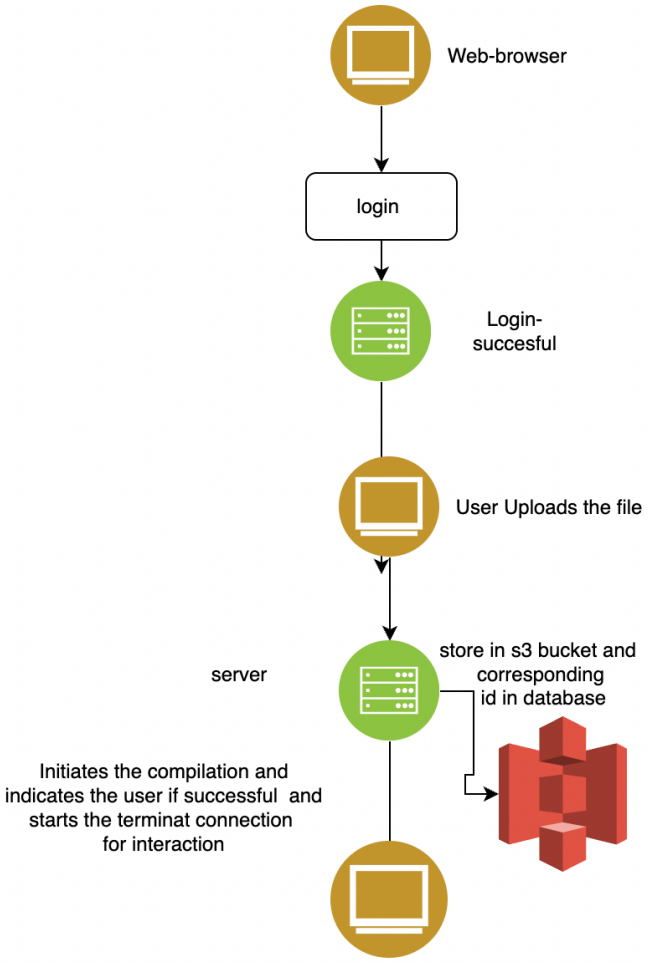
Project Final Report –

Problem Definition –

We propose an online python compiler. This will be a web application and the terminal would be connected to AWS EC2 instance which would be a standard Amazon Linux Machine environment. The front end of the web app would be developed using Angular JS. For backend we would be using Node JS and database as Mongo DB. The user can edit the code, compile, and interact with the final output on the web terminal. User has an option to edit, change, compile and interact with the code. The entire platform would be implemented on EC2 servers. There would security mechanisms such as preventing the user from entering some commands that can expose the backend server.

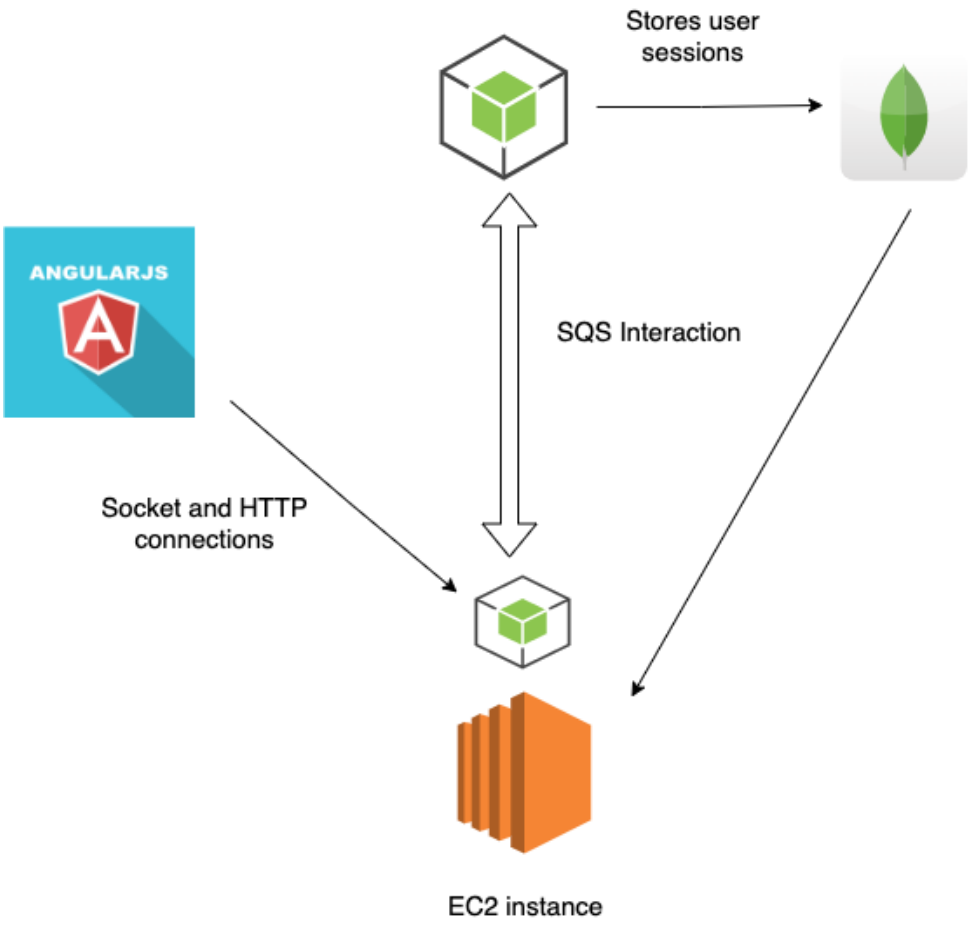
Conceptual design –

Following is the overview of our system's conceptual design –



Implementation description –

The above system has been implemented using various technologies. The front-end part of the system has been implemented using Angular JS. The code in the angular then connects with Node JS in the backend which will in turn use MongoDB for storing the user sessions. Mongo offers a great ease when developing applications using cloud infrastructure. The flow diagram for our system is as follows –



Our main project structure is divided into two separate folders – ui and server folder. First we will talk about the server folder. It has a configuration file called “config.js” where it contains all the configuration details related to the database connection. App.js is the main running file which contains all the file and folder paths required for routing, authentication code, etc. It also has a middleware folder wherein we have placed our authentication service file. Libraries which are required for development are placed in lib folder. The ui folder has the main source folder called “src”. It contains all the files related to user interface like css and html along with all the image files.

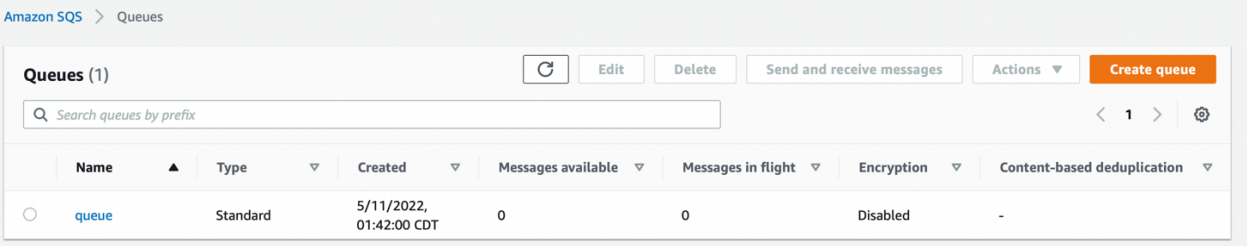
User guide –

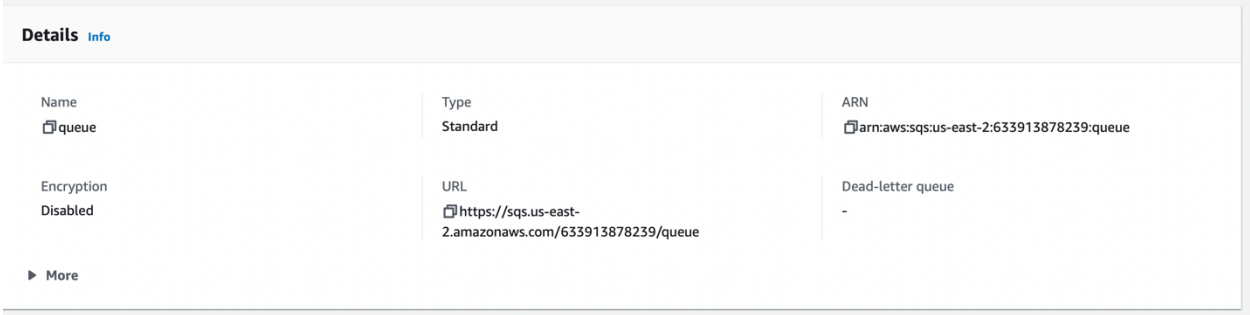
Follow the below steps to run our system. The installation of the individual components would vary depending on the machine you use.

Install MongoDB on your machine
Install latest nodeJS
Install angularJS
Install python with version of 2.7+

Once the prerequisites are installed then we setup our amazon account.

Login to your aws account. If you don't have one, then sign up.
Create a new SQS queue in your account.
Select all the default options for the queue.
After creating the queue, you should be able to see the configuration details as follows.





Start the node server in ui and the server folder located in the main project. Go the desired port and run the system.

Self-Evaluation –

Aditya: I think from the system design perspective we have implemented everything. The main part of the project was to connect the nodeJS to ec2 instance hosted. I think the implementation of SQS concept also proved to be super effective. The non-interactive mode which we implemented also benefits user when interactive mode is down.

Lohith:

I think we have achieved the system design goals. We implemented two parts in our system with an “Interactive version” and “non interactive version. If at any time if the interactive version is not available due to aws services failure or delay/lag, the user has an option to use non interactive mode. This enables a duality and expands the use case to our users.

Future Work scope –

We can make improvement by connecting the containers to ECS and running them. We can also make the docker image to be pushed into ECR. We can add some more security measures such as encrypting and decrypting the messages. Also, we can make architectural changes such as making a new Node server.

References –

https://docs.aws.amazon.com/sqs/index.html
https://faun.pub/aws-simple-queue-service-sqs-how-does-it-work-61d5d41c4b16
https://stackabuse.com/message-queueing-in-node-js-with-aws-sqs/
https://www.bogotobogo.com/DevOps/AWS/aws-Amazon-SQS-Simple-Queue-Service-with-NodeJS-AWS-SDK.php
https://github.com/microsoft/node-pty
https://www.npmjs.com/package/node-pty
https://nodejs.org/en/
https://angular.io/

Contributions from Group Members –

Aditya: Ng terminal, socket connections in frontend, UI and architecture, Login forms, containerizing using docker, Pty services and other frontend backend activities.

Lohith: Sockets in backend, SQS connection, Authentication services for login, Database connections, Pty services, routing and other backend activities