Milestone 3 - CODE **Matriculation: 1310870**
Name: Muhammad Adnan

Front-End

CODE

chat_user/chat-user.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Socket } from 'ngx-socket-io';
import { ChatDetails, ChatInsert, ChatList } from '../_models/chat-model';
import { AuthenticationService } from '../_services/authentication.service';
import { ChatService } from '../_services/chat.service';

declare var scrollToBottomFunction: any;

@Component({
  selector: 'app-chat-user',
  templateUrl: './chat-user.component.html',
  styleUrls: ['./chat-user.component.css'],
})
export class ChatUserComponent implements OnInit {
  txtChat: string;
  chatList: ChatList;
  chatDetail: ChatDetails;
  loggedInUserId: string = '';
  prodId: string;
  receiverId: string;
  activatedChatSessionID: string = '';

  constructor(
    private _chatService: ChatService,
    private activatedRoute: ActivatedRoute,
    private socket: Socket,
    private _authService: AuthenticationService
  ) {}

  ngAfterViewChecked() {
    let scrollDiv = <HTMLInputElement>document.getElementById('scrollDiv');
    scrollDiv.scroll({ top: scrollDiv.scrollHeight, behavior: 'smooth' });
  }
  ngOnInit(): void {
    ///SOCKET IO TO UPDATE CHAT
    this.socket.on('clientChatUpdate', (data) => {
      this.getChatList();
    });
    ///////////////////////////

    this.activatedRoute.paramMap.subscribe((x) => {
      this.prodId = x.get('prodId') || '';
      this.receiverId = x.get('rec') || '';

      let loggedInUser = this._authService.currentUser;

      console.log(loggedInUser);

      this.loggedInUserId = loggedInUser.user.id.toString();

      if (this.prodId != '' && this.receiverId != '') {
        this._chatService
          .checkAndInsertChatSession(
            this.prodId,
            this.loggedInUserId,
            this.receiverId
          )
          .subscribe((data) => {
            if (data.chat) {
              //Got a new Chat Session OR get old one if exists, check by stored procedure
              //Stored Procedure Name: CheckAndInsertChatSession
              this.activatedChatSessionID = data.chat[0].chatSessionId;
            }

            this.getChatList();

            //SOCKET IO
            this.socket.emit('updateChat', '');
            ///////////////////////
          });
      } else {
        //Get list and activated first chat
        this.getChatList();
      }
    });
  }

  getChatList() {
    //TODO ADNAN

    this._chatService.getChatList(this.loggedInUserId).subscribe((data) => {
      this.chatList = data;

      if (data) {
        if (this.activatedChatSessionID == '' && data.chat[0]) {
          this.activatedChatSessionID = data.chat[0].id;
        }
        if (data.chat[0]) this.getChatDetail(this.activatedChatSessionID);
      }
    });
  }

  sendChat() {
    if (this.txtChat && this.txtChat.trim()) {
      let chatObj = {} as ChatInsert;
      chatObj.message = this.txtChat.trim();
      chatObj.receiverId =
        this.chatDetail.chat[0].ReceiverID == this.loggedInUserId
          ? this.chatDetail.chat[0].SenderID
          : this.chatDetail.chat[0].ReceiverID;
      chatObj.senderId = this.loggedInUserId;
      chatObj.chatSessionID = this.chatDetail.chat[0].chatSessionID;

      this._chatService.insertChat(chatObj).subscribe((data) => {
        this.txtChat = '';

        //SOCKET IO
        this.socket.emit('updateChat', '');
        ///////////////////////
      });
    }
  }

  getChatDetail(chatSessionId) {
    this.activatedChatSessionID = chatSessionId;
    this._chatService
      .getChatHistoryById(chatSessionId, this.loggedInUserId)
      .subscribe((data) => {
        this.chatDetail = data;

        this._chatService
          .updateReadBit(chatSessionId, this.loggedInUserId)
          .subscribe((data) => {
            this._chatService.getNotification(this.loggedInUserId);
          });
      });
    return false;
  }
}
```

_model/chat-model.ts

```
export interface ChatList {
  status: string;
  chat: ChatListDetail[];
}
export interface ChatListDetail {
  id: string;
  user1ID: string;
  user2ID: string;
  ProductName: string;
  ProductID: string;
  topMessage: string;
  opponentUserName: string;
  unreadMessages: number;
}

export interface ChatDetails {
  status: string;
  chat: ChatDetailsDetails[];
}

export interface ChatDetailsDetails {
  id: string;
  Message: string;
  Date: Date;
  SenderID: string;
  ReceiverID: string;
  chatSessionID: string;
  isRead: boolean;
  opponentUserName: string;
  myName: string;
  ProductID: string;
}

export interface ChatInsert {
  message: string;
  senderId: string;
  receiverId: string;
  chatSessionID: string;
}

export interface CheckAndInsertChatSession {
  status: string;
  chat: CheckAndInsertChatSessionDetail[];
}

export interface CheckAndInsertChatSessionDetail {
  chatSessionId: string;
}

export interface ChatNotification {
  status: string;
  chat: ChatNotificationDetail[];
```

```
}

export interface ChatNotificationDetail {
  totalCount: string;
}
```

_services/chat.service.ts

```
import { HttpClient, HttpParams } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable } from 'rxjs';
import { environment } from 'src/environments/environment';
import {
  ChatDetails,
  ChatInsert,
  ChatList,
  ChatNotification,
  CheckAndInsertChatSession,
} from '../_models/chat-model';

@Injectable({
  providedIn: 'root',
})
export class ChatService {
  private baseUrl = environment.apiUrl;
  chatNotification = new BehaviorSubject<ChatNotification>(
    {} as ChatNotification
  );

  constructor(private http: HttpClient) {}

  public getChatHistoryById(
    chatSessionId: string,
    loggedInUserId: string
  ): Observable<ChatDetails> {
    const params = new HttpParams()
      .set('chatSessionId', chatSessionId)
      .set('loggedInUserId', loggedInUserId);

    return this.http.get<ChatDetails>(
      this.baseUrl + '/chat/getChatHistoryById',
      { params }
    );
  }

  public getChatList(receiverId: string): Observable<ChatList> {
    const params = new HttpParams().set('receiverId', receiverId);
    return this.http.get<ChatList>(this.baseUrl + '/chat/getChatList', {
      params,
    });
  }

  public checkAndInsertChatSession(
    productId: string,
    senderId: string,
    receiverId: string
  ): Observable<CheckAndInsertChatSession> {
    const params = new HttpParams()
      .set('productId', productId)
      .set('senderId', senderId)
      .set('receiverId', receiverId);

    return this.http.get<CheckAndInsertChatSession>(
      this.baseUrl + '/chat/checkAndInsertChatSession',
      { params }
    );
  }

  public updateReadBit(
    chatSessionID: string,
    receiverId: string
  ): Observable<any> {
    const params = new HttpParams()
      .set('chatSessionID', chatSessionID)
      .set('receiverId', receiverId);

    return this.http.get<any>(this.baseUrl + '/chat/updateReadBit', { params });
  }

  public getNotification(receiverId: string) {
    const params = new HttpParams().set('receiverId', receiverId);

    this.http
      .get<ChatNotification>(this.baseUrl + '/chat/getNotification', { params })
      .subscribe((data) => {
        this.chatNotification.next(data);
      });
  }

  public insertChat(chatObj: ChatInsert): Observable<any> {
    return this.http.post<any>(this.baseUrl + '/chat/insertChat', chatObj);
  }
}
```

HTML:

chat_user/chat-user.component.html

```
<main class="content">
  <div class="container p-0">
    <h1 class="h3 mb-3">Chat</h1>

    <div class="card">
      <div class="row g-0">
        <div class="col-12 col-lg-5 col-xl-3 border-right">
          <a
            href="#"
            class="list-group-item list-group-item-action border-0"
            [ngClass]="{
              'chat-message-active': c.id == activatedChatSessionID
            }"
            *ngFor="let c of chatList?.chat"
            (click)="getChatDetail(c.id)"
          >
            <!-- <div class="badge bg-success float-right">5</div> -->
            <div class="d-flex align-items-start">
              <img
                src="assets/images/profile-user.png"
                class="rounded-circle mr-1"
                alt="Vanessa Tucker"
                width="40"
                height="40"
              />
              <div class="flex-grow-1 ml-3">
                {{ c.ProductName }} ({{ c.opponentUserName }})
                <span
                  *ngIf="c.unreadMessages != 0"
                  class="badge badge-primary badge-pill"
                  >{{ c.unreadMessages }}</span
                >
                <div class="small">
                  <span class="fas fa-circle chat-online"></span>
                  {{ c.topMessage }}
                </div>
              </div>
            </div>
          </a>

          <hr class="d-block d-lg-none mt-1 mb-0" />
        </div>
        <div class="col-12 col-lg-7 col-xl-9">
          <div class="position-relative">
            <div class="chat-messages p-4" id="scrollDiv">
              <!-- <app-loader-animation [isShow]="(chatDetail | json) == '{}'"></app-loader-
animation> -->

              <div
                [ngClass]="
                  loggedInUserId == chatDetail.SenderID
                    ? 'chat-message-left'
                    : 'chat-message-right'
                "
                class="pb-4"
                *ngFor="let chatDetail of chatDetail?.chat"
              >
                <div>
                  <img
                    src="assets/images/chat_user.png"
                    class="rounded-circle mr-1"
                    alt="Chris Wood"
                    width="40"
                    height="40"
                  />
                  <div class="text-muted small text-nowrap mt-2">
                    {{ chatDetail.Date | date: "HH:mm" }}
                  </div>
                </div>
                <div class="flex-shrink-1 bg-light rounded py-2 px-3 mr-3">
                  <div
                    class="font-weight-bold mb-1"
                    *ngIf="loggedInUserId == chatDetail.SenderID"
                  >
                    You
                  </div>
                  <div
                    class="font-weight-bold mb-1"
                    *ngIf="loggedInUserId != chatDetail.SenderID"
                  >
                    {{ chatDetail.opponentUserName }}
                  </div>

                  {{ chatDetail.Message }}
                </div>
              </div>
            </div>
          </div>

          <div class="flex-grow-0 py-3 px-4 border-top">
            <div class="input-group">
              <input
                type="text"
                class="form-control"
                placeholder="Type your message"
                [(ngModel)]="txtChat"
                (keyup.enter)="sendChat()"
              />
              <button class="btn btn-primary" (click)="sendChat()">Send</button>
            </div>
```

```
          </div>
        </div>
      </div>
    </div>
  </div>
</main>
```

CSS

chat_user/chat-user.component.css

```
body{margin-top:20px;}

.chat-online {
    color: #34ce57
}

.chat-offline {
    color: #e4606d
}

.chat-messages {
    display: flex;
    flex-direction: column;
    max-height: 400px;
    overflow-y: scroll;
    min-height: 400px;
}

.chat-message-left,
.chat-message-right {
    display: flex;
    flex-shrink: 0
}

.chat-message-left {
    margin-right: auto
}

.chat-message-right {
    flex-direction: row-reverse;
    margin-left: auto
}
.py-3 {
    padding-top: 1rem!important;
    padding-bottom: 1rem!important;
}
```

```css
.px-4 {
    padding-right: 1.5rem!important;
    padding-left: 1.5rem!important;
}
.flex-grow-0 {
    flex-grow: 0!important;
}
.border-top {
    border-top: 1px solid #dee2e6!important;
}

.chat-message-active{
background-color:#ccc;
}
```

Back-end

backend/chat.js

```
const router = require("express").Router();
const { json } = require("body-parser");
const { stringify } = require("querystring");
const sqlManager = require("./sql");
const config = require("./config");
const multer = require("multer");
const upload = multer({ dest: "uploads/" });
var util = require("util");

router.get("/getChatHistoryById", function (req, res) {
  console.log(req);
  sqlManager.getChatHistoryById(
    req.query.chatSessionId,
    req.query.loggedInUserId,
    function (err, result) {
      if (err) {
        res.status(500).json({ status: "Failed", message: err.message });
        return;
      }
      if (result.length == 0) {
        res.status(200).json({ status: "Success", chat: [] });
        return;
      }
      res.status(200).json({ status: "Success", chat: result });
    }
  );
});

router.get("/getChatList", function (req, res) {
  console.log(req);
  sqlManager.getChatList(req.query.receiverId, function (err, result) {
    if (err) {
      res.status(500).json({ status: "Failed", message: err.message });
      return;
    }
    if (result.length == 0) {
      res.status(200).json({ status: "Success", chat: [] });
      return;
    }
    res.status(200).json({ status: "Success", chat: result });
  });
});

router.get("/checkAndInsertChatSession", function (req, res) {
  console.log(req.body);

  sqlManager.checkAndInsertChatSession(
    req.query.productId,
    req.query.senderId,
    req.query.receiverId,
    function (err, result) {
      if (err) {
        res.status(500).json({ status: "Failed", message: err.message });
        return;
      }
      if (result.length == 0) {
        res.status(200).json({ status: "Success", chat: {} });
        return;
      }
      res.status(200).json({ status: "Success2", chat: result[0] });
    }
  );
});

router.get("/updateReadBit", function (req, res) {
  console.log(req.body);

  sqlManager.updateReadBit(
    req.query.chatSessionID,
    req.query.receiverId,
    function (err, result) {
      if (err) {
        res.status(500).json({ status: "Failed", message: err.message });
        return;
      }

      res.status(200).json({ status: "Success" });
    }
  );
});

router.get("/getNotification", function (req, res) {
  console.log(req.body);

  sqlManager.getNotification(req.query.receiverId, function (err, result) {
    if (err) {
      res.status(500).json({ status: "Failed", message: err.message });
      return;
    }

    if (result.length == 0) {
      res.status(200).json({ status: "Success", chat: {} });
      return;
    }

    res.status(200).json({ status: "Success", chat: result });
  });
});

router.post("/insertChat", function (req, res) {
  console.log("adnan");
  console.log(req.body);

  sqlManager.insertChat(req.body, function (err, result) {
    if (err) {
      res.status(500).json({ status: "Failed", message: err.message });
      return;
    }
    if (result.length == 0) {
      res.status(404).json({ status: "Success" });
      return;
    }
    res.status(200).json({ status: "Success" });
  });
});

module.exports = router;
```

backend/sql.js

```
//////////////////////////////////////////////CHAT//////////////////////////////////////////////
////
function getChatHistoryById(chatSessionId, loggedInUserId, cb) {
  var queryString =
    `SELECT c.*,
    U.name as opponentUserName,
    MU.name as myName
        FROM dbo.chat c
        inner join User U on (U.id = c.senderId and c.senderId <> ` +
    loggedInUserId +
    `) OR (U.id = c.receiverId and c.receiverId <>` +
    loggedInUserId +
    `)
        inner join User MU on (MU.id = c.senderId and c.senderId <> U.Id) OR (MU.id = c.recei
verId and c.receiverId <> U.Id)
    WHERE chatSessionID = ` +
    chatSessionId +
    `
    ORDER BY Date`;
  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

function getChatList(userId, cb) {
  var queryString =
    `
    SELECT
        CS.id,
        CS.user1ID,
        CS.user2ID,
        P.title AS ProductName,
        P.id AS ProductID,
        (select count(*) as totalCount from chat where ifnull(isRead, 0) = 0 and receiverId =
 ` +
    userId +
    ` and chatSessionID = CS.id) as unreadMessages,
        SUBSTRING(
                (SELECT
                    message
                FROM chat tm
                WHERE ((tm.SenderID = CS.user1ID AND tm.ReceiverID = CS.user2ID) OR (tm.Sende
rID = CS.user2ID AND tm.ReceiverID = CS.user1ID))
                AND tm.chatSessionID = CS.id
                ORDER BY tm.Date DESC
                limit 1), 1, 50
            ) AS topMessage,
            U.name as opponentUserName
        FROM chatSession CS
        INNER JOIN Product P ON P.id = CS.productID
        inner join User U on (U.id = CS.user1ID and CS.user1ID <> ` +
    userId +
    `) OR (U.id = CS.user2ID and CS.user2ID <> ` +
    userId +
    `)
        where (CS.user1ID = ` +
    userId +
    `) OR (CS.user2ID = ` +
    userId +
    `)
        order by CS.createdDate desc`;
  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

function checkAndInsertChatSession(productId, senderId, receiverId, cb) {
  //body.message = body.message.replaceAll("'", "\'");
  var queryString =
    `call CheckAndInsertChatSession(` +
    productId +
    `,` +
    senderId +
    `, ` +
    receiverId +
    `)`;
  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

function updateReadBit(chatSessionID, receiverId, cb) {
  //body.message = body.message.replaceAll("'", "\'");
  var queryString =
    `update chat
        set isRead = 1
        where receiverId = ` +
    receiverId +
    ` and chatSessionID = ` +
    chatSessionID;

  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

function getNotification(receiverId, cb) {
  //body.message = body.message.replaceAll("'", "\'");
  var queryString =
    `select count(*) as totalCount from chat
    where ifnull(isRead, 0) = 0
    and receiverId = ` + receiverId;

  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

function insertChat(body, cb) {
  console.log(body);

  //body.message = body.message.replaceAll("'", "\'");
  var queryString =
    `
INSERT INTO dbo.chat
( Message ,
  Date ,
  SenderID ,
  ReceiverID ,
  chatSessionID,
  isRead
```

```
)
VALUES  ( '` +
    body.message +
    `' , -- Message - varchar(max)
  now() , -- Date - datetime
  ` +
    body.senderId +
    `  , -- SenderID - int
  ` +
    body.receiverId +
    `  , -- ReceiverID - int
  ` +
    body.chatSessionID +
    `,  -- chatSessionID - int
    false  -- isRead - bool

)`;
  connection.query(queryString, function (err, rows) {
    if (err) cb(err);
    else cb(undefined, rows);
  });
}

/////////////////////////////////////////////////////////////////////////////////////////////
////
```

sql/stored procedure/CheckAndInsertChatSession

```
CREATE DEFINER=`admin`@`%` PROCEDURE `CheckAndInsertChatSession`(
    IN productId int,
    IN senderId int,
    IN receiverId int)
    BEGIN
    if (select not Exists(select 1 from chatSession CS where CS.productId = productId and ((C
S.user1ID = senderId) OR (CS.user2ID = senderId))
    and ((CS.user1ID = receiverId) OR (CS.user2ID = receiverId)) limit 1)) then
        INSERT INTO dbo.chatSession
                ( user1ID, user2ID, productID, createdBy, createdDate )
        VALUES  ( senderId, -- user1ID - int
                  receiverId, -- user2ID - int
                  productId,  -- productID - int
                  senderId,
                  now());

        SET @chatSessionId =  @@identity;

        INSERT INTO dbo.chat
            ( Message ,
              Date ,
              SenderID ,
              ReceiverID ,
              chatSessionID,
              isRead
            )
        VALUES  ( 'Hi' , -- Message - varchar(max)
              now() , -- Date - datetime
              senderId , -- SenderID - int
              receiverId , -- ReceiverID - int
              @chatSessionId,  -- chatSessionID - int
              false
            );

        select @chatSessionId as chatSessionId;

    else

    select CS.id as chatSessionId from chatSession CS where CS.productId = productId and ((CS.user1ID = senderId) OR (CS.user2ID = senderId))
    and ((CS.user1ID = receiverId) OR (CS.user2ID = receiverId)) limit 1;

    end if;
    END
```