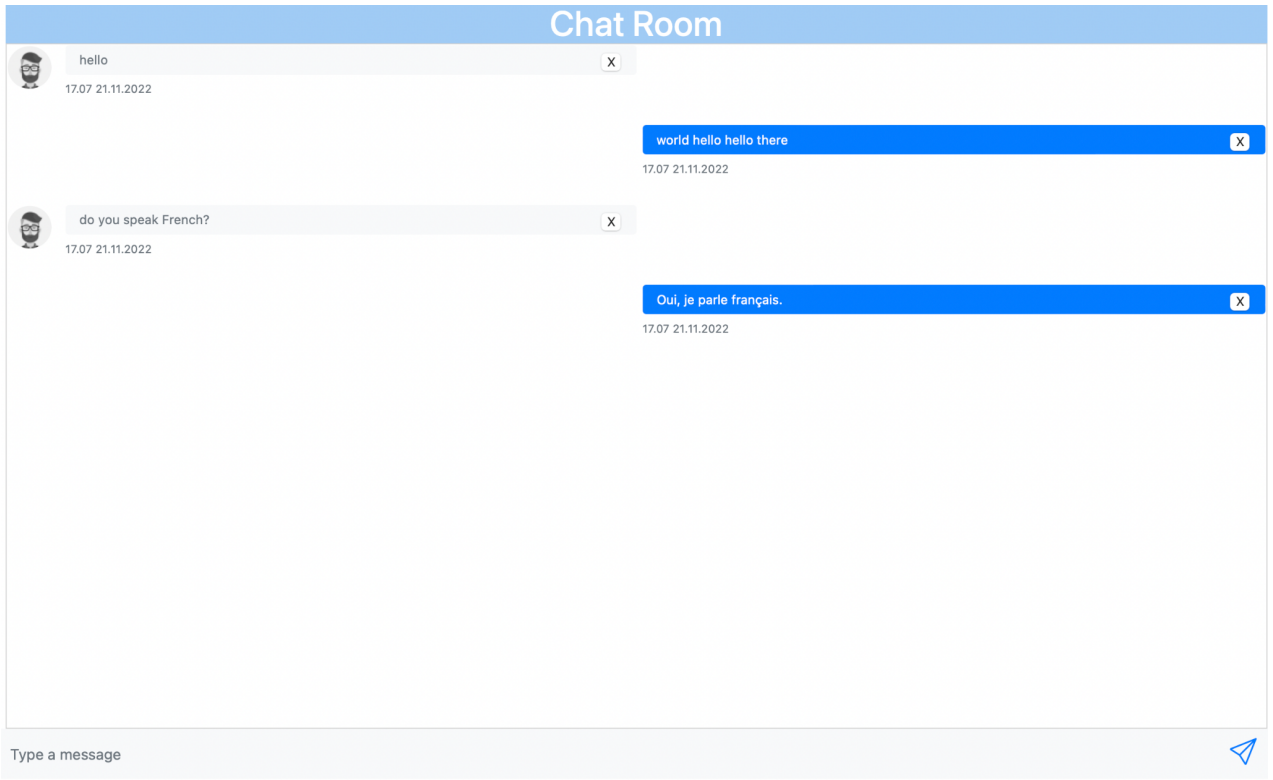
Assignment 3 – Chat Room

The goal of this assignment is to create **a chat room** based on the knowledge you gathered during this course. We are engineers, so of course this will not be a typical chatroom! The questions that the user will ask, will be answered by an AI model.



Problem Description

In this Chat Room, the user will be able to write and send a message. An AI model will then read the message and will reply with an answer.
The chatroom shall present all messages send and received, with their date, and also allow the user to delete them. For the implementation of this chat room, you will need to use a Spring Boot + React project and integrate the OpenAI.

Use the **Assignment3startingcode.zip.** that is uploaded in the blackboard. The folder includes the starting code for your project.

Part 1 – Implement the Java Classes

Step 1 - Creating the Message Class.

Step 1.1: You will find a file called Message.java . This file represents an **entity** of message and contains the following attributes.

- **int id** An id that will be automatically generated.
- **String messageText** The message's text
- **String messageDate** The date and time the message was generated.
- **String messageType** A text to distinguish between **"Question"** or "**Answer"** messages.

Step 1.2 Implement the **getters** and **setters** methods for each of the attributes.

Step 2 - Implementing the MessageController.
The Message controller includes the methods you will need for adding a new message, and deleting a message based on id.

Step 2.1 Implement the **addMessageandResponse** method.
When a user sends a text message, the AI model shall use the text and send a response. Therefore, this method should take as a parameter the message's text that the user wants to send.
Then:

- You need to create a **message** object and save it into the repository
- Make an api request to **OpenAI** with the message's text and receive the answer. **check a previous practical on how to do this!!
- Create a new **message** object with the AI response and save it to the repository as well.
- Remember, that you need to set the **type** of the message as "**Question"** for the message of the user and "**Answer"**

```
@GetMapping("/addMessageandResponse")
public RedirectView addMessageandResponse(@RequestParam final String message) {
```

Step 2.2 Implement the **deleteMessage** method.

This method should take as a parameter the integer id which is the id of the message. Then should proceed with deleting the message from the repository based on the given id.

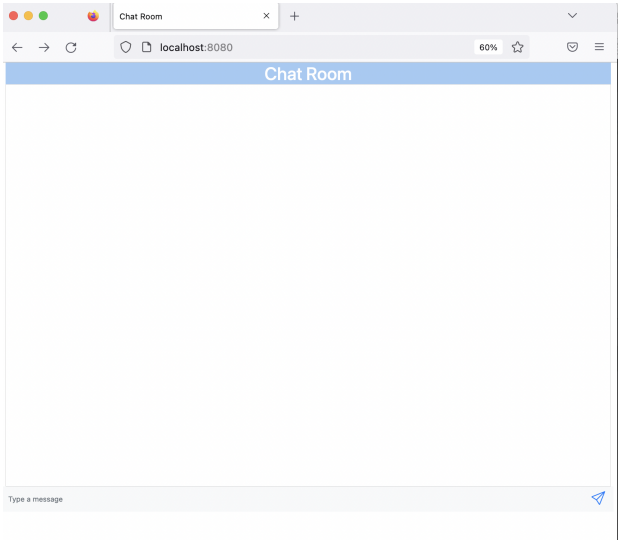
```
@GetMapping("/deleteMessage")
public RedirectView deleteMessage(@RequestParam Integer id) {
```

Step 2.3 Implement the **MessageList** method. This method shall return all messages saved in the repository.

```
@GetMapping("/MessageList")
public Iterable<Message> getMessages() {
```

Part 2 – React implementation and UI

Step 1 - Starting up.
In the src/main/**resources**/**static**/ package you will see a file called **Index.html.** This is the main HTML page of your project. If you open this file in a browser, you will see the following screen.



This file includes the title, the MessageArea and a form that you can use to send messages. Explore the file and understand the elements.

Step 1.1 Send Form.
In Index.html there is a Form with id `sendMessageForm` . Add the appropriate `action` and `method.` So that the text that you write in the input box will be added into your repository, generate an answer and save the answer.

TEST: Navigate to **http://localhost:8080/MessageList** all added messages shall be presented here in a JSON format.

Step 1.2 Implement the **MessageArea** component.
In Main.js Implement the

```
class MessageArea extends React.Component {
 }
```

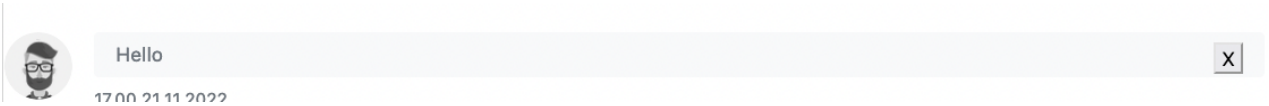
1. This component shall a) include the `messages: []` in its state, and b) When **mount,** to **fetch** all the messages from the repository.
1. In the render method, it shall return a new object of the **Messages** component and pass as argument `messages` the `messages` array.

Step 1.3 Implement the **Messages** component.
We have already provided some starting code for this method. Check the code and make sure that you understand it.
In this step you need to add the required code, so that
when the `(message.type==="Question")` it returns a new object of the `QuestionMessage` and passes the appropriate parameters, otherwise it returns an object of the `QuestionMessage` and passes the appropriate parameters.

Step 1.4 Correct `AnswerMessage` and `QuestionMessage` functions to display the text and date properties. At the moment, a static “hello” is shown for all messages. Adjust the code, so that it can display the provided props.

Step 1.4 Enable the user to delete a message in `AnswerMessage` and `QuestionMessage` functions.

At the moment, we only allow the sending of messages. In this step, we want to enable users to delete messages. Identify the best way to do this.



Part 3 – Improve the UI & Extras!

You are free to make your own changes to the ui of this application to make it look better and give to the user a better experience.

Add extra functionality on your assignment to get a **bonus**!! Maybe add the option to select a different model provided by OpenAI? Allow the user to change its picture? It is up to you!

Part 4 – Test and Screenshot

Step 1. Make sure that your application is running, that you can send and receive messages and that you can delete a message.

Step 2. Send and receive multiple messages and **Take a screenshot of your webpage.**

Submission

Export your project folder into a zip file, add the screenshot and compress them again. Upload the zip that contains both your project and the **screenshot** on Assignment 3 on Blackboard.

Marking

Implement the Java Classes	**5**
Chat Room functionality	**5**
Bonus	**2**

How the Bonus works
For every additional feature, you get 1 mark in Bonus! This can lead your mark to up to 12/10! However, your total mark for the assignments cannot exceed the 30% of the mark for this course.