CodeSync - A collaborative online Integrated Development Environment (IDE)

Suprateek Chatterjee
sc10344@nyu.edu

Karmanya Mendiratta
km6296@nyu.edu

Ananya Kumar Gangver
akk8368@nyu.edu

Aditya Ojha
ao2612@nyu.edu

***Abstract*—The efficacy of traditional technical interviews has been declining in the rapidly evolving software industry, where real-time collaboration and problem-solving are paramount. Traditional methods, such as phone screenings and isolated coding challenges, often fail to accurately assess a candidate's ability to code in a collaborative environment, leaving companies unable to effectively gauge true on-the-job performance. CodeSync, an innovative cloud-based Integrated Development Environment (IDE), has been developed to address these challenges. It transforms the technical interview process by enabling real-time collaborative coding sessions within a robust and user-friendly interface.**

Leveraging a suite of AWS services, CodeSync offers a secure, scalable, and highly responsive platform. It incorporates Amazon Cognito for stringent user authentication, AWS Lambda for efficient serverless computing, Amazon DynamoDB and S3 for reliable data storage and retrieval, AWS Amplify for hosting the web application and Amazon EC2 for scalable container management. These technologies collectively support a seamless coding environment where interviewers and candidates can interact dynamically, sharing code and results in real-time.

The deployment of CodeSync demonstrates a successful use case of combining various AWS services to create a solution that not only meets the high demands of technical recruiting but also significantly improves the interview and assessment process. This platform thus represents a significant advancement in the domain of technical hiring, providing an effective bridge between the needs of employers and the capabilities of prospective employees.

***Index Terms*—AWS, Cloud Computing, [specific AWS services used]**

I. Introduction

A. *Project Context*

In recent years, the software industry has seen a significant shift towards more dynamic and collaborative work environments. As teams become more global and projects more complex, the ability to effectively collaborate and code in real-time has become crucial. However, traditional technical interviews often fail to capture this dynamic, relying instead on asynchronous coding tasks or over-simplified coding problems that do not reflect the true nature of modern software development. This gap between the interview process and actual job demands can lead to ineffective hiring decisions, affecting the productivity and innovation capabilities of organizations. To address these challenges, there has been a growing emphasis on enhancing the technical interview process through innovative solutions. Online coding interviews that leverage cloud-based Integrated Development Environments (IDEs) are becoming increasingly popular as they allow for a more accurate assessment of a candidate's coding skills and their ability to collaborate in real-time. Such environments not only provide the tools necessary for coding and testing but also mirror the collaborative and fast-paced nature of today's tech workplaces.

B. *Benefits of Cloud-Based Solutions in Technical Assessments*

Cloud-based solutions offer numerous advantages over traditional local environments, especially in the context of technical assessments:

- **Scalability:** Cloud environments can easily scale to accommodate a large number of users or increase in resource demand, making it feasible to conduct multiple interviews simultaneously without degradation in performance.
- **Accessibility:** With internet access, cloud-based IDEs allow candidates and interviewers to engage in coding interviews from anywhere in the world. This accessibility broadens the talent pool, enabling companies to attract and evaluate global talent effortlessly.
- **Real-Time Collaboration:** Cloud-based platforms can facilitate real-time interactions, where interviewers can observe and interact with the candidate as they code. This setup provides deeper insights into the candidate's thought process, problem-

solving approach, and ability to communicate effectively under pressure.

- **Security and Integrity:** Using cloud-based platforms ensures that the code, data, and results of coding sessions are securely stored and managed. Advanced security protocols can protect sensitive data and maintain the integrity of the interview process.
- **Cost-Effectiveness:** Cloud platforms eliminate the need for organizations to invest in physical infrastructure and maintenance of testing environments. This reduction in capital expenditure makes cloud-based assessments an economical choice for companies.
- **Uniformity in Testing Environment:** Cloud-based IDEs ensure that all candidates are tested under the same software conditions, with the same tools and resources available. This uniformity helps in fair assessment and comparison of candidates' skills.

II. System Architecture Overview

The architecture of CodeSync is designed to leverage the power and scalability of various AWS services to create a robust, secure, and scalable cloud-based IDE. This section details the specific AWS services utilized in the project and their roles within the overall system architecture.

A. Services Used

- **Amazon Cognito:** Utilized for user authentication and authorization. Cognito provides user sign-up, sign-in, and access control functionalities, ensuring that only authenticated users can access their coding sessions and personal data.
- **AWS Lambda:** Serverless computing service used to execute backend code in response to events such as HTTP requests from Amazon API Gateway. Lambda functions handle business logic, including user and session management, data retrieval, and other backend processes, reducing the need for physical server management.
- **Amazon API Gateway:** Acts as the entry point for all backend services. It routes incoming API requests to the appropriate Lambda functions and is integral for creating scalable and secure APIs.
- **Amazon DynamoDB:** A NoSQL database service used for storing user and room data efficiently. DynamoDB offers quick data retrieval and scalability, which is critical for maintaining smooth operation and user experience during peak loads.
- **Amazon S3:** Used for storing and retrieving user-generated content and session data. S3 provides durable, highly available object storage, ensuring that user data is safely stored and persistently accessible.
- **Amazon Simple Queue Service (SQS):** Manages message queuing for decoupling internal services and reliably transmitting event data between different components of the application, such as between Lambda functions and ECS containers.
- **Amazon Elastic Cloud Compute (EC2):** Manages Docker containers that host the application's runtime environments and dependencies. ECS ensures efficient container orchestration, allowing for scaling and management of application deployment.
- **AWS Simple Email Service (SES):** Manages the sending of emails for notifications and communications to users, enhancing the platform's engagement capabilities.
- **Docker Container:** Responsible for running code at the backgrounf in individual sandbox environment.
- **Web Sockets:** Responsible for real time updates for both the clients within time constraints.

B. Architecture Diagram

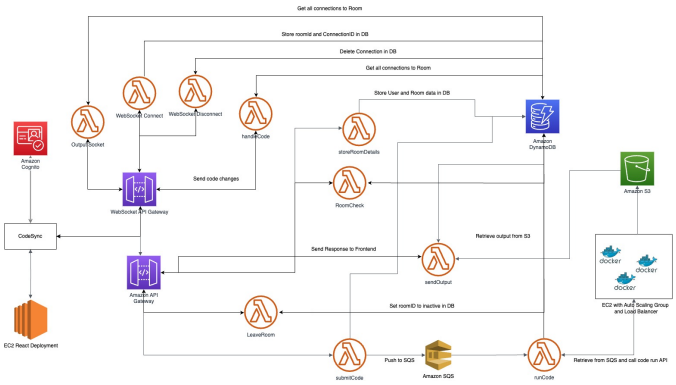

Fig. 1. Architecture of CodeSync

III. Back-end and Front-end Implementation

A. User Authentication Using AWS Cognito and CodeSyncPool

AWS Cognito is integral to our web application, CodeSync, a real-time coding interview platform. The 'CodeSyncPool' user pool within AWS Cognito is specifically configured to streamline the sign-in and sign-up processes, ensuring a seamless user experience and robust security for both candidates and interviewers.

*1) **Functionality of CodeSyncPool:*** 'CodeSyncPool' manages critical tasks for CodeSync, including user registration via email or social accounts, secure authentication with optional MFA, and autonomous password recovery for enhanced user independence.

*2) **Integration with Frontend Components:*** 'CodeSyncPool' is tightly integrated with CodeSync's frontend components to ensure a seamless user experience:

- signin.jsx: Handles user authentication during sign-in.
- signup.jsx: Manages new user registrations and initial setup.
- home.jsx: Retrieves user ID details post-authentication to personalize settings and preferences.

B. Code Execution in an Isolated Environment

The platform executes user-submitted Python code within isolated Docker containers, providing a secure and controlled runtime environment. Key features include:

- Docker Containers: Each code execution occurs in a standalone Docker container, ensuring isolation and preventing any potential interference between different users' processes.
- Restricted Python Environment: Only specific, whitelisted Python functions are allowed, safeguarding against malicious attempts to execute harmful operations.
- Execution Time Limit: Each execution session is constrained to a maximum of three minutes, automatically terminating longer running scripts to free up system resources.
- Custom Libraries: Users can specify necessary Python libraries via a `requirements.txt` file, which are installed in the Docker container before code execution.

*1) **Scalability and Load Management:*** To handle variable load efficiently, the platform utilizes AWS services for dynamic scaling and load balancing:

- Auto Scaling Group (ASG): Automatically scales the number of EC2 instances based on CPU utilization, ensuring that the load is always within optimal limits.
- Application Load Balancer (ALB): Distributes incoming API requests across the available instances evenly, enhancing the responsiveness and reliability of the code execution API.
- Scaling Policy: Configured to initiate scaling actions once CPU utilization crosses an 80% threshold, guaranteeing performance stability during peak loads.

*2) **Microservice Architecture:*** The system is structured around a microservice architecture, enabling modular development and deployment of functionalities:

- Lambda Functions:
  - Submit-Code Function: Receives code submissions, stores them in S3, signals SQS, and updates DynamoDB with the code execution status.
  - Run-Code Function: Polls SQS for new tasks, invokes the code execution API, stores outputs in S3, and updates the execution status in DynamoDB.
  - Send-Output Function: Retrieves execution results from S3 and delivers them to the requesting frontend application.
- REST API: Managed through API Gateway, exposing various endpoints for the frontend to interact with backend services securely.

*3) **Data Management and Messaging:***

- Amazon S3: Used for storing scripts and execution outputs, offering scalable and secure data storage solutions.
- Amazon SQS: Manages the order of code execution requests, ensuring that submissions are processed sequentially.
- Amazon DynamoDB: Stores and manages execution statuses, providing fast and efficient state management across components.

C. Use of Web Sockets

- In our collaborative coding environment, the WebSocket flow is essential for enabling real-time communication and synchronization among users. This integration involves several AWS Lambda functions that manage user connections, broadcast code changes, execute code, and share results.
- When a user creates a room, a unique roomId is generated, allowing other users to join the room using this identifier. When a user joins a room, a WebSocket API call triggers the WebSocketConnect Lambda function. This function extracts the connectionId and roomId from the incoming request and stores them in the DynamoDB table named WebSocketConnections thus establishing a socket connection.

- As users interact with the code editor and make changes, these updates are sent to the WebSocket API, which triggers the handleUpdateCode Lambda function. This function retrieves all connectionIds associated with the roomId from DynamoDB. It then broadcasts the code changes to all connections, except the one that initiated the change, ensuring that the code editor remains synchronized across all users in the room in real-time. This mechanism allows multiple users to see each other's edits instantaneously.
- Upon pressing the "Run Code" button, the code execution flow gets initiated. The code is executed, and the resulting output is stored in an S3 bucket. The output is then sent to the WebSocket API, triggering the outputSocket Lambda function. This function retrieves the output and broadcasts it to all connections in the room. Consequently, all users can view the execution results simultaneously.
- If a user chooses to leave the room, pressing the "Leave Room" button triggers the WebSocketDisconnect Lambda function. This function removes the user's connectionId from the WebSocketConnections table in DynamoDB, effectively disconnecting the user from the room and terminating their session.

D. *Front-End Development with React and Vite*

The front-end of CodeSync, designed for real-time coding interviews, was developed using React and Vite. React was chosen for its efficient, component-based architecture, allowing for dynamic and responsive user interfaces. Its virtual DOM optimizes re-rendering, essential for the high performance required during live sessions.

Vite, a modern build tool, significantly enhances the development workflow with features like hot module replacement (HMR) and efficient production builds. It rebuilds only changed modules during development, speeding up the process. Vite's integration with React streamlined setup and deployment, providing out-of-the-box optimizations such as asset compression and code splitting. This setup ensures quick load times and a smooth user experience on the AWS EC2 deployed application, making the platform robust and responsive.

IV. Challenges Faced

During the development of our real-time coding interview platform, CodeSync, we encountered several significant challenges that tested our technical capabilities and problem-solving skills. Here's a brief overview of these challenges and the solutions we implemented:

- **Web Sockets Integration:** Managing real-time communication via Web Sockets was complex, especially ensuring data consistency and timely delivery across multiple client sessions.
- **Running Code in a Secure and Isolated Environment:** We utilized Docker containers to execute user-submitted code securely, isolating each session to prevent security breaches.
- **Displaying Output for All Users:** Synchronizing code output for all participants in a session was achieved through Web Sockets, ensuring everyone saw real-time results simultaneously.
- **Storage and Retrieval of Outputs:** Outputs were stored in AWS S3 buckets with optimized access and retrieval strategies to maintain performance.
- **Real-Time Output Display:** After storing outputs in S3, updates were pushed to all users via Web Sockets, triggered by notifications sent through SQS.
- **Deploying the React App on EC2:** We faced challenges in deploying and scaling our React application on EC2, which we overcame by automating our deployment processes using AWS CodeDeploy for better load management.

These challenges were instrumental in enhancing the robustness and efficiency of the CodeSync platform, providing invaluable learning experiences in handling advanced cloud-based architectures and real

V. User Interface

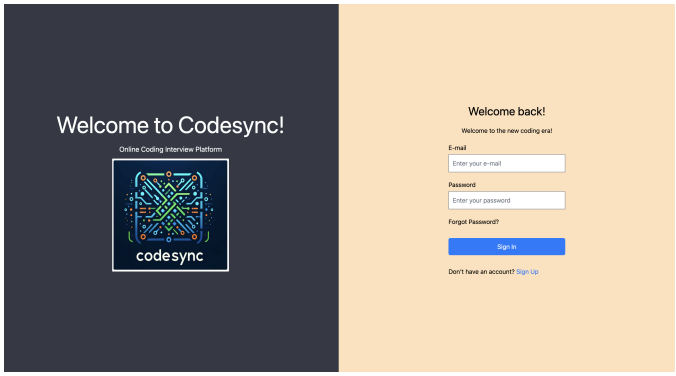

Fig. 2. Sign-In Page

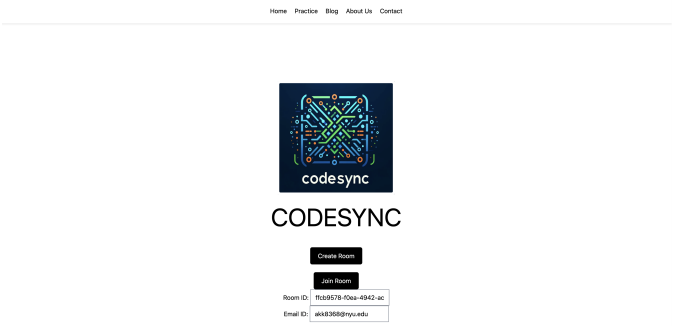

Fig. 3. Home Page

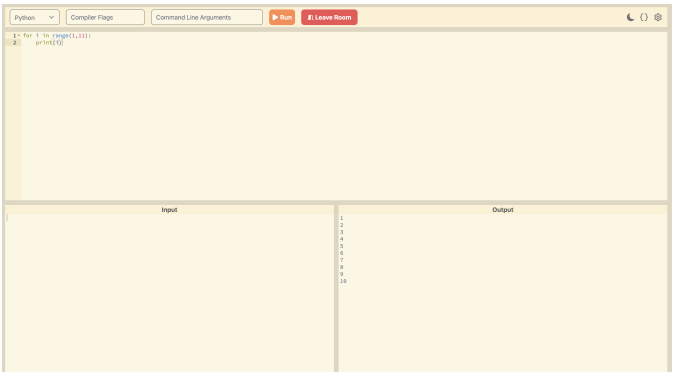

Fig. 4. Code editor

VI. Conclusion and Future Work

A. *Conclusion*

Throughout the development of CodeSync, our team has successfully addressed multiple challenges associated with real-time coding interviews. By integrating technologies such as AWS Cognito, Docker, and Web Sockets, we have created a secure and efficient platform that enhances the interviewing process for both interviewers and candidates. The use of React and Vite for the front-end has ensured that the user interface is both responsive and intuitive, providing a seamless experience during live coding sessions.

B. *Future Work*

Looking forward, there are several enhancements and expansions we aim to implement to further improve CodeSync:

- **Advanced Code Analysis Tools:** Integration of more sophisticated code analysis tools to provide real-time feedback on code quality and efficiency, which can help users improve their coding skills on the spot.
- **Expanded Language Support:** Although currently supporting Python, future versions will include support for more programming languages, broadening the platform's applicability to a wider audience.
- **Mobile Compatibility:** Developing a mobile-friendly version of the platform to allow users to participate in interviews from mobile devices, enhancing accessibility and convenience.
- **AI-Driven Insights:** Implement artificial intelligence algorithms to analyze coding patterns and suggest improvements or predict the difficulty level of coding challenges.
- **Global Scaling:** Focus on optimizing infrastructure to support global scaling, ensuring smooth performance regardless of user location, potentially integrating global CDN solutions for static assets.

These enhancements will not only extend the platform's capabilities but also contribute to a more engaging and comprehensive coding interview experience. Through continuous improvement, we aim to establish CodeSync as the leading tool for tech talent assessment worldwide.