SOUND MINGLE

TEAM MEMBERS:

Ashith Shetty (21bcs019)-Frontend

Mandar Dighe (21bcs033)-Frontend/Backend

Ratnesh Kherudkar (21bcs091)-Design/Frontend

Shravya Kanalli (21bcs113)-Design

Shreyansh Tiwari (21bcs114)-Frontend/Backend

AIM

As music is one of the very ways in which a lot of people in today's world connect. We the team of sound mingle came up with this idea to provide a platform like no other to make connecting with random people and discovering new music becomes very easy.

TARGET AUDIENCE:

Our main audience are all the Music lovers who would use the platform to discover and listen to new music, connect with other users, and share their own music preferences.

Musicians and artists can also use this platform to promote their music. Where our platform would be a place where they can promote their music organise group jamming sessions and increase their reach.

Music labels and distributors are the Companies that would use the platform to promote their artists and music catalogues, and to connect with new audiences.

Advertisers and sponsors that would use the platform to advertise their products or services, or to sponsor events or promotions related to music. This is also one of our revenue generation models.

SOFTWARE DEVELOPMENT METHODOLOGY USED:

We used the spiral methodology This model is a combination of the waterfall and prototyping models and involves a cyclic approach to software development.

In this the project is divided into smaller parts and each part is developed, evaluated, and tested before moving on to the next part. This model is best suited for complex and large projects that require regular evaluation and feedback.

We held meetings only when required that was usually during the start of each spiral but with the help of Github we could keep collaboratively work on the project.

And with prototyping platform we used like framer designing the UI also could be done collaboratively where we could each view the changes and design made by each other.

SOFTWARE DEVELOPMENT LIFECYCLE:

1) Requirements analysis:
   We started with finalising our idea the major features our website would have and all the technical requirements to implement all the features were discussed upon.

   <u>Conclusion of requirement analysis:</u>

   a) ALL the features our website would consist of:

      i) DUO: In this feature two random individuals can connect and listen to the song that they would like to recommend. Here you have a real time music listening feature where both the connected users would be able to listen to the same song in real time.
      There is an integrated real time chat feature as well where the individuals can discuss the song, the artist and anything else they would like to talk about and get to know each other and their music tastes.

ii) GROUP: In this feature there would be several music rooms that would be created on the basis of genre, artists, raga, ect. I can pick any music room that would like to join where I can add in my own song suggestions which get added to the current Queue.

iii) PLAYLIST CREATION: The songs that have been played in the duo sessions and the group sessions automatically get created into a playlist that gets saved to your Spotify account.

b) MAJOR REQUIREMENTS:

i) We used the Spotify music player as it is currently one of the most widely used and has a very good range of music.
For the above we used the Spotify API.

ii) The home page has the use of Open AI API where there is lyrics of the songs of the top 3 artists according to your songs listened and liked being generated randomly which makes our home page more indulging.

c) THE TECH STACK USED:
We used the MERN stack(mongoDB, Express Js, React Js, Node Js).

2) DEVELOPMENT:

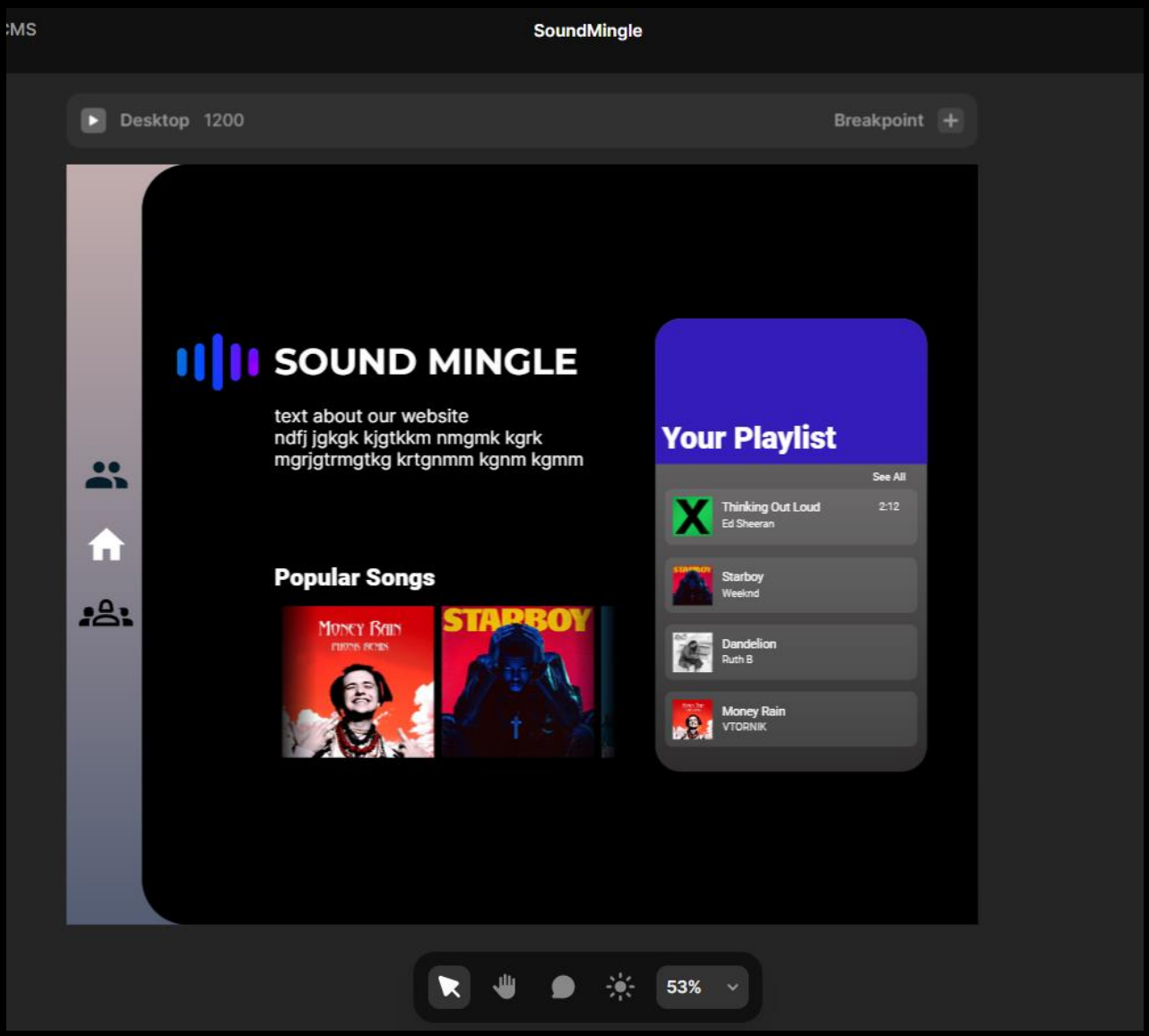
a) Wireframing:

The very first was to have a very rough idea about how our website would look. For this we used the very basic pen and paper method for drawing down the wireframe ideas.

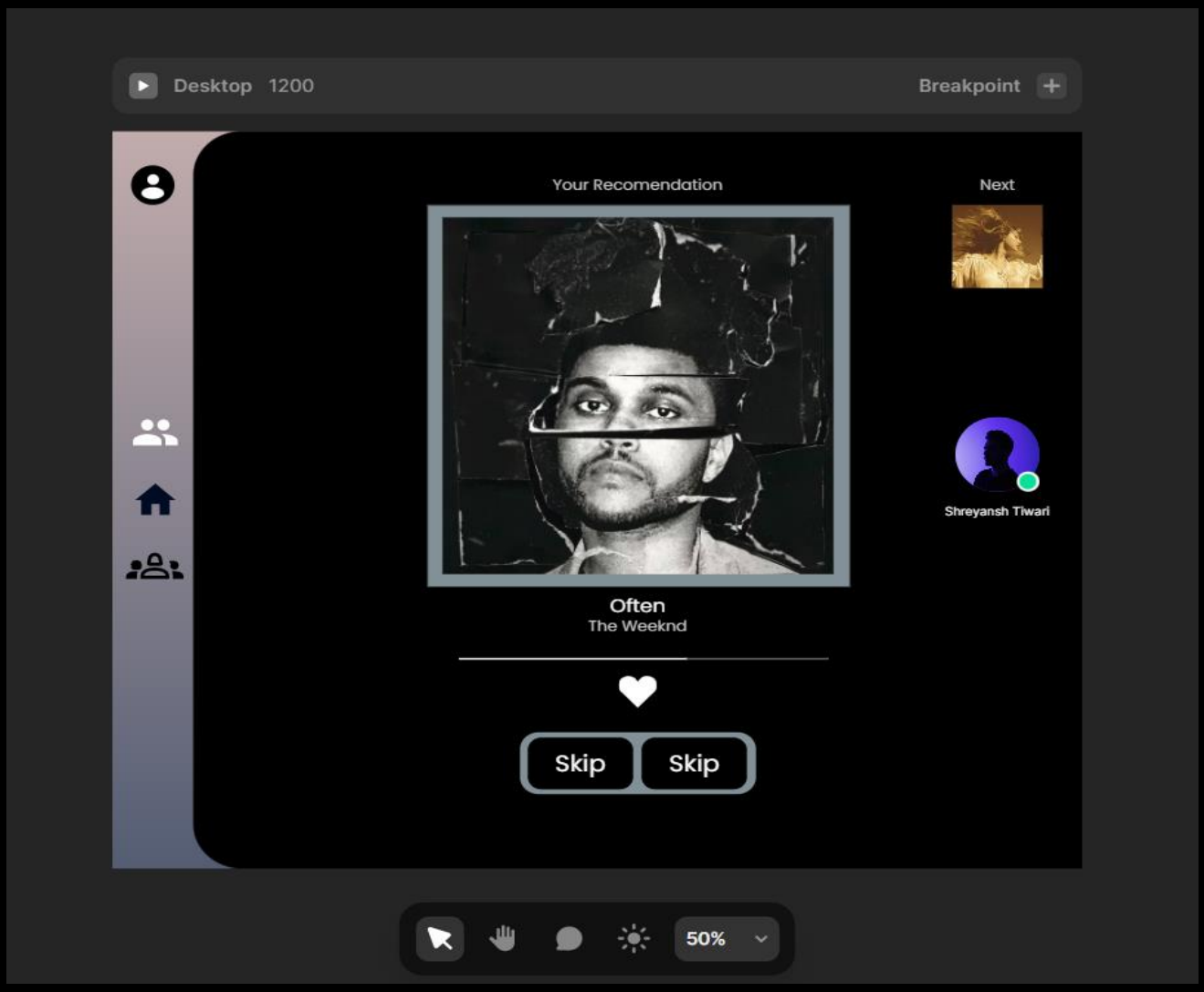
b) Prototype:

Next was to design the UI of our website by creating a high-fidelity prototype for which we used Framer which is a one of the many platforms for prototype designing.

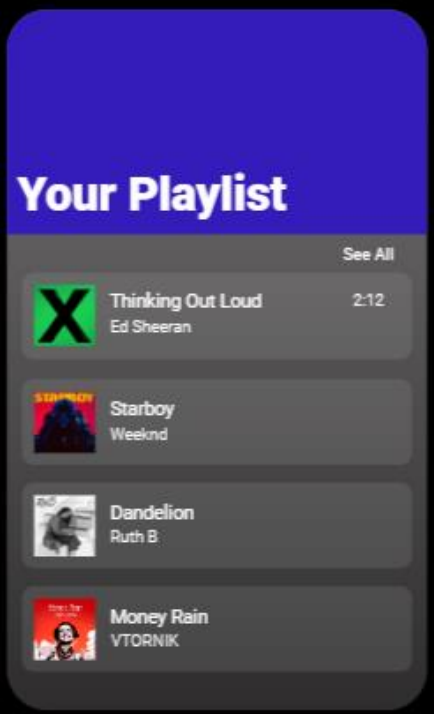
HOME PAGE:



DUO:



PLAYLIST:



c) Frontend Development:
After having the basic view of our website designed with the help of the prototype. We coded the design and generated the frontend of the website.

d) Backend and API Connections:
As mentioned earlier we used the Spotify API and The Open AI API in our website we had to connect these with our website so that the features that we decided on could be executed.

<u>The below is the algo that we wrote for the random matching in the:</u>

```
const express = require("express");
const app = express();
const http = require("http");
const { Server } = require("socket.io");
const cors = require("cors");
const { v4: uuidv4 } = require("uuid");
const fs = require("fs");
const fsp = require("fs").promises;
const { log } = require("console");
const createRoom = require("./controllers/createroom");
const { loadavg } = require("os");
const path = require("path");

app.use(cors());

const server = http.createServer(app);

const io = new Server(server, {
  cors: {
    origin: "http://localhost:3000",
    methods: ["GET", "POST"],
  },
});

io.on("connection", (socket) => {
  console.log(`User Connected: ${socket.id}`);

  socket.on("join", async (data) => {
    const filePath = path.join(__dirname, "data", "rooms.json");
```

```
    const fileData = await fsp.readFile(filePath, "utf8");
    const roomsData = JSON.parse(fileData);

    const incompleteRooms = roomsData.rooms.filter((room) => !room.isfull); // filter the
incomplete rooms
    const incompleteRoomIds = incompleteRooms.map((room) => room.room_id); // extract the
room_ids of the incomplete rooms
    var curRoomId = null;

    if (incompleteRoomIds.length === 0) {
      const newRoomId = uuidv4();
      await createRoom(data, newRoomId);
      socket.join(newRoomId);
      curRoomId = newRoomId;
    } else {
      roomID = incompleteRoomIds[0];
      rooms = roomsData.rooms;
      const room = rooms.find((room) => room.room_id === roomID);

      room.user_2 = data;
      if (room.user_1 && room.user_2) {
        room.isfull = true;
      }
      socket.join(roomID);
      curRoomId = roomID;
      io.to(curRoomId).emit("roomsData", room);

      await fsp.writeFile("./data/rooms.json", JSON.stringify({ rooms }));
    }
  });

  socket.on("skipped", async (data) => {
    const filePath = path.join(__dirname, "data", "rooms.json");
    const a = await fsp.readFile(filePath, "utf8");
    const userData = JSON.parse(a);
    const rooms = userData.rooms;
    const room = rooms.find((r) => r.room_id === data.roomid);

    // if the room was found, update the value of skipped for the specified user

    if (room.user_1.userName === data.userName) {
      room.user_1.skipped = true;
    } else if (room.user_2 && room.user_2.userName === data.userName) {
      room.user_2.skipped = true;
    } else {
      console.log("Room not found.");
    }
    await fsp.writeFile("./data/rooms.json", JSON.stringify({ rooms }));

    // check if both users have skipped
    if (room.user_1.skipped && room.user_2.skipped) {
      io.to(room.user_1.socketid)
        .to(room.user_2.socketid)
        .emit("closeConnection");
```

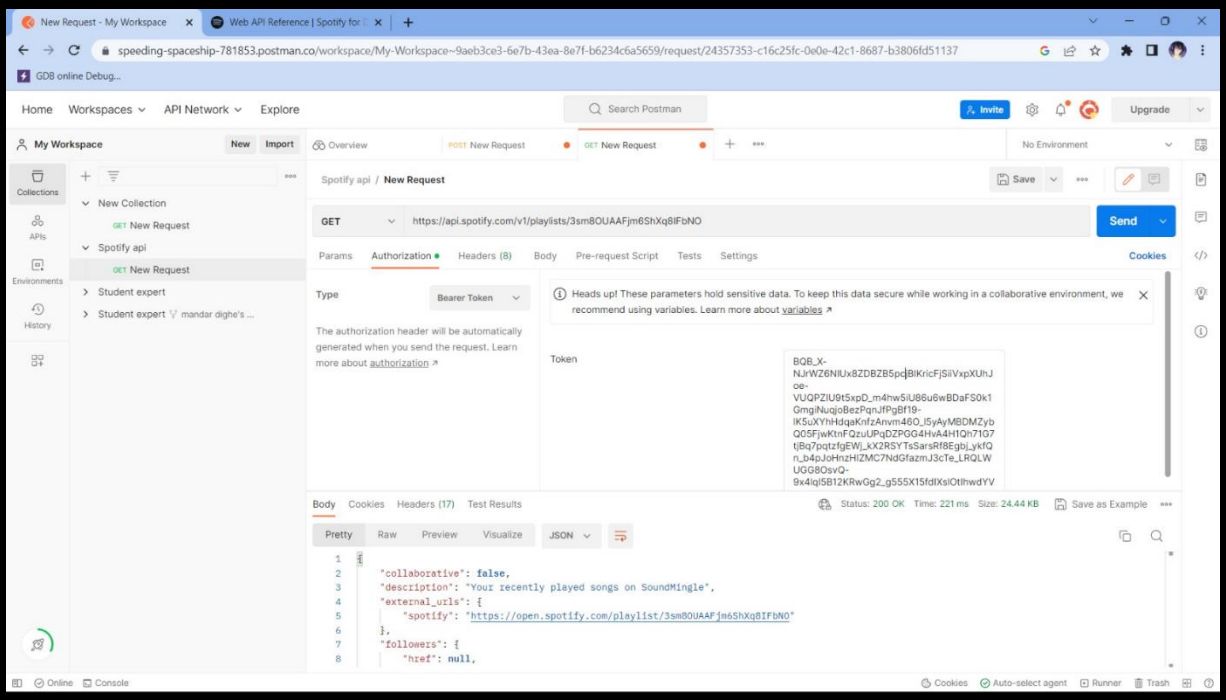
```
    } else {
      io.to(room.user_1.socketid).to(room.user_2.socketid).emit("turnRed");
    }
  });

  socket.on("disconnect", () => {
    console.log("user disconnected");
  });

  socket.on("message", (data) => {
    console.log(data);

    io.emit("message", data);
  });

  socket.on("creategroup", async (userdata, roomdata) => {
    const newRoomId = uuidv4();
    socket.join(newRoomId);
    roomdata.room_id = newRoomId;

    roomdata[userdata.userid] = userdata;
    const jsondata = await fsp.readFile("data/grouprooms.json", "utf8");
    const obj = await JSON.parse(jsondata);
    obj.rooms.push(roomdata);

    const newData = JSON.stringify(obj);
    await fsp.writeFile("data/grouprooms.json", newData);
  });

  socket.on("joingroup", async (groupId, socketId, userdata) => {
    const jsondata = await fsp.readFile("data/grouprooms.json", "utf8");
    const obj = await JSON.parse(jsondata);
    const group = obj.rooms.find((g) => g.group_id === parseInt(groupId));

    if (!group) {
      console.log(`Group with ID ${groupId} not found`);
      return;
    }

    // Add the user ID to the group's users array
    group.users.push(userdata.userid);

    group[userdata.userid] = userdata;
    obj.rooms.push(group);

    const newData = JSON.stringify(obj);
    await fsp.writeFile("data/grouprooms.json", newData);
  });
});

server.listen(3001, () => {
  console.log("SERVER IS RUNNING");
});
```

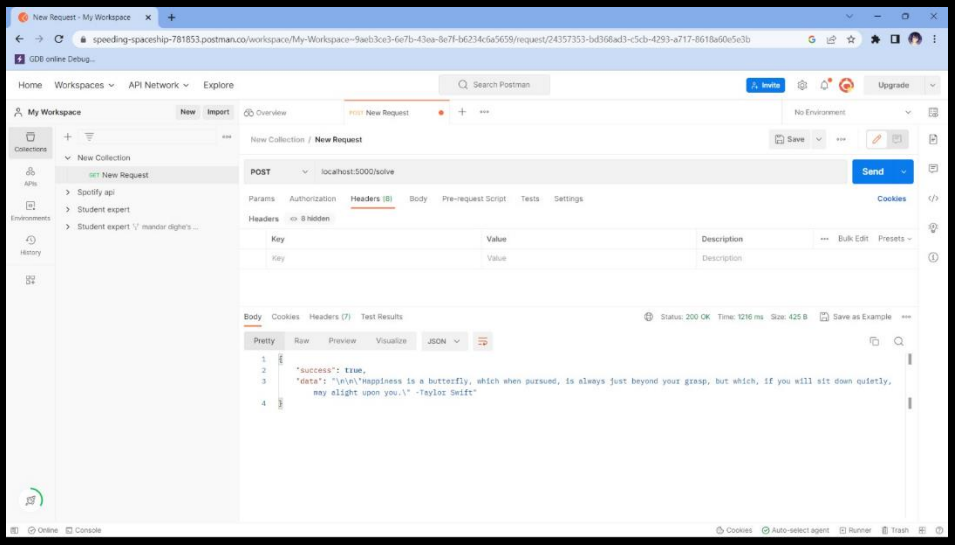
Explanation of the above code:

When a user selects a song and clicks match the username and song data goes to
the backend.
When another user click on matches it tries to search open rooms which have
“isfull” parameter as false if it does not find any open room it creates a new room Id
and waits until other user matches it.
As the two users are matched, we create an array in backend which contains the track
uri of both the songs selected by users as element then this array is passed to both
the user and the music player take this array as input and play the songs in same
order for both the users.
When a user 1 clicks skip, a message is sent to backend that user 1 skipped and the
skip button in the users 2 website becomes red and when user2 clicks skip a
message is sent to backend that user 2 skipped when the backend code gets to
know that both users skipped the it ends the room.
When the room end the array of songs goes into the Soundmingle playlist in both the
users Spotify account.

e) Testing:

We used Postman for testing the API connections as this is one of the major aspects of our project.





BUSINESS MODEL:

Market Analysis:

The music industry is a vast and growing market. According to a report by IFPI, the global music industry grew by 7.4% in 2020, with a total revenue of $21.5 billion. In addition, the streaming revenue of the music industry grew by 19.9%, which shows the demand for music streaming platforms. This indicates that there is a significant market for a music discovery and social platform that offers a new way to connect with other music lovers.

This makes a good market space to enter into.

REVENUE GENERATION MODEL:

1) Advertisements: One of the major methods of our revenue generation model would be through the ads played on our site. But if you would like to avoid the ads you may take our subscription.

2) Music Artists: The musicians who would like to promote their music and create a private room just so that they can organise a jamming session with their music followers or to promote their music through ads.