

Team Project Sprint 3

Course Number: CPS 592

Course Name: Cloud Computing and Applications

Instructor: Dr.Ahmed EI Ouadrhiri

ID: 101740195

Name: Usha Sankalamaddi

Email: sankalamaddiu1@udayton.edu

ID: 101739921

Name: Harika Kunaparaju

Email: kunaparajus1@udayton.edu

1. Introduction:

In Sprint1 we have implemented microservice development and cloud deployment on Azure and Heroku. In microservice1 we have created a web app service in Azure. In microservice2 we have installed heroku .

First we have developed a code in Express.js that takes input as a year and gives the age and Chinese zodiac animal and deployed it on microservice1. Microservice1 here is we have created a new web app service in azure and deployed to it.

Next we have developed a code in Express.js that takes two dates as input and calculates the no of days between two dates and no of weeks between the two dates and deployed it on microservice2. Here microservice2 is Heroku and deployed to it.

Lastly we have developed a front-end UI(that uses HTML and CSS) which includes both microservice1 and microservice2 which will invoke both the services from Azure and Heroku and displays the output.

URL of Microservice1 (Azure): <https://cca-sankalamaddiu1-kunaparajus1.azurewebsites.net/>

URL of Microservice2 (Heroku): <https://cca-team-8microservice2.herokuapp.com/>

URL of Front-end UI: <https://cca-sankalamaddiu1-kunaparajus1.bitbucket.io/>

In Sprint2 we have reused the code from individual assignment and the main requirements in this sprint is to:

- i) Display the current and list of online (authenticated) users.
- ii) Private chats with authenticated users.
- iii) Displaying each private chat in separate chat windows.

Firstly we have reused the login and Signup microservices code from individual assignment, and created a new database for the team project and we have changed the format of returning the message when the user is authenticated.

Then we have successfully deployed those microservices on azure web services.

Secondly, we also reused the chat server code from the individual assignment and implemented the list of authenticated users and private chats with authenticated users. Then we have deployed the code in azure web services.

URL of Login Microservice: <https://cca-sankalamaddiu1-kunaparajus1-accountservices.azurewebsites.net/login/username/password>

URL of Signup Microservice: <https://cca-sankalamaddiu1-kunaparajus1-accountservices.azurewebsites.net/signuptest>

In Sprint3 we have reused the code from sprint2 and extended the functionality of storing the private chats and implemented a chatbot as well.

Firstly, we have stored the users private chat history in the mongodb database as a collection and then retrieve the history for the user when requested again by the user.

Then we have implemented a chatbot which can respond automatically to the messages sent by the user. For the chatbot we have reused the two microservices from sprint1.

2. Design and Implementation:

In Sprint1 Front-end UI includes both Microservice1 and Microservice2.

For Microservice1 we have taken one input field for the year and a button to call Microservice1. So the Microservice1 is called with the input value. Code is implemented in Express.js and we have enabled CORS. Then API Age_ChineseZodiac is invoked and from the query input year is taken and corresponding age and animal is calculated. Then the result from the call is displayed as output.

For Microservice2 we have taken two input fields for two dates and a button to call Microservice2. So the Microservice2 is called with the two input values. Code is implemented in Express.js and we have enabled CORS. Then API datediff is invoked and from the query two dates is taken as input and no of days and weeks is calculated. Then the result from the call is displayed as output.

In Sprint2 firstly, we have reused the login and Signup microservices code from individual assignment. For two microservices we have developed the code in Express.js where in get method is for the Login and post method for signup. Using get method we are using the user details and retrieving the user details if the user is already present. But here we have changed the format of retrieving the users details and we have created a new database for the team project. And using the post method we have successfully inserting the new user details in to the database. Then we have deployed the two microservices code in the azure web services.

Secondly, we have reused the chat server code from individual assignment and we have implemented in the front end UI, when the unauthorized user tries to login then we are prompting the user as unauthorized. And also whenever we click on login we have also implemented the loading functionality, it will display like “please wait” and then if the user is authenticated it will open the chat server page. Also, when the new user tries to register he should give the email that ends with “@gmail.com” otherwise an alert will pop up.

We have also used some pre-existing libraries and displayed the list of online authorized users and displays the welcome message whenever the user is authenticated. If the connected users wants to chat privately, if they click on the user then a chat window will pop up for that user to chat. We have also developed a logout button where in the user returns to the login page if they clicks it.

Then I have successfully deployed the code on azure web services, where in the link to access it is:

<https://cca-teamproject-sprint2.azurewebsites.net/>

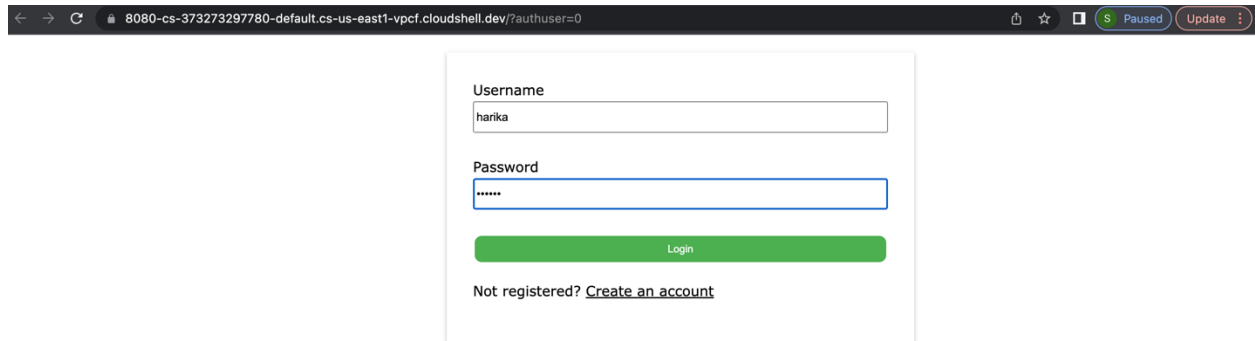
In Sprint3, we have reused the code from sprint2 and extended the functionality of storing the chat history in the database and loading the history whenever requested by the user. We have implemented two APIs functions. One is to store a private message and the other is to retrieve the private chat message.

Then we have integrated a chatbot into the chat system. A chatbot is an intelligent program that automatically replies to a message. For this chatbot implementation we have reused the two microservices developed in sprint1 Age and Chinese zodiac animal service and Days and weeks Service. Where in the chatbot will call these microservices accordingly and return the result to the chatbot UI automatically. For example, if the message contains the year then it returns the age and zodiac sign according to the year. If the message contains two dates then it returns the difference between the dates.

Then we have deployed this Sprint in Azure where the accessible link is:

<https://cca-project-sprint3.azurewebsites.net>

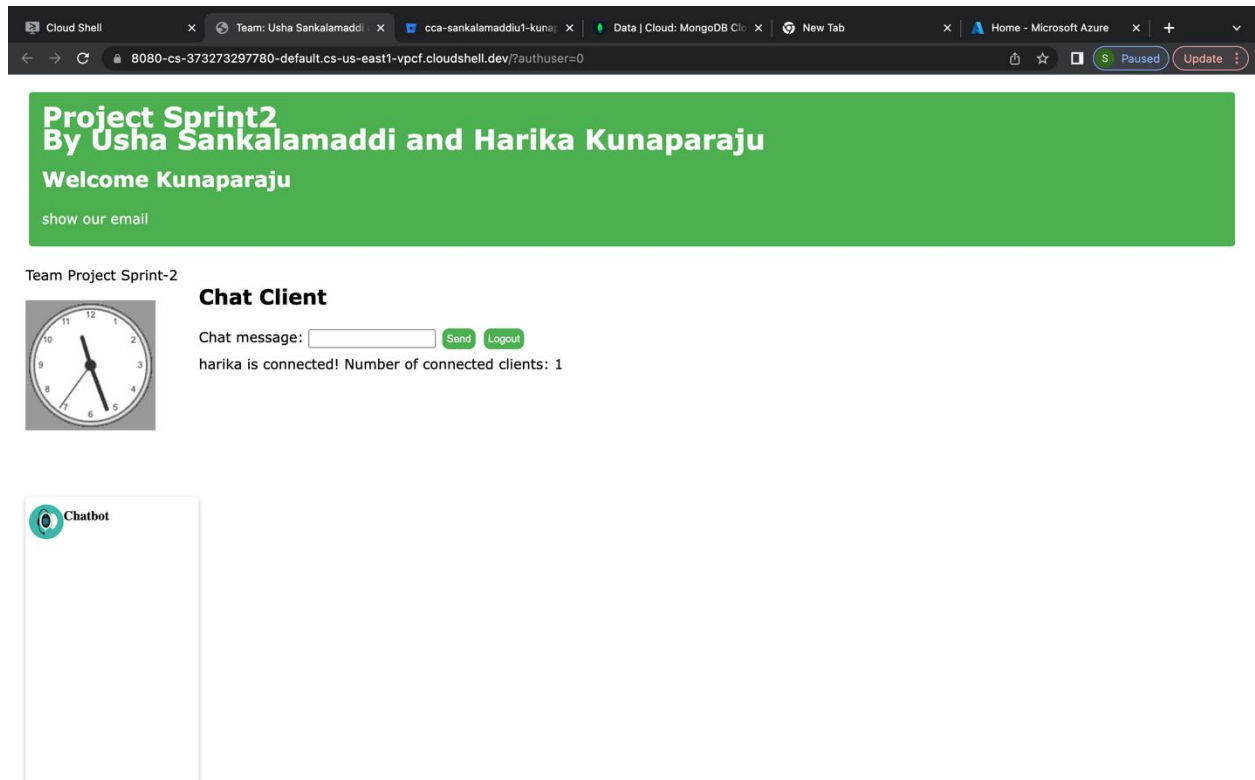
3. Demo (Screenshots):



A screenshot of a web browser showing a login form. The browser's address bar displays the URL: 8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev/?authuser=0. The login form is centered and contains the following elements:

- A "Username" label above a text input field containing the text "harika".
- A "Password" label above a password input field containing six asterisks "*****".
- A green "Login" button.
- A link below the button that reads "Not registered? [Create an account](#)".

Here the user tries to login with the login credentials.



A screenshot of a web application interface after successful login. The browser's address bar shows the same URL as the previous screenshot. The page features a large green header with the text "Project Sprint2 By Usha Sankalamaddi and Harika Kunaparaju" and "Welcome Kunaparaju". Below the header, there is a "show our email" link. The main content area is titled "Team Project Sprint-2" and "Chat Client". It includes a clock icon, a "Chat message:" input field, and "Send" and "Logout" buttons. Below the input field, it displays the message "harika is connected! Number of connected clients: 1". At the bottom, there is a "Chatbot" section with a circular icon and a text input field.

Then if the user is authenticated then he will be able to see the chat server page.

Create an account'."/>

8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev/?authuser=0

Paused Update

Username
usha

Password

Login

Not registered? [Create an account](#)

Here another user is trying to login in the system with login credentials.

Cloud Shell Team: Usha Sankalam Team: Usha Sankalam cca-sankalamaddi Data | Cloud: MongoDB New Tab Home - Microsoft Az

8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev/?authuser=0

Paused Update

Project Sprint2
By Usha Sankalamaddi and Harika Kunaparaju
Welcome reddy
show our email

Team Project Sprint-2

Chat Client

Chat message: Send Logout

usha is connected! Number of connected clients: 2

Kunaparaju
Chatbot

Then if the user is authenticated he will be able to the list of users connected.

Cloud Shell | Team: Usha Sankalam | cca-sankalamaddi1 | Data | Cloud: Mongo | New Tab | Home - Microsoft Az | 8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev?authuser=0


Project Sprint2

By Usha Sankalamaddi and Harika Kunaparaju

Welcome Kunaparaju

show our email


Team Project Sprint-2



Chat Client

Chat message: Send Logout

usha is connected! Number of connected clients: 2



reddy

You said:

Hi usha

Here user1 tries to chat privately with the user2.

Cloud Shell | Team: Usha Sankalam | cca-sankalamaddi1 | Data | Cloud: Mongo | New Tab | Home - Microsoft Az | 8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev?authuser=0


Project Sprint2

By Usha Sankalamaddi and Harika Kunaparaju

Welcome reddy

show our email


Team Project Sprint-2



Chat Client

Chat message: Send Logout

usha is connected! Number of connected clients: 2



Kunaparaju

Kunaparaju said:

Hi usha

You said:

hello harika

Then user2 receives the message and responds to user1

Cloud Shell Team: Usha Sankala... cca-sankalamaddu1... Data | Cloud: Mongo... New Tab Home - Microsoft Az... 8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev?authuser=0

Project Sprint2

By Usha Sankalamaddi and Harika Kunaparaju

Welcome Kunaparaju


show our email

Team Project Sprint-2

Chat Client

Chat message:

usha is connected! Number of connected clients: 2



reddy

You said:

how are you doing?

reddy said:

am doing good. what about you?

You said:

pretty good

Some discussion is happening between the users.

Cloud Shell Team: Usha Sankala... cca-sankalamaddu1... Data | Cloud: Mongo... New Tab Home - Microsoft Az... cloud.mongodb.com/v2/636d93c99ae2bf41038031c3#metrics/replicaSet/636d95cdc86b384b8b9b5805/explorer/chatclient/chat-history/find

Atlas University of... Access Manager Billing All Clusters Get Help cca-lab1

Project 0 Data Services App Services Charts

+ Create Database

Search Namespaces

chatclient

chat-history

chat-users

chatclient.chat-history

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.24KB TOTAL DOCUMENTS: 21 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

```
{
  "_id": ObjectId('6391686344a1e0da6ffa7675'),
  "sender": "harika",
  "receiver": "usha",
  "message": "Hi usha",
  "time": 2022-12-08T04:30:27.449+00:00
}
```

```
{
  "_id": ObjectId('6391687b44a1e0da6ffa7676'),
  "sender": "usha",
  "receiver": "harika",
  "message": "hello harika",
  "time": 2022-12-08T04:30:51.094+00:00
}
```

1-20 of many results

System Status: All Good

©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Then the chat history is stored successfully in mongoDB database.

Cloud Shell

Team: Usha Sankala

Team: Usha Sankala

cca-sankalamaddi1

Data | Cloud: Mongo

New Tab

Home - Microsoft Az

8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev?authuser=0

Paused

Update


Project Sprint2

By Usha Sankalamaddi and Harika Kunaparaju

Welcome reddy

[show our email](#)


Team Project Sprint-2




Chat Client

Chat message: [Send](#) [Logout](#)

harika is connected! Number of connected clients: 2



Kunaparaju



Chatbot

Kunaparaju

Kunaparaju said:

how are you doing?

You said:

am doing good. what about you?

Kunaparaju said:

pretty good

If the users disconnected and tries to login again then they can be able to see their chat history

Cloud Shell

Team: Usha Sankala

Team: Usha Sankala

cca-sankalamaddi1

Data | Cloud: Mongo

New Tab

Home - Microsoft Az

8080-cs-373273297780-default.cs-us-east1-vpcf.cloudshell.dev?authuser=0

Paused

Update


Project Sprint2

By Usha Sankalamaddi and Harika Kunaparaju

Welcome Kunaparaju

[show our email](#)


Team Project Sprint-2




Chat Client

Chat message: [Send](#) [Logout](#)

usha is connected! Number of connected clients: 2



reddy



Chatbot

Chatbot

Don't Understand

You said:

I born in 1998

Chatbot said:

you are 24 years old and you are a Tiger

reddy

You said:

how are you doing?

reddy said:

am doing good. what about you?

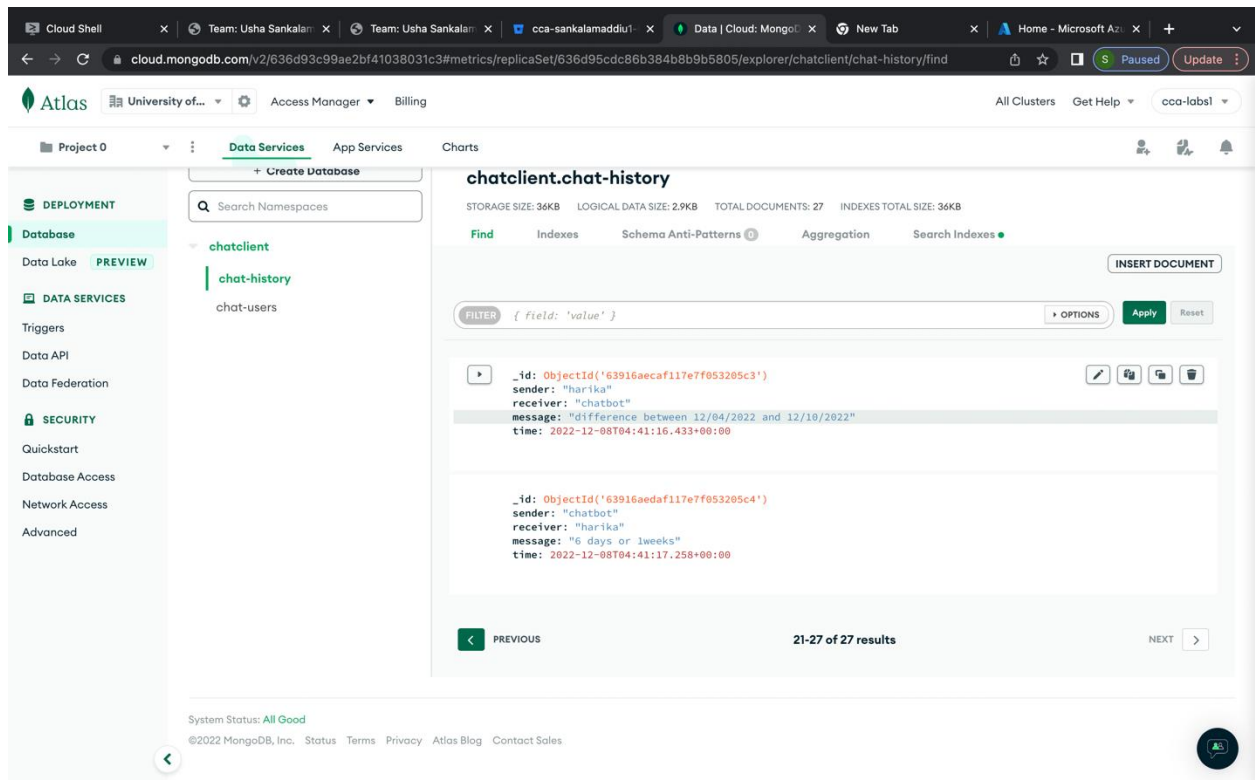
You said:

pretty good

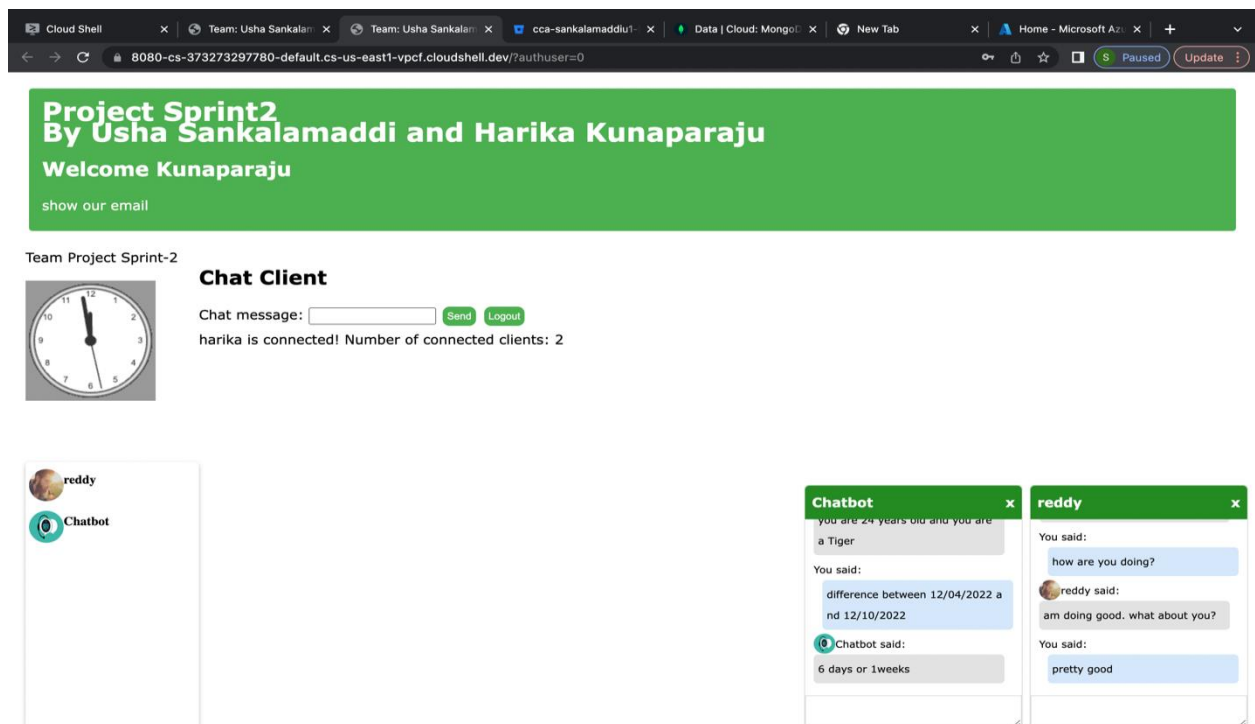
Then the user is interacting with chatbot, if he enters the year then it automatically calls the first microservice in sprint1 and returns the result as age and zodiac animal.

The screenshot shows a web browser window with multiple tabs. The active tab is titled 'cca-sankalamaddu1'. The address bar shows a URL from 'cloudshell.dev'. The main content area has a green header with the text 'Project Sprint2 By Usha Sankalamaddi and Harika Kunaparaju' and 'Welcome Kunaparaju'. Below this is a 'show our email' link. The main section is titled 'Team Project Sprint-2' and 'Chat Client'. It features a clock icon, a 'Chat message:' input field with 'Send' and 'Logout' buttons, and a status message 'usha is connected! Number of connected clients: 2'. On the left, there is a list of users: 'reddy' and 'Chatbot'. On the right, there are two chat windows. The 'Chatbot' window shows a conversation where the user asks for the difference between two dates, and the chatbot responds with '6 days or 1weeks'. The 'reddy' window shows a conversation where the user asks 'how are you doing?' and the chatbot responds 'am doing good. what about you?'. The user then responds 'pretty good'.

Here if the user gives two dates then it automatically calls the microservice2 from sprint1 and returns the result as difference between them.

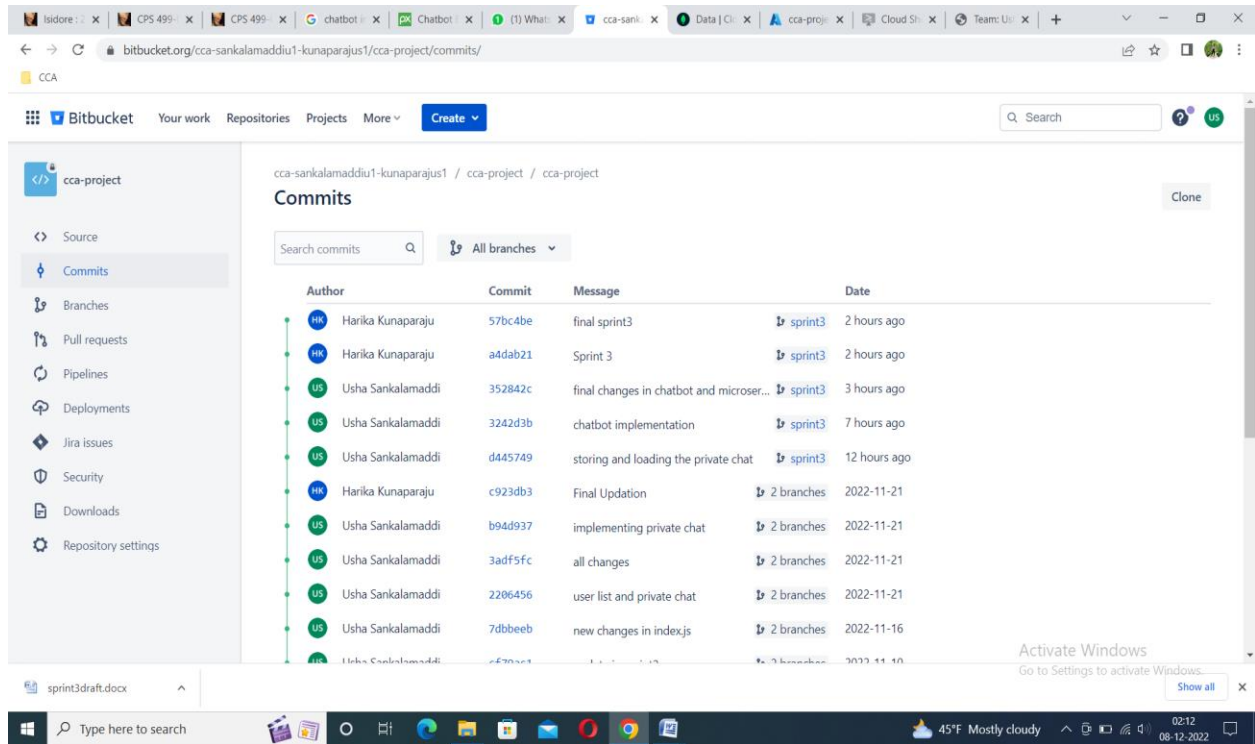


Here we can see all the chat history with the chatbot is also stored in the database.



Here if the users are disconnected and tried to connect again they can see the chat history of chatbot as well.

Appendix A- Team Contributions



Here are the latest commits in sprint3 branch in cca-project repository.

Appendix B - Source Code

Index.js

```
const express = require('express')
const app = express()
const cors = require('cors')
const axios = require('axios');
const chatdb = require('./chatdb.js');
var port = process.env.PORT || 8080;
app.use(express.static('static'))
app.use(express.urlencoded({extended: false}))
app.use(cors())

const http = require('http').Server(app).listen(port);

app.get('/', (req, res) => {
```

```

    res.sendFile( __dirname + '/static/chatclient/index.html')
  })

const io = require('socket.io')(http);
console.log('socket.io server is running on port ' + port);
app.get('/', (req, res) => {
  res.sendFile( __dirname + '/static/chatclient/index.html')
})

io.on('connection', socketclient => {
  console.log('A new client is connected!');
  socketclienthandler(socketclient);
})

function login(username, password, callback) {
  axios.get('https://cca-sankalamaddiul-kunaparajus1-
accountservices.azurewebsites.net/login/' + username + '/' + password)
    .then(response => {
      // console.log("Response from
microservice:" + JSON.stringify(response.data,null,2));
      var status=response.data&&response.data.status;
      if(status=='authenticated')
      {
        return callback(true,null,response.data.profile);
      }
      else
      {
        return callback(false,status,null);
      }
    });
}

function register(data, callback) {
  axios.post('https://cca-sankalamaddiul-kunaparajus1-
accountservices.azurewebsites.net/signup', {
    username: data.username,
    password: data.password,
    fullname: data.fullname,

```

```

        email: data.email
    })
    .then(response => {
        if (response.data) {
            return callback(response.data);
        } else {
            return callback();
        }
    });
}

function onlineClients() {
    var olClientsCount = 0;
    var onlineClients = io.sockets.sockets;
    for (var id in onlineClients) {
        const sClient = onlineClients[id];
        if (sClient && sClient.authenticated) {
            olClientsCount++;
        }
    }
    return olClientsCount;
}

function BroadcastAuthenticatedClients(event,message) {
    var sockets = io.sockets.sockets;
    for (var id in sockets) {
        const socketclient = sockets[id];
        if (socketclient && socketclient.authenticated) {
            socketclient.emit(event,message);
        }
    }
}

function authenticatedClients(){
    var sockets=io.sockets.sockets;
    let onlineUsers=new Map();
    for(var id in sockets){
        const socketclient=sockets[id];

```

```

        if(socketclient && socketclient.authenticated && socketclient.profile){
            onlineUsers.set(socketclient.username, socketclient.profile);
        }
    }
    return onlineUsers;
}

function authenticatedClientsJSON(onlineUsers ){
    let onlineUsersJSON=[];
    onlineUsers.forEach((value,key) => {
        onlineUsersJSON.push(value);
    });
    return onlineUsersJSON;
}

function socketclienthandler(socketclient) {
    socketclient.on("login", (username, password) => {
        console.log(`Listening login:: ${username}, ${password}`);
        login(username, password, (authenticated, message, account) => {
            if (authenticated) {
                socketclient.authenticated = true;
                socketclient.username =username;
                socketclient.password=password;
                socketclient.profile=account;
                console.log(`${username} is authenticated`);

                const connectedClients=authenticatedClients();
                const
connectedClientsJSON=authenticatedClientsJSON(connectedClients);

                var chatbot = {username: "chatbot", fullname: "Chatbot", avatar:
'./chatbot-4071274_960_720.jpg'};
                connectedClientsJSON.push(chatbot);

                socketclient.emit('authenticated',account,connectedClientsJSON);
                BroadcastAuthenticatedClients('updateUserList',
connectedClientsJSON);

                var welcomeMessage = `${socketclient.username} is connected!
Number of connected clients: ${onlineClients()}`;

```



```

        BroadcastAuthenticatedClients('online', welcomeMessage);
    }
    else
    {
        console.log(`${username} is not authenticated`);
        socketclient.emit("loginfailed", message);
    }
    });
});

socketclient.on("privateChat", (receiver, message) => {
    var sockets = io.sockets.sockets;
    const sender = socketclient.username;

    if (receiver === 'chatbot') {
        const twoDatesRegex = new RegExp(/[0-9]{2}V[0-9]{2}V[0-9]{4}\s\w+\s[0-9]{2}V[0-9]{2}V[0-9]{4}/);
        const yearRegex = new RegExp(/\d{4}/);

        if (twoDatesRegex.test(message)) {
            let pattern = new RegExp(/[0-9]{2}[V][0-9]{2}[V][0-9]{4}/);
            let scpattern = new RegExp(/[0-9]{2}[V][0-9]{2}[V][0-9]{4}$/);

            var firstDt = pattern.exec(message)[0];
            var scndDt = scpattern.exec(message)[0];
            chatdb.storePrivateMessage({ sender, receiver, message });
            axios.get('https://cca-sankalamaddiul-kunaparajus1-microservice2.azurewebsites.net/datediff?from_date=' + firstDt + '&to_date=' + scndDt)
                .then(response => {
                    chatdb.storePrivateMessage({ sender: 'chatbot', receiver: sender, message: response.data });
                    socketclient.emit("privateChat", 'chatbot', response.data);
                });
        }
        else if (yearRegex.test(message)) {
            var yr = yearRegex.exec(message)[0];
            chatdb.storePrivateMessage({ sender, receiver, message });
            axios.get('https://cca-sankalamaddiul-

```

```

kunaparajus1.azurewebsites.net/Age_ChineseZodiac?year=' + yr)
    .then(response => {
        chatdb.storePrivateMessage({sender: 'chatbot', receiver: sender,
message: response.data});
        socketclient.emit("privateChat", 'chatbot', response.data);
    })
    .catch(e => {
        console.log(e);
        return e;
    });
}
else
{
    chatdb.storePrivateMessage({sender, receiver, message});
    var res = "Don't Understand";
    chatdb.storePrivateMessage({sender: 'chatbot', receiver: sender,
message: res});
    socketclient.emit("privateChat", 'chatbot', res);
}
}
else
{
    for (var id in sockets) {
        const socketclient = sockets[id];
        if (socketclient && socketclient.authenticated && socketclient.username
== receiver) {
            chatdb.storePrivateMessage({sender,receiver,message});
            socketclient.emit("privateChat", sender, message);
        }
    }
}
});

socketclient.on("register", (username, password, fullname, email) => {
    console.log(`register:: ${username}, ${password}`);
    register({username, password, fullname, email}, (data) => {
        if (data.msg) {
            io.emit("regform");
        } else {

```

```

        io.emit("errsignup", data.err);
    }

    });
});

socketclient.on("disconnect", () => {
    if (socketclient.authenticated) {
        var byeMessage = `${socketclient.username} is disconnected!! Number of
connected clients: ${onlineClients()}`;
        console.log(byeMessage);

        const connectedClients=authenticatedClients();
        const
connectedClientsJSON=authenticatedClientsJSON(connectedClients);

        BroadcastAuthenticatedClients('online',byeMessage);
        BroadcastAuthenticatedClients('updateUserList', connectedClientsJSON);
    }
});

socketclient.on("message", data => {
    if (socketclient.authenticated) {
        console.log("data from a client: " + data);
        BroadcastAuthenticatedClients("message", data);
    }
});

socketclient.on("typing", data => {
    if (socketclient.authenticated) {
        var userName = socketclient.username;
        var sockets = io.sockets.sockets;
        for (var id in sockets) {
            const socketclient = sockets[id];
            if (socketclient && socketclient.authenticated && socketclient.username
!= userName) { // excluding one who is typing
                socketclient.emit("typing", `${socketclient.username} is typing ...`);
            }
        }
    }
}

```

```

    }
  });

  socketclient.on("loadchat", (receiver) => {
    //console.log('999999')
    const sender = socketclient.username;
    chatdb.loadPrivateMessage(sender, receiver, (chathis) => {
      //console.log('aaa ', chathis);
      if (chathis.length > 0) {
        var sockets = io.sockets.sockets;
        for (var id in sockets) {
          const socketclient = sockets[id];
          if (socketclient && socketclient.authenticated &&
socketclient.username == sender) {
            socketclient.emit("sendchathis", chathis, sender);
          }
        }
      } else {
        socketclient.emit("sendchathis", null, sender);
      }
    })
  });
}

```

Chatdb.js

```

const MongoClient = require('mongodb').MongoClient;
const mongourl = "mongodb+srv://cca-labs1:harikausha@cca-sankalamaddiu1-
kuna.yzjrgop.mongodb.net/chatclient?retryWrites=true&w=majority";
const dbClient = new MongoClient(mongourl, {useNewUrlParser: true,
useUnifiedTopology: true});
dbClient.connect(err => {
  if (err) throw err;
  console.log("Connected to the MongoDB cluster!");
});
const db = dbClient.db();

module.exports.storePrivateMessage = (privatechat) => {

```

```

const {sender, receiver, message} = privatechat;
const chatHis = {sender, receiver, message, time: new Date()};

db.collection("chat-history").insertOne(chatHis, (err, result) => {
  if (err) {
    console.log("Error: While inserting a chat history" + err);
    return 0;
  } else {
    console.log("Inserted chat history into chatmsgs collection...!")
    return 1;
  }
});
}

module.exports.loadPrivateMessage = async (sender,receiver, callback) => {
  //console.log('hhhh');
  const msgs = await db.collection("chat-history").find({$or:[
    {sender: sender, receiver: receiver},
    {sender: receiver, receiver: sender}
  ]}).sort({time:1}).limit(10).toArray();
  return callback(msgs);
}

```

Chatbox.js

```

if(!${}){
  alert("This chatbox UI needs jQuery library, please include it and try again!");
}
var currentUserListPositionID = currentUserListPositionID || 'menubar';
var currentUserList = [];
activeUserList = [];
var currentUser="";

function setCurrentUser(user) {
  if(!user || !user.username){
    //Only the current user, then no other users is online
    chatboxAlert('Developers: You must provide a user account in a JSON object
of: {username:"", fullname:"",avatar:""}');

```

```

    return;
}
currentUser= user.username;
}

function displayUserList(userlist){
    currentUserList=userlist;
    renderUserList();
};

function displaySelfMessage(to_user,message){
    var user_chatbox = $("#chatbox_" + to_user);
    if (message.trim().length != 0) {
        var rel = $(user_chatbox).attr("rel");
        $('<span>You said: </span><div class="msg-right">' + message +
        '</div>').insertBefore('[rel="' + rel + "]" .msg_push');
        //Auto scroll the messages
        $('.msg_body').scrollTop($('.msg_body')[0].scrollHeight);
    }
}

function displayNoChatHistory(to_user){
    var user_chatbox = $("#chatbox_" + to_user);
    var rel = $(user_chatbox).attr("rel");
    $('<div>No chat history earlier.</div>').insertBefore('[rel="' + rel + "]"
    .msg_push');
    //Auto scroll the messages
    $('.msg_body').scrollTop($('.msg_body')[0].scrollHeight);
}

function displayFriendMessage(username, message) {
    //If chat box has not been displayed. Display
    var friendFullname;
    var friendAvatar;
    //Find user in list
    for (var idx in currentUserList) {
        user = currentUserList[idx]
        if (user.username == username) {
            friendFullname = user.fullname || user.username;
            friendAvatar = user.avatar || 'https://i.pravatar.cc/150?u='+username;

```

```

        break;
    }
}
if ($("#chatbox_" + username).length == 0) {
    renderChatbox(username, friendFullname);
    rearrangeChatbox();
    trigger_requestChatHistory(username); //version 2
    return;
}
var user_chatbox = ($("#chatbox_" + username);

if (message && message.trim().length != 0) {
    var rel = $(user_chatbox).attr("rel");
    $('<span>' + friendFullname + ' said: </span><div class="msg-left">' + message +
'</div>').insertBefore('[rel="' + rel + '"].msg_push');
    //Auto scroll the messages
    $('<div class="msg-body">').scrollTop($('<div class="msg-body">')[0].scrollHeight);
}
}

function updateUserList(userlist) {
    displayUserList(userlist);
    if(!activeUserList || activeUserList.length == 0){
        return;
    }
    $.each(activeUserList, function (index, username) {
        if ($("#chatbox_" + username).length == 1 && !currentUserList[username]) {
            removeChatbox(username);
        }
    });
}

/**
 * This function will render Friend List on Side Bar
 */

function renderUserList() {
    $("#chat-sidebar").empty();

```

```

    //alert("chatbox: " + currentUserList);
    if(!currentUserList || currentUserList.length == 0 ||
!currentUserList[0].username){
        //Only the current user, then no other users is online
        chatboxAlert('Developers: You must provide a userlist in a JSON array of:
[{"username":""," fullname":"","avatar":""},...]');
        return;
    }
    if(!currentUserList || currentUserList.length == 1){
        //Only the current user, then no other users is online
        $("#chat-sidebar").append('<span>You are the only online user</span>');
        clearAllChatbox();
        return;
    }
    for (var idx in currentUserList) {
        const user = currentUserList[idx];
        //console.log(currentUser)
        if(user.username && user.username != currentUser){
            const avatar = user.avatar || 'https://i.pravatar.cc/150?u='+user.username;
            const fullname = user.fullname || user.username;
            const newusr = '<div id="sidebar-user-box" class="' + user.username + ">'
+
            '' +
            '<span id="' + user.username + ">' + fullname + '</span>' +
            '</div> '
            $("#chat-sidebar").append(newusr);
        }
    }
}

/**
 * This function will render a chatbox with a specific username
 */
function renderChatbox(username, fullname) {
    if ($("#chatbox_" + username).length == 0) {
        chatPopup = '<div class="msg_box" style="right:270px" id="chatbox_' +
username + '" rel="' + username + ">' +
            '<div class="msg_head">' + fullname +
            '<div class="close">x</div> </div>' +

```



```

        '<div class="msg_wrap"> <div class="msg_body"> <div
class="msg_push"></div> </div>' +
        '<div class="msg_footer"><textarea id="message" class="msg_input"
rows="2"></textarea></div></div> </div>';
        $("body").append(chatPopup);
        activeUserList.push(username);
    }else{
        $("#chatbox_" + username).show();
    }
}

function clearAllChatbox() {
    if(!activeUserList || activeUserList.length == 0){
        return;
    }
    $.each(activeUserList, function (index, username) {
        var chatbox = $("#chatbox_" + username);
        if (chatbox != null) {
            chatbox.remove();
            //Remove from active List
            var rel = chatbox.attr('rel');
            const index = activeUserList.indexOf(rel);
            if (index > -1) {
                activeUserList.splice(index, 1);
            }
        }
    });
}

function removeChatbox(username) {
    var chatbox = $("#chatbox_" + username);
    if (chatbox != null) {
        chatbox.remove();
        //Remove from active List
        var rel = chatbox.attr('rel');
        const index = activeUserList.indexOf(rel);
        if (index > -1) {
            activeUserList.splice(index, 1);
        }
        rearrangeChatbox();
    }
}

```

```

}

/**
 * This function will re-arrange the Chatbox
 */

function rearrangeChatbox() {
    i = 10; // start position
    j = 260; //next position
    //alert(activeUserList);
    if(!activeUserList || activeUserList.length == 0){
        return;
    }
    $.each(activeUserList, function (index, username) {
        if ($("#chatbox_" + username).length == 1) {
            if (index < 4) {
                $('[rel="' + username + "']").css("right", i);
                $('[rel="' + username + "']").show();
                i = i + j;
            }
            else {
                $('[rel="' + username + "']").hide();
            }
        }
    });
}

/**
 * Check if chat-sidebar exists , Not not , append to the body
 */

if ($("#chat-sidebar").length == 0) {
    //console.log("Automatically Adding chat-sidebar");
    $("##" + currentUserListPositionID).append('Online users: <div id="chat-  
sidebar">');
}

//Handle Event clicking on user name on friend-list
$(document).on('click', '#sidebar-user-box', function () {
    var username = $(this).attr("class");

```

```

var fullname = $(this).children().text();
if ($("#chatbox_" + username).length != 0) {
    $("#chatbox_" + username).show();
    return;
}
renderChatbox(username, fullname);
rearrangeChatbox();
//trigger_requestChatHistory(username); //version 2
});

$(document).on('click', '.close', function () {
    //Find and remove the box
    var chatbox = $(this).parent().parent();
    if (chatbox != null) {
        chatbox.remove();
        //Remove from active List
        var rel = chatbox.attr('rel');
        const index = activeUserList.indexOf(rel);
        if (index > -1) {
            activeUserList.splice(index, 1);
        }
    }
    //Re-arrange the chatbox
    rearrangeChatbox();
});

//This will hide chat box
$(document).on('click', '.msg_head', function () {
    var chatbox = $(this).parents().attr("rel");
    $('[rel="' + chatbox + '"].msg_wrap').slideToggle('slow');
    return false;
});

/**
 * Handle when a message is typed and entered
 */

$(document).on('keypress', '.msg_footer textarea', function (e) {
    if (e.keyCode == 13) {
        var msg = $(this).val();
        var receiver = $(this).parent().parent().parent().attr("rel");
    }
});

```

```

//Clean up the box
$(this).val("");

if (msg.trim().length != 0) {
    // Display the message
    displaySelfMessage(receiver,msg)
    //Send to Socket
    if (typeof sendPrivateChat === "undefined" ) {
        console.log("TODO: Implement function sendPrivateChat(receiver,msg).
Function name is case-sensitive.");
        chatboxAlert("Developers: Implement function
sendPrivateChat(receiver,msg). Function name is case-sensitive.")
    }else{
        sendPrivateChat(receiver,msg);
    }
}
});

function chatboxAlert(msg){
    var alertmessage='<div class="alert"> <span class="closebtn"
onclick="this.parentElement.style.display=\'none\';">&times;</span>'+
    msg +'</div>';
    $('body').append(alertmessage);
}

function trigger_requestChatHistory(receiver){
    if (typeof requestChatHistory === "undefined" ) {
        console.log("TODO: Implement the function requestChatHistory(receiver).
Function name is case-sensitive.");
        chatboxAlert("Developers: Implement the function
requestChatHistory(receiver). Function name is case-sensitive.")
    }else{
        requestChatHistory(receiver);
    }
}

```

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title> Team: Usha Sankalamaddi and Harika Kunaparaju</title>
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"
crossorigin="anonymous"></script>
    <script src="/socket.io/socket.io.js"></script>
    <script src="./email.js"></script>
    <script src="./chatbox.js"></script>
    <link href="https://udayton-cloud.bitbucket.io/chatbox.css" rel="stylesheet">
    <script src="https://udayton-cloud.bitbucket.io/clock.js"></script>
    <link rel="stylesheet" href=https://udayton-cloud.bitbucket.io/style1.css>
    <script>
      $(document).ready(function(){
        $("#chatclient").hide();
        $("#regform").hide();
        $("#chat-sidebar").hide();
        $(".waiting").hide();
        $("#errssignup").hide();
        $("#loginres").hide();
      });
    </script>
    <style>
      .button {
        background-color: #4CAF50;
        border: none;
        color: white;
        padding: 5px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 12px;
        margin: 4px 2px;
        cursor: pointer;
      }
      .round { border-radius: 8px;}
      #response { background-color: #ff9800;}
```

```
#errsignup, #loginres{
    color: green;
    padding: 10px;
}
#loginform, #regform {
    width: 40%;
    min-width: 320px;
    max-width: 475px;
    background: #fff;
    position: relative;
    top: 50%;
    left: 35%;
    box-shadow: 0px 2px 5px rgb(0 0 0 / 25%);
    padding: 30px;
}

#login, #register {
    width: 100%;
    display: grid;
}

#newusername, #newpassword, #loginid, #reg-username, #reg-password,
#full-name, #email {
    height: 30px;
}

.loader{
    border: 16px solid #ff9800;
    border-top: 16px solid #4CAF50;
    border-radius: 50%;
    width: 25px;
    height: 25px;
    animation: spin 2s linear infinite;
    margin-left: 200px;
}

@keyframes spin{
    0% {transform: rotate(0deg);}
    100% {transform: rotate(360deg);}
}
```



```

    <form id="register">
        <label>Username</label>
        <input type="text" placeholder="Enter Username" name="regusername"
id="reg-username" required=""><br>
        <label>Password</label>
        <input type="password" placeholder="Enter Password"
name="regpassword" id="reg-password" required=""><br>
        <label>Full Name</label>
        <input type="text" placeholder="Enter Full name" name="fullname"
id="full-name" required=""><br>
        <label>Email</label>
        <input type="email" pattern=".+\\@.+\\.com" placeholder="Enter Email"
onkeypress="checkLoginEnter(event)" name="email" id="email"
required=""><br>
        <input id="regid" class="button round" type="button" value="Register"
onclick="register()">
    </form>
    <p>
        Already registered?
        <a href="#"
onclick="$('#regform').hide();$('#loginform').show()">Signin</a>
    </p>
</div>

```

```

<div id="chatclient">
    <div class="container-wrapper">
        <div id="top">
            <h1> Project Sprint2 </h2>
            <h1> By Usha Sankalamaddi and Harika Kunaparaju </h2>
            <h2 id="welmsg"></h2>
            <div id="email" onclick="showhideEmail()">show our email</div>
        </div>
        <div class="wrapper">
            <div class="menubar" style="height: 5px;">
                <p> Team Project Sprint-2</p>
                <canvas id="analog-clock" width="150" height="150"
style="background-color:#999;"></canvas>
            </div>
            <div style="margin-left: 200px;border:0px; margin-bottom: 140px;">

```



```

<div id="digital-clock"></div>

<h2> Chat Client </h2>
Chat message: <input name="data"
onkeypress="checkEnter(event)" onkeyup="socketio.emit('typing')" id="data">
  <button class="button round" type="submit" form="data"
value="Send" onclick="send()">Send</button>
  <button class="button round" type="submit" form="data"
value="Logout" onclick="logout()">Logout</button>
<div id="typing"></div>
<div id="online"></div>
<div id="response"></div>

<script>
var socketio = io();

socketio.on("online", data => {
  $("#online").html(data + "<br>");
})

socketio.on("message", data => {
  $("#response").append(data + "<br>");
});
socketio.on("typing", data => {
  $("#typing").html(data);
  setTimeout(() => {
    $("#typing").html("<br>");
  }, 3000);
});

function checkEnter(event) {
  if (event.keyCode == 13) send();
}

function send() {
  var input = $("#data").val();

```

```

        if (input.length == 0) return;
        socketio.emit("message", input);
        $("#data").val("");
    }

function logout(){
    $("#loginform").show();
    $("#chatclient").hide();
    $("#chat-sidebar").hide();
    $(".msg_box").hide();
    $("#newusername").val("");
    $("#newpassword").val("");
}

var canvas = document.getElementById("analog-clock");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
setInterval(drawClock, 1000);

function drawClock() {
    drawFace(ctx, radius);
    drawNumbers(ctx, radius);
    drawTime(ctx, radius);
}
</script>
</div>
</div>
</div>
</div>

<div id="chat-sidebar">
    <div id="list-group"></div>
</div>
<script>
    function checkLoginEnter(event){
        if(event.keyCode==13) login();

```

```

}
function login() {
    $("#loginres").hide();
    var username = $("#newusername").val();
    var password = $("#newpassword").val();

    if (!username || !password) {
        alert("Please provide all details!");
    }
    $(".waiting").show();
    socketio.emit("login",username,password);
}

function register() {
    $("#errsignup").hide();
    var username = $("#reg-username").val();
    var password = $("#reg-password").val();
    var fullname = $("#full-name").val();
    var email = $("#email").val();

    if (!username || !password || !fullname || !email) {
        alert('Invalid inputs, please retype');
        return;
    }

    var mailRegex=new RegExp(".*\@.*\.com");
    if(!mailRegex.test(email)){
        alert("Email address is not valid!")
        return;
    }
    else{
        socketio.emit("register",username,password, fullname, email);
    }
}

socketio.on("authenticated", (account,connectedClientsJSON) => {
    $(".waiting").hide();
    $("#loginform").hide();
    $("#chatclient").show();
    $("#chat-sidebar").show();

```

```

    var currentUserListID='list-group';
    let profile=account;
    $("#welmsg").html('Welcome '+ profile.fullname);

    setCurrentUser(profile);
    displayUserList(connectedClientsJSON);
  });

  function requestChatHistory(receiver) {
    socketio.emit("loadchat", receiver);
  }

  socketio.on("sendchathis", (chathistory, username) => {
    console.log('sending chat his');
    if (chathistory) {
      for (let idx in chathistory) {
        var chat = chathistory[idx];

        if (chat.sender == username) {
          displaySelfMessage(chat.receiver, chat.message);
        } else {
          displayFriendMessage(chat.sender, chat.message);
        }
      }
    } else {
      displayNoChatHistory(username);
    }
  })

  function sendPrivateChat(receiver, msg)
  {
    socketio.emit("privateChat", receiver, msg);
  }

  socketio.on("privateChat", (sender, message) => {
    displayFriendMessage(sender, message);
  });

  socketio.on("updateUserList", connectedClientsJSON => {

```

```
        displayUserList(connectedClientsJSON);
    });

    socketio.on("regform", () => {
        $("#errsignup").show();
        $("#returnmsg").css("background-color", "blue");
        $("#returnmsg").html("Registered successfully! Try Login now..!");

        $("#reg-username").val("");
        $("#reg-password").val("");
        $("#full-name").val("");
        $("#email").val("");
    });

    socketio.on("errsignup", (msg) => {
        $("#errsignup").show();
        $("#returnmsg").css("background-color", "blue");
        $("#returnmsg").html(msg);

        $("#reg-username").val("");
        $("#reg-password").val("");
        $("#full-name").val("");
        $("#email").val("");
    });

    socketio.on("loginfailed", (msg) => {
        $("#loginres").show();
        $(".waiting").hide();
        $("#lastmsg").css("background-color", "blue");
        $("#lastmsg").html(msg);
        $("#username").val("");
        $("#password").val("");
    });

</script>
</body>
</html>
```

Video Demo link:

https://udayton.zoom.us/rec/share/SX606fVHiKDz5voMWSfZoyvnQjK3R7ZD0xz42c3SHLcLoAJPZ_Qtl6E0UcqnBpLM.-TmCJfeIuVvDkAi9?startTime=1670485875000

cca-project repository link:

<https://bitbucket.org/cca-sankalamaddiu1-kunaparajus1/cca-project/src/sprint3/>