Duet: connecting out-of-state undergrads through music.

Product description (20%)

People need to make connections, meet others through music tastes.

Priorities:

1. See music others enjoy
2. Protect private information until both opt in to share it (DM lock, private info, location)
3. Propose matches immediately, and give better matches over time (with usage)

In order to accomplish this, we'll start with a Webapp and port it to a mobile app. Using `flutter` can allow us to develop for both.

Our users need to

P0

- Link their Spotify Account, give API access
- Feed of profiles (every user has a profile), shown by compatibility. Be able to mark profiles as positive, neutral, negative. This leads into direct messaging after mutual interaction.
- Must have user profile; main page with profiles; matching, DM; concerts near me
- Matching algorithm: music distribution, total songs listened to, algo work
- Must have a personalizable bio on profile
- Conversations page, same as any messaging app
- reporting, blocking feature

P1

- Show the two user group size; show them *why* they were connected (overlap in music listening)
- Users could also create taste profile if not much spotify data: list their top 10 current favorite songs, genre (todo, this changes often for some people?). Sometimes listen to a meditation before bed, but do not want to inform my profile. Emphasize this is for fun, not commited one way door.
- L/R swipe for "what are you in the mood for right now?", reprompt every now and then "what are you listening to --> what do you want to represent you". Goes through top k (15) weighted songs, user swipes to decide what informs their profile.
- Icebreakers, "a song I would play at X" prompts; both users have those shown in profiles. Start convo by **replying** to those prompts, starts the conversation.
- Let users decide what they want to look for: similar music, different music, preferred/minimum filters, location, timezone/message reply history; user-led filtering

P2

- expand to Apple Music, Soundcloud if have time / need more predictive signal

- Concert feature, go as a group, users that go to a concert are placed to group chat to coordinate before/after (attend/not going as a FB event)
- concert groups, divided to ensure small enough size? coordinate by location/preference/other

Use cases (functional requirements) (30%)

Eclectic Tastes

Andrey Risukhin

- Goal: match with people who also enjoy the Portal 2 OST, Minecraft OST, and instrumental accordion music
- Actors: Seeker, system, matchee.
- Triggers: Seeker installs the app, opens it for the first time.
- Preconditions: Matchee exists in the system database.
- Postconditions: Seeker and Matchee both exist in the system, and they have opened a DM.
- Steps
    1. Seeker creates their profile, selecting the songs (which **may not be auto-suggested**) that they want to seek for.
    2. System takes their preferences, returns matched candidates shown one at a time.
    3. Seeker annotates the profiles shown as (+), (0), (-): want to connect, maybe next time, never want to see again.
    4. System alerts each profile with (+) that they have been (+)'d be Seeker.
    5. Matchee recieves this alert in their notifications tab, clicks on it to open the Seeker's profile. 5.0.1 System marks this notification as "read" but does not delete. 5.1.1 Matchee marks Seeker as (+) 5.1.2 System opens a DM betweek Matchee and Seeker
- Variations 5.2.1 Matchee marks Seeker as (0) 5.2.2 System closes the Seeker profile, some internal change on ranking to show later 5.3.1 Matchee marks Seeker as (-) 5.3.2 System never shows this profile again
- Exceptions
    - The match fails to be displayed, as if marked (-) and if no other profiles are shown (undefined behavior). Resolution: a "no users to match with remain" message is displayed, perhaps allow users to reconsider their (-) marked profiles.

Concert Meetup

Jacob He

- Goal: Coordinate attending a concert together with other users with similar music tastes
- Actors:
    - Primary Actor: User A (initiates the meetup by expressing interest in a concert).
    - Secondary Actors: Users B, C, and D (potential concert-goers with overlapping music tastes).
- Triggers: User A sees a notification about an upcoming concert featuring an artist they like and selects the option to “Find Concert Buddies.”
- Preconditions:
    - User A has linked their Spotify account, and the app knows their favorite artists.
    - The concert feature is enabled, with event details for the user's area.
    - Users B, C, and D have also expressed interest in the same artist or genre.

- Postconditions (Success Scenario):
    - A group of compatible users (Users A, B, C, and D) is formed.
    - The group coordinates logistics (e.g., transportation, meetup location) in a group chat.
    - The attendees enjoy the concert together and optionally share their experience in the app.
- List of Steps (Success Scenario):
    - Discovering the Concert:
        - User A logs into the app and sees a banner for an upcoming concert by one of their favorite artists.
        - User A selects the “Find Concert Buddies” option.
    - Group Formation:
        - The app scans other users with overlapping music preferences who have expressed interest in the same artist or concert.
        - The app proposes a group of Users A, B, C, and D, along with compatibility highlights (e.g., “You all listen to 80% of the same songs by this artist!”).
        - User A approves the group formation.
    - Icebreakers and Planning:
        - A group chat is created for Users A, B, C, and D, with conversation starters like:
        - “What’s your favorite track from this artist?”
        - “What’s your go-to pre-concert hype song?”
        - Group members coordinate logistics (e.g., “Who’s driving?” “Let’s meet at [location] before the show.”).
    - Concert Meetup:
        - The group meets at the concert, having already broken the ice through the app.
        - They enjoy the concert and share live photos or highlights in the group chat.
    - Post-Concert Engagement:
        - After the concert, the app prompts Users A, B, C, and D to rate their experience together.
        - Users can continue chatting, share playlists, or plan future meetups through the app.
- Extensions/Variations of the Success Scenario:
    - Customizing the Group Size:
        - If User A prefers smaller groups, the app proposes a subset of 2-3 users instead of a larger group.
    - Post-Concert Playlist Creation:
        - The app automatically generates a playlist combining the group’s favorite songs by the artist and similar tracks.
    - Shared Interests Beyond Music:
        - The group discovers shared interests (e.g., favorite food spots) through their profiles, enhancing the meetup experience.
    - Event-Specific Badges:
        - Users who attend the concert together earn a special badge for their profiles (e.g., “[Artist] 2025 Tour Attendee”).
- Exceptions (Failure Conditions and Scenarios):
    - No Interested Users Nearby:
        - The app fails to find enough users interested in the concert within a reasonable distance.

    - Resolution: The app suggests alternative concerts or virtual watch parties for live-streamed events.
    - Incompatible Group Dynamics:
        - Users in the group don’t engage in the chat or drop out of the meetup.
    - Resolution: The app allows User A to disband the group and retry with new matches.
    - Safety Concerns or Inappropriate Behavior:
        - A group member exhibits inappropriate behavior in the chat or at the concert.
    - Resolution: The app enables reporting, and the user is removed from the group and reviewed for potential account action.

Discover new Music and Artist

Selim Saridede

- Goal: Allow users to discover new music and artists based on their preferences and listening habits.
- Actors: User, system
- Triggers:
    - User opens the app and navigates to the "Discover" section.
    - User listens to a new song or artist, and the system recommends similar music.
    - User matches with a new person and checks their favorites
- Preconditions:
    - User is logged into their account.
    - The system has access to the user’s listening history or preferences.
    - The system has a recommendation algorithm based on user preferences.
- Postconditions (success scenario):
    - The user discovers new music or artists they enjoy.
- List of steps (success scenario):
    - User opens the app and navigates to the "Discover" or "Recommended" section.
    - The system displays a list of music recommendations based on the user’s preferences.
    - User scrolls through the recommendations and clicks on a song or artist they find interesting.
    - If the user likes the music, they can save it to their favorites and people can see their new music taste.
    - User matches with other people who also liked similar genre/music.
- Extensions/variations of the success scenario:
    - The user can share music or artists they discover with friends through the app’s social features, expanding the discovery experience.
- Exceptions: failure conditions and scenarios:
    - No Recommendations Found: If the system cannot generate recommendations due to insufficient data, it may ask the user to provide music preferences or suggest popular music to get started.

Icebreaker Conversations Based on Shared Song Prompts

Belem Barrientos-Guevara

- Goal: Allow users to start conversations by responding to shared, music-related prompts visible on their profiles.
- Actors:
    - Primary Actor: User initiating the conversation.
    - Secondary Actor: User with a profile containing answered prompts.
    - System: Displays prompts and promotes messaging.
- Triggers:
    - A user views another user's profile and notices an interesting song prompt.
    - The user sends a DM to start a conversation related to it.
- Preconditions:
    - Both users have active accounts.
    - The profile owner has answered at least one icebreaker prompt.
    - The viewer has access to the profile (ex: not blocked).
- Postconditions (Success Scenario):
    - A conversation is initiated directly tied to the icebreaker prompt, creating a new friendship 😃
- List of Steps (Success Scenario)
    - The user views another user's profile.
    - The system displays answered prompts on the profile (ex: "A song I'd play in a high speed chase").
    - The user selects a specific prompt that piques their interest.
    - The system displays a chat window appears with the option to start a conversation.
    - The user sends their message tied to the selected prompt.
    - The system notifies the profile owner of the new message ("[User] sent you a DM!").
    - Both users can continue the conversation within the app.
- Extensions/Variations of the Success Scenario:
    - Seasonal prompts, like "Your favorite vacation song," can be featured for a limited time.
    - Users can answer prompts with a short voice message, creating more personal connections.
    - Users can favorite certain prompts to revisit and engage with later.
- Exceptions: Failure Conditions and Scenarios:
    - Exception 1: The profile owner hasn't answered any prompts.
        - System Response: Display a message encouraging the user to complete their profile or suggest another profile with answered prompts.
    - Exception 2: A user repeatedly ignores icebreaker messages.
        - System Response: Allow the system to deprioritize the uninterested user in future interactions.

Select Top k Songs

Ramon Costa-Patel

- Goal: Making a representative top k favorite songs on the user profile
- Actor: User
- Triggers:
    - User creates profile
    - User selects "redo top k songs" in profile page
- Preconditions:
    - System has access to Spotify data

    - User has a username and password
- Postcondition: User has a representative list of top k favorite songs
- Success case:
    1. User accesses "choosing top k songs" screen
    2. System shows most listened song by user on spotify
    3. User swipes the song right
    4. System adds song to list of top k favorite songs, then shows the next most listened song
    5. Repeat steps iii and iv until a list of size k is done
- Variation: starting at step iii
    - User swipes song left
    - System doesn't add song to list of top k favorite songs, then shows next most listened song Continue as in success case
- Exceptions: user spotify data couldn't be retrieved

Music Taste Tracking System

Keegan Tran

- Goal: Allow the app to track a user’s listening history, identify changes in their music taste, and prompt them to update their top k songs to better reflect their current preferences.
- Actors:
    - *Primary Actor*: User (who owns the profile).
    - *System*: Tracks listening history and provides notifications/prompts.
- Triggers
    - The system detects a significant shift in the user’s listening habits over a set period (e.g., increased frequency of listening to a new genre or artist).
    - The user manually opts to update their top k songs via the profile settings.
- Preconditions
    - The user’s Spotify account is linked, and the app has access to their listening history.
    - The system tracks and analyzes the user’s listening habits continuously.
- Postconditions (Success Scenario)
    - The user updates their top k songs to reflect their current listening preferences.
    - The updated list is visible on their profile, and the matching algorithm incorporates the changes.
- List of Steps (Success Scenario)
    1. The system analyzes the user’s Spotify listening history over the past month (or configurable period).
    2. The system identifies a significant deviation in the user’s most listened-to genres, artists, or songs.
    3. The system notifies the user with a message: **“Your music taste has evolved! Would you like to update your top k songs?”**

4. The user clicks the notification and is directed to the **Top k Songs Update** interface.
5. The system displays a ranked list of the user's most listened-to songs during the period.
6. The user reviews the list and swipes right on songs they wish to add to their updated top k list or left to skip songs they don't feel represent their taste.
7. Steps 5 and 6 repeat until the user selects k songs.
8. The system saves the new top k list, updates the user's profile, and adjusts the matching algorithm accordingly.
9. The system displays a confirmation message: **"Your top k songs have been updated! Check out profiles of people with similar tastes."**

- Extensions/Variations of the Success Scenario
    - User Declines the Update Notification
        - *System Response*: The system waits until the next significant taste shift to prompt the user again.
    - User Adds Fewer Than k Songs
        - *System Response*: The system saves the partial list and prompts the user later to complete it.
    - User Customizes the Period of Analysis
        - *System Response*: The system allows the user to select a custom time frame (e.g., "last week," "last 3 months") for identifying taste changes.
    - Playlist Integration
        - *System Response*: The system suggests creating a new playlist from the updated top k songs.

- Exceptions: Failure Conditions and Scenarios
    - Insufficient Spotify Data
        - *System Response*: N/A as no taste change could be detected.
    - Technical Error During the Update Process
        - *System Response*: Save the user's progress and retry automatically after resolving the issue.
    - User Ignores or Dismisses Multiple Prompts
        - *System Response*: Gradually reduce the frequency of prompts and only notify after major shifts in taste.

Non-functional requirements (10%)

Describe at least three non-functional requirements of your product, e.g., related to scalability, usability, security and privacy, etc.

- Login/authentication system
- secure messaging APIs, ex: Signal protocol
- Terms of Service (don't get sued)
- Scalability (YouTube music, Soundcloud, Apple Music), 1k UW students (subreddit, snapchat, other social media, post on EdBoard) for good UW experience (incentives to share installs, usage, rewards later (tiebreaker of referrals in match priority))
- Safety (blocking, reporting profiles)

- Usability (crash avoid 9.99% for 1000 ms) = track user sessions; easy to navigate (few buttons), low friction to use. Webapp looks good on mobile and desktop!

External requirements (10%)

In addition to the requirements stated above, the course staff imposes the following requirements on your product:

- The product must be robust against errors that can reasonably be expected to occur, such as invalid user input.
- disallow repeat usernames
- flagging offensive stuff
- The product must be installable by a user, or if the product is a web-based service, the server must have a public URL that others can use to access it. If the product is a stand-alone application, you are expected to provide a reasonable means for others to easily download, install, and run it.
- Flutter handles public URL,
- The software (all parts, including clients and servers) should be buildable from source by others. If your project is a web-based server, you will need to provide instructions for someone else setting up a new server. Your system should be well documented to enable new developers to make enhancements.
- Documentation
- The scope of the project must match the resources (number of team members) assigned.
- We get frequent feedback from TAs on this, use scrum to track feasibility

Team info (10%)

Provide a concise summary of the project team and project artifacts. Specifically:

Identify each team member and high level role.

Jobs to be done:

- Store data from Spotify API, other sources. Create matches between profiles using that data. Collect usage data per-user. Use that to inform algorithm feedback.
- User DMs

Roles!

- Belem - Frontend
- Selim - Backend
- Andrey - Data, Algorithms, probably Spotify API; begs for resources
- Keegan - Frontend
- Jacob - Backend
- Ramon - Messaging, DMs, notifications; scrum manager

Link to each project relevant artifact such as your git repo.

- All artifacts on our git repository: `https://github.com/karlabelem/DUET`

List communication channels/tools, expectations, and meeting plans that will keep your team in sync.

- Communicate via Discord
- If need response, ping. If pinged, respond within 24 hours (sooner better! 😃
- Two synchronous meeting slots per week: Tuesday class, Friday 2:30pm - 3:30pm

Team process description (20%)

Schedule

- Week 4 (P0) - profile creation, store profiles, save/load profiles, link user account, store their spotify data in DB.
- Week 5 (P0) - roughly finalized conversations page, mostly done profile feed/recommendations UI and Algorithm. Login screen for frontend. Block profiles. Feedback on P0: Try with friends.
- Week 6 (P1) - icebreakers, top k songs, reasoning for matches shown to users (interpretability). Filtering preferences for searching.
- Week 7 (P1) - rest of P1. Feedback on P1: Try with friends.
- Week 8 (P2) -
- Week 9 (P2) -

Major Risks

1. Single point of failure / tribal knowledge, someone gets sick etc. Mitigate by documenting code, high level plan, keeping track of work in sprints. Pull Requests required, review by 1 other (check comments are clear etc).
2. Cost of resources, PR (Andrey) ask for credits and resources from companies
3. Misestimation of work to be done - flexible schedule, we have P1 weeks, P2 weeks that can be extended with P0 as needed.

External Feedback

- Weeks 5, 7 (end of P0, P1 weeks) user feedback; stakeholder feedback each Thursday with Connor (make him use it)

Modules

- Data: Python, some analytics service (Amplitude?), telemetry (alerter)
- Frontend: Flutter (web/app easy, VS Code extension). Typescript.
- Backend: Firebase (db platform through Google, like Azure)

Process

- Agile, Scrum
- Client meetings on Thursday
- Two weekly standups - use Discord for urgent issues
- Ramon as the Scrum Manager (has experience before)