Limeade

Event Meetup App

Problem we are solving

- Connecting people to social experiences that require or are better with multiple participants
- Making memories and connections from finding something to do together

Who are the users of this solution?

- People who would be interested in events but feel awkward going alone
  - People looking for social interactions and friends through shared interests
- Advertisers and businesses looking to turn services into experiences

...

Demo

Process Discussion

- As a team we communicated using Discord as a platform, since it was the one everyone was most active on, through standup messages, general messages, and questions about each of our stories
- We then relayed the standup messages to our TA through Slack
- As for Git/Github, we made branches for features and we had a development and main branch
    - During the sprint, every feature branch is created based on the development branch and gets merged back to the development branch
    - At the end of the sprint, development gets merged to main
- We mostly divided tasks based on volunteering, and we didn't really have any problems allocating tasks
- During the first sprint, we were just setting up the application, so there wasn't enough work to spread around for all of us but this problem was fixed when there were more features to implement

Technical Discussion

[Youngjae] One lesson I learned was generally how to use websocket to implement communication between the client and the server. Some libraries that are available in Angular didn't seem to work for me and weren't very intuitive as to how to use them. However, just using the WebSocket object in the client and express ws in the server seemed to work well.

[Amy] Something I learned during the project was the difficulty of implementing events with an exact date. Time zones aren't certain even if the information comes from the browser, and the only way to find the time zone is to ask the user. Date objects in MongoDB and Javascript are slightly different and to convert to each needed a little bit of string manipulation and formatting, and even then, often ended up looking different due to the default time zone MongoDB settles on.

Technical Discussion continued

[Kai] One important lesson that I learned about working in a software development team is the necessity of effective communication and collaboration. In a team setting, clear and concise communication is essential for the success of the project. This includes regular check ins, sharing information, and assigning clear roles and responsibilities.

```
ws.on("message", async (message) => {
  // Parse the incoming message
  const data = JSON.parse(message);

  // Create a new chat message with senderName and message fields
  const currentDate = new Date();
  const formattedDate = currentDate.toLocaleString("en-US", {
    year: "numeric",
    month: "long",
    day: "numeric",
    hour: "2-digit",
    minute: "2-digit",
    second: "2-digit",
    hour12: false,
  });

  const chatMessage = {
    senderName: data.senderName,
    message: data.message,
    date: formattedDate,
  };

  try {
    // Add the chat message to the chat room's messages array
    chatRoom.messages.push(chatMessage);

    // Save the chat room object with the new message
    await chatRoom.save();

    // Broadcast the new message to all clients in the chat room
    const clients = chatRoomClientsMap.get(id);
    clients.forEach((client) => {
      if (client.readyState === WebSocket.OPEN) {
        client.send(JSON.stringify([chatMessage]));
      }
    });
  } catch (error) {
    console.error("Error saving or broadcasting chat message:", error);
  }
});

// Handle WebSocket disconnection
ws.on("close", () => {
  // Remove the WebSocket connection from the chat room's clients
  const clients = chatRoomClientsMap.get(id);
  if (clients) {
    clients.delete(ws);
    if (clients.size === 0) {
      chatRoomClientsMap.delete(id);
    }
  }
});
```

```
ngOnInit() {
  this.api.getMe().subscribe((next) => {
    this.user = next;
  });
  const websocketUrl = `${this.serverUrl}/${this.roomId}`;
  this.webSocket = new WebSocket(websocketUrl);

  // WebSocket event listeners
  this.webSocket.onopen = () => {
    console.log('WebSocket connection established');
  };

  this.webSocket.onmessage = (event) => {
    const message = JSON.parse(event.data);
    console.log(this.messages);
    console.log('Received message:', message);
    for (const item of message) {
      this.messages.push(item);
    }
    this.messageText = '';
  };

  this.webSocket.onclose = () => {
    console.log('WebSocket connection closed');
    // Perform any necessary clean-up or reconnection logic
  };

  this.webSocket.onerror = (error) => {
    console.error('WebSocket error:', error);
    // Handle any errors that occur during the WebSocket connection
  };
}

ngOnDestroy() {
  if (this.webSocket) {
    this.webSocket.close();
  }
}

sendMessage() {
  if (this.messageText === '') {
    return;
  }
  if (this.webSocket.readyState === WebSocket.OPEN) {
    this.webSocket.send(
      JSON.stringify({
        senderName: this.user.username,
        message: this.messageText,
      }),
    );
  }
}
}
```

Software Architecture (MEAN Stack)

- Frontend was built with Angular using Typescript as the language of choice
  - Styling and “animations” were done using plain CSS
  - Frontend components were split into components and pages — pages were pages that the user could navigate to and components were reusable portions of code used in multiple pages
  - Multiple api services were used to communicate with backend, divided by use
- Backend was built using Node.js, Express and MongoDB as the database
  - Two main models (event, user) had their own routers along with staff users since they had the most additional features
  - Mongoose was used as ORM for MongoDB
  - Authentication was done using sessions and important variables were stored in an .env file

Individual Contribution

- **Youngjae:** creating and editing profiles, usersearch, chat rooms and inviting
- **Lukas:** user filtering by interest, reporting through chat and profile, editing account settings, premium signup
- **Gary:** documentation and organization, blocked list on profile + unblock users, Assignment 2 CI/CD pipelines
- **Amy:** Event filtering, event pages, parts of event form, premium event advertising, styling, fixing bugs
- **Kai:** Event home page, event info page, event forms, joining, creating, and editing events