Angular Assignment- 2

Assignment: Minimal Chat Application (Frontend)

Introduction

In this assignment, you will be building a minimal chat application frontend using Angular and Typescript. The goal is to create an Angular frontend application that interfaces with already available APIs. You will create Angular modules, components, services and routing that allow users to register, authenticate, initiate conversations, send messages, retrieve message history, and apply sorting and paging mechanisms.

Tech Stack

- Angular 15+
- Typescript
- HTML/CSS

Requirements

User registration (Route: /register)

Create a registration component where users can enter their email, name, and password. On submission, make a POST request to the backend API **/api/register** to register the user.

Registration page should have an option for "Sign up with Google", that allows the user to register to the system using their Google account.

On successful response, redirect user to login. On failure response, show relevant error message to user.

User login (Route: /login)

Create a login component where users can enter their email and password. On submission, make a POST request to the backend API **/api/login** to authenticate the user. If the login is successful, store the JWT token in the browser's local storage for future API requests, and redirect user to /chat route. If login fails, display relevant error message to user.

Note: JWT token needs to be sent in Authorize headers for all further API calls to authenticate user.

Login page should have a button for "Login with Google", which should allow the user to login to the system with their Google account.

User List (Route: /chat)

Imagine the UI available in Microsoft Teams or Slack where the user list is displayed on the left portion of the UI. Create a user list component that retrieves the list of users from the backend API **/api/users** using a GET request. Display the list of users with their names and email addresses.

Conversation History (Route: /chat/user/{userId})

Create a conversation history component where users can view their conversation history with a specific user. Upon clicking specific user from the User List component, users can initiate a conversation with specific user. Make a GET request to the backend API **/api/conversations/{userId}** to retrieve the conversation history. Display the messages with their content and timestamps in chronological order.

Conversation history should only display last 20 messages (see sort parameter in API).

When user scrolls beyond 20 messages, it should retrieve and display the next 20 messages by providing “before” parameter in API.

Send Message

In conversation history component, there will be a textbox and send button in the bottom area where user can write a message content and send it to specific user. On click of send button, make a POST request to the backend API **/api/messages** to send the message to the selected recipient. After the message is successfully sent, append relevant message in conversation history. In case of failure, display relevant error message to user.

Edit Message

In conversation history component, upon right clicking on particular message sent by current user, user will be displayed an edit option. Once user choose edit option, there will be an inline editor textbox where user can edit existing message. There will be a button to accept or decline existting message. If user accepts edited message, make a PUT request to the backend API **/api/messages/{messageId}** to update the message content. Upon success, update the message in conversation history, upon failure, display relevant error message to user.

Delete Message

In conversation history component, upon right clicking on particular message sent by current user, user will be displayed a delete option. Once the user chooses the delete option, display them a confirmation dialog and after confirming, make a PUT request to the backend API **/api/messages/{messageId}** to update the message content. Upon success, delete the message from conversation history, upon failure, display relevant error message to user.

Realtime Communication

In conversation history component, if the other person is sending any messages right now, it should automatically be received and appended to the conversation history without having to reload the page. This can be achieved by connecting to the SignalR hub in the backend and implementing the necessary methods of SignalR.

Search Conversation

The user should be able to search for a particular message by using a string. So, when the user enters the string to search for, the system should show the list of messages that contains the string. This can be achieved by making an API call to **/api/conversation/search?query={{search string}}**. Upon success, it should return a list of messages that contains said string. In case of failure, display an appropriate message to the user.

There should be a search box at the top of the screen as shown below. On a successful search, the results should show up as a list on the left pane (replacing the user list temporarily). This temporary panel should have a close button to remove it and get back to the original screen layout.

Request Logging

Create a page for viewing request logs

- Should display data in the form of a table with columns
- Should allow us to select timeframe of logs to show
    - Last 5 mins (default)
    - Last 10 mins
    - Last 30 mins
    - Custom – Allow inserting specific time range
        - This needs a time picker
- Should allow us to select which columns to show

Scoring

- User Registration - 10
- User Login - 15
- User List - 10
- Conversation History - 25
- Send Message - 5
- Edit Message – 5
- Delete Message – 5
- Realtime communication – 15
- Search Conversation – 5
- Request Logging – 5

Timeline

You have a timeline of **4** days to complete this.

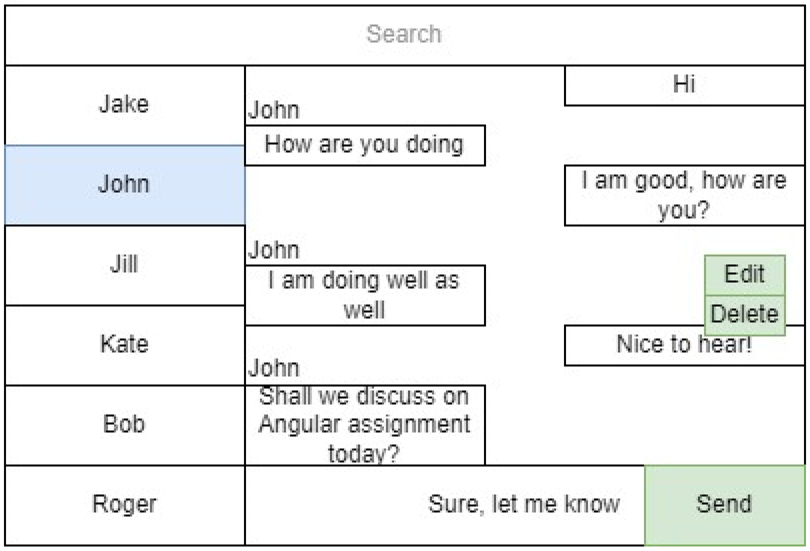
Ground Rules

- Code must be pushed to GitHub repository before leaving for the day.
- Make sure to follow all points mentioned in the Requirements section. Scoring will be based on adherence of all conditions mentioned in requirements.

Submission Guidelines

- Share link of public GitHub Repository to Reporting Person
- Create a short screen recording of the features as mentioned in the document and share it with the Reporting Person

/chat/user/23



Good Luck!