Lunch With Friends

Amanda Zhang, Amir Hegazy, Brenna Chen, Jaemyung Choi, Soumika Guduru

What is Lunch With Friends?

It's a web application designed to help you meet people who are interested in going to the same restaurants.

Flow of Our App

- Log in with Google
- Search for a restaurant
- Click the restaurant you'd like to visit
- Select a group or single option
- Get matched to another user
- Enter a chat room with them!

Continuing Professional Development

- **Server endpoints**
- **Path parameters**
- JSON Object
- Object serialization into a database
- Session variables
- Google Maps API
- Google Sign-in/OAuth 2.0 API
- Google Cloud Platform

Server Endpoints + Path Parameters

```java
package CSCI201_LunchWithFriends;

import java.io.IOException;

@ServerEndpoint(value = "/chatroomServer/{userID}/{room}")
public class ChatRoomServer {
    /*
     * Map of key: room number  value: set of sessions (iow set of users in a chat)
     * in easier terms- the key is the chatName/roomName, and the value is the set of users in that chat
     */
    private static Map<String, Set<Session>> chatRoomMap = (Map<String, Set<Session>>)Collections.synchronizedMap(new HashMap<String, Set<Session>>());

    @OnOpen
    public void open(Session session, @PathParam("userID") String userID, @PathParam("room") String room) throws IOException {
        session.getUserProperties().put("userID", userID);
        session.getUserProperties().put("room", room);
        /*
         * getChat --> Either finds the room this user will chat inside
         *      if no such room exists --> Creates chatroom + stores it inside chatRoomMap as <room, this users session>
         * chatroom will be a set of user sessions who can message each other
         *      -->in this case, the set of other users this user session with chat with
         */
        Set<Session> chatroom = getChat(room);
        /*
         * Adding current user's session into created chatroom
         */
        chatroom.add(session);
        session.getBasicRemote().sendText(makeText("Chat System", "you are now connected as "+userID+" in ChatRoom: "+room+"!"));
    }

    @OnMessage
    public void recievedMessage(String message, Session session) throws IOException
    {
        String userID = (String) session.getUserProperties().get("userID");
        String room = (String) session.getUserProperties().get("room");
        /*
         * find the chatroom the user is in */
        Set<Session> chatroom = getChat(room);
        /*
         * Sends message to all chat users in the same chatroom */
        for(Session rs : chatroom) {
            if(rs.isOpen()) {
                rs.getBasicRemote().sendText(makeText(userID, message));
            }
```

Continuing Professional Development

- Server endpoints
- Path parameters
- **JSON Object**
- Object serialization into a database
- Session variables
- Google Maps API
- Google Sign-in/OAuth 2.0 API
- Google Cloud Platform

JSONObject

Server Endpoint

```
private String makeText(String userID, String msg) {
    JsonObject jObj = Json.createObjectBuilder().add("msg", userID+": "+msg).build();
    StringWriter sWriter = new StringWriter();
    try(JsonWriter jWriter = Json.createWriter(sWriter)){
        jWriter.write(jObj);
    }
    return sWriter.toString();
}
```

Chat Client (chatBasic.jsp)

```
var getp1 = "${userID}";
var p1 = String(getp1);
var getp2 = "${room}";
var p2 = String(getp2);

var socket = new WebSocket(
        "ws://localhost:8080/LunchWithFriends/chatroomServer" + "/" + p1
                + "/" + p2);

socket.onmessage = function processMessage(recievedMessage) {
    var mdata = JSON.parse(recievedMessage.data);
    if (mdata.msg != null)
        msgTextArea.value += mdata.msg + "\n";
}
```

Continuing Professional Development

- Server endpoints
- Path parameters
- JSON Object
- **Object serialization into a database**
- Session variables
- Google Maps API
- Google Sign-in/OAuth 2.0 API
- Google Cloud Platform

Continuing Professional Development

- Server endpoints
- Path parameters
- JSON Object
- Object serialization into a database
- **Session variables**
- Google Maps API
- Google Sign-in/OAuth 2.0 API
- Google Cloud Platform

Continuing Professional Development

- Server endpoints
- Path parameters
- JSON Object
- Object serialization into a database
- Session variables
- **Google Maps API**
- Google Sign-in/OAuth 2.0 API
- Google Cloud Platform

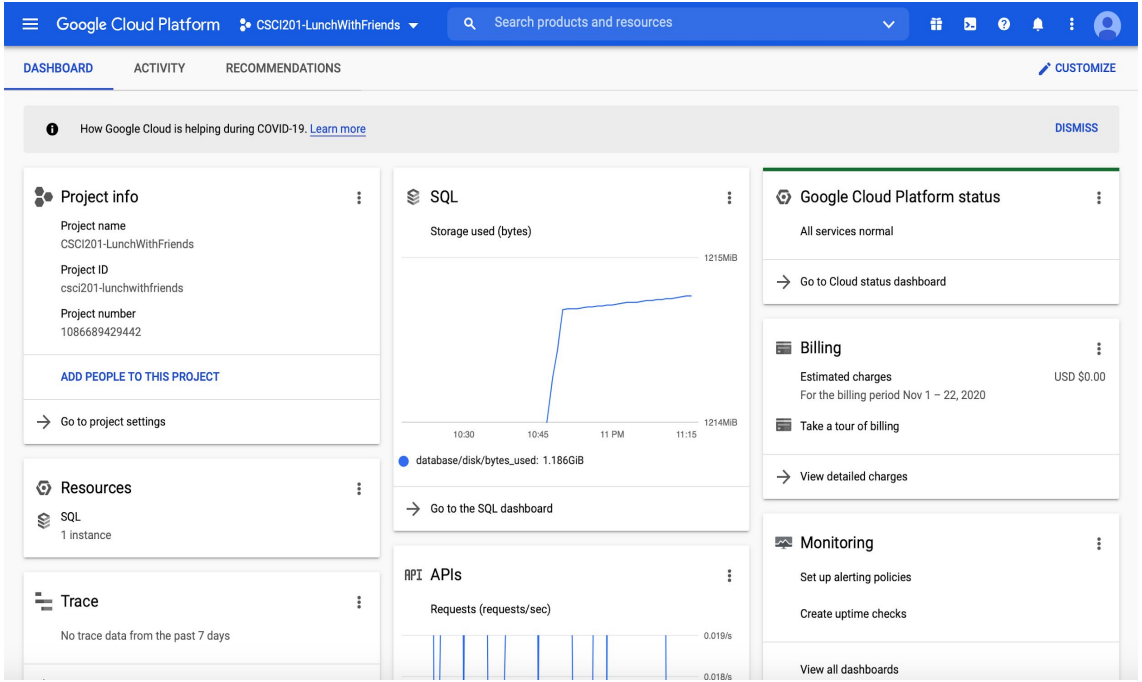
Continuing Professional Development

- Server endpoints
- Path parameters
- JSON Object
- Object serialization into a database
- Session variables
- Google Maps API
- **Google Sign-in/OAuth 2.0 API**
- Google Cloud Platform

Continuing Professional Development

- Server endpoints
- Path parameters
- JSON Object
- Object serialization into a database
- Session variables
- Google Maps API
- Google Sign-in/OAuth 2.0 API
- **Google Cloud Platform**

Google Cloud Platform



Skills and Tools

- Eclipse
- Apache Tomcat
- GSON
- Javax
- Google Cloud Platform
- MySQL
- Yelp API
- Google Maps API
- OkHttpClient

Design and Development Decisions

What worked

- Strong communication + helping each other on tasks
- Clear GUI mockup & database schema to base front & back-end
- Clear step-by-step process for user to follow in detailed design document
- Flexible project design to tailor to issues/time constraints

What didn't work

- Difficulty connecting parts together, not enough familiarity with team members' code
- Testing was difficult b/c many parts were interdependent
- Not setting hard deadlines for coding portions -> lots of delay due to interdependent parts

Outside Courses

- CS103: Introduction to Programming
- CS104: Data Structures and OOP
- CS270: Introduction to Algorithms
- ITP104: Web Publishing
- EE109: Introduction to Embedded Systems

Data Structures

- HashMap
- ArrayList
- Synchronized Map
- Synchronized Set
- Map
- Set

Why did we choose these data structures?

Multithreading + Networking

- Chat Server Endpoint
- Synchronized Java Collections
- Apache Tomcat + Servlets
  - Servlet containers use threading to serve the same servlet to different requests

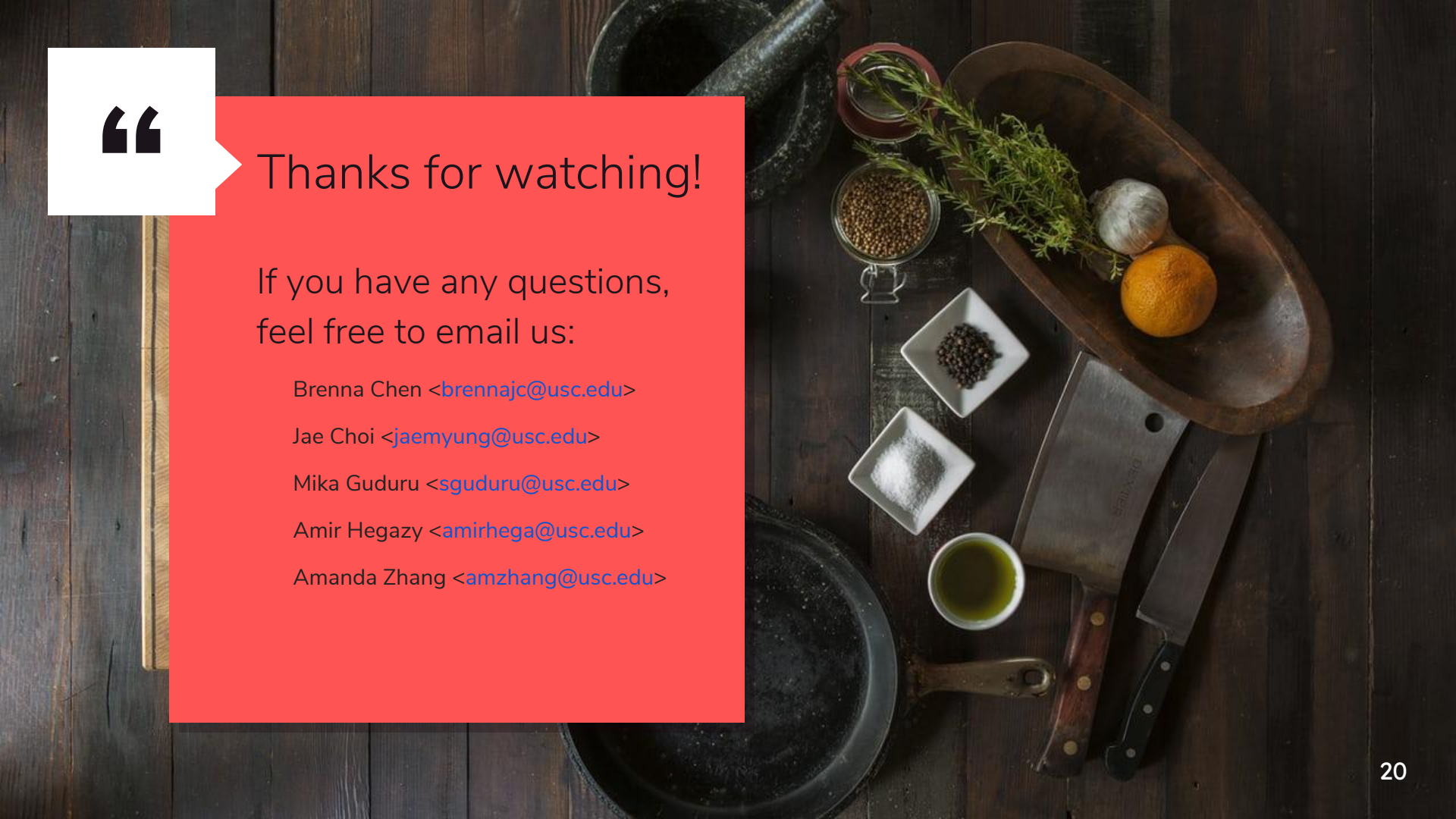
User Login Functionality

Guest User

- Can search & view restaurants

Authenticated User (Log-in through Google)

- Can search & view restaurants
- Can see other people interested in the same restaurant
- Can be matched to other users
  - A chatroom will be created between the users



Thanks for watching!

If you have any questions, feel free to email us:

Brenna Chen <brennajc@usc.edu>

Jae Choi <jaemyung@usc.edu>

Mika Guduru <sguduru@usc.edu>

Amir Hegazy <amirhega@usc.edu>

Amanda Zhang <amzhang@usc.edu>

Code Demo