**Computer Engineering Department**

**CX: Cloud Computing**

**COE 453: Cloud and Edge Computing**

**Final Course Project**

**Term 222**

This is the URL of our web application [**https://api-qfxsxjjz5q-ww.a.run.app**](https://api-qfxsxjjz5q-ww.a.run.app) you are welcome to test it. It is running well.

|  |  |
| --- | --- |
| **Project Title** | المجلس |
| **Team Members**  **(Name + ID)** | Ahmed Abdullah Alshehri 201814760  Saud Khalid Alkhulaifi 201837340 |
| **Project Idea** | This project is just a part of a bigger project about a video streaming platform like Twitch. However, to do a possible project we are thinking about making just the live chat of each session (Stream). The application will give an interface for the clients to send and receive messages. The messages will be saved in a database to be retrieved later |
| **My feedback on your course project proposal** | Accepted.  But, the concept is wrong. Chat happens between two parties. The log of chat messages is saved into the database. Remember that your frontend is going to be a single-page application in React. |
| **Services** | |  |  | | --- | --- | | **Service Name** | **Description** | | API | This service will handle all GraphQl requests. It will also handle Rest API calls for posting messages and getting messages based on session ID. Our React application connects to this API using WebSocket to get and send messages. This service also redirects the clients to the React server. | | Get Messages | This service will get all messages for this chat room to the client from the MongoDB server. | | Send Message | This service will send the message of the client to the MongoDB server. | | React server | This service will return the main page of our web application | | MongoDB Server | This service will be connected to the database, and it will store new messages and retrieve messages based on a session ID. In this service the timestamp will be recorded.  The Database server is NoSQL database that will store each message as document in a collection of messages. Each message will have username, message, sessionID (i.e. room number), and a time stamp. | |
| **REST APIs** | Please use the same format as in the Midterm (Check Key Solution). Document all REST endpoints in your architecture (not only the REST endpoints for your services)   |  |  |  | | --- | --- | --- | | **Service** | **Endpoint (HTTP Verb + Path)** | **JSON File** | | API Service | Put /PostUri | {  "uri": “https://postdoc-qfxsxjjz5q-ww.a.run.app”  } | | Put /getUri | {  "uri": “https://getdoc-qfxsxjjz5q-ww.a.run.app”  } | | Get /Messages?sessionID=1 | [  {  "sessionID": 4,  "username": "Mohammed",  "message": "السلام عليكم ",  "timeStamp": 1683884468882  }  ] | | Post /Message | {  "sessionID":2,  "Message":"Hello Message",  "username":"Saud",  "timeStamp": 1683884468882  } | | Get / | Redirect to React app | | Get Messages | Get /?sessionID=1 | [  {  "sessionID": 4,  "username": "Mohammed",  "message": "السلام عليكم ",  "timeStamp": 1683884468882  }  ] | | Post message | Post / | {  "sessionID":2,  "Message":"Hello Message",  "username":"Saud",  "timeStamp": 1683884468882  } | | Mongodb server | Get /?sessionID=1 | [  {  "sessionID": 4,  "username": "Mohammed",  "message": "السلام عليكم ",  "timeStamp": 1683884468882  }  ] | | Post / | {  "sessionID":2,  "Message":"Hello Message",  "username":"Saud",  "timeStamp": 1683884468882  } | |
| **GraphQL API** | |  | | --- | | **Object Type** | | type Message {  username: User  message: String!  session: Int!  timestamp: Float!  }  type Session {  sessionID: Int!  messages: [Message]  } | | **Query Type** | | type Query {  session(sessionID: Int!): Session  } | | **Mutation Type** | | type Mutation {  CreateMessage(username: String, message: String, session: Int, timestamp: Int): Message  } | | **Queries** | | query{  session(sessionID: 2) {  sessionID  messages {  username  message  sessionID  timeStamp  }  }  } | | **Mutations** | | mutation {  message(sessionID: 2, message: "Hello", username: "Graphql") {  username  message  sessionID  timeStamp  }  } | |
| **Workload Characterization** | |  |  | | --- | --- | | **REST Endpoint** | **Average Response Time** | | Get /Messages?sessionID=1 | 200ms | | Post /Message | 1291ms | | **GraphQL** | **Average Response Time** | | query{  session(sessionID: 2) {  sessionID  messages {  username  message  sessionID  timeStamp  }  }  } | 195ms | | mutation {  message(sessionID: 2, message: "Hello", username: "Graphql") {  username  message  sessionID  timeStamp  }  } | 51ms | |

This is our architecture in Google Cloud Provider

