, one for each level (junior, mid, senior). Retrieve the model's score for each candidate. Compare the results with expected values. Repeat this process for multiple occasions.				
Expected	Senior-level candidates should score higher than mid-level, and mid-level candidates should score higher than junior.				
Date-Result	29.02.2025 - Initial trial of different levels has shown accuracy.				

Test-ID	T-3	**Category**	**Model Validation**	**Severity**	**High**
Objectives	Validate that the model correctly identifies when a candidate applies for a role unrelated to their experience. Edge Cases: CVs with mixed experience across multiple fields.				
Steps	Submit a Data Scientist CV for a Cybersecurity Engineer position. Observe the model's response.				
Expected	The system should fairly evaluate the CV based on the level, junior may				

	get a high score, whereas senior can get lower.
Date-Result	01.03.2025 - In senior level CVs model answers with a lower experience and skill score based on the job explanation, however junior level CVs are evaluated higher as it is normal to have a junior inexperienced.

Test-ID	T-4	**Category**	**Frontend-Backend Model Integration**	**Severity**	**High**
Objectives	Ensure that the frontend correctly displays the model's evaluation score and integrate with it. Edge Cases: Model API delay, incorrect frontend parsing of the score.				
Steps	Submit multiple CVs for evaluation. Check if the frontend correctly retrieves and displays the score.				
Expected	The frontend should show the same score returned by the model.				
Date-Result	5.03.2025 - Scalability and stress testing is successful on model				

Test-ID	T-5	**Category**	**Candidate Ranking**	**Severity**	**High**
Objectives	Validate that the ranking of candidates updates dynamically when a new, better candidate is added.				
Steps	Submit a highly qualified candidate. Observe if the system adjusts the ranking accordingly.				
Expected	The new candidate should be placed in a higher rank, pushing down less qualified candidates.				
Date-Result	1.05.2024 - Successful				

Test-ID	T-6	**Category**	**Filtering Feature**	**Severity**	**Medium**
Objectives	Ensure that HR users can filter candidates based on experience or other skills. Edge Cases: Gaps in employment history, freelancers with multiple short-term contracts.				
Steps	Apply a filter for candidates with e.g. 5+ years of experience. Check if only qualified candidates are shown.				
Expected	Only candidates with 5+ years of experience should appear in the filtered results.				

Date-Result	22.04.2025 - Sucessful

Test-ID	T-7	**Category**	**AI Question Generation**	**Severity**	**High**
Objectives	Validate that the system generates relevant interview questions based on the job role and CV content. Edge Cases: CVs with multiple roles, missing skill descriptions.				
Steps	Submit a DevOps Engineer CV. Observe the generated interview questions.				
Expected	Questions should be role-specific (e.g., Kubernetes, CI/CD pipelines) and tailored to the candidate's experience.				
Date-Result	22.04.2025 - Questions are generated based on specifications				

Test-ID	T-8	**Category**	**Test Evaluation**	**Severity**	**High**
Objectives	Ensure that coding assignments are evaluated properly Edge Cases: Compiler issues				
Steps	Upload a code response.				
Expected	Judge0 should return the result properly based on test cases which work in Azure VM.				
Date-Result	22.04.2025 - Successful				

Test-ID	T-9	**Category**	**System Performance**	**Severity**	**High**
Objectives	Measure response time for processing a batch of 1000 CVs. Edge Cases: High server load, large file sizes.				
Steps	Submit 1000 CVs in a single request. Measure time taken for processing.				
Expected	System should process all CVs within acceptable time limits (e.g., under 5 minutes).				
Date-Result	22.04.2025 - Model responded immediately to all candidates.				

Test-ID	T-10	**Category**	**Security**	**Severity**	**High**
Objectives	Ensure that unauthorized users cannot access candidate data or system functions. Edge Cases: Token expiration, session hijacking attempts.				
Steps	Attempt to access candidate scores without logging in. Try logging in as a candidate and accessing HR data.				
Expected	Unauthorized users should be blocked from restricted data.				
Date-Result	27.02.2025 - Role based login is working correctly, candidate is not shown HR or company private data or vice versa.				

Test-ID	T-11	**Category**	Functional	**Severity**	High
Objectives	Ensuring user will upload correct type of document as a resume				
Steps	1-Inform the user about the accepted file type 2-Enabling user to upload the file 3-If user uploads correct file type, then user will be informed 4-If user tries to upload something different file type, warn the user about the expected file type				
Expected	User must not be able to upload any file type except pdf				
Date-Result	23.02.2025-Successful				

Test-ID	T-12	**Category**	Functional	**Severity**	High
Objectives	Ensuring that user will upload correct file type as an image				
Steps	1-Warn the user about the accepted file type 2-Allow user to select the file 3-control the file 4-If the image is .png or .jpeg, then user is informed about the success of the upload 5-If the file is different than mentioned, user is informed that operation is failed				
Expected	User should only upload .jpeg or .png				
Date-Result	2.3.2025 - Successful				

Test-ID	T-13	**Category**	Functional	**Severity**	High
Objectives	Send mails to reset password				
Steps	1-Ask user an input his/her new password 2-Ask them to type the password again 3-Ask them to type the verification code 4-If the new password and retype password don't match, send appropriate warning message 5-If verification code is not true, send appropriate message 6-If the retyped password is match with the new password and verification code is true, then change the password and send appropriate message				
Expected	If the conditions satisfied, user can change its password				
Date-Result	2.3.2025 - Successful				

Test-ID	T-14	**Category**	Functional	**Severity**	High
Objectives	Send mail to verify the account				
Steps	1-User enters his/her email to register 2-Verification code is sent 3-If the user correctly enters the code, then user account will open 4-If the user not correctly enters the code, then appropriate message is sent				
Expected	If the condition satisfied, user can open new account				
Date-Result	2.3.2025 - Successful				

Test-ID	T-15	**Category**	Functional	**Severity**	High
Objectives	UI components should look fine on different screen				
Steps	1-Develop the screens 2-Test them on different monitors				
Expected	In different-sized monitors, screens should look delicate and not hinder the functionality.				
Date-Result	1.05.2025 - Every screen sees pages based on dynamic width and height.				

Test-ID	T-16	**Category**	Non-Functional	**Severity**	High

Objectives	User should not wait for too long for the functionalities
Steps	1-Develop the functionalities 2-Measure the time 3-If the waiting time is too long (5-30 seconds, depend on the functionality) then reconsider the implementation
Expected	All functionalities should not exceed the boundary time
Date-Result	1.05.2025 - For small deployment the durations are acceptable i.e. 30 seconds maximum.

Test-ID	T-17	**Category**	Functional	**Severity**	High
Objectives	Verification code should be generated randomly				
Steps	1-The code generating random numbers tested several times 2- Generated codes are observed				
Expected	No obvious relation between those code sequences is found				
Date-Result	2.3.2025 - Successful				

Test-ID	T-18	**Category**	Functional	**Severity**	High
Objectives	Verification code should be stored				
Steps	1-Generated numbers are stored in database 2-Those generated numbers should match with the users				
Expected	Generated sequences are observable in the database				
Date-Result	2.3.2025 - Successful				

Test-ID	T-19	**Category**	Functional	**Severity**	High
Objectives	Verification code should be deleted in database after 5 minutes				
Steps	1-Generated numbers stored in the database 2-5 minute after generation, the code must be deleted in the database				
Expected	Generated codes should be deleted after 5 minutes				
Date-Result	2.3.2025 - Successful				

Test-ID	T-20	**Category**	Functional	**Severity**	**High**

Objectives	Ensure candidates, HR experts, and company representatives can successfully register and log in.
Steps	1.Navigate to the registration page. 2.Fill in the required details (name, email, password, role selection). 3.Submit the registration form. 4.Check for a confirmation email. 5.Log in using the registered credentials. 6.Verify successful redirection to the correct user dashboard.
Expected	Users can register and log in without issues. Incorrect credentials should result in an “Invalid credentials” error message.
Date-Result	1.05.2025 - Successful

Test-ID	T-21	**Category**	Functional	**Severity**	High
Objectives	Ensure company representatives can successfully post a job listing				
Steps	1.Navigate to the "+" sign to the section for “Add New Job”. 2.Fill in all required fields (job title, company, location, requirements, etc.). 3.Submit the form. 4.Verify that the job appears correctly in the “Jobs” section. 5.Try editing and deleting the job post.				
Expected	The job post should be created, displayed, editable and deletable				
Date-Result	10.04.2025 - Successful				

Test-ID	T-22	**Category**	Functional	**Severity**	High
Objectives	Ensure candidates can apply for job listings				
Steps	1.Navigate to the “Jobs” page. 2.Select a job and click "Apply." 3.Upload a resume and enter additional required information. 4.Submit the application. 5.Verify that the application appears in the candidate's "Results" section. 6.Verify that the application appears in the employer’s "Jobs" section.				
Expected	Candidates should successfully submit applications and employers should be able to see them.				
Date-Result	10.04.2025 - Successful				

Test-ID	T-23	**Category**	Functional	**Severity**	High

Objectives	Ensure HR Experts can schedule interviews with candidates
Steps	1.HR logs in and accesses the list of applicants. 2.Selects a candidate and clicks “Schedule Interview.” 3.Chooses a date and time, adds interview details, and confirms. 4.Candidate receives an interview invitation notification. 5.Candidate accepts or declines the interview.
Expected	Scheduled interviews should be visible to both HR and candidates, notifications should be correctly sent.
Date-Result	10.04.2025 - Successful

Test-ID	T-24	**Category**	Functional	**Severity**	Medium
Objectives	Ensure candidates can efficiently search and filter jobs				
Steps	1.Navigate to the “Jobs” search page. 2.Use the search bar to find a job by title or company name. 3.Apply filters (location, company, etc.). 4.Verify that results update based on applied filters. 5.Reset filters and ensure all jobs reappear.				
Expected	Search and filtering should return accurate results and not cause errors.				
Date-Result	10.04.2025 - Successful				

Test-ID	T-25	**Category**	Functional	**Severity**	High
Objectives	Ensure HR Experts can review and manage job applications				
Steps	1.HR logs in and navigates to the "Jobs" section. 2.Selects an application to view details. 3.Adds notes or comments to the application. 4.Changes the application status (e.g., "Under Review," "Accepted," "Rejected"). 5.Verify that the candidate is notified about the status change.				
Expected	HR should be able to review, update and manage applications and candidates should be notified about these changes.				
Date-Result	22.04.2025 - Successful				

Test-ID	T-26	**Category**	Functional	**Severity**	Medium
Objectives	Ensure candidates can edit and update their info and profile				
Steps	1.Candidate logs in and navigates to the "Profile" section.				

	2.Updates personal details (name, contact information, etc.). 3.Uploads a new resume or replaces the existing one. 4.Saves changes and logs out. 5.Logs back in and verifies that the updates are saved.
Expected	Candidates should be able to edit their profiles and changes should be saved and reflected manually.
Date-Result	24.04.2025 - Successful

Test-ID	T-27	**Category**	Non-Functional (Performance)	**Severity**	High
Objectives	Ensure system can handle high traffic without crashing				
Steps	1.Simulate multiple users (candidates, HR, company representatives) logging in simultaneously. 2.Perform job searches and apply for jobs concurrently. 3.Observe response times and system behavior.				
Expected	The platform should remain responsive and stable.				
Date-Result	25.04.2025 - Successful				

Test-ID	T-28	**Category**	Non-Functional (Security)	**Severity**	High
Objectives	Ensure passwords are securely stored and managed.				
Steps	1. Attempt to log in using an incorrect password multiple times. 2. Attempt to retrieve a password using the "Forgot Password" feature. 3. Verify that password reset links expire after a set time.				
Expected	Secure password management should be enforced and working.				
Date-Result	29.03.2025 - Successful				

Test-ID	T-29	**Category**	Non-Functional (Accessibility)	**Severity**	Low
Objectives	Ensure platform meets accessibility standards				
Steps	1.Enter and tab buttons should work on the keyboard.				

	2.Verify color contrast and font sizes for readability. 3.Check if images can be easily seen and readable.
Expected	The platform should be accessible.
Date-Result	10.04.2025 - On multiple screens, the readability is clear.

Test-ID	T-30	**Category**	Functional	**Severity**	High
Objectives	HR experts should be able to download the resume of the candidates				
Steps	1-Required UI elements are created 2-After clicking the button, resume should be downloaded				
Expected	Resume is downloaded in HR computer				
Date-Result	23.02.2025 - Successful				

Test-ID	T-31	**Category**	Functional	**Severity**	High
Objectives	Companies must be able to create job listings.				
Steps	1-) Tested via sending curl through terminal 2-) Tested the linkage to the frontend via interacting with relevant UI				
Expected	Job listings are created by companies				
Date-Result	02.02.2025 - Successful				

Test-ID	T-32	**Category**	Functional	**Severity**	High
Objectives	Companies create interviews for job listings and assign/manage interview participants				
Steps	1-) Tested via sending curl requests 2-) Tested by interacting with with relevant UI				
Expected	Companies can create meetings for having interviewers assess candidates. Each meeting will be specific for a candidate. An interviewer can have multiple experts as participants, but only a specific interviewer will be responsible for scheduling.				
Date-Result	05.02.2025 Successful				

Test-ID	T-33	**Category**	Functional	**Severity**	High
Objectives	Hosting meetings through Jitsi, by having our own deployment instance.				

Steps	Joined from different machines on our AWS EC2 instance's URL, and successfully participated in meetings we created.
Expected	Meetings can be created and participated by users through Recruit4Me's Jitsi instance.
Date-Result	18.02.2025 - Successful

Test-ID	T-34	**Category**	Non Functional (Logs)	**Severity**	**Medium**
Objectives	Accessing logs of interviews, for tracking who joined meetings when necessary.				
Steps	1-) Configuring cfg.lua file to print logs 2-) Manual inspection through the Jitsi instance				
Expected	Log files can be accessed manually when an issue arises by end users.				
Date-Result	26.02.2025				

Test-ID	T-35	**Category**	Non Functional	**Severity**	High
Objectives	Configure Jitsi to only enable Recruit4Me interviewers to create meeting rooms, and automate fetching host credentials to Jitsi from backend.				
Steps	1-) Wrote a mock lua script imitating data sent from backend to Jitsi. 2-) Execute the script in Jitsi, to see if hosts are registered. 3-) Created rooms by using newly registered host credentials				
Expected	The script must register new interviewers' credentials periodically				
Date-Result	27.02.2025 - successful				

Test-ID	T-36	**Category**	Non Functional	**Severity**	Medium
Objectives	User passwords must be hashed to reduce leakage in case of database interception.				
Steps	1-) Implemented hashing for registration and login comparison 2-) Tested via interacting application's UI and inspecting database entries				
Expected	The passwords must stored as hashed				
Date-Result	12.02.2025 (Successful)				

Test-ID	T-37	**Category**	Functional	**Severity**	**Medium**
Objectives	Candidates applied to a job listing must be fetched through database, and user must be able to filter out rejected or pending applications				
Steps	Tested via sending curl				
Expected	Candidate data must be returned correctly from the backend				
Date-Result	06.02.2025 (Successful)				

Test-ID	T-38	**Category**	**Functional**	**Severity**	**Medium**
Objectives	Companies must be able to send hire invites to interviewer experts, by specifying an amount of payment per interview.				
Steps	Tested via sending curl				
Expected	Companies' requests must be successfully registered to the database				
Date-Result	24.02.2025 - Successful				

Test-ID	T-39	**Category**	**Functional**	**Severity**	**Medium**
Objectives	Interviewers must be able to accept or reject hire invites sent by companies. Upon rejection, they should be able to specify a text response. And with respect to that rejection, the company could reoffer a new hire invite.				
Steps	Tested via sending curl.				
Expected	Interviewers can evaluate hire invites, and companies can resend invites accordingly.				
Date-Result	25.02.2025 - Successful				

Test-ID	T-40	**Category**	**Functional**	**Severity**	**High**
Objectives	Ensure that HR personnel and company representatives can only access candidate data relevant to their job listings and cannot view or edit applications outside their authorization scope.				
Steps	HR logs in and navigates to the applicant list for a job posted by their company. Attempts to view, edit, or shortlist candidates from a job posting created by another company. Attempts to directly access a candidate's data by modifying API requests				

	(e.g., changing job_id in the request URL). Company representatives try to access candidate profiles that belong to other companies' job postings. Monitor the system logs for unauthorized access attempts.
Expected	HR personnel should only access candidates applying to their company's job listings and not those from other companies. Unauthorized API requests should return an "Access Denied" (403 Forbidden) error. Logs should capture unauthorized access attempts for auditing purposes. No candidate data should be leaked to unauthorized users.
Date-Result	9.3.2025 - Successful

Test-ID	T-41	**Category**	**Security**	**Severity**	**High**
Objectives	Ensure that API rate limiting is enforced to prevent abuse.				
Steps	Attempt to send a large number of requests (e.g., 1000 API requests in a short period). Monitor system response to detect throttling mechanisms. Ensure that excessive requests return an appropriate HTTP 429 error.				
Expected	The system should throttle excessive API calls and prevent abuse.				
Date-Result	1.04.2025 - System can handle large number of request				

Test-ID	T-42	**Category**	**Functional**	**Severity**	**Medium**
Objectives	Ensure job postings expire after a set duration and are no longer available for applications.				
Steps	Post a job with an expiration date. Check if the job remains visible before the expiration date. Verify that the job automatically disappears after the expiration date.				
Expected	Expired jobs should be hidden from candidates and prevent new applications.				
Date-Result	10.02.2025 - Successful				

Test-ID	T-43	**Category**	**Functional**	**Severity**	**High**
Objectives	Ensure that rejected candidates cannot reapply for the same job.				
Steps	A candidate applies for a job.				

	HR rejects the application. Candidate attempts to reapply after rejection. System prevents the reapplication and provides an appropriate message.
Expected	Candidates should not be able to reapply after rejection
Date-Result	16.04.2025 - Successful

Test-ID	T-44	**Category**	**Non-Functional (Performance)**	**Severity**	**High**
Objectives	Ensure the system can handle concurrent user activity without performance degradation.				
Steps	Simulate 500 users logging in, applying for jobs, and submitting CVs at the same time. Monitor system response times and CPU/memory usage. Check for slowdowns or crashes under load.				
Expected	The system should remain stable and responsive under heavy load.				
Date-Result	1.04.2025 - With help of Azure VM the system is stable under high load.				

Test-ID	T-45	**Category**	**Functional**	**Severity**	**High**
Objectives	Ensure that candidates receive a confirmation email after successfully submitting an application.				
Steps	Candidate submits an application for a job. System triggers a confirmation email. Candidate checks the email inbox for the confirmation message. Attempt to apply again and check if a duplicate confirmation is sent.				
Expected	Candidates should receive a confirmation email after successful application submission.				
Date-Result	27.02.2025 - Successful				

Test-ID	T-46	**Category**	**Functional**	**Severity**	**Medium**
Objectives	Ensure that HR experts can send automated interview reminders to candidates.				
Steps	HR schedules an interview for a candidate. System generates and sends a reminder email 24 hours before the interview. Candidate receives the reminder and confirms attendance.				

Expected	Interview reminders should be sent automatically before scheduled interviews.
Date-Result	11.02.2025 - Successful

Test-ID	T-47	**Category**	**AI Model**	**Severity**	**High**
Objectives	Ensure that the AI model does not introduce bias based on gender, ethnicity, or other non-relevant attributes in candidate evaluation.				
Steps	Submit identical CVs with different names (e.g., male vs. female names) and compare scores. Submit CVs from candidates of different nationalities or ethnic backgrounds with the same experience and evaluate score consistency. Analyze the AI-generated interview questions for bias (e.g., different difficulty levels based on the candidate's background). Conduct statistical analysis on model outputs across a diverse dataset.				
Expected	Candidate scores should only be affected by experience, education, and job relevance—not gender, nationality, or other biases. AI-generated interview questions should remain fair and job-specific, avoiding discriminatory patterns. System logs bias-related concerns and provides explainability metrics where possible.				
Date-Result	1.03.2025 - Initial tests showed no bias				

Test-ID	T-48	**Category**	**Data Consistency & Integrity**	**Severity**	**High**
Objectives	Ensure that candidate test scores and rankings remain accurate and are not overwritten or lost due to system errors.				
Steps	Submit multiple candidates for a job role, ensuring all receive initial scores. Introduce a system failure or restart the server during an ongoing evaluation. Check if previously processed scores remain intact after the system recovers. Ensure no candidate scores are reset to default or lost. Compare the final ranking list before and after system failures to ensure consistency.				
Expected	Candidate scores should be accurately stored and persist even after system crashes. Rankings should not change unexpectedly unless new candidates are				

	added. System logs should track all updates to candidate scores.
Date-Result	1.05.2025 - With regular backups the system remains consistent.

Test-ID	T-49	**Category**	**Fraud Prevention & Duplicate Application Handling**	**Severity**	**High**
Objectives	Ensure that candidates cannot submit multiple applications for the same job using different accounts to manipulate rankings.				
Steps	A candidate creates an account and submits an application for a job. The same candidate creates a new account using a different email and applies for the same job again. Check if the system detects duplicate applications based on resume content, IP address, or other unique identifiers. Attempt to manipulate the system by modifying slight details in the second application (e.g., changing a single word in the resume). Monitor system logs and candidate ranking changes to see if duplicate detection is effective.				
Expected	The system should flag duplicate applications and notify HR personnel. A candidate should not be able to submit multiple applications under different accounts for the same job. HR should have the ability to merge or reject duplicate applications. Logs should record attempts at duplicate submissions.				
Date-Result	22.04.2025 - Successful				

Test-ID	T-50	**Category**	**Data Loss Prevention & Auto-Save Functionality**	**Severity**	**Medium**
Objectives	Ensure that partially completed job applications, test answers, and interview scheduling details are automatically saved and recoverable if the session is interrupted.				
Steps	A candidate starts filling out a job application but does not submit it. The candidate closes the browser or refreshes the page unexpectedly. Reopen the application page and check if the previous progress is still there. A candidate is taking an online coding test, and their internet connection drops for a few minutes. Once reconnected, check if the test progress and answers are intact. HR personnel start scheduling an interview, but the session times out				

	before submission. After logging in again, check if the previous scheduling details are still available.
Expected	Job applications, test responses, and interview scheduling data should be auto-saved periodically. After a session timeout or browser closure, users should be able to resume where they left off without losing progress. The system should have a button for easy recovery.
Date-Result	23.04.2025 - Successful

6. Maintenance Plan and Details

Effective software maintenance is critical to keep Recruit4Me reliable, secure, and up-to-date. According to IEEE Std 1219-1998, software maintenance is defined as “modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment”. Our plan follows this standard’s process model (problem identification, analysis, design, implementation, testing, delivery) and classifies maintenance work into the three canonical types:

- **Corrective maintenance:** fixing defects and bugs to restore intended functionality. Whenever errors are reported (via user feedback or monitoring alerts), these are logged in Azure DevOps and prioritized for immediate fix.
- **Adaptive maintenance:** modifying the system to remain compatible with changes in its environment. For example, updating APIs, libraries, or cloud platform versions (Azure/AWS service updates) falls under this category. We regularly review platform changes and adapt Recruit4Me accordingly.
- **Perfective maintenance:** enhancing performance, usability, or features based on user feedback. This includes refactoring code for efficiency, improving UI responsiveness, or adding small requested features.

Our development team handles all maintenance requests through a formal change-control process: issues are logged in Azure Boards, classified by type, estimated, and either scheduled for the next release or handled immediately if critical. Scheduled updates (such as quarterly security patches or feature releases) are announced in advance, while urgent security fixes are deployed as emergency patches.

Key elements of our maintenance framework include:

- **CI/CD Automation:** We use Azure DevOps Pipelines to automate build, test, and deployment processes. Each commit to our Git repository triggers a build pipeline that

runs static analysis and unit/integration tests. Successful builds automatically deploy to a staging environment, where end-to-end tests are run, and then to production with only minimal manual approval. Infrastructure-as-code (using ARM templates or Terraform) ensures that environments are versioned and reproducible.

- **Monitoring & Scaling:** We employ cloud monitoring tools to track system health. For Azure services, we use Azure Monitor and Application Insights to collect metrics (CPU, memory, response times) and logs in real time; for any AWS-hosted components (Jitsi), we use AWS CloudWatch. Dashboards display key indicators and automated alerts notify the team if thresholds (e.g. high error rate or latency) are exceeded. The application is hosted on scalable infrastructure (e.g. Azure Kubernetes Service or AWS Elastic Kubernetes Service), with autoscaling rules that add or remove instances based on load. This elastic setup allows Recruit4Me to handle traffic spikes (such as surges in job applications) without manual intervention.
- **Error Tracking:** We integrate an error-monitoring service (such as Sentry or App Insights) into our CI/CD pipeline. Runtime exceptions and crashes are automatically captured, and an issue is created in our tracking system linked to the corresponding code commit. This traceability dramatically speeds up corrective maintenance, as developers can see exactly which change introduced a fault.
- **Backup & Disaster Recovery:** We maintain a robust backup and recovery policy for all critical data (databases, user profiles, job postings, model artifacts). Azure components use Azure Backup with a Recovery Services vault, which takes scheduled snapshots to geo-redundant storage. AWS components use AWS Backup or native snapshot capabilities (EBS/RDS) with cross-AZ replication, providing “11 9s” durability for stored data. We define Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) for each service (e.g. RTO of 1 hour, RPO of 24 hours for mission-critical data) and enforce backup retention policies (daily, weekly, monthly snapshots) to meet compliance needs. Importantly, we regularly test recovery procedures: partial restores and full disaster-recovery drills are conducted (at least semi-annually) to verify that services can be restored within targets. A documented recovery playbook outlines roles and step-by-step procedures to follow in the event of a catastrophic failure.

By combining IEEE 1219–compliant processes with automated cloud tooling and disciplined procedures, our maintenance plan ensures that Recruit4Me can evolve gracefully. Continuous monitoring and CI/CD deployment let us detect issues early and deliver fixes promptly, while scaling and backup mechanisms protect against downtime. In summary, this comprehensive approach to corrective, adaptive, and perfective maintenance provides a robust foundation for long-term software quality and availability.

7. Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

7.1.1 Constraints

7.1.1.1 Implementation Constraints

- System was designed using React for front-end, Express.js for backend development.
- Predefined APIs were needed to utilize speech-to-text processes such as Google Speech-to-text or AWS Transcribe.
- Parsing resumes require skill extraction thus integration of external tools that we use like Azure Document Intelligence.
- A secure database MySQLto store candidate records and test results.
- Platform should be scalable to handle large data of candidates and companies.
- Protocols making the application secure are used such as OAuth2 or SSO.
- System is hosted by Azure to increase availability and reliability.
- In a certain(now undefined) period video data must be deleted from the database to ensure high availability in the backend.

7.1.1.2 Economic Constraints

For the moment, the Azure Service Credits are utilized which were given by Microsoft, however, for long-run other potential services should be observed.

- In resume parsing, with support of NLP and predefined models from the team, potential API costs are minimized.
- Open-source solutions are implemented first to see effectiveness before using large scale APIs like Google Speech-to-text, AWS Transcribe, Azure's API.
- Google cloud platform is an option, which utilizes Compute Engine instances and BigQuery for data analysis.
- Amazon Web Services(AWS) offers free EC2 instances, Lambda functions and RDS for a limited duration, which are analyzed to ensure if it is proper for early-stage development.
- Google Speech-to-text requires approximately 50 TL per hour for real-time transcription. AWS Transcribe has a similar pricing, lower volumes can be managed however large scale candidate pools require budget evaluation.
- Efforts should be made to optimize these resources by scheduling GPU usage only during model training or evaluation processes.
- A modestly scaled deployment (e.g., 2 vCPUs, 8GB RAM, and 50GB storage) on AWS or GCP costs around 650 to 2000 TL per month. Adding auto-scaling capabilities for peak periods could slightly increase this price.

- Managed database services like AWS RDS or GCP Cloud SQL cost approximately 1000 to 3500 TL per month depending on storage and performance needs. These are options to use when our free credits from Azure are finished which also have a Cloud Service for deployment for initial release.
- Regular updates, monitoring, and bug fixes require developer hours. Allocating team resources efficiently is crucial to avoid unnecessary expenditures. We managed to handle testing with automated tools as it doesn't have a big load on us.

7.1.1.3 Ethical Constraint

- System must ensure that all evaluations and eliminations are objective and free from bias based on factors such as gender, ethnicity, nationality or other non-technical factors. Which means that grading should be based on skill sets, experiences and test performances.
- Communication skills ratings must be standardized by using well-defined criteria, such as clarity, grammar, and coherence, or some factors explicitly indicated by the company to avoid subjective judgments and potential biases.
- AI models should be regularly checked for unintentional bias in the predefined algorithms and data handling to ensure fair treatment of all candidates.
- All candidate data, including resumes, test results, and recordings, must comply with GDPR or other relevant data protection regulations, depending on the region.
- Recordings should be stored for no longer than a certain period, as specified, and deleted securely afterward.
- Notifications sent to candidates about results, feedback, or interviews must respect their privacy preferences and include clear opt-out options for future communication.
- Encryption should be used to protect sensitive data during transmission and storage, ensuring the highest level of security.
- For a possible built-in messaging in application encryption should be made to protect privacy.
- For candidates eliminated by the system, the reasons must be explained in a transparent and professional manner, including data points such as skill gaps, test scores, or evaluation criteria.
- All third-party tools integrated into the system, such as resume parsers, speech-to-text APIs, or scheduling platforms, must comply with strict ethical standards and avoid misuse of candidate data.

7.1.2. Standards

We use the Agile technique in the development of Recruit4Me HR Automation, dividing the project into two-week sprints that encourage flexibility in responding to changing requirements and iterative progress. Every sprint is devoted to the autonomous development of particular modules, which are then combined into the system as a whole at the end of each cycle. This strategy guarantees ongoing delivery of useful components and permits prompt

modifications in response to stakeholder input. Recruit4Me HR Automation is carefully crafted to fully adhere to the ISO/IEC 25010 software quality model, which covers essential qualities including functionality, dependability, usability, efficiency, maintainability, and portability, in order to provide the highest standards of software quality [1]. At the same time, the web application component closely complies with Web Content Accessibility Guidelines (WCAG) 2.1, guaranteeing that the platform satisfies international accessibility standards and is usable by all users, including those with disabilities [2]. We have chosen UML 2.5.1 as our main modeling standard for system modeling, which makes it easier to create thorough and consistent diagrams that clearly convey the architecture, parts, and interactions of the system. Furthermore, the requirements documentation is painstakingly created in compliance with IEEE 830 criteria, guaranteeing that all functional and non-functional needs are precisely stated, organized, and simple enough for all parties involved to understand [3]. This combination of Agile practices and adherence to industry-recognized engineering standards ensures that Recruit4Me HR Automation is developed in a structured, high-quality manner, capable of meeting user needs, maintaining flexibility throughout its lifecycle, and delivering a robust, accessible, and maintainable HR automation solution.

7.2 Ethics and Professional Responsibilities

We integrated ethical considerations throughout Recruit4Me. In our machine learning usage, we proactively mitigated algorithmic bias: we deliberately excluded sensitive attributes (race, gender, etc.) from our training data and designed the model to focus on objective qualifications. We plan to regularly audit model outputs for fairness (e.g. checking that recommendations do not disproportionately disadvantage any group). Privacy and user consent were also priorities: all personal data is encrypted and accessed only by authenticated users, and we conform to privacy laws and best practices in storing and processing user data. We emphasize transparency about our algorithms: documentation and user help clearly explain how candidate-job matching works and how interview recommendations are generated. These practices follow widely accepted AI ethics principles – for example, Google's AI guidelines advocate avoiding unfair bias and promoting user privacy. By aligning with these professional ethics (fairness, accountability, transparency), we aim to maintain user trust and social responsibility in our platform.

7.3 Teamwork Details

7.3.1 Contributing and Functioning Effectively on the Team

In order to keep track of the tasks and the situation of those tasks, we utilize Notion. With the help of the Notion, everyone sees the contribution of both himself and other contributors. We met on a weekly basis to find answers on the problems that we are facing. Another benefit of these weekly meetings is identifying the next step in the project. Cross functional team members organize the integrity of the ML and application sides of the project. Apart from those

weekly meetings, we have also heavily used other communication tools to enhance team collaboration such as WhatApp and Zoom.

7.3.2 Helping Create a Collaborative and Inclusive Environment

Proper teamwork is essential for collaboration. Therefore, noting the tasks are necessary. Because of this, we have decided to use planning applications. Notion is the sprint planning application where noting the tasks and sprint planning is possible. This application also has filters showing tasks based on the due time or person. With those filters, we can determine the workload of each person in the project. Another tool that we used is discord. With the help of the discord, we manage to make meetings even if we are far from each other. Another benefit of the discord is grouping the different tasks in different chat rooms. This feature of discord eases document management. We also have a WhatsApp group to enhance the daily communication among team members. This group has a huge positive effect on the relationship among team members. With the help of WhatsApp messaging, we are able to manage the positive energy among the team members. Lastly, workload is increasing and deliverable deadlines are coming up. We meet face to face to enhance the communication and keep motivation high among team members. With the help of face to face interactions, we are able to handle the tasks efficiently and faster.

7.3.3 Taking Lead Role and Sharing Leadership on the Team

Since we keep in touch very frequently, there is no need for the leader or any kind of leadership. However, based on the experiences for the particular work, some of us take the leadership for specific tasks for a limited amount of time. Thanks to this approach, everyone in the team has the opportunity to enhance managing skills. Furthermore, since everybody knows the importance of the tasks that they did during the whole project, everybody does those tasks on time.

7.3.4 Meeting Objectives (Mapping Against Original Project Plan)

We continuously tracked our progress against the original plan. We set milestones at the start of the project (e.g., completing the ML match model by week 6, video chat integrated by week 10) and had a Kanban board where we ticked off items as they were completed. At the end, all of the major objectives in our plan were met on schedule: job posting/search, candidate matching, and video interviewing were all built and tested. When minor deviations occurred (e.g., it took a little longer than expected to tweak the ML model), we adjusted subsequent sprint plans and team assignments accordingly to compensate. In each case, tasks were rearranged to ensure deliverables remained on track. This rigorous tracking and flexibility prove that our team met project goals effectively and showed very good project management.

7.4 New Knowledge Acquired and Applied

The project was a major learning experience. Team members gained new technical skills and process knowledge, including:

- **Machine Learning:** We learned about applied NLP and ML concepts in matching resumes and jobs. Using libraries like scikit-learn and TensorFlow, we played around with preprocessing text (e.g. tokenization, vectorization) and with training a classification model. We learned these concepts by following online tutorials and documentation and used them to build our first recommendation engine.
- **React and Front-End Development:** Some members of the team learned React expertise. We learned how to build dynamic user interfaces with components, hooks, and state management, and added UI libraries (such as Material-UI) for a polished look. Along the way, we also added accessibility features following WCAG guidelines. Documentation and code samples (React official documentation and community resources) were key study aids, and pair programming helped transfer knowledge.
- **Video Integration (Jitsi):** We learned how to incorporate Jitsi Meet for live video interviews. By learning the official Jitsi API documentation and experimentation, we learned how to incorporate one-to-one video calls into our web application, handling authentication and call setup. This provided us with a better understanding of WebRTC and real-time communication in web development.
- **CI/CD and DevOps:** We learned how to apply continuous integration/deployment. We developed Azure DevOps pipeline definitions (YAML) and learned how to automate releases, testing, and builds. Through Microsoft's Learn tutorials and trial-and-error, we tuned our deployment process so that releases could be done in a timely fashion. We learned how to practice infrastructure-as-code, which solidified our grasp of reproducible deployments.
- **Process and Collaboration:** We improved our software process skills. We adopted Agile/Scrum practices (writing user stories, sprint planning, retrospectives) that made the development more iterative and responsive. We created an SRS and test cases using IEEE best practices, which taught us how to document requirements and testing in a systematic manner. Through version control and code reviews, we learned effective team collaboration workflows.

In the process of all this, we utilized different ways of learning: online tutorials and official documentation (Azure docs, React guides, Jitsi reference, IEEE standards, etc.) were foremost sources, supported by code samples and developer communities. Pair programming sessions and reviewing code as a team allowed real-time learning from each other. We also came up with a team wiki and design logs in order to make decisions and back learning further. Overall, this project offered the team beneficial real-world exposure to modern web and ML development, CI/CD, and collaborative engineering practices.

8. Conclusion and Future Work

In total, the Recruit4Me project managed to meet its objectives by applying the intended functionality with professional proficiency. We have built an end-to-end recruitment portal where employers can post and manage job vacancies, candidates can search and apply for vacancies, and our ML-based engine conducts automatic candidate matching for vacancies. We have added live video interviewing functionalities (with Jitsi) and adhered to our non-functional requirements (security, usability, etc.). All of the core features identified in the project plan were

created and tested within schedule, demonstrating that project deliverables were meeting our original objectives.

In the future, we have identified several improvements to maximize the value of the platform:

- **Recruiter-Side Analytics:** We plan to include employer dashboards and reporting features. These would include analytics such as application volume by job, candidate demographics, and time-to-fill metrics. By having visualized insights, recruiters would be in a position to make data-driven decisions and have a clearer picture of their hiring pipelines.
- **Advanced ML/NLP Models:** We would like to improve the matching engine leveraging more advanced natural language processing. For example, using most recent Transformer-based models (like BERT) to represent resumes and job descriptions would capture more semantic meaning. We would obtain more training data and fine-tune our algorithms so that candidate suggestions are more relevant and accurate.
- **Global Expansion:** In order to cater to an international market, Recruit4Me can be localized for various regions. This includes supporting additional languages (interface translation and job description translation), compatibility with national job taxonomies, and compliance with regional legislation (privacy policy, labor law, etc.). Localization of the platform will enable us to grow Recruit4Me's user base and meet the needs of global recruiters and applicants.

These advancements are grounded on our current establishment. Incorporating employer analytics, enhancing our ML models, and expanding our markets to international territories, Recruit4Me will be a more effective and efficient tool. Successfully completing the project and the knowledge gained provide a strong foundation for these advancements, and Recruit4Me will be able to evolve with future recruitment issues.

9. Glossary

- **HR (Human Resources):** HR department in an organization is responsible for managing employee-related functionalities, such as recruitment, training, rating performance and workplace policies. In the Recruit4Me application, HR is a major part of the system, handling large volumes of candidate applications and assisting automation. This system facilitates CV parsing, rating candidates, test generation and execution, scheduling interviews which minimize manual interventions. HR can focus on efficiently hiring a candidate with assistance of this automation. AI and ML further enhance HR operations by providing unbiased data and evaluation of applicants.

- **AI (Artificial Intelligence):** Artificial intelligence aids in developing the systems which are capable of doing tasks which mostly require human intelligence, such as decision-making, natural language processing(NLP), and recognition of patterns. In Recruit4Me, AI is crucial in automatization of the candidate evaluation process. Speech-to-Text for online interview response evaluation, generating questions based on

level and domain of the candidate with the assistance of multiple APIs such as OpenAI. These ensure HR workflows stay consistent and unbiased.

- **ML (Machine Learning):** ML is an AI component which focuses on building algorithms, models to learn from data and make predictions beforehand without explicit programming. In Recruit4Me, ML is utilized in the CV evaluation pipeline, where the Model Evaluation Engine is used. These compose of tasks such as analyzing candidate CVs for requested qualifications by the company, rating candidates based on a predefined criteria, and give them ranks to develop assessment for further states of the laboring. ML algorithms provide feedback for improvement in addition, and continuously refine the evaluation methods as more data is gained, increasing accuracy and reliability of the system.

10. References

[1] “ISO 25010.” Online. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Accessed 1-May-2025].

[2] “Web Content Accessibility Guidelines (WCAG) 2.1,” Sep. 2023. Online. Available: https://www.w3.org/TR/WCAG21/. [Accessed 1-May-2025].

[3] IEEE Computer Society and Software Engineering Standards Committee, IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.