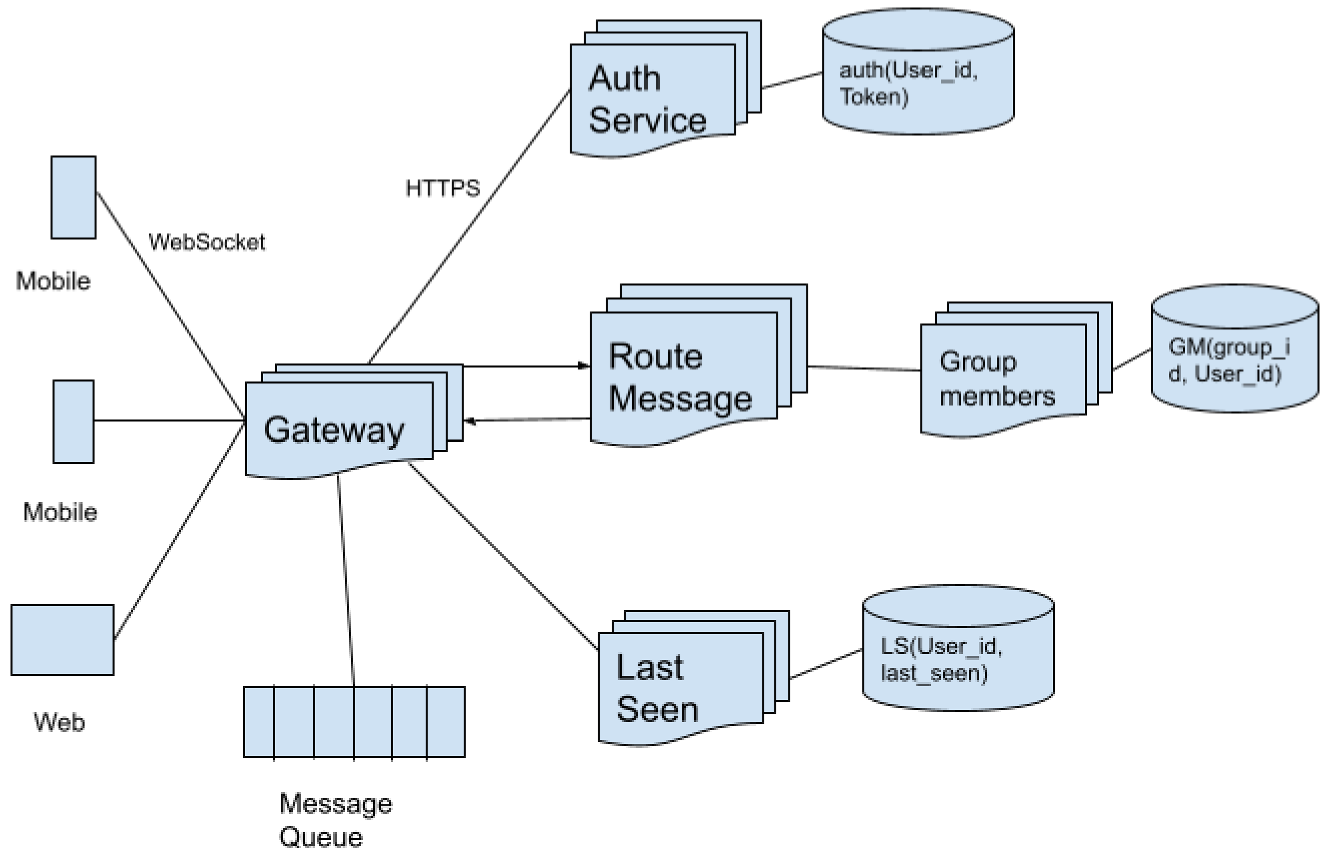
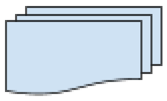
Name: Madhur H. Kadam
Roll No: N20111005

A. Brief information about Chat application

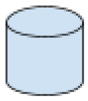
1) PUCSD Chat application is simply a replica of Whatsapp.
2) Features:
    a) Real-time messaging.
    b) Sent, Delivered, Seen receipts.
    c) Last Seen

B. Detailed architecture diagram





: Microservice



: Database

Details:

1) There will be multiple Microservices which will handle different features.
2) Gateway Microservice:
    a) Gateway and Web/Mobile devices will have **WebSocket** connection between them.
    b) Gateway and other microservices will have **HTTPS** connection between them.
    c) Gateway will receive different types of requests according it will use other microservices.
3) Route Message Microservice:
    a) Once Gateway receives a message from Client, it will use Route Message microservice to resolve the address of message receiving client.
4) Group members Microservice:
    a) This microservice is used by Route message microservice.
    b) When a user sends a message in a group, then Group microservice will return a list of Users of that group.
5) Auth Microservice:
    a) This is used to perform user Login, Logout, SignUp.
    b) It assigns a unique Token to every client for every session.
6) Last Seen Microservice:
    a) This microservice will keep track of the last seen time of the user.
    b) Last seen will be updated for every request or response from that user.
    c) This microservice will also check if the user is Online or Not through Websocket connection and accordingly update last seen.
7) Message Queue:
    a) If the receiving user of the message is not Online then temporarily the message will be stored in Message Queue.
    b) Once a user sends a request for Websocket connection, then messages in Message Queue related to that user will be sent to him.
8) Sent, Delivered, Seen receipts:
    a) These receipts will be treated as normal messages.
    b) Once the message is received by the gateway it will send a SentReceipt to Sender.
    c) Once the Receiver receives the message it will send a DeliveredReceipt to gateway, then gateway will route that to sender and same for SeenReceipt.

C) Technology/Tools Stack Used:

1) Back-End: Node.js, Express.js, WebSockets.
2) Front-End: Web: React, Mobile: React-Native
3) DevOps:
   i) Microservices management: Docker, Kubernetes, kubernetes ingress controller.
   ii) CI/CD: Github, Github Actions.
4) Database: MongoDB
5) Host: AWS

D) Log Management and Audition Details:

1) We will keep a log of messages being sent. (logs should not contain message data)
2) Thus in peak hours we can take required actions.
3) We will also keep track of frequency of messages for every user, thus we can observe logs and can determine if someone is just continuously spamming messages.
4) There can be a Bot sending frequent messages and using too much server resources, thus we need to keep analysing logs and find such activity and need to Block Bots account.

E) Incident Management in Chat application:

1) If due to some error Gateway is not able to redirect a message to the receiver then it will keep the message in Message Queue and will retry sending the message after some time.
2) Delivered receipts are used to determine if the receiver received that message or not.

F) How your chat application will manage load during peak hours/events in the society:

1) This app's architecture consists of multiple microservices.
2) It is designed as Horizontally scalable.
3) Microservices are Created using Node.js and Express.js.
4) These will be docker containers.
5) Kubernetes will perform Orchestration of these containers.
6) Kubernetes ingress controller is used for Load-Balancing for microservices.
7) App will be Scale Up or Scale down automatically according to requirement.
8) For very high load we can turn off Read, Delivered, Seen receipt and Last seen feature as these are low priority features.