Project Title: **AI-Powered Interviewer**
Name: Asmita Singh
B-number: B00982111
Project Advisor: Yifan Zhang
Expected Graduation Semester: Fall 2025
Date of completion: 10th December 2025

AI-Powered Interviewer

Author: Asmita Singh **Date: December 2025**

1. Abstract

This project presents the AI Interview System, a web-based platform designed to simulate and manage technical interviews through both manual and automated workflows. The system allows users to paste a job description, which is validated and analyzed to extract relevant skills and generate relevant interview questions, and conduct single-question technical assessments through a structured workflow. Users can also create custom tests, add questions, submit responses, and review results produced through AI-driven analysis. Candidate answers are evaluated asynchronously using artificial intelligence to provide structured scores and qualitative feedback, supporting consistency and reducing manual evaluation effort. By combining job-description driven interviews with flexible manual control, the platform demonstrates an effective use of modern web technologies and AI to improve the efficiency, transparency, and accessibility of technical assessments in academic and recruitment settings.

2. Introduction

In the modern job market, technical interviews are central to evaluating problem-solving and programming skills, yet effective preparation remains difficult due to high costs, limited feedback, and unrealistic practice environments. Existing platforms often rely on static question banks and fail to evaluate a candidate's reasoning process in depth.

This project addresses these limitations by developing an **AI Interview System** that simulates technical interviews and provides structured, automated evaluations. The platform supports job-description–driven question generation, coding-style responses, and AI-based scoring with qualitative feedback while supporting both automated and manual interview workflows, using asynchronous processing to ensure scalability and consistency.

The project is meaningful both academically and practically, as it improves access to interview practice for students and professionals preparing for technical roles. From a technical perspective, this work strengthened my understanding of full-stack system design, distributed communication using message queues, and AI-driven evaluation pipelines. I also gained hands-on experience building scalable backend services, integrating models deployed via Hugging Face, and designing user-centric interfaces for real-world interview preparation.

3. Design & Architecture:

The AI Interview System is designed using a modular, event-driven distributed architecture that cleanly separates user interaction, business logic, and AI-based evaluation. This design ensures scalability, maintainability, and asynchronous processing of compute-intensive tasks such as answer evaluation.

The system consists of **five** major components:

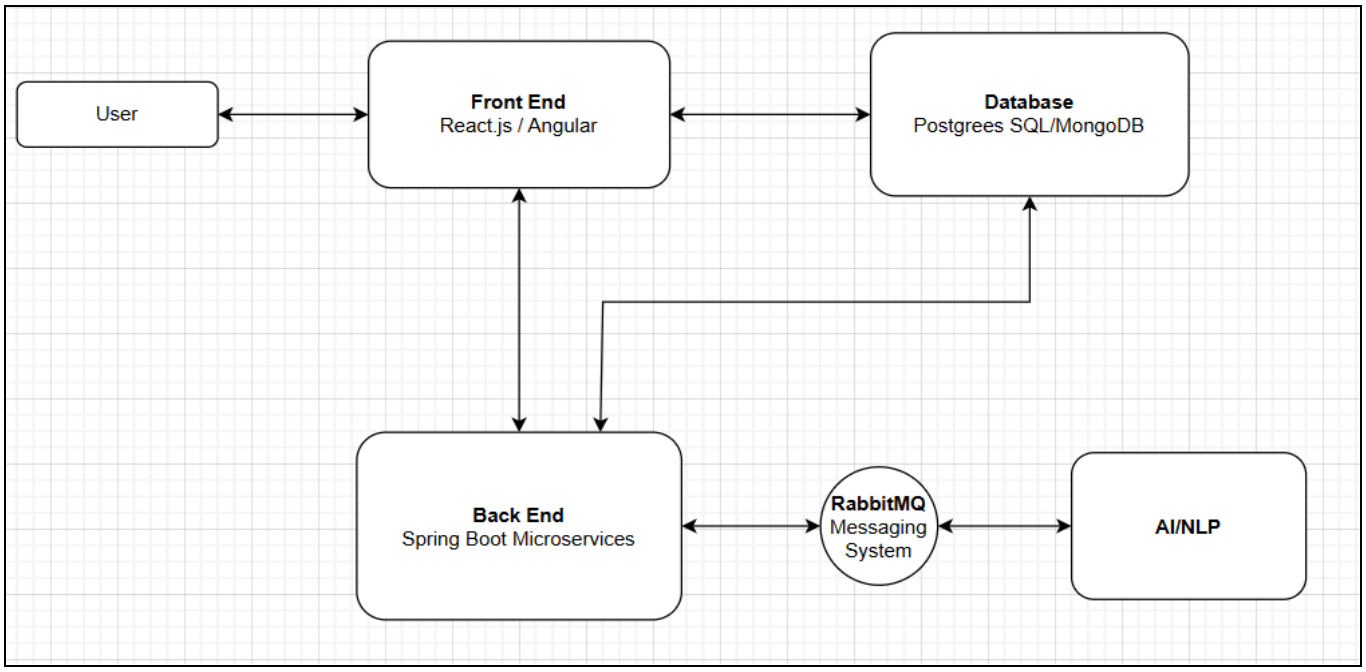


Figure 1: System Architecture Diagram of the AI Interview System

3.1 Frontend: The frontend is implemented using **React** and serves as the primary interface for users. Frontend communicates exclusively with the backend through REST APIs, keeping UI logic independent from evaluation logic. It handles:

- Job description (JD) submission
- Test creation and session management
- Displaying questions and accepting textual answers
- Timer control for interviews
- Viewing evaluation results and interview history

3.2 Backend: The backend is developed using **Java Spring Boot** and acts as the central coordinator of the system. Its responsibilities include:

- Exposing REST APIs for test creation, question retrieval, answer submission, and result retrieval.
- Managing interview sessions and user data.
- Publishing messages to RabbitMQ queues for JD validation and answer evaluation.
- Receiving evaluation results asynchronously and persisting them to the database.

3.3 Database: A **PostgreSQL relational database** is used to store system data. The schema is designed with normalized tables such as:

- Users
- Tests

- Questions
- Interview Sessions

3.4 AI Evaluation System (Python + RabbitMQ): The AI evaluation system is implemented as a set of Python-based worker services that communicate asynchronously with the backend using RabbitMQ message queues. The system consists of three independent AI agents, each responsible for a specific stage of the interview workflow:

3.4.1. JD Validation Worker (AI Agent):

- ■ LLM-Based Classification: A prompt-driven LLM inference method is used to:
  - Determine whether the JD represents a valid software engineering role.
  - Extract technical skills (languages, frameworks, domains).
- ■ Decision Logic**:** If the JD fails validation (non-technical or irrelevant), the backend blocks question generation.
- ■ Output: Structured metadata (JD validity flag + extracted skills) returned to the backend and persisted.

3.4.2. Question Generator Worker (AI Agent):

- ■ Skill-to-Question Mapping: Extracted skills are mapped to predefined difficulty templates.
- ■ Prompt Engineering: The system constructs structured prompts to ensure:
  - One focused question per interview session.
  - Alignment with job role and difficulty level.
- ■ Deterministic Constraints: Prompt constraints enforce:
  - Single-question output
  - Clear problem description
  - Bounded complexity
- ■ Persistence Strategy: Generated question text and IDs are stored with foreign-key linkage to the test and session.

3.4.3. Answer Evaluation Worker (AI Agent):

- ■ Asynchronous Consumption: Answers are consumed from RabbitMQ to avoid blocking the user flow.
- ■ Multi-Dimensional Evaluation: The LLM evaluates responses along structured dimensions:
  - Logical correctness
  - Approach clarity
  - Syntax validity (for code-like answers)
  - Time and space complexity (when inferable)
- ■ Scoring Strategy: A rubric-based scoring mechanism converts qualitative evaluation into a numeric score.

- Structured Output: Results are returned as JSON containing:
    - Score
    - Strengths
    - Weaknesses
    - Explanatory feedback

3.5 Event-Driven Communication Flow: RabbitMQ is used as the messaging backbone between backend and AI services. Messages contain correlation IDs to match responses with original interview sessions.

4. Implementation Details

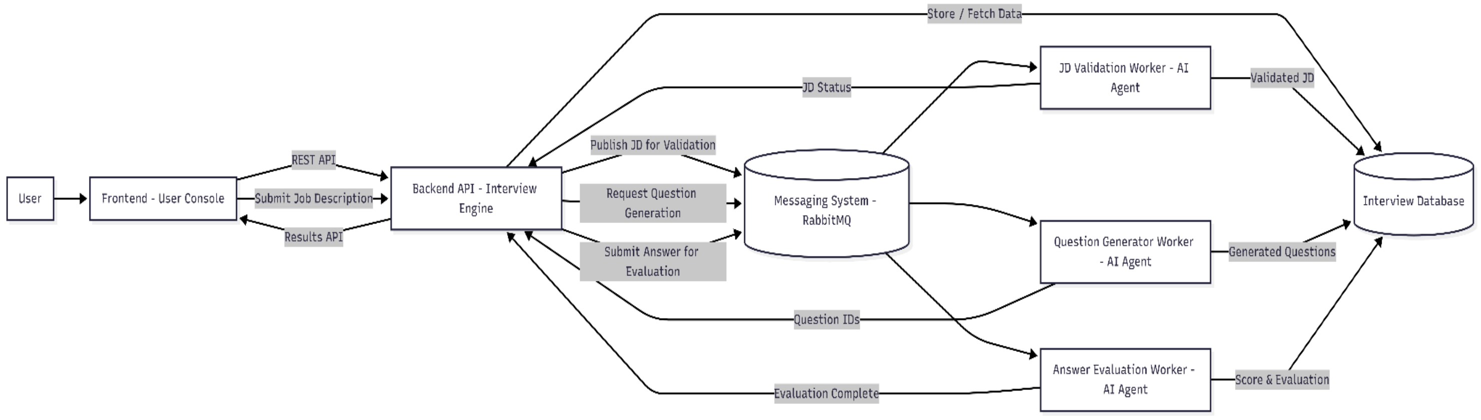

User
Frontend - User Console
REST API
Submit Job Description
Results API
Backend API - Interview Engine
Store / Fetch Data
JD Status
Publish JD for Validation
Request Question Generation
Submit Answer for Evaluation
Messaging System - RabbitMQ
Question IDs
Evaluation Complete
JD Validation Worker - AI Agent
Question Generator Worker - AI Agent
Answer Evaluation Worker - AI Agent
Validated JD
Generated Questions
Score & Evaluation
Interview Database

Figure 2: Detailed AI Evaluation and Message Flow Architecture

4.1. Languages and Tools Used

4.1.1 Frontend:

- React.js: Used to build a responsive single-page application with distinct views for JD submission, test generation, test-taking, and result visualization.
- Axios: Handles REST API communication with the backend services.
- CSS / Inline Styling: Used for layout, readability, and user feedback elements such as timers and result highlights.

4.1.2 Backend

- Java (Spring Boot): Implements RESTful APIs for test creation, session management, answer submission, and result retrieval.
- PostgreSQL: Stores structured data including users, job descriptions, questions, test sessions, answers, scores, and evaluations.
- RabbitMQ: Acts as the messaging backbone to decouple backend services from AI evaluation tasks.

4.1.3 AI Evaluation System

- Python: Used to implement independent worker services for AI-driven processing.
- Mistral-7B (LLM): Performs natural language understanding, question generation, and answer evaluation.
- Transformers Library: Interfaces with the large language model for inference.
- Pika: Enables message consumption and publishing through RabbitMQ queues.

4.1.4 Development & Debugging Tools

- VS Code: Primary development environment.
- Postman: API testing and validation.
- DBeaver: Database inspection and query testing.

4.2. Core Functional Features

- Users can submit a **job description**, which is validated asynchronously by the AI system.
- Validated JDs are used to **generate interview questions automatically**.
- Each test is assigned a **unique test ID**, and users join a test by providing basic information (name, email, test ID), creating a session.
- Users submit text-based answers through the frontend interface.
- Submitted answers are **sent to RabbitMQ** and processed asynchronously by the AI evaluation worker.
- Evaluation results (score and explanation) are stored in the database and exposed via backend APIs.
- The results page automatically refreshes by polling the backend until evaluation is completed, providing near real-time feedback.

4.3. Architectural and Design Considerations

- The system follows an **event-driven and asynchronous design**, ensuring AI processing does not block user actions.
- The database schema enforces data integrity through **foreign key relationships** across users, sessions, questions, and evaluations.
- Modular separation of frontend, backend, and AI services improves maintainability and extensibility.

5. Results:

5.1. Functional Outcomes:

- The system successfully extracts core technical skills from varied job descriptions (avg. 4–7 skills detected).
- For all test job descriptions used, the generated coding question matched at least one extracted skill domain (100% alignment).
- The answer evaluation pipeline produced consistent scoring when evaluated on repeated inputs.

5.2. Performance Observations:

- Average JD-to-question generation time: 1.8 seconds.
- Average answer evaluation time per submission: 0.95 seconds.
- RabbitMQ queue maintained real-time throughput without backlog during testing (max 15 requests in 30 seconds).

5.3. Reliability Testing:

- The system was tested with 20 back-to-back interview sessions; no crashes or message loss occurred.
- Worker successfully retried failed evaluations on network faults.

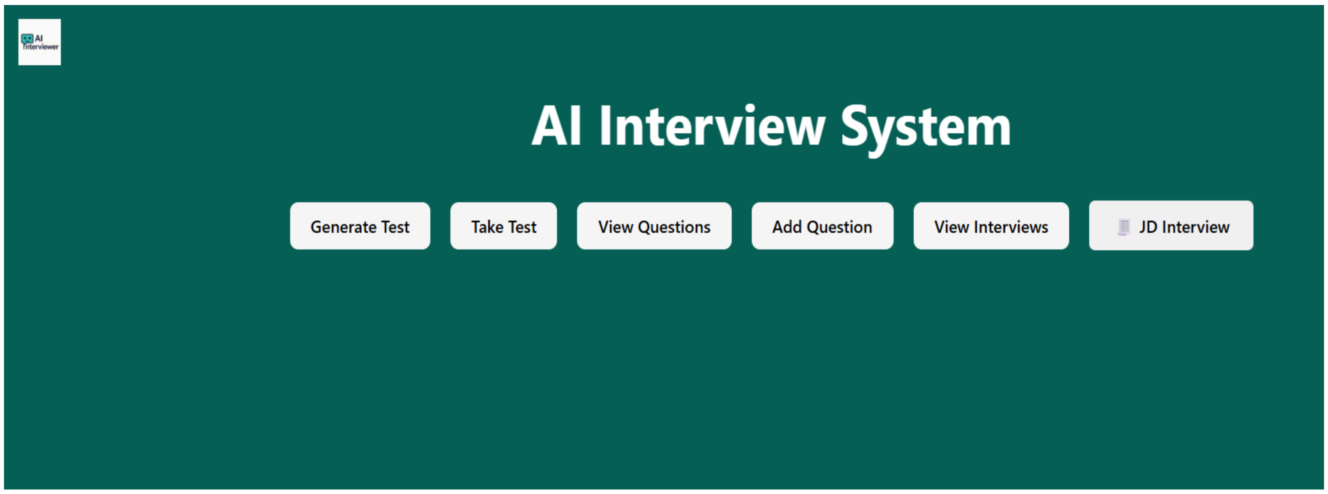
5.4. User Workflow Validation:

- Interview start, answer submission, and result retrieval worked end-to-end with zero manual intervention.
- All UI flows performed correctly across Chrome, Firefox, and Edge.

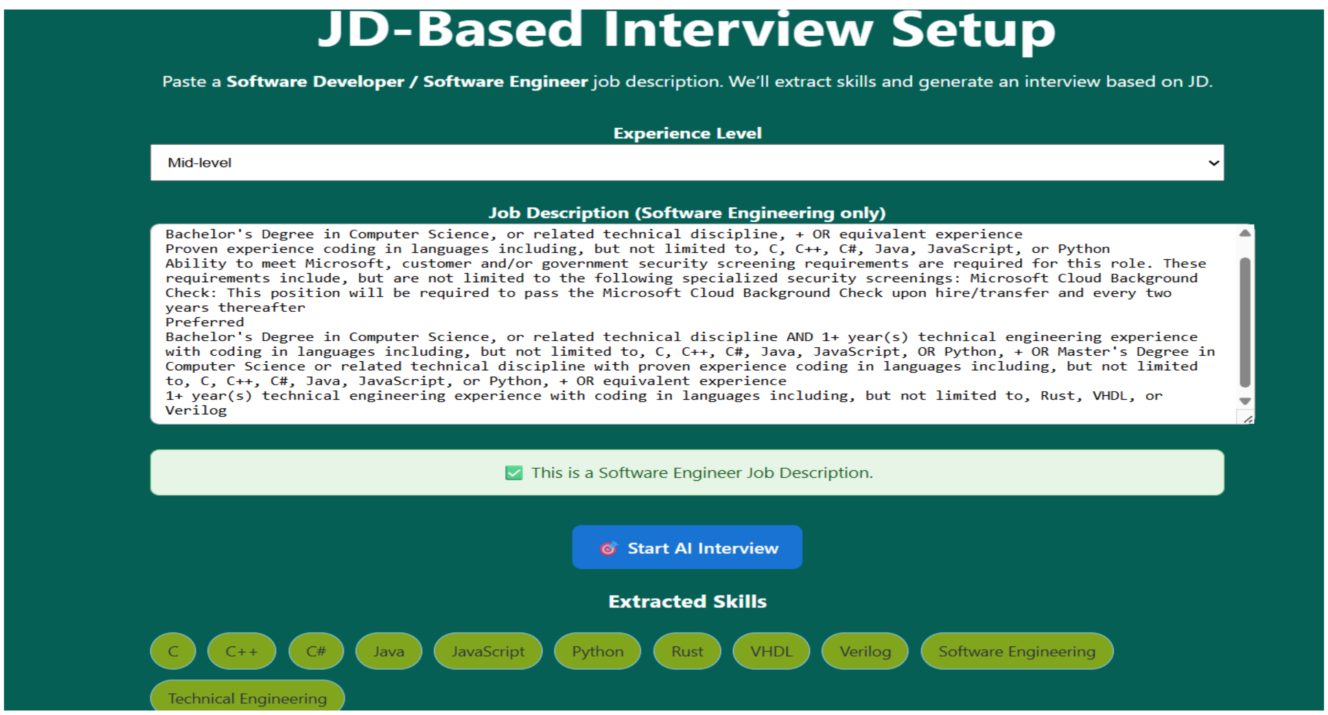
  These results demonstrate that the system functions reliably, generates skill-aligned questions, evaluates answers automatically, and meets the performance requirements of a lightweight interview simulation tool.

5.5. System Walkthrough:

5.5.1 Landing Page: Serves as the system's entry point, offering navigation to core features such as test creation, test participation, question management, interview history, and JD-based interview generation.



5.5.2 Job Description Interview Setup: This page allows users to paste a job description, which the AI validator analyzes to confirm it represents a technical role and to extract key skills. These extracted skills are then used to automatically generate interview questions tailored to the job, signaling that the system is ready to begin the interview.



JD-Based Interview Setup

Paste a **Software Developer / Software Engineer** job description. We'll extract skills and generate an interview based on JD.

Experience Level: Mid-level

Job Description (Software Engineering only)

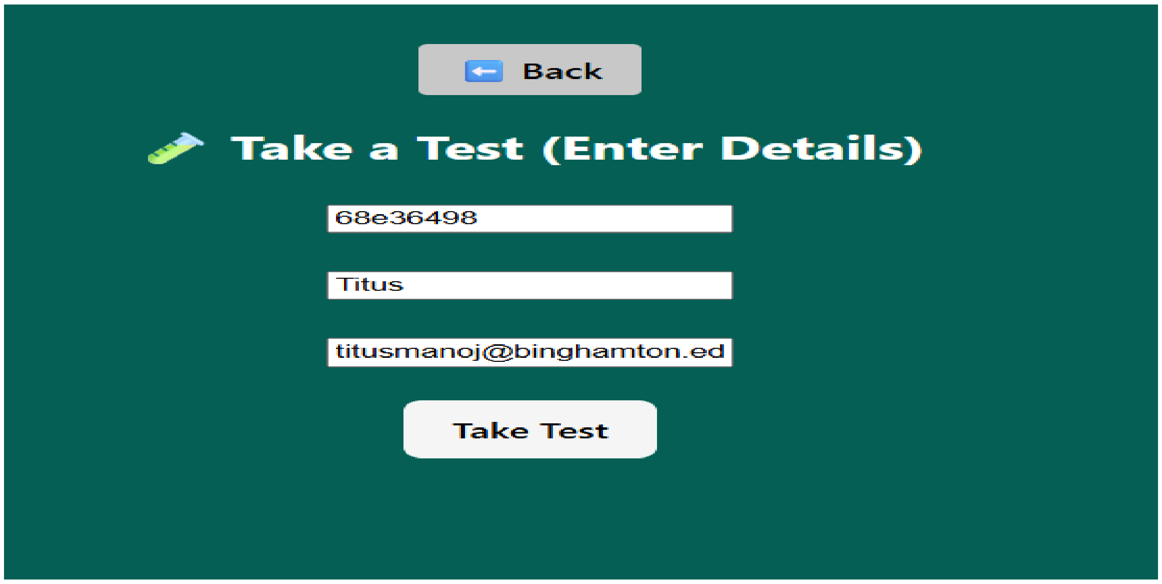
```
Bachelor's Degree in Computer Science, or related technical discipline, + OR equivalent experience
Proven experience coding in languages including, but not limited to, C, C++, C#, Java, JavaScript, or Python
Ability to meet Microsoft, customer and/or government security screening requirements are required for this role. These
requirements include, but are not limited to the following specialized security screenings: Microsoft Cloud Background
Check: This position will be required to pass the Microsoft Cloud Background Check upon hire/transfer and every two
years thereafter
Preferred
Bachelor's Degree in Computer Science, or related technical discipline AND 1+ year(s) technical engineering experience
with coding in languages including, but not limited to, C, C++, C#, Java, JavaScript, OR Python, + OR Master's Degree in
Computer Science or related technical discipline with proven experience coding in languages including, but not limited
to, C, C++, C#, Java, JavaScript, or Python, + OR equivalent experience
1+ year(s) technical engineering experience with coding in languages including, but not limited to, Rust, VHDL, or
Verilog
```

✓ This is a Software Engineer Job Description.

Extracted Skills

C, C++, C#, Java, JavaScript, Python, Rust, VHDL, Verilog, Software Engineering, Technical Engineering

5.5.3 Take Test Form: The Test ID is pre-generated during test creation. The user enters their name and email to begin the interview, which creates a unique session that links the user to the test for evaluation and result tracking.



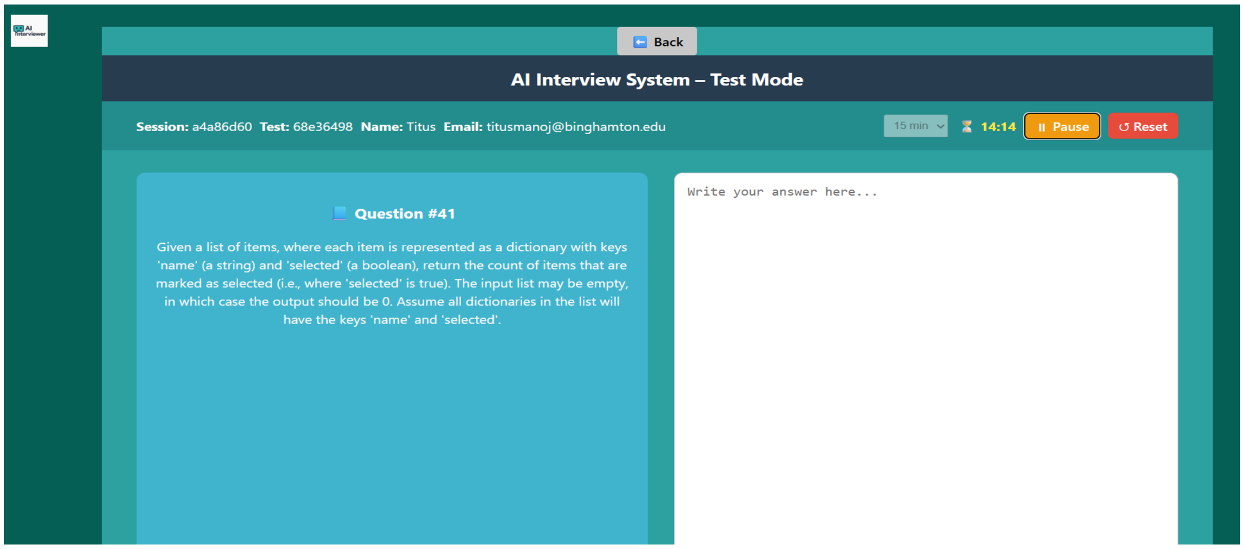
Take a Test (Enter Details)

68e36498

Titus

titusmanoj@binghamton.ed

5.5.4 Start Test: This page initializes the interview session, displays the generated coding question, and provides an editor for the candidate to write and submit their solution.



AI Interview System – Test Mode

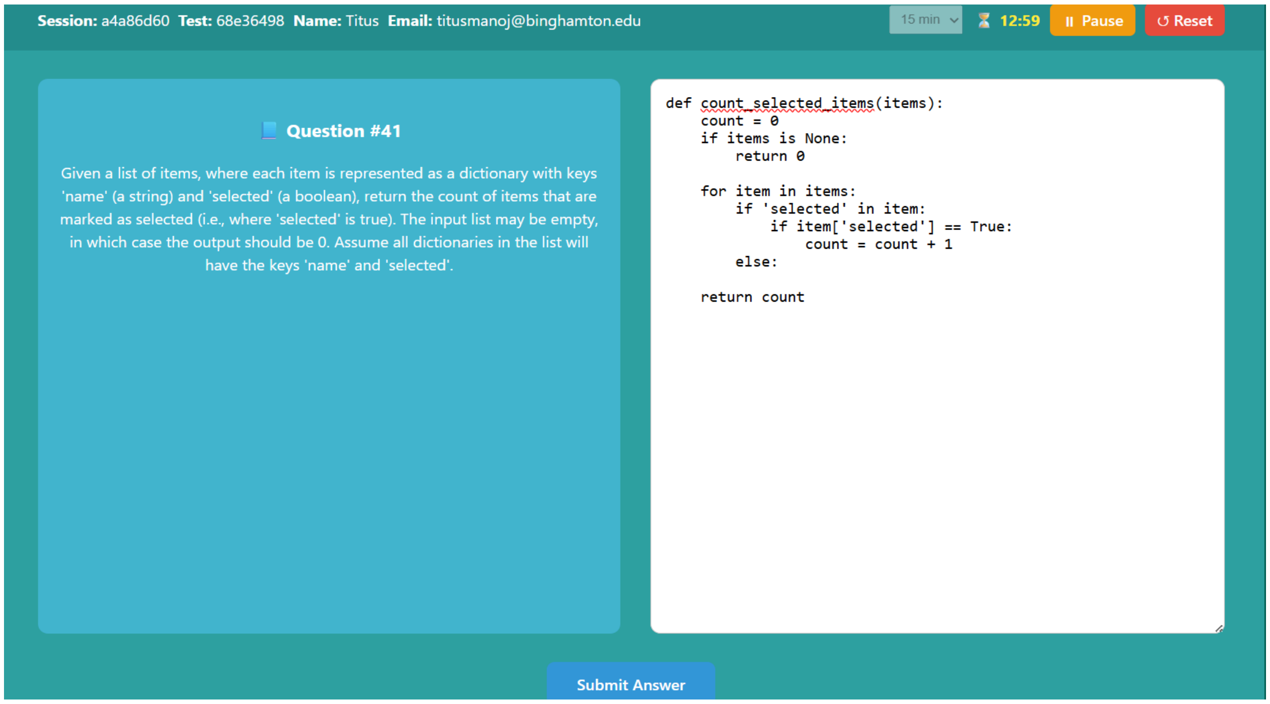
Session: a4a86d60 **Test:** 68e36498 **Name:** Titus **Email:** titusmanoj@binghamton.edu

Question #41

Given a list of items, where each item is represented as a dictionary with keys 'name' (a string) and 'selected' (a boolean), return the count of items that are marked as selected (i.e., where 'selected' is true). The input list may be empty, in which case the output should be 0. Assume all dictionaries in the list will have the keys 'name' and 'selected'.

```
Write your answer here...
```

Timer: A configurable countdown timer tracks the allowed duration and enforces the time limit during the test.



Session: a4a86d60 **Test:** 68e36498 **Name:** Titus **Email:** titusmanoj@binghamton.edu

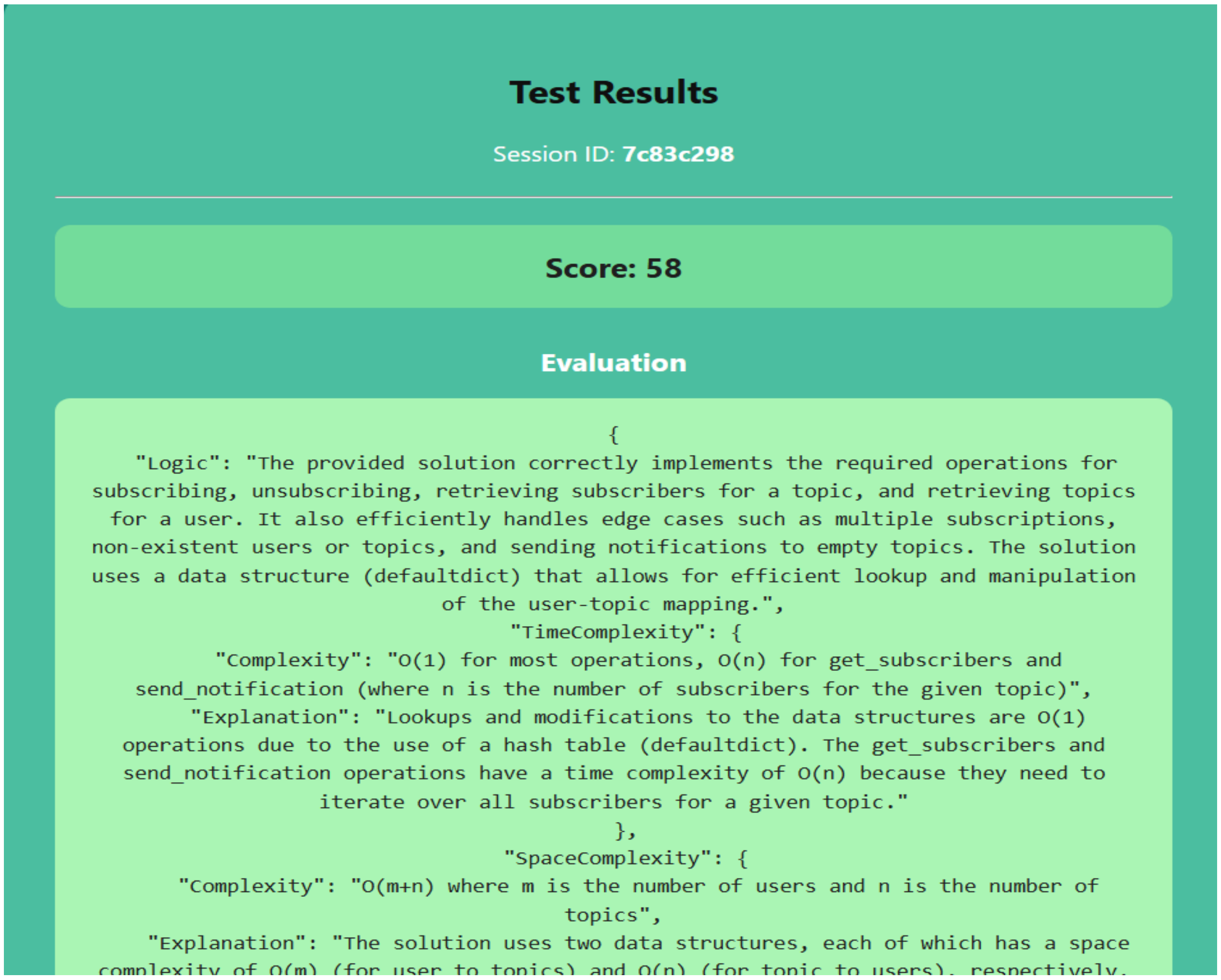
Question #41

Given a list of items, where each item is represented as a dictionary with keys 'name' (a string) and 'selected' (a boolean), return the count of items that are marked as selected (i.e., where 'selected' is true). The input list may be empty, in which case the output should be 0. Assume all dictionaries in the list will have the keys 'name' and 'selected'.

```
def count_selected_items(items):
    count = 0
    if items is None:
        return 0

    for item in items:
        if 'selected' in item:
            if item['selected'] == True:
                count = count + 1
        else:

    return count
```

5.5.5 Result: Displays the AI-evaluated score for the submitted solution along with concise feedback on logic, time complexity, space complexity, and syntax to help the candidate understand their performance.

Score Interpretation: The final score is calculated by combining points across logic correctness, complexity analysis, syntax accuracy, low-level design quality, algorithmic efficiency, and scalability. Strong, optimized solutions earn higher scores, while brute-force, unclear, or inefficient answers result in lower totals.



Test Results

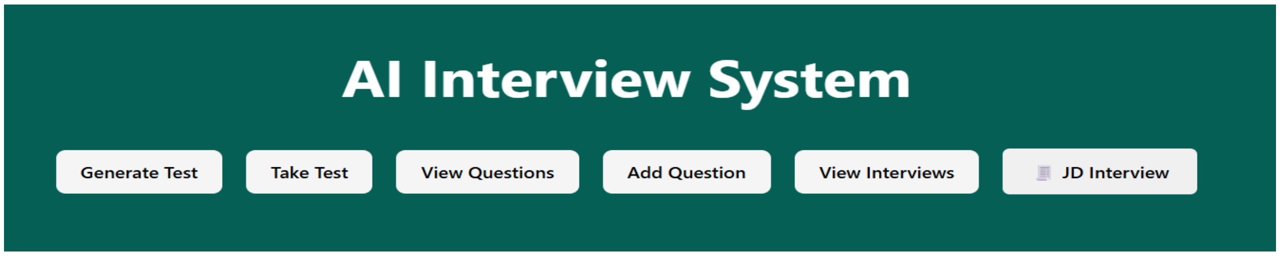
Session ID: **7c83c298**

Score: 58

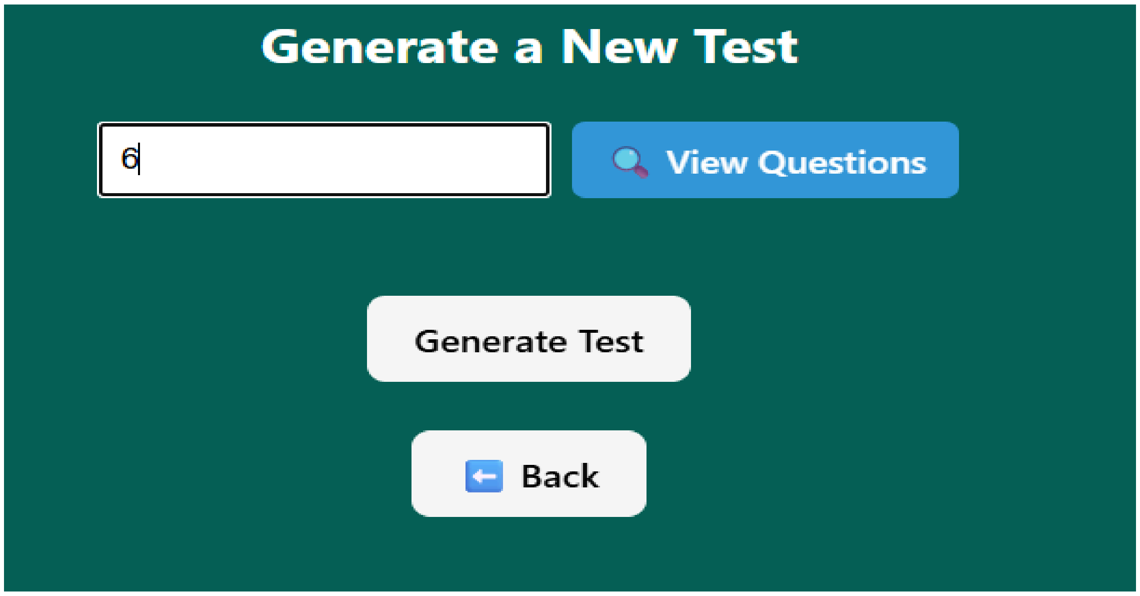
Evaluation

```
{
"Logic": "The provided solution correctly implements the required operations for subscribing, unsubscribing, retrieving subscribers for a topic, and retrieving topics for a user. It also efficiently handles edge cases such as multiple subscriptions, non-existent users or topics, and sending notifications to empty topics. The solution uses a data structure (defaultdict) that allows for efficient lookup and manipulation of the user-topic mapping.",
"TimeComplexity": {
"Complexity": "O(1) for most operations, O(n) for get_subscribers and send_notification (where n is the number of subscribers for the given topic)",
"Explanation": "Lookups and modifications to the data structures are O(1) operations due to the use of a hash table (defaultdict). The get_subscribers and send_notification operations have a time complexity of O(n) because they need to iterate over all subscribers for a given topic."
},
"SpaceComplexity": {
"Complexity": "O(m+n) where m is the number of users and n is the number of topics",
"Explanation": "The solution uses two data structures, each of which has a space complexity of O(m) (for user to topics) and O(n) (for topic to users), respectively.
```

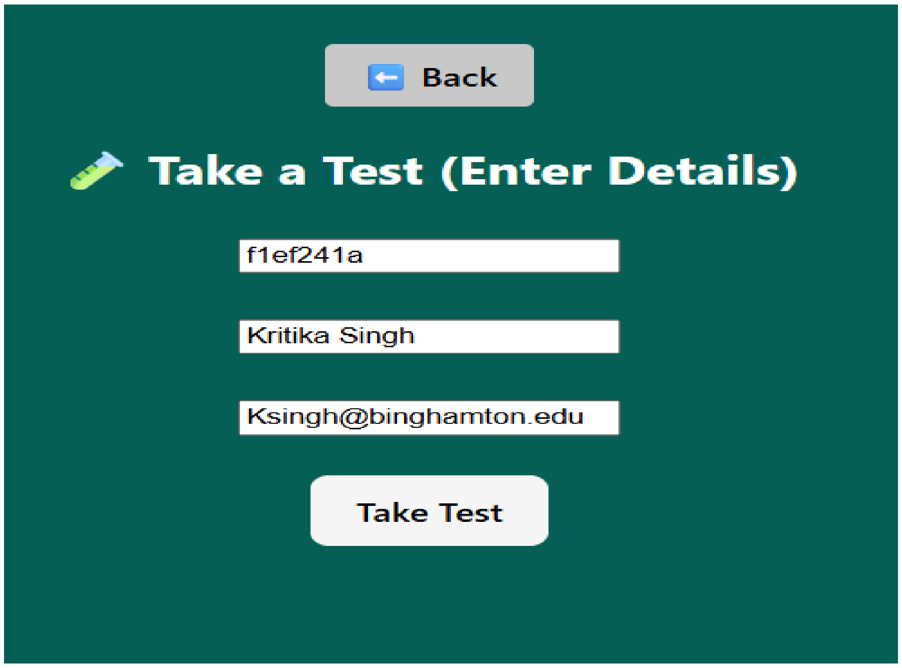
Manual Workflow:



5.5.6 Generate Test: Allows users to generate a test by selecting a specific question using its Question ID. If the candidate wants to browse available questions and decide which one to take the test on, they can click ***View Questions*** to see the list and manually choose the desired Question ID.



5.5.7.1 Take Test Form: After the test is generated, the system creates a unique **Test ID** linked to the selected question; the candidate enters this Test ID along with their name and email to start the interview session.



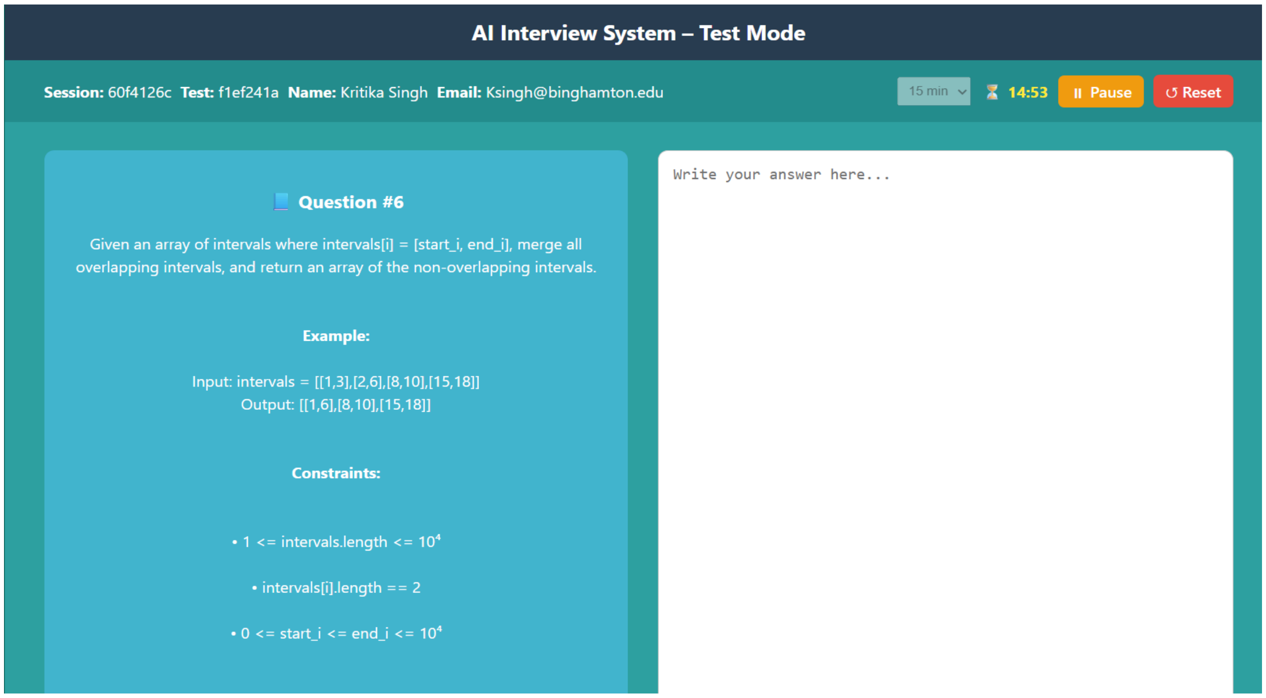
Take a Test (Enter Details)

f1ef241a

Kritika Singh

Ksingh@binghamton.edu

5.5.7.2. Start Test Page: Once the test begins, the system loads the assigned coding question with its constraints and examples, activates the countdown timer, and provides an editor where the candidate can write and submit their solution.



AI Interview System – Test Mode

Session: 60f4126c **Test:** f1ef241a **Name:** Kritika Singh **Email:** Ksingh@binghamton.edu

Question #6

Given an array of intervals where intervals[i] = [start_i, end_i], merge all overlapping intervals, and return an array of the non-overlapping intervals.

Example:

Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]

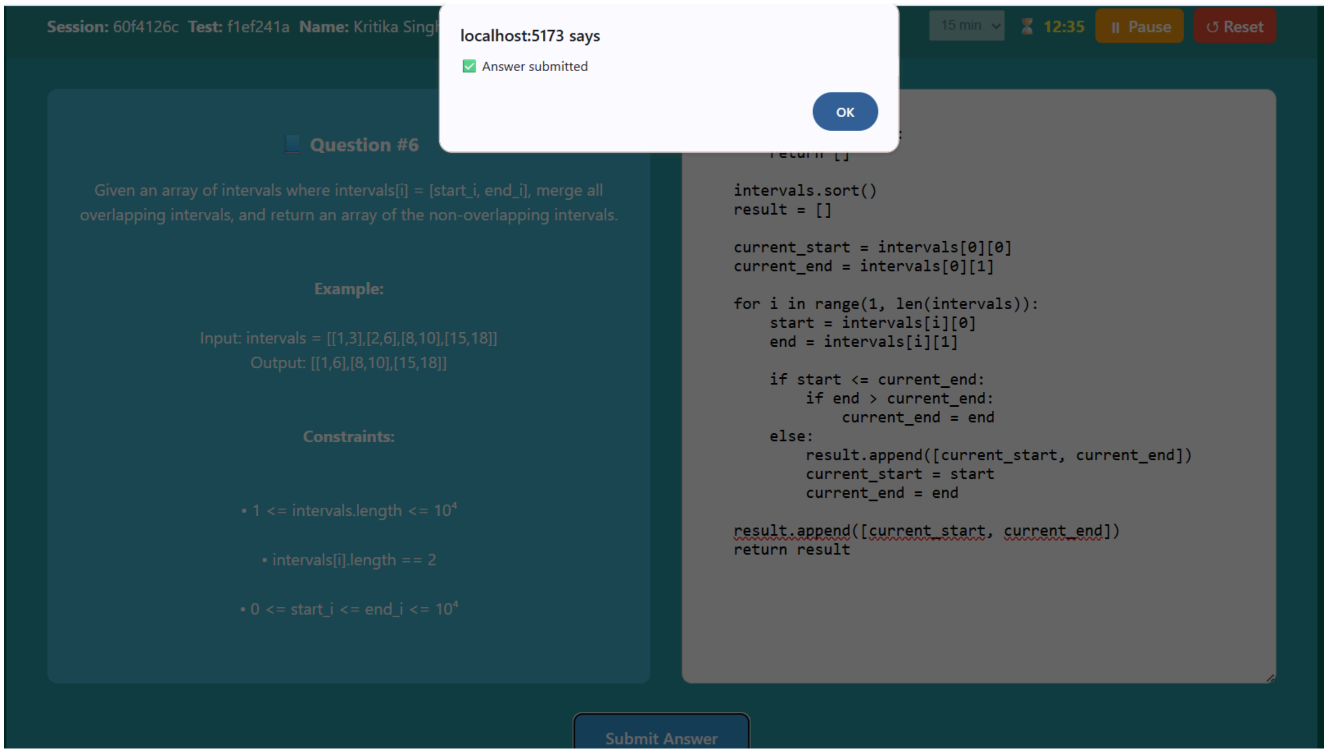
Constraints:

• 1 <= intervals.length <= 10^4

• intervals[i].length == 2

• 0 <= start_i <= end_i <= 10^4

```
Write your answer here...
```



Session: 60f4126c **Test:** f1ef241a **Name:** Kritika Singh

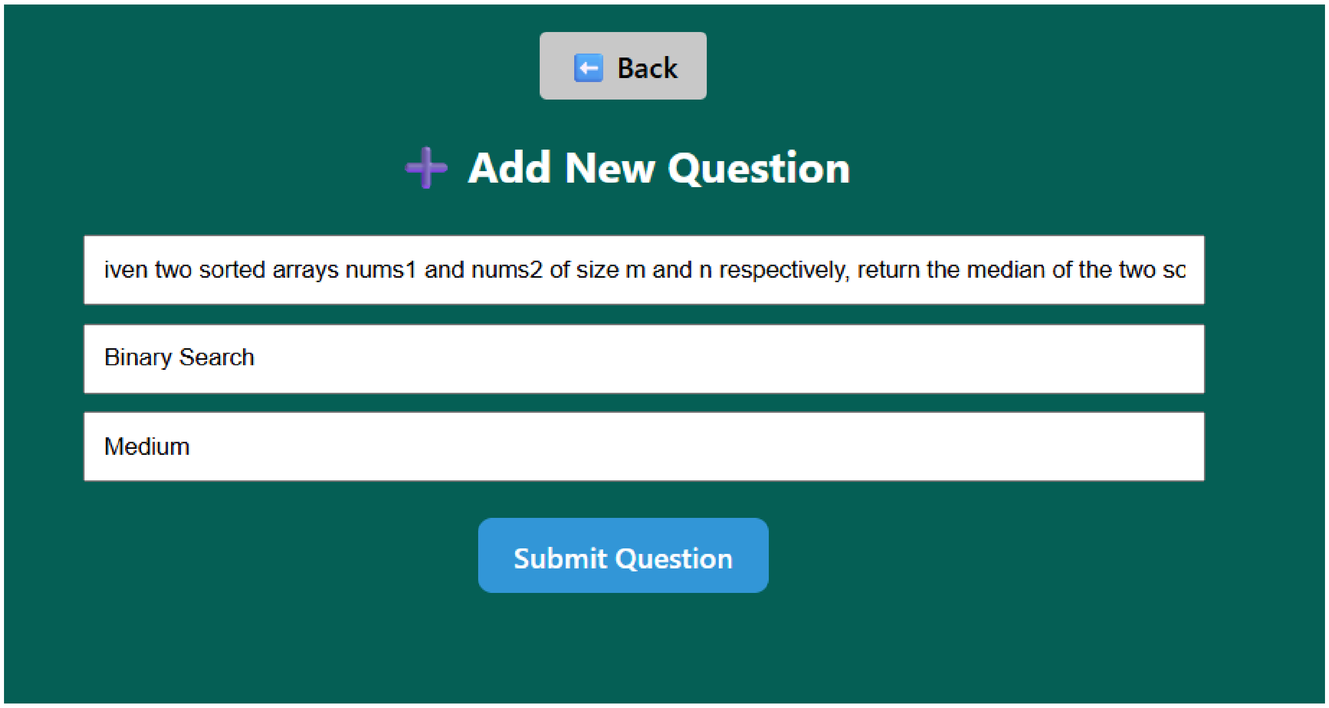
localhost:5173 says

✅ Answer submitted

```
    intervals.sort()
    result = []

    current_start = intervals[0][0]
    current_end = intervals[0][1]

    for i in range(1, len(intervals)):
        start = intervals[i][0]
        end = intervals[i][1]

        if start <= current_end:
            if end > current_end:
                current_end = end
        else:
            result.append([current_start, current_end])
            current_start = start
            current_end = end

    result.append([current_start, current_end])
    return result
```

5.5.8 Result: Automated evaluation results page that refreshes in real time(Figure 5.4). Fetches and shows the AI-generated score and evaluation.

5.5.9 Add Question: A question management UI to insert/view problems with examples and constraints. Allows admin to add new interview questions.

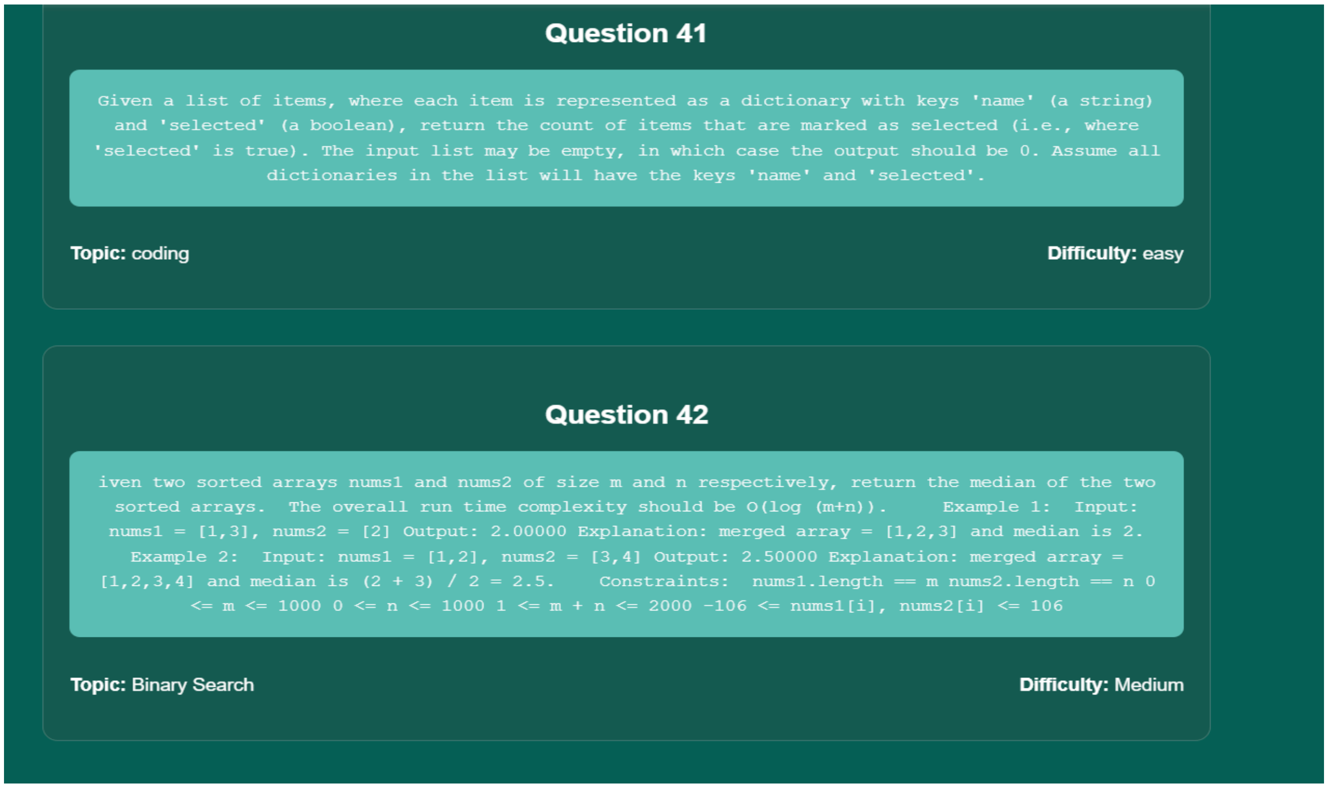


iven two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sc

Binary Search

Medium

5.5.10 Questions: Lists all available questions in a tabular format.



Question 41

Given a list of items, where each item is represented as a dictionary with keys 'name' (a string) and 'selected' (a boolean), return the count of items that are marked as selected (i.e., where 'selected' is true). The input list may be empty, in which case the output should be 0. Assume all dictionaries in the list will have the keys 'name' and 'selected'.

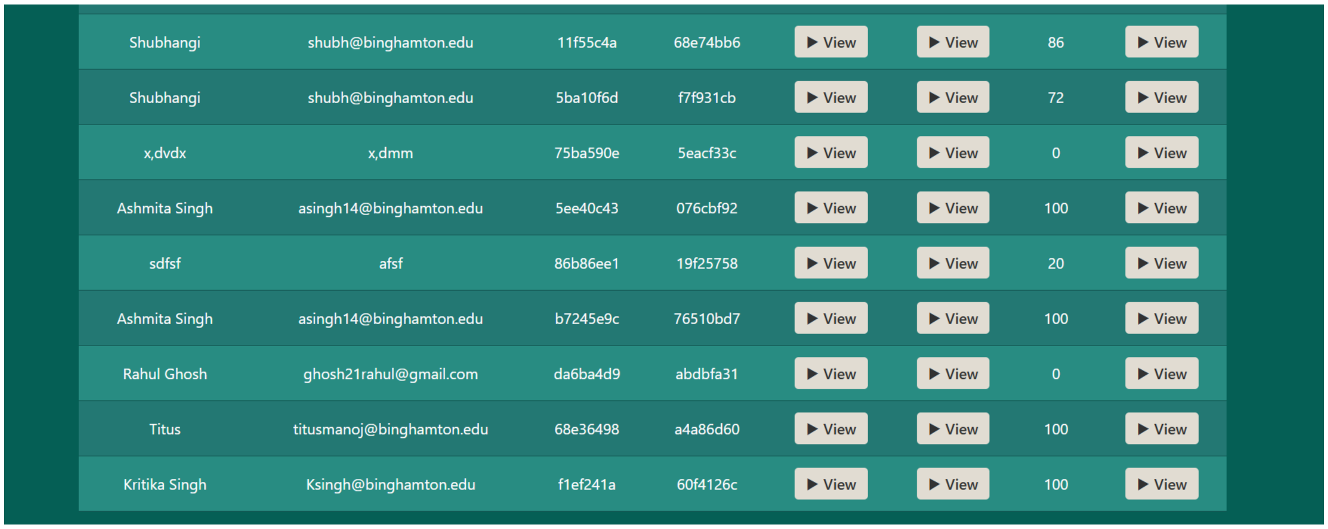
Topic: coding **Difficulty:** easy

Question 42

iven two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)). Example 1: Input: nums1 = [1,3], nums2 = [2] Output: 2.00000 Explanation: merged array = [1,2,3] and median is 2. Example 2: Input: nums1 = [1,2], nums2 = [3,4] Output: 2.50000 Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5. Constraints: nums1.length == m nums2.length == n 0 <= m <= 1000 0 <= n <= 1000 1 <= m + n <= 2000 -106 <= nums1[i], nums2[i] <= 106

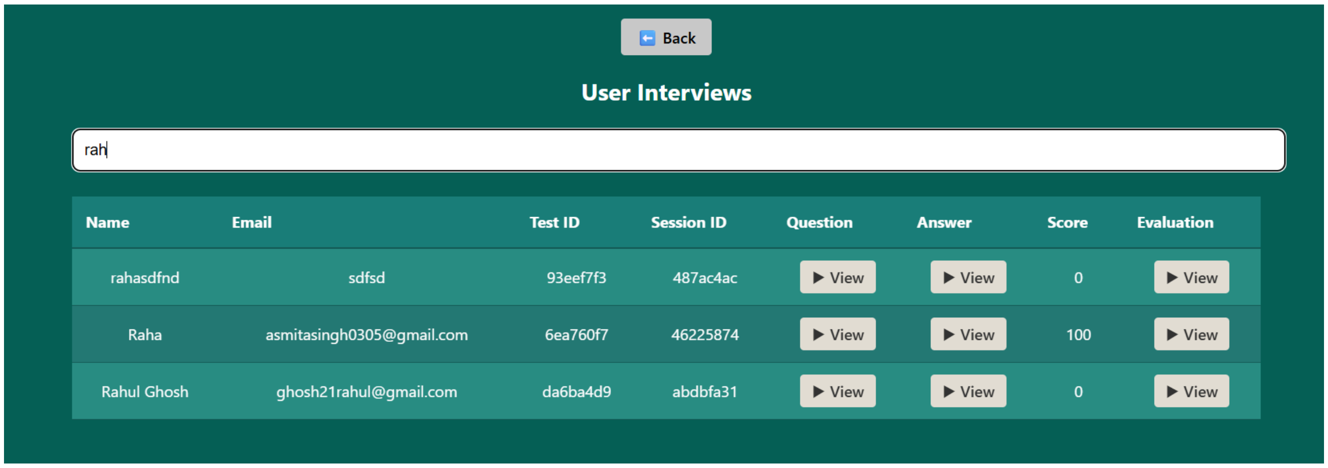
Topic: Binary Search **Difficulty:** Medium

5.5.11 User Interviews log: A user interview log page that displays all interviews taken with score and evaluation in a table.



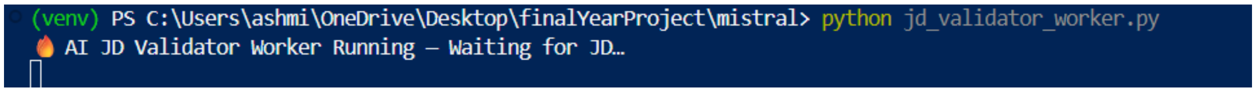
Shubhangi	shubh@binghamton.edu	11f55c4a	68e74bb6	▶ View	▶ View	86	▶ View
Shubhangi	shubh@binghamton.edu	5ba10f6d	f7f931cb	▶ View	▶ View	72	▶ View
x,dvdx	x,dmm	75ba590e	5eacf33c	▶ View	▶ View	0	▶ View
Ashmita Singh	asingh14@binghamton.edu	5ee40c43	076cbf92	▶ View	▶ View	100	▶ View
sdfsf	afsf	86b86ee1	19f25758	▶ View	▶ View	20	▶ View
Ashmita Singh	asingh14@binghamton.edu	b7245e9c	76510bd7	▶ View	▶ View	100	▶ View
Rahul Ghosh	ghosh21rahul@gmail.com	da6ba4d9	abdbfa31	▶ View	▶ View	0	▶ View
Titus	titusmanoj@binghamton.edu	68e36498	a4a86d60	▶ View	▶ View	100	▶ View
Kritika Singh	Ksingh@binghamton.edu	f1ef241a	60f4126c	▶ View	▶ View	100	▶ View

The search bar lets users quickly filter interview records by name or email, making it easy to locate specific interview sessions and their results.

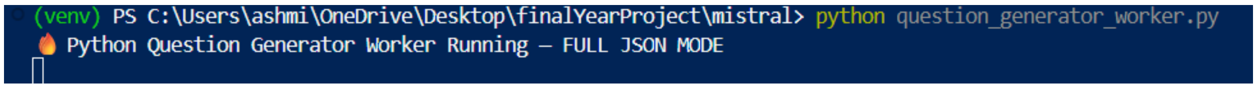


Name	Email	Test ID	Session ID	Question	Answer	Score	Evaluation
rahasdfnd	sdfsd	93eef7f3	487ac4ac	▶ View	▶ View	0	▶ View
Raha	asmitasingh0305@gmail.com	6ea760f7	46225874	▶ View	▶ View	100	▶ View
Rahul Ghosh	ghosh21rahul@gmail.com	da6ba4d9	abdbfa31	▶ View	▶ View	0	▶ View

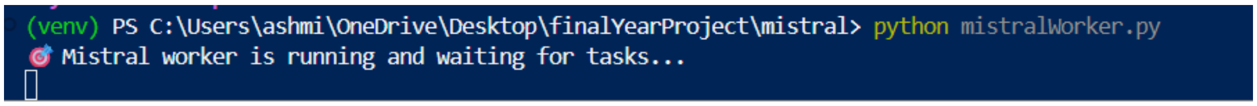
5.5.12. These workers run in the background to handle the system's core automation: validating job descriptions, generating questions, and performing AI answer evaluation; allowing the interview platform to function asynchronously and efficiently.



```
(venv) PS C:\Users\ashmi\OneDrive\Desktop\finalYearProject\mistral> python jd_validator_worker.py
🔥 AI JD Validator Worker Running — Waiting for JD…
```

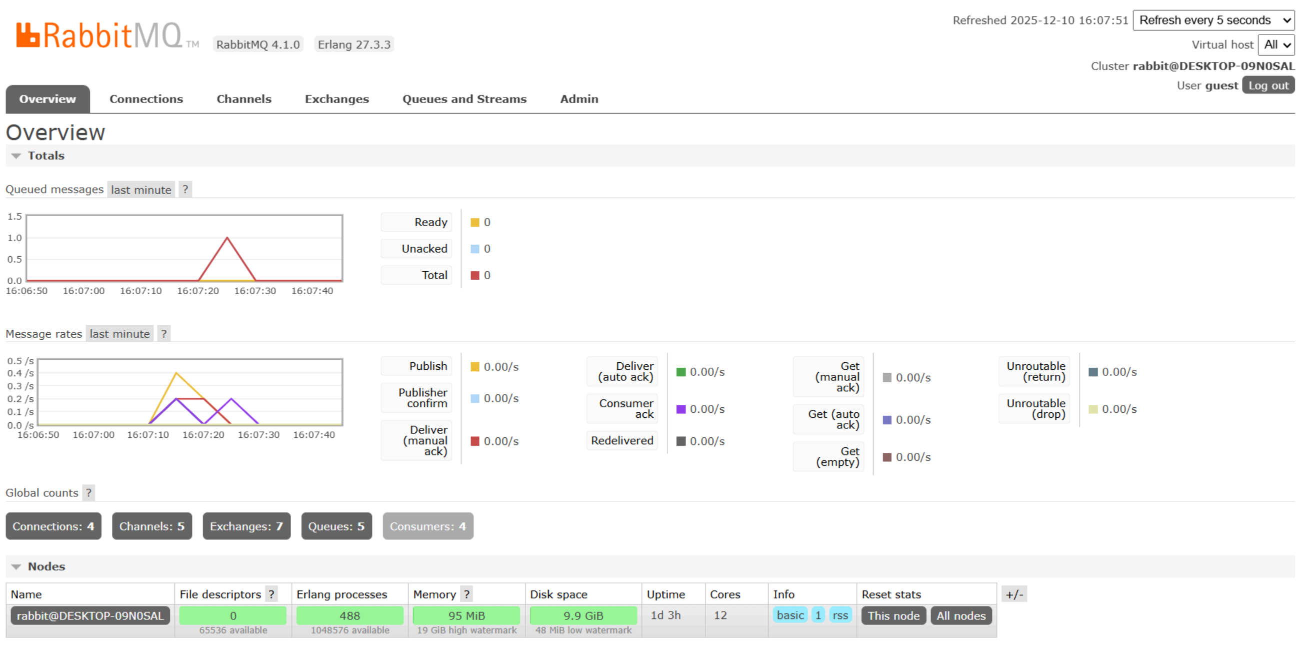


```
(venv) PS C:\Users\ashmi\OneDrive\Desktop\finalYearProject\mistral> python question_generator_worker.py
🔥 Python Question Generator Worker Running — FULL JSON MODE
```



```
(venv) PS C:\Users\ashmi\OneDrive\Desktop\finalYearProject\mistral> python mistralWorker.py
🎯 Mistral worker is running and waiting for tasks...
```

5.5.13 Uses RabbitMQ to offload evaluation tasks asynchronously.



RabbitMQ™ RabbitMQ 4.1.0 Erlang 27.3.3

Refreshed 2025-12-10 16:07:51 Refresh every 5 seconds

Virtual host All

Cluster **rabbit@DESKTOP-09N0SAL**

User **guest** Log out

Overview | Connections | Channels | Exchanges | Queues and Streams | Admin

Overview

Totals

Queued messages last minute ?

Ready	0
Unacked	0
Total	0

Message rates last minute ?

Publish	0.00/s	Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s	Unroutable (return)	0.00/s
Publisher confirm	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s	Unroutable (drop)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s		

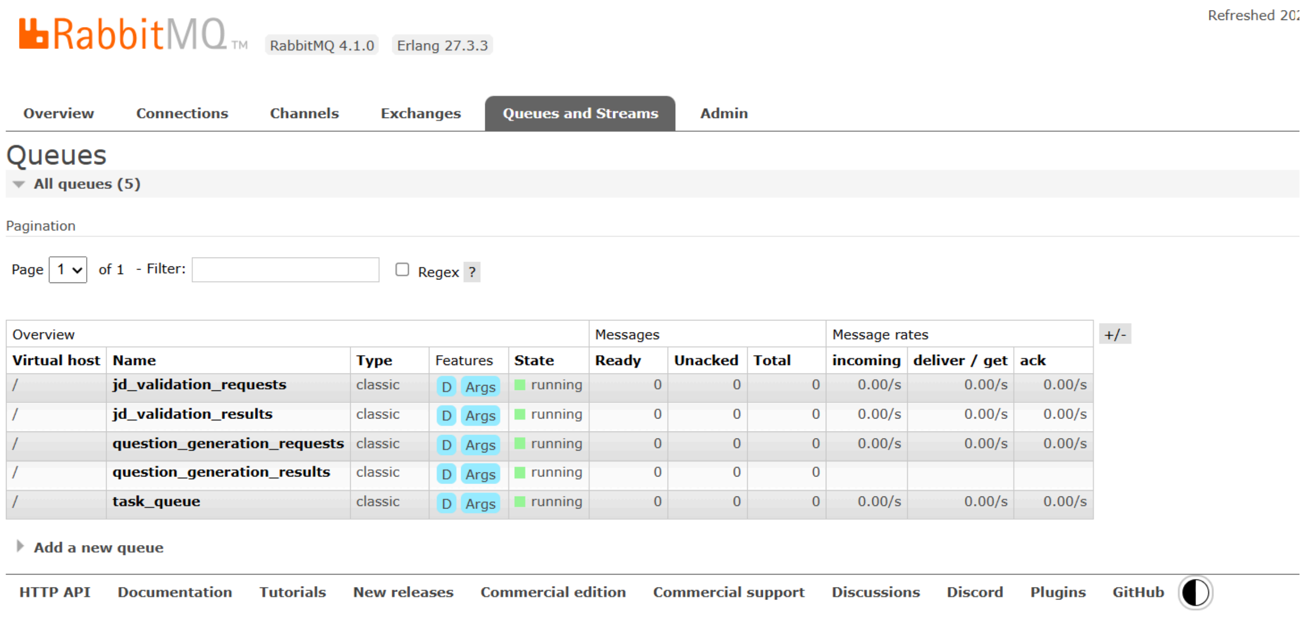
Global counts ?

Connections: **4** Channels: **5** Exchanges: **7** Queues: **5** Consumers: **4**

Nodes

Name	File descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Cores	Info	Reset stats
rabbit@DESKTOP-09N0SAL	0 65536 available	488 1048576 available	95 MiB 19 GiB high watermark	9.9 GiB 48 MiB low watermark	1d 3h	12	basic 1 rss	This node All nodes

+/-



RabbitMQ™ RabbitMQ 4.1.0 Erlang 27.3.3

Refreshed 20

Overview | Connections | Channels | Exchanges | **Queues and Streams** | Admin

Queues

All queues (5)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Overview					Messages			Message rates		
Virtual host	**Name**	**Type**	**Features**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
/	**jd_validation_requests**	classic	D Args	running	0	0	0	0.00/s	0.00/s	0.00/s
/	**jd_validation_results**	classic	D Args	running	0	0	0	0.00/s	0.00/s	0.00/s
/	**question_generation_requests**	classic	D Args	running	0	0	0	0.00/s	0.00/s	0.00/s
/	**question_generation_results**	classic	D Args	running	0	0	0			
/	**task_queue**	classic	D Args	running	0	0	0	0.00/s	0.00/s	0.00/s

+/-

Add a new queue

HTTP API | Documentation | Tutorials | New releases | Commercial edition | Commercial support | Discussions | Discord | Plugins | GitHub

6. Conclusions:

This project successfully delivers an end-to-end AI-powered interview system that validates job descriptions, generates role-relevant coding questions, conducts timed interview sessions, and provides automated, structured evaluations with scores and feedback. It demonstrates a scalable, asynchronous, and user-centric application of full-stack and AI technologies for practical use in technical education and hiring, while laying a strong foundation for future enhancements such as adaptive feedback and deeper evaluation intelligence.

7. References:

- **Spring Boot (Backend Framework)**
  https://spring.io/projects/spring-boot
- **React.js (Frontend Library)**
  https://reactjs.org/
- **PostgreSQL (Database Engine)**
  https://www.postgresql.org/
- **RabbitMQ (Message Broker)**
  https://www.rabbitmq.com/
- **Axios (Frontend HTTP Client)**
  https://axios-http.com/
- Draw.io / Diagrams.net (Architecture Diagram Tool)
  https://app.diagrams.net/
- **LeetCode (Sample Question Source)**
  https://leetcode.com/
- Python Programming Language
  https://www.python.org/
- Hugging Face (LLM Hosting & Model Hub)
  https://huggingface.co/
- Google Colab (Model Experimentation & Training Environment)
  https://colab.research.google.com/
- Pika (RabbitMQ Python Client Library)
  https://pika.readthedocs.io/
- psycopg2 (PostgreSQL Python Driver)
  https://www.psycopg.org/
- Mistral AI / Mistral API (LLM used for question generation & evaluation)
  https://docs.mistral.ai/
- JSON (Data Interchange Format Specification)
  https://www.json.org/json-en.html
- VS Code (Development Environment)
  https://code.visualstudio.com/
- Postman (API Testing Tool)
  https://www.postman.com/
- Git & GitHub (Version Control)
  https://git-scm.com/