NYU

Final Project
CodeSync

- Karmanya Mendiratta (km6296)
- Suprateek Chatterjee (sc10344)
- Aditya Ojha (ao2612)
- Ananya Kumar Gangver (akk8368)

05.14.24

PART 0

Contents:

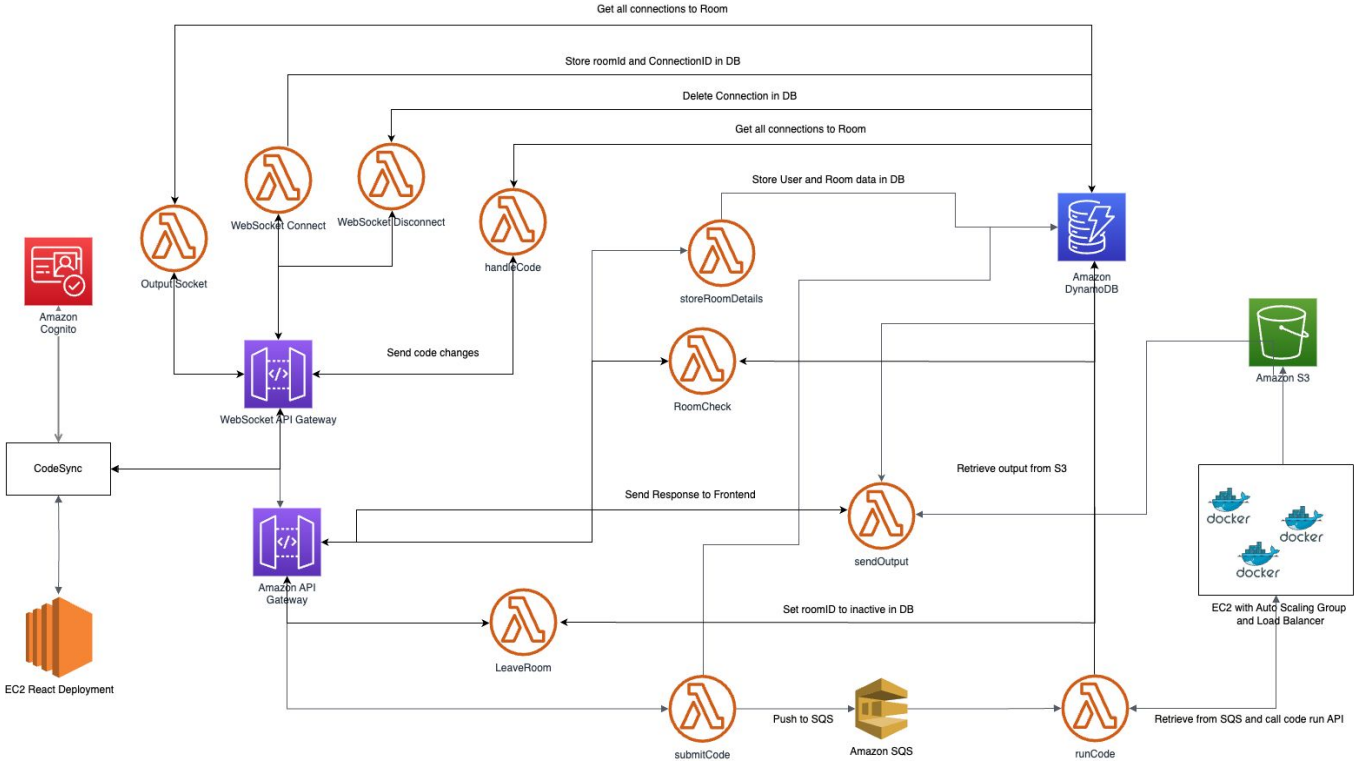
About Codesync

Codesync is a cutting-edge **live coding platform** designed to facilitate real-time collaboration and coding among developers. Users can create rooms where they can code together, with live updates and synchronization enabled through WebSockets.

Key Features:

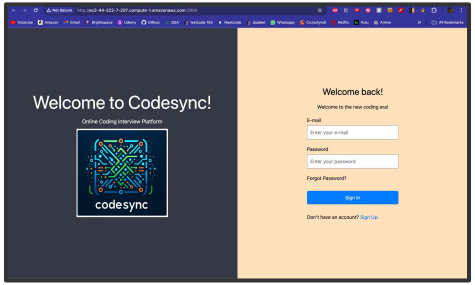
- **Real-time Collaboration:** Seamless live coding sessions where multiple users can code together in real-time. WebSockets (AWS WebSocket API) ensure that all participants in a room see the same code simultaneously, providing an interactive and synchronized coding experience.
- **Secure Code Execution:** Codesandbox isolated environments are used to run code inputted by users. This ensures that code is executed securely and independently, preventing interference and maintaining the integrity of the coding environment.
- **Scalable Infrastructure:** Built on a robust microservice architecture leveraging AWS services for backend and frontend deployment, ensuring high availability and performance.
- **Secure and Efficient:** Utilizes AWS Cognito for user authentication, DynamoDB for data storage, and EC2 instances for frontend deployment.

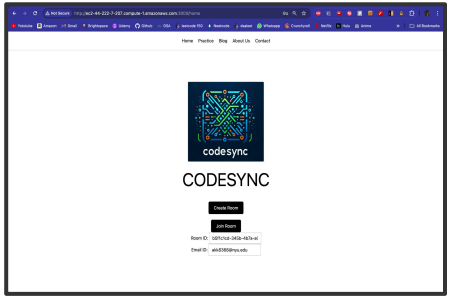
Architecture Diagram



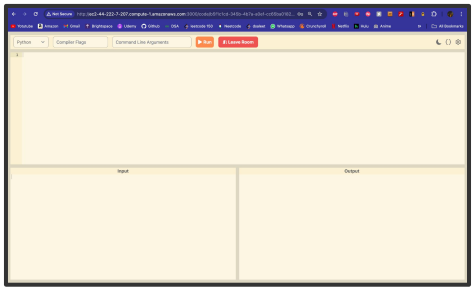
Frontend and AWS Cognito

- The frontend of Codesync is built with **React**, providing a dynamic and interactive user interface. Deployed on **AWS EC2**, it ensures scalability and reliability, allowing the platform to handle increasing user demand. Key features include an intuitive user interface for easy navigation and real-time collaboration using WebSockets, which ensures instant code updates across all participants.





- For user authentication, **AWS Cognito** is utilized to provide secure sign-up, login, and user management. It supports scalable user pools and offers multi-factor authentication for enhanced security. AWS Cognito also integrates seamlessly with other AWS services, ensuring both security and efficiency in managing user identities and access.



NYU

Web Sockets Integration in Rooms

- **User Joins Room:**
  - WebSocket connection established using `CollaborativeIDEWebSocket` method `$connect`
  - Enables real-time collaboration and synchronization
- **Code Updates:**
  - `Updatecode` method broadcasts changes to all connected users
- **User Leaves Room:**
  - `LeaveRoomAPI` triggers `leaveRoom` Lambda function
  - Checks if user is the room owner
  - Closes room session and terminates all WebSocket connections if the user is the owner

How Room Logic is being stored and handled

APIs:

- **Create Room:** The Python Code Processing API triggers the `createRoom` method, which invokes the `store-roomDetails` Lambda function. This function records details in the DynamoDB `Room-details` table, including room ID, creation time, user ID, and active status.
- **Join Room:** When a user wants to join a room, the "`RoomCheck`" API's `check-room` method verifies if the room exists in the `Room-details` table and if it is active. If the checks pass, the user is connected to the room associated with that room ID.

DynamoDB Tables:

- Room-details: Stores information about each room, such as room ID, creation time, user ID, and active status.

Submitted Code Backend Flow

- When user submits a code, the code script is stored in an **S3 bucket** and a signal is sent to **SQS** with **room_id** as the key
- '**code-execution-status**' table is updated in db as "**Pending**"
- **run-code** function polls the SQS queue, calls the code run API and stores the output in an S3 bucket
- 'code-execution-status' table is updated in db as "**Completed**"
- Once the execution status is updated, **send-output** function fetches the output from S3 and sends it to the frontend

Code Run API

- Deployed as a **Flask API** running in a docker container
- **Isolated Sandbox Environment** to ensure security
- User can set code execution timeout
- User can define custom requirements/libraries to include in the code environment
- Configured with an **Auto Scaling Group** and an **Application Load Balancer** for scalability

THANK YOU

NYU