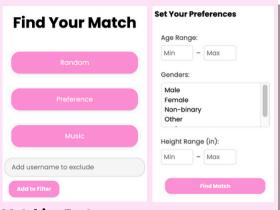
CS4530 Final Project

Group 208: Toto, Sam, Zhaorui

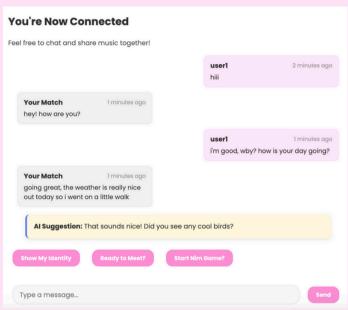
Pawfect Match is a platform for the "single-looking-to-mingle" community. We look to create a space where singles can find companionship over shared interests, engage in meaningful discussions, and stay connected with prospective lovers.





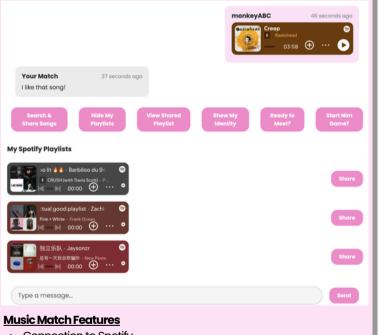
Matching Features

- 3 different match types: (1) random; (2) preference-based; (3) music-based (via Spotify)
- Able to avoid matching with specified users by entering their usernames



Date Room Features

- · Sends users to a "date room" once match is made
- Supports anonymous mode. Users can choose to show identity when they feel comfortable
- · Al chat bot that gives suggestions on responses is available
- "Ready to Meet" button that sends a connection request to the match
- Button to start a Nim game with match



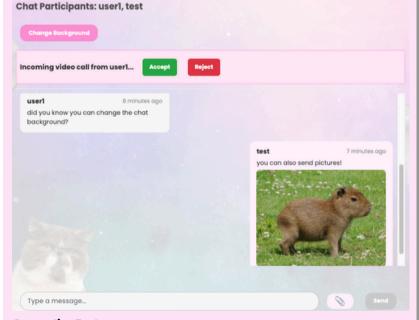
- · Connection to Spotify
- Matches users based on their top Spotify genres
- Able to send songs and playlists to match
- Can view shared playlist

Tech Stack:

Pawfect Match was built using the MERN stack (MongoDB, Express.js, React, and Node.js). The backend is powered by Express and Node.js, while the frontend is built with React. We use Socket.IO to enable realtime features such as live messaging and matchmaking, and MongoDB serves as our database to store user data, chat history, and match information.

Future Work:

- Add feature to enable real-time music playing in chat
- Add a calendar view to schedule video call dates
- Enable connection with other music apps like Apple Music
- Support more variety of games



Connection Features

- Supports follower / following connections to stay in touch with matches
- Able to send pictures in chat
- Able to change chat background image
- Ability to voice & video call

Overall Design Decisions:

We utilized modular controller-service architecture in the backend for separation and maintainability. We ensured our database schemas were designed with flexibility to accommodate the different match types and evolving user preferences. On our frontend, we implemented reusable hooks and components to easily extend new features without disrupting the UI. Our team leveraged multiple APIs (like Spotify Web API, Toastify) to enhance user experience.

Our codebase is available on **GitHub** Our demo is available on Render









