Code Sensei - Distributed AI Programming Assessment Handler

Aaron Bengochea, Allan Thekkepeedika, Janardhana Reddy, Timothy Cao, Yusen Li, Yuyang Tan
Computer Science Department, New York University
Email: (ab6503, ajt444, jr6922, tc4088, ly2602, yt2079)@nyu.edu

Abstract

Programming assessments and technical interviews remain pivotal yet often suffer from rigid question flows, limited personalization, and delayed feedback. We introduce Code Sensei, an AI driven programming assessment platform that dynamically adapts to each candidate's performance by leveraging large language model powered recommendation engines. Code Sensei indexes a question bank in OpenSearch for sub second retrieval, issues problems via an asynchronous API driven workflow, and evaluates submitted code in real time providing both correctness validation and fine grained performance metrics (execution time, memory usage, etc.). Under the hood, a modular, cloud-native architecture combines serverless Lambdas for orchestration, AWS Fargate for safe containerized execution, and VPC resident Bedrock endpoints to minimize latency and maximize security. Data driven insights fuel an AI enabled personalized question flow and an integrated assessment assistant, transforming each assessment into an interactive learning experience rather than a static test. Preliminary benchmarks demonstrate Code Sensei's ability to scale seamlessly to thousands of concurrent sessions while maintaining strict isolation and low latency. By unifying adaptive problem selection, immediate feedback, and scalable infrastructure, Code Sensei offers a next generation solution for technical hiring and learning.

I. Problem Statement

Technical interviews and programming assessments are cornerstone activities for both hiring and skills development, yet existing solutions too often fall short in three critical areas:

1. **Static, One-Size-Fits-All Question Flows**
   Traditional assessment platforms present a fixed sequence of problems with no adaptation based on a candidate's real time performance. This rigidity leads to either trivial questions that bore strong candidates or overly difficult challenges that frustrate less experienced ones, resulting in poor engagement and an inability to accurately gauge ability.
2. **Delayed, Superficial Feedback**
   Many systems only signal "pass" or "fail" after code submission, without granular insights into execution time, memory consumption, or test case coverage. This lack of detailed, immediate feedback deprives candidates of learning opportunities and prevents interviewers or educators from diagnosing specific areas for improvement.
3. **Scalability and Security Trade-offs**
   High volume assessments require sandboxed execution environments to protect platform integrity and candidate data. Existing on premise or monolithic solutions often struggle to elastically scale to meet peak demand, or they compromise isolation, increasing the risk of code injection exploits and cross tenant interference.

These shortcomings highlight the need for an assessment platform that can dynamically personalize question selection, provide real time, performance driven feedback, and scale securely to thousands of concurrent sessions. Code Sensei addresses this gap by integrating LLM powered recommendations, containerized code execution, and fine grained telemetry into a unified, cloud-native architecture.

II. Motivation

In an era where software drives innovation across every industry, finding and developing top engineering talent has become both critical and challenging. Companies invest significant time and resources in coding interviews and technical assessments, yet cumbersome, one-size-fits-all processes frequently lead to candidate frustration, interviewer fatigue, and lost opportunities to identify truly exceptional problem solvers. At the same time, learners seeking to improve their skills often lack timely, actionable feedback that bridges the gap between theory and practice.

By harnessing AI to tailor question selection in real time and deliver fine grained performance insights, Code Sensei addresses these pain points head on. Personalized assessment flows keep candidates engaged and motivated, while immediate visibility into execution efficiency and test coverage accelerates learning and reduces interviewer overhead. Moreover, a cloud-native, serverless architecture ensures that organizations of any size can scale assessments up or down on demand without sacrificing security or reliability. Collectively, these advances promise to transform technical hiring and skills development into a more efficient, equitable, and educational experience.

III. Existing Solutions

HackerRank:

HackerRank is one of the most widely adopted online coding assessment platforms, offering a large library of algorithmic and data structure challenges across multiple languages and difficulty levels. It provides employers with customizable screening tests, real time code compilation, and detailed candidate performance analytics.

LeetCode:

LeetCode is an online coding interview preparation platform that features thousands of community contributed problems, weekly contests, and mock interview functionality. Solutions are evaluated on both correctness and execution performance, with leaderboards to benchmark candidates against peers.

Codility:

Codility specializes in automated technical screening with an emphasis on work sample and real world coding tasks delivered in a browser based IDE. It includes built in plagiarism detection and supports custom test cases to simulate production scenarios.

CodeSignal:

CodeSignal's Pre-Screen and Skills Assessment products offer JavaScript, Python, and C++ challenges, UI based real world tasks, and AI driven scoring for high volume hiring pipelines. Their platform integrates with major ATS tools and provides predictive analytics on candidate success metrics.

InterviewBit:

InterviewBit combines bite sized practice problems with an automated "Interview Simulator" environment, interactive hinting, and configurable proctoring settings. It supports subjective and project based tasks in addition to standard algorithmic questions, with real time feedback and plagiarism prevention.

Common Limitations

Despite their popularity, these platforms share several challenges:

- **One-size-fits-all flows**: Static question sequencing does not adapt to individual performance trends, potentially under or over challenging candidates.
- **Limited contextual feedback**: While most platforms report execution time and memory usage, they lack deep, personalized insights on code style, algorithmic choices, or learning pathways.
- **Scale vs. proctoring trade-offs**: High volume automated assessments risk cheating without robust, real time AI driven proctoring integrated at the question level.

How Code Sensei Addresses These Gaps

Code Sensei introduces an adaptive, performance driven question flow powered by LLM recommendations that consider both current assessment progress and detailed past performance metrics. By embedding an AI assessment assistant directly into each question session, it delivers contextual hints, code review suggestions, and personalized learning guidance in real time. Furthermore, Code Sensei's infrastructure is architected for seamless horizontal scaling, integrating secure containerized execution environments with AI enabled privacy preserving telemetry and robust identity based access controls.

IV. System Architecture

Code Sensei's system architecture is built on a modular, cloud-native foundation to ensure high availability, fault tolerance, and efficient resource utilization. Compute intensive tasks such as code compilation and sandboxed execution are offloaded to AWS Fargate, a serverless container engine that automatically scales containers and eliminates server management overhead reducing costs and accelerating deployment. Event driven orchestration is handled by AWS Lambda, enabling low latency, real time processing of assessment workflows with built in automatic scaling and minimal operational complexity. Problem indexing and retrieval employ Amazon OpenSearch Service, a fully managed, scalable search engine offering sub second indexing and query performance over large question datasets. Foundation model inference occurs via Amazon Bedrock through private VPC interface endpoints, delivering secure, low latency access to LLMs without exposing traffic to the public internet. Together, these components operate within a unified VPC backed microservices framework that achieves seamless horizontal scalability to thousands of concurrent sessions while maintaining strict isolation, observability, and cost effectiveness. Figure 1 illustrates Code Sensei's overall system architecture and it's major components.

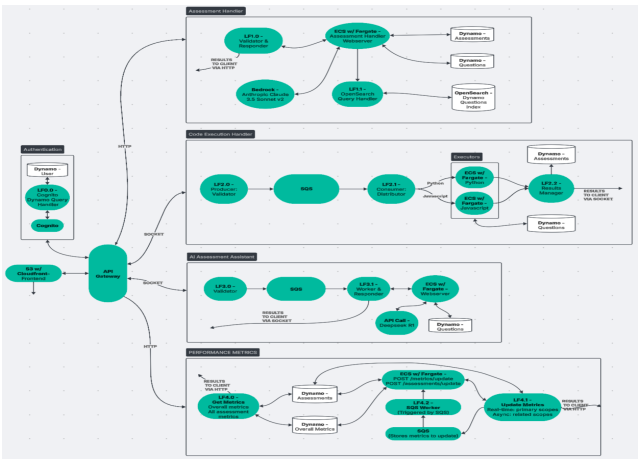

Figure 1: Code Sensei's overall system architecture and it's major components

Overview of System Components

The following components work together seamlessly to deliver a robust and user friendly adaptive coding assessment experience.

Client Layer

The Client Layer is the candidate's primary interface into Code Sensei. It is implemented as a modern Single Page Application (SPA) and provides the following key capabilities:

- **Custom Assessment Configuration:** Candidates (or interviewers) select topics, difficulty levels, total number of questions, and overall duration before beginning an assessment. All settings are sent to the backend via a secure REST call to initialize a new assessment session.
- **Integrated Code Editor:** A browser based editor lets users write, edit, and run code in real time. Submissions are executed against hidden test suites via WebSocket, with results streamed back immediate correctness, stdout/stderr, and detailed failure information.
- **Rich Problem Display:** Each question pane shows title, full description, topic tags, difficulty badge, and sample input/output.
- **AI Powered Assessment Assistant:** A chat widget ingests the user's current code snapshot, then leverages the Bedrock backed assistant to answer questions, suggest debugging strategies, or provide targeted examples. All hint requests and responses occur over the same WebSocket channel to preserve context.
- **Live Performance Metrics:** A dashboard panel updates after each run with execution time, memory usage, and test case coverage. Historical metrics from prior runs in the current assessment are visualized to help users identify regressions or performance improvements over time.

API Layer

The API Layer is built on AWS API Gateway, offering both HTTP REST and WebSocket interfaces secured by AWS Cognito JWT authorizers. Client calls for assessment actions and performance metrics are handled via RESTful HTTP connections, while code submissions and AI assistant interactions utilize WebSocket for low latency, bidirectional messaging. All requests are validated and throttled by API Gateway before being forwarded to small "validator" Lambdas that perform authentication context extraction and input sanitization. Detailed CloudWatch logging and custom metrics ensure end to end observability.

- **Protocol Routing:** Assessment Handler and Performance Metrics services use RESTful HTTP; Code Execution Handler and AI Assessment Assistant use WebSocket.
- **Authentication & Authorization:** Cognito User Pools validate JWTs for all routes.
- **Request Validation & Throttling:** API Gateway models and mapping templates enforce schema checks; built in throttling and quotas protect against abuse.
- **Observability & Monitoring:** CloudWatch metrics and logs capture latency, error rates, and connection counts.
- **Lambda Integration:** Validator Lambdas extract auth context, sanitize inputs, and route requests to downstream services via direct invocation or SQS/EventBridge.

Figure 2 illustrates the system design of our API Gateway and Security.

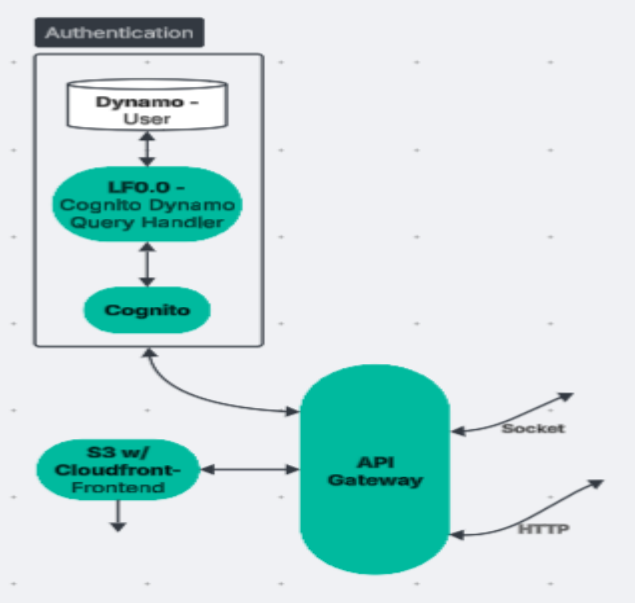

Figure 2: System design - API Gateway and Security

Assessment Handler

The Assessment Handler provides the core orchestration for managing the "begin," "ongoing," and "end" phases of each coding assessment. Incoming REST requests are first validated and sanitized by a lightweight AWS Lambda, which enforces authentication and schema checks before handing off to a purpose built service running on AWS Fargate. Within this containerized environment, the service:

- **State Management:** Persists and updates the assessment and the question state in DynamoDB.
- **Question Retrieval:** Delegates lookups to an OpenSearch backed index via the OpenSearch init Lambda.
- **Adaptive Recommendation:** Invokes Amazon Bedrock LLMs over a private VPC endpoint to compute the next topic–difficulty pairing. This recommendation is driven by the candidate's real time performance metrics (execution time, memory usage, correctness) and the overall topics and difficulty levels initially selected by the client.

Once the next action is determined, whether issuing a new problem, updating progress, or finalizing the session, the handler composes a concise JSON payload and returns it to

the client, ensuring low latency feedback and clear separation of concerns. Figure 3 illustrates the system design of the Assessment Handler service.

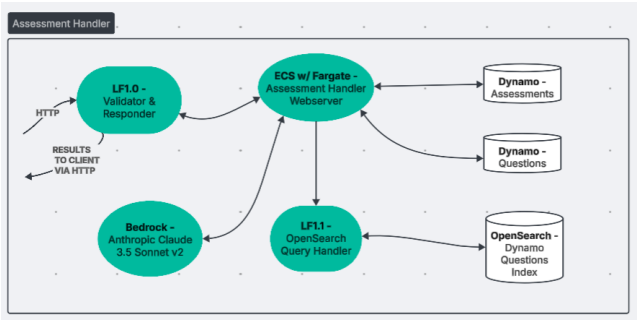

Figure 3: System design - Assessment Handler

Code Execution Handler

The Code Execution Handler orchestrates secure, language specific execution of candidate code against hidden test suites in real time. Submissions arrive via WebSocket and are validated by an AWS Lambda function, which enqueues each request in Amazon SQS.

- **Task Queuing:** Validated code submissions are placed on an SQS queue to decouple client interaction from execution.
- **Language-Specific Execution Clusters:** A consumer retrieves tasks, fetches test cases from the Question Bank in DynamoDB, and dispatches code to AWS Fargate containers within dedicated ECS clusters (e.g., Python, JavaScript), which automatically scale based on queue depth.
- **Result Processing:** Upon completion, results including pass/fail status, stdout/stderr output, execution time, and memory usage are recorded in the Assessments DynamoDB table.
- **Real-Time Feedback:** A dedicated function streams execution feedback back to the candidate over WebSocket.

This event driven, multi stage pipeline ensures low latency, secure code evaluation with comprehensive error handling and full observability via Amazon CloudWatch metrics and logs. Figure 4 illustrates the system design of the Code Execution Handler service. Figure 4 illustrates the system design of the Code Execution Handler service.

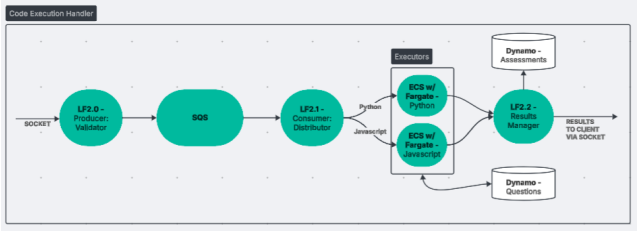

Figure 4: System design - Code Execution Handler

AI Assessment Assistant

The AI Assessment Assistant orchestrates dynamic, real time evaluation of client submissions by ingesting submissions via WebSocket connections, utilizing both the current state of the client's assessment and the newly asked question to generate contextual feedback, examples, or hints. Leveraging Amazon Bedrock's generative models (using Deepseek R1) to assess correctness and generate feedback.

- **Submission Ingestion:** Receives each client's submission and contextual metadata (e.g., question ID, difficulty level, past performance) via WebSocket.
- **Contextual Feedback:** Incorporates the current assessment state and the client's new question state to provide targeted feedback, illustrative examples, or hints.
- **Intermediate Streaming:** Streams intermediate evaluation states back to the client over WebSocket connections.
- **Audit Logging:** Persists both inputs and model outputs in the Assessments DynamoDB table for full auditability.

This event driven, multi stage pipeline ensures low latency, transparent, and scalable assessments with full observability via Amazon CloudWatch metrics and logs. Figure 5 illustrates the system design of the AI Assessment Assistant service.

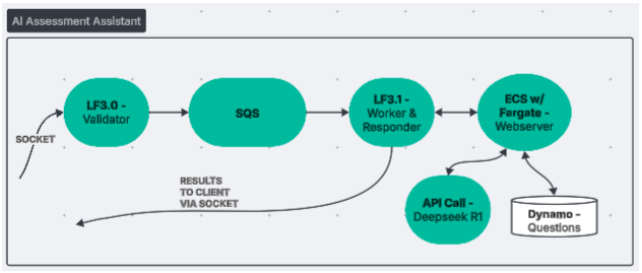

Figure 5: System design - AI Assessment Assistant

Performance Metrics

The Performance Metrics component tracks and analyzes client performance by computing average metrics (e.g., time taken) per topic and difficulty scope. It embeds all question level data within each Assessment record and leverages an AssessmentQuestionLocator table to map user+scope combinations (e.g., easy#graphs) to question indices across assessments for efficient lookup.

- **Data Modeling & Lookup:** Embeds question data in Assessment and uses AssessmentQuestionLocator to map user+scope combinations to their question indices.
- **Real Time Scope Update:** On question completion, updates scoped totals in the Assessment record and computes the immediate average for that specific scope (e.g., time for easy#graphs).
- **Asynchronous Scope Enqueue:** Pushes related scopes (e.g., easy#all, all#graphs, all#all) onto an SQS queue for deferred processing.

- **Background Aggregation:** A worker consumes SQS messages, uses the locator table to aggregate metrics across assessments, and updates precomputed averages in the Metrics DynamoDB table.
- **Low Latency Retrieval:** Exposes a lightweight HTTP endpoint that reads precomputed values from the Metrics table to serve dashboard queries with minimal delay.

This architecture balances real time feedback with scalable asynchronous aggregation, minimizing compute and database load while ensuring fast metric retrieval for dashboards. Figure 6 illustrates the system design of the Performance Metrics service.

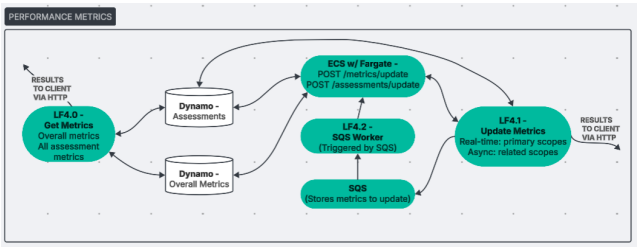

Figure 6: System design - Performance Metrics

V. Results

The Code Sensei platform has been rigorously validated through extensive testing and real world candidate interactions. Below, we highlight the system's core capabilities adaptive problem selection, real time code execution, AI driven guidance, and performance analytics each illustrated with accompanying figures and metrics.

Landing Page

Upon arriving at Code Sensei's landing page, users are greeted with a clean, dark themed interface and a striking hero graphic that instantly conveys the platform's focus on AI enhanced coding. A concise tagline summarizes key benefits real time assessments, performance metrics, and personalized learning while a prominent "Sign in" call to action directs candidates to authenticate before proceeding. The left hand navigation remains disabled until login, reinforcing that access to the dashboard and assessments is gated behind user authentication.

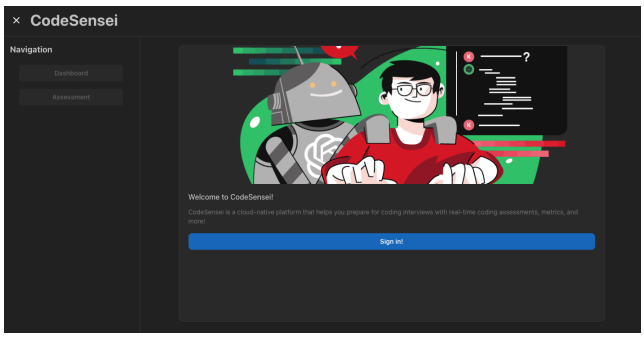

Figure 7: Code Sensei - Landing Page

Login Page

The login page provides a minimalistic interface backed by AWS Cognito for secure authentication, enabling users to seamlessly sign in with enterprise or social credentials before gaining access to the platform.

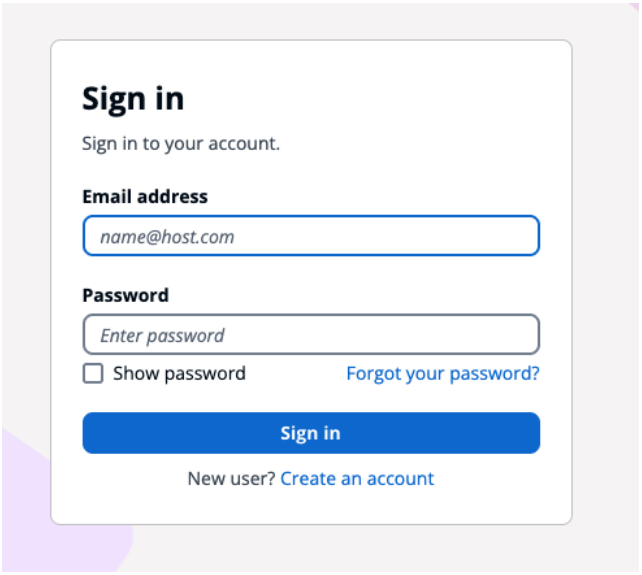

Figure 8: Code Sensei - Sign In Page

Main Dashboard

The assessment configuration panel enables users to select one or more topics, choose desired difficulty levels, and specify the total number of questions before beginning a session. A persistent left-hand navigation menu provides direct access to both the Assessment setup and the Performance Metrics dashboard, streamlining movement between configuration and results.

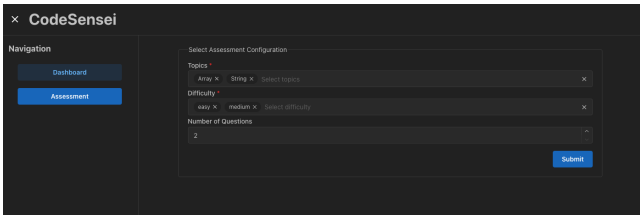

Figure 9: Code Sensei - Main Dashboard

Assessment View

During an active assessment, candidates work in a full featured code editor that supports both Python and JavaScript via a simple language toggle, with problem details (title, description, topics, difficulty) displayed alongside. Users may execute their solution against hidden test cases with the submit button or opt to skip to the next question at any time, allowing for a seamless and flexible workflow.

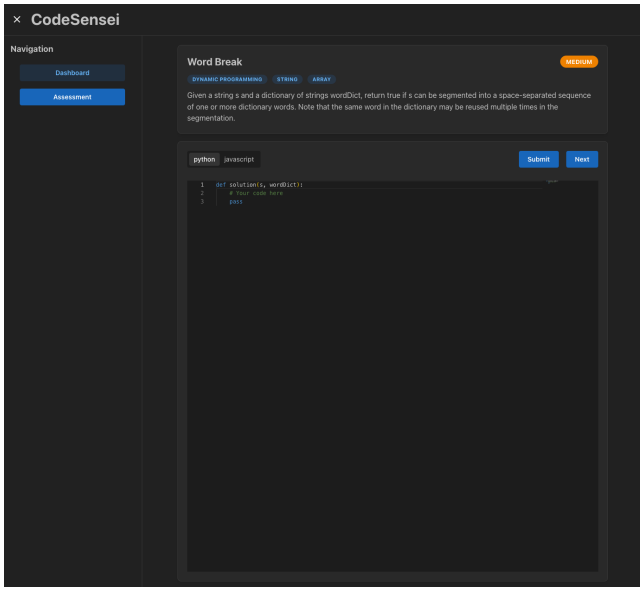

Figure 10: Code Sensei - Assessment View

Output Window

The output pane presents each hidden test case alongside its expected result and the user's code output in a clear, line by line format, enabling immediate visibility into correctness and facilitating rapid iteration.

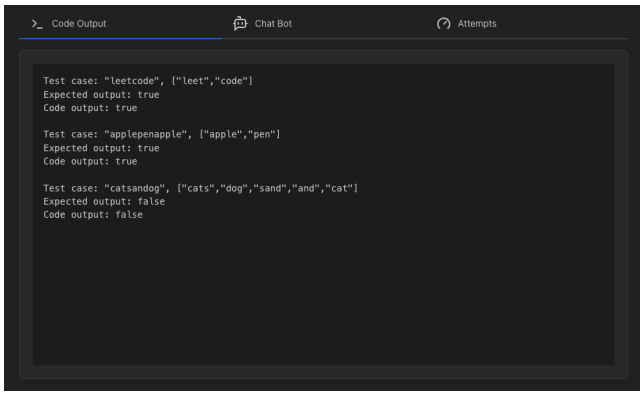

Figure 11: Code Sensei - Output Window

AI Assessment Assistant

The integrated AI Assessment Assistant lets candidates interactively ask questions about the current problem or their code, drawing on context from the active editor and the user provided question. Powered by Deepseek R1, it delivers targeted hints, suggestions, and design guidance to help users overcome challenges in real time.

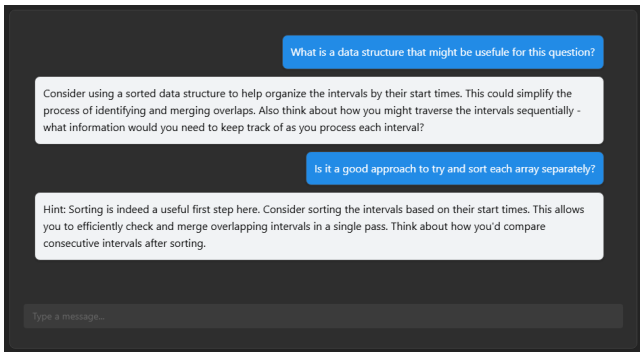

Figure 12: Code Sensei - AI Assessment Assistant

Performance Metrics Dashboard The Metrics Dashboard offers both per assessment and aggregate performance insights, displaying key metrics such as time spent, execution time, and memory usage broken down by topic difficulty pairs. Users can drill into individual assessments for detailed analysis or view their cumulative performance trends across all completed sessions to track progress and identify areas for improvement.

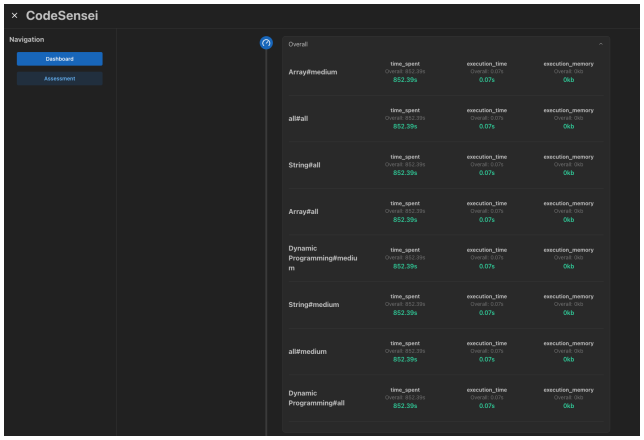

Figure 13: Code Sensei - Performance Metrics Dashboard

VI. Conclusion and Future Work

In this work, we introduced a fully serverless, event driven AI Assessment platform that (1) ingests client submissions in real time via WebSockets, (2) dynamically constructs context aware prompts using both current assessment state and newly asked questions, (3) leverages Amazon Bedrock to deliver immediate correctness judgments, rich feedback and hints, and (4) persists all interactions for full audit and observability. By combining in place, per scope updates with asynchronous batch aggregation for performance metrics, the system achieves low latency responses without overloading compute or database resources. Our multi stage pipelines provide a scalable, transparent, and secure foundation for personalized AI driven assessments.

Future Work

- **Adaptive Learning Paths:** Automatically adjust question sequences and difficulty based on each client's historical performance and topic mastery.
- **Rubric Customization UI:** Expose a web interface for instructors to tweak rubric weightings, feedback tone, and hint generation parameters.
- **Multimodal Support:** Extend beyond code and text to handle diagram based questions or short audio/video submissions.
- **LMS & API Integrations:** Build connectors for common learning management systems (Canvas, Moodle) and open API endpoints for downstream analytics.
- **A/B Testing of Feedback Strategies:** Experiment with different prompt templates or model variants to determine which feedback styles maximize learning gains.
- **Mobile & Offline Clients:** Offer a lightweight mobile app with cached question sets and deferred sync for low bandwidth or intermittent connectivity scenarios.
- **Expanded Language Support:** Add execution runtimes for additional languages (e.g., Java, C++, Ruby) beyond the current Python and JavaScript clusters to broaden assessment coverage.