Friendrr

CMPT 362 Final Presentation

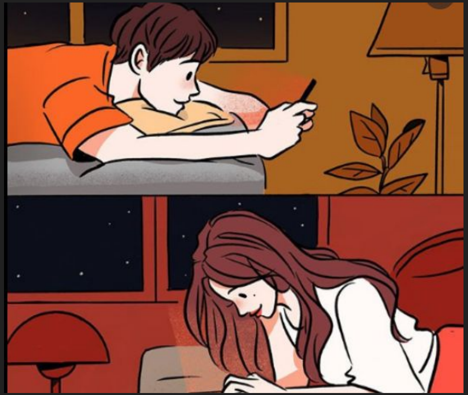
Sterling Tamboline
Benjamin Reedel
Martin Lau
Andrew Fang
Feng Wu

Agenda

- Friendrr Idea
- Reason To Pursue This Idea
- System Architecture
- Thread Design Diagrams
- Core Features List
- Technical aspect of features
    - User matching and becoming friends
    - Search algorithm
    - Usability and UI
    - Real time messaging
- Demo
- Challenges
- Assigned work
- Lessons Learnt

Introduction: Friendrr Idea

- It can be difficult to meet new people
- Feeling lonely
- Find people with similar hobbies
- Our app allows people to connect together in a fun and interactive approach

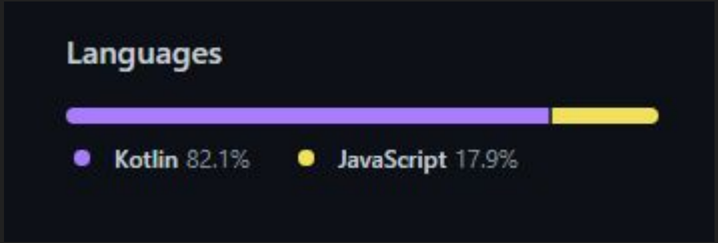


Introduction: Reason to pursue this idea

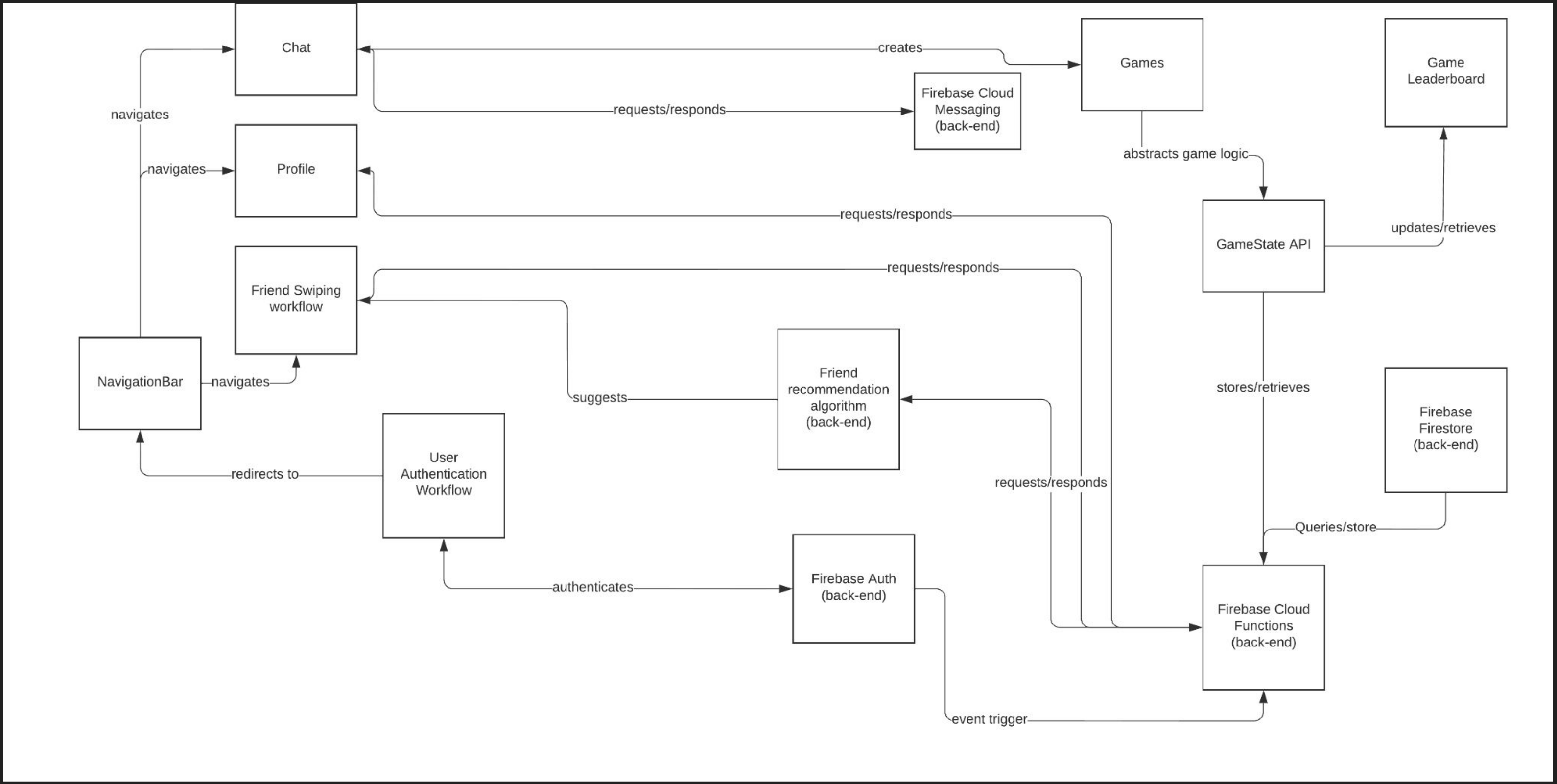
- Many people are using dating apps to meet new friends
  - Defeats the purpose of dating apps
- Friend matching application do not have novelty
  - Facebook social media clones
- Technical complexity
  - Requires cloud technologies
- Inspirations:
  - Tinder: Swipe feature
  - Tiktok: Self-serving recommendation algorithm
  - Facebook messenger: messenger with mini games

Tools/Technologies Used

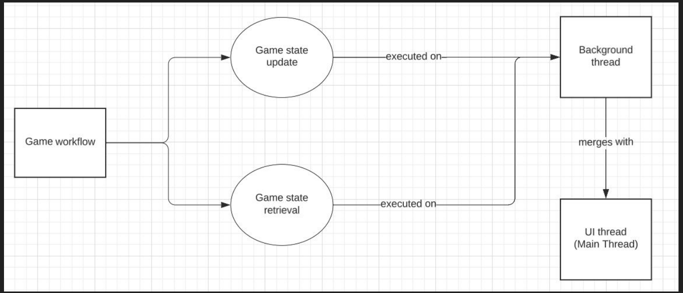
- Cloud technologies
  - Firebase Firestore database
  - Firebase Cloud functions
  - Firebase Authentication
- Technologies
  - Node.js
- Languages
  - Kotlin
  - Javascript
- Additional Tools
  - Figma
  - Git(Github)

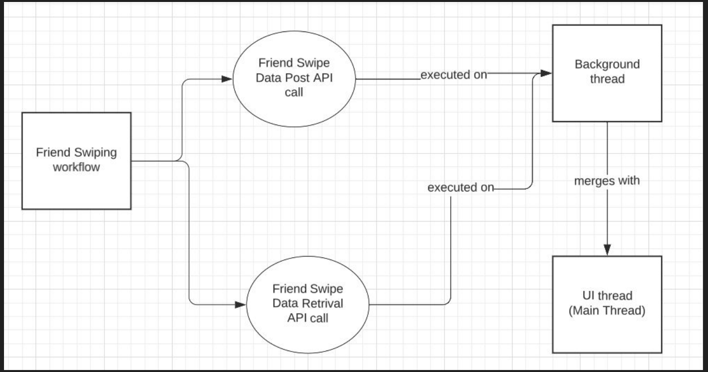


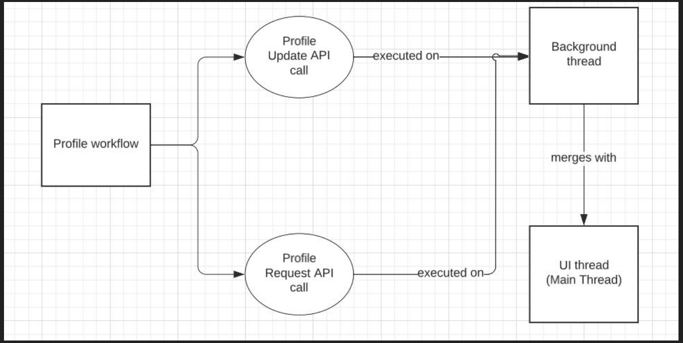
System Architecture(Updated)

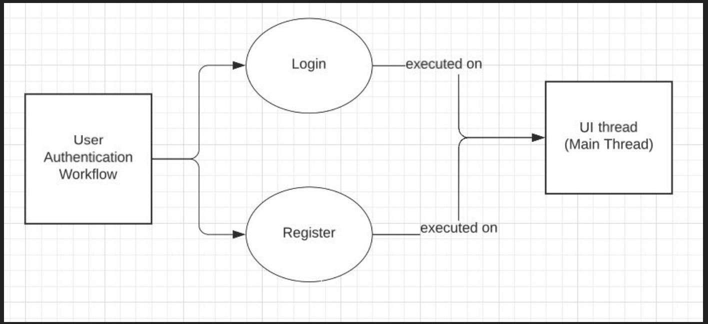


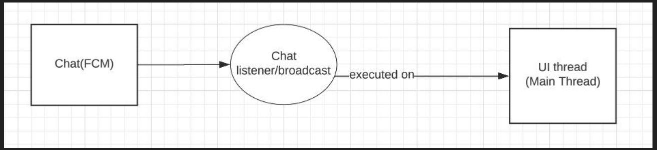
Thread design diagram











Core Feature List

1) Swipe/Match users workflow with automatic audio playback
2) Search algorithm
3) View match history list
4) Add/Remove matched users and friends workflow
5) Real time chat messaging
6) Mini games against matched users and friends
   a) Rock paper scissor
   b) Card matching
   c) Photo guessing game
7) Gamestate API for storing/retrieving multiple games
8) Global leaderboard for games
9) Profile Setup/Edit workflow
   a) Voice recording
   b) Image capturing(with image compression)

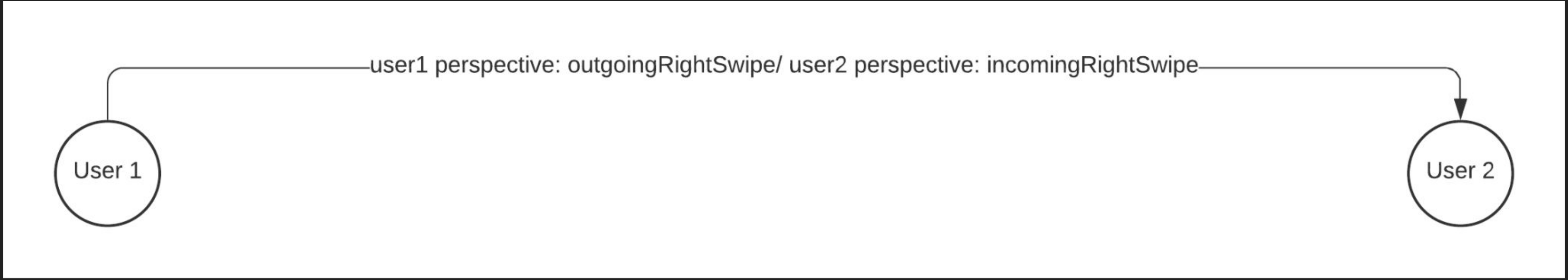
Technical Aspect: Matching and becoming friends part 1

- User matching and adding friends workflow can be abstracted using graphs
- Suppose we have two users represented with two vertices, they are not matched so they do not have any edges

User 1
User 2

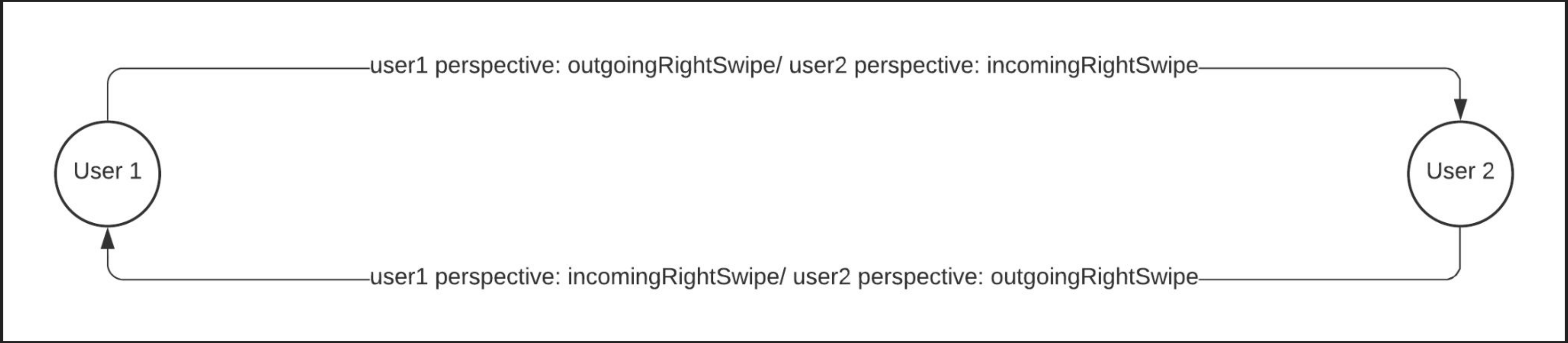
Technical Aspect: Matching and becoming friends part 2

- When user 1 swipes right after seeing user_2 in the match workflow, user_1 wants to match with user_2
- The directed edge from user_1 to user_2 indicates the matched progression of the two users’ relationship status
- Back-end API function performs cycle detection when EACH edge has been added


user1 perspective: outgoingRightSwipe/ user2 perspective: incomingRightSwipe
User 1
User 2

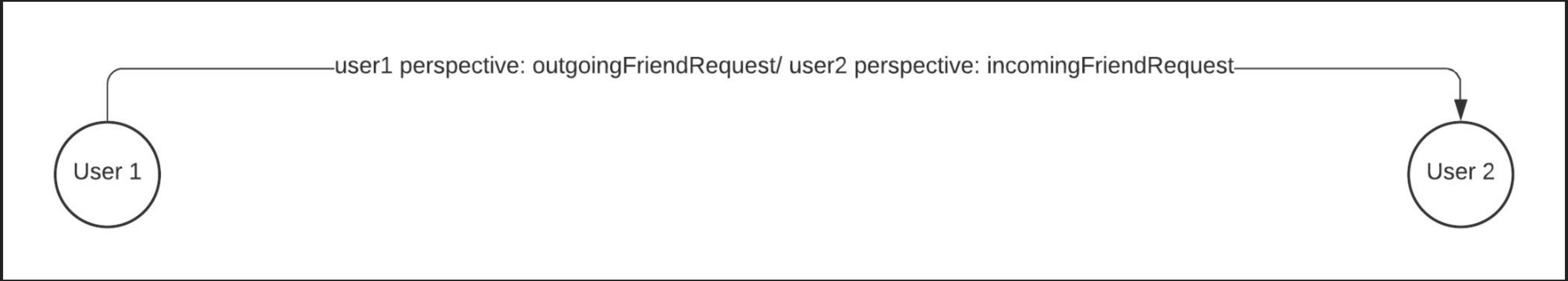
Technical Aspect: Matching and becoming friends part 3

- Consider the scenario where user_2 also swipes right on user_1
- This means both users wants to be matched
- Cycle has been detected -> both users will be placed in a list called **candidateMatchedList** -> both directed edges will be removed afterwards
- Can played a **limited types** of games after being matched


user1 perspective: outgoingRightSwipe/ user2 perspective: incomingRightSwipe
User 1
User 2
user1 perspective: incomingRightSwipe/ user2 perspective: outgoingRightSwipe

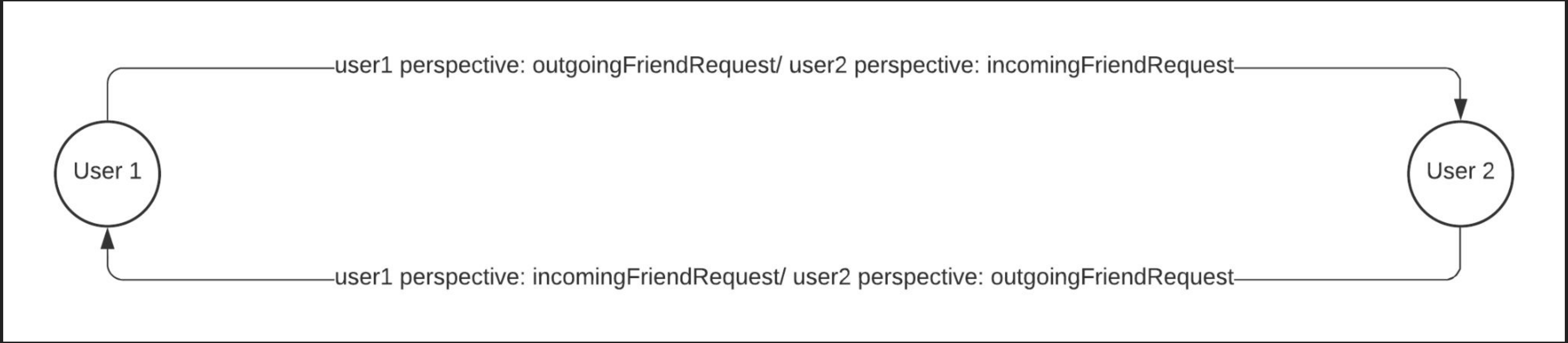
Technical Aspect: Matching and becoming friends part 4

- Consider scenario when user_1 and user_2 are chatting and having a good time **AFTER** being matched
- They now want to be friends. User_1 sends adds user_2 as a potential friend


user1 perspective: outgoingFriendRequest/ user2 perspective: incomingFriendRequest
User 1
User 2

Technical Aspect: Matching and becoming friends part 5

- When user_2 sends a friend request, both users will finally be friends and they will be added into the **friendList** in the database
- In short, **two types of cycles** are needed for the two users to be friends

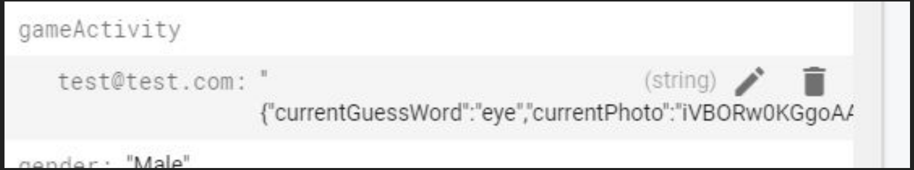

user1 perspective: outgoingFriendRequest/ user2 perspective: incomingFriendRequest
User 1
User 2
user1 perspective: incomingFriendRequest/ user2 perspective: outgoingFriendRequest

Technical Aspect: Search algorithm(Back-end feature)

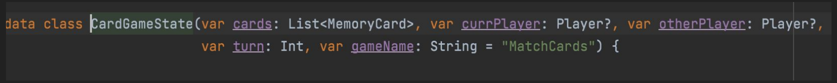
- Filters users based on:
    - Age range
    - Hobbies incommon
    - User active usage level
        - Use friendr app more -> higher appearance occurrences
- Handling special scenarios
    - What if user has swiped right on this specific user(meaning that they want to match the specific user)
    - What if user has already matched with this specific user
    - What if user is already friends with this specific user

Technical Aspect: Gamestate API(Back-end feature)

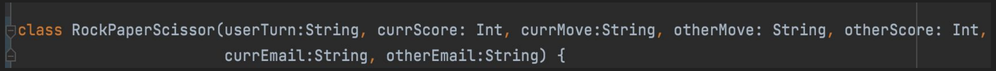
- Different games have different types of attributes to store
- Need an approach to unified attributes of a given game to store in the database
    - Otherwise, additional API have to be created for EACH game
- Gamestate API helps **unified** all game states for EACH game and stores it in the database regardless of the attributes are needed for the game
- Reduces the complexity for those who are implementing the game
    - Abstracts back-end and only need to focus on game logic



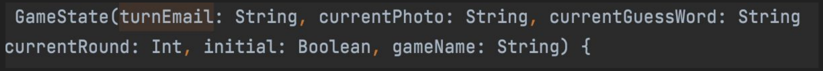
```
gameActivity
    test@test.com: "                                        (string)
        {"currentGuessWord":"eye","currentPhoto":"iVBORw0KGgoA/
gender: "Male"
```



```
data class CardGameState(var cards: List<MemoryCard>, var currPlayer: Player?, var otherPlayer: Player?,
                         var turn: Int, var gameName: String = "MatchCards") {
```



```
class RockPaperScissor(userTurn:String, currScore: Int, currMove:String, otherMove: String, otherScore: Int,
                       currEmail:String, otherEmail:String) {
```

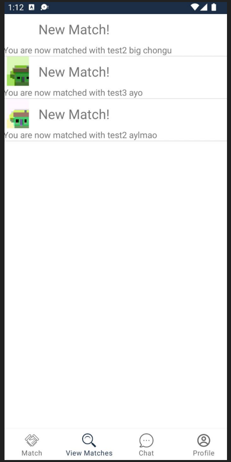


```
 GameState(turnEmail: String, currentPhoto: String, currentGuessWord: String
currentRound: Int, initial: Boolean, gameName: String) {
```

Technical Aspect: Improving Usability And User Interface

- Good UI and usability is IMPORTANT
- Gradual improve each iteration
- Balancing creating new features and enhancing/polishing current features
- Example UI: - Login Screen->

  - View match Screen



New Match!

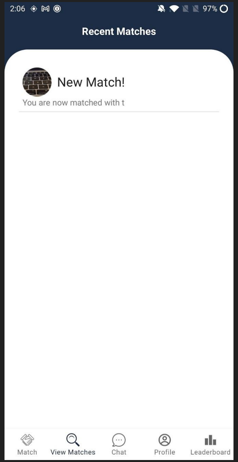
You are now matched with test2 big chongu

New Match!

You are now matched with test3 ayo

New Match!

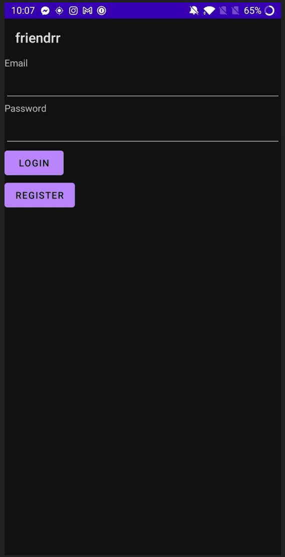
You are now matched with test2 aylmao



Recent Matches

New Match!

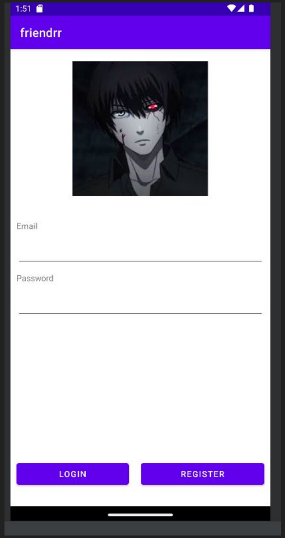
You are now matched with t



friendrr

Email

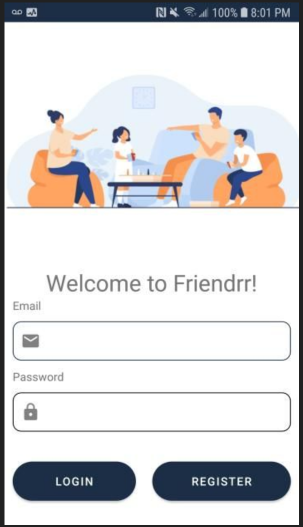
Password



friendrr

Email

Password



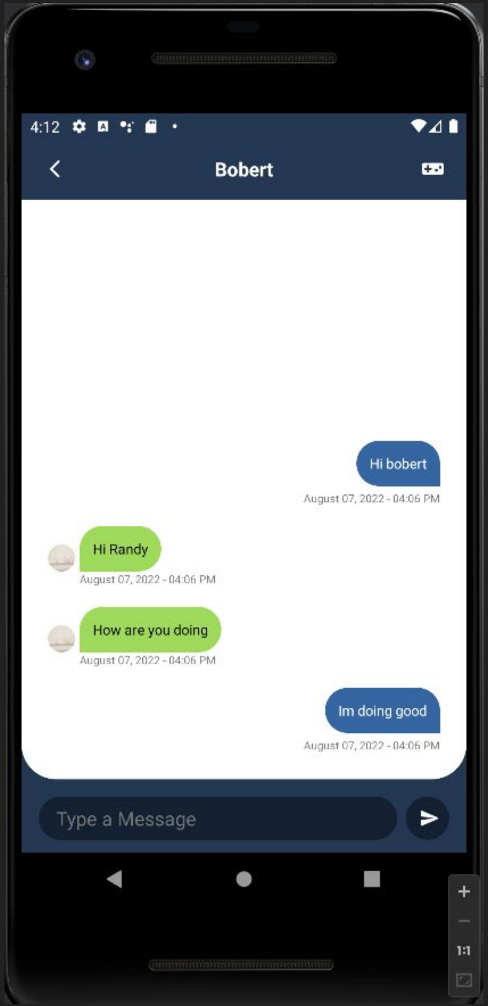
Welcome to Friendrr!

Email

Password

Technical Aspect: Real time messaging

- We are storing each message within a table in our Firestore database.
- Each message contains the sender's email, the receiver's email, the timestamp of the message, and the text itself.
- When entering the chat activity, we send a query to the messages collection to return all messages between the current user's email and the receiving user's email.
- These messages are then sorted according to their timestamps, so we can show them in the order they were sent.
- A RecyclerView with a number of custom drawables and containers are used for the actual visuals
- A snapshot listener on the database collection allows the messages to update in real time, as either user sends messages to one another.


4:12

Bobert

Hi bobert

August 07, 2022 - 04:06 PM

Hi Randy

August 07, 2022 - 04:06 PM

How are you doing

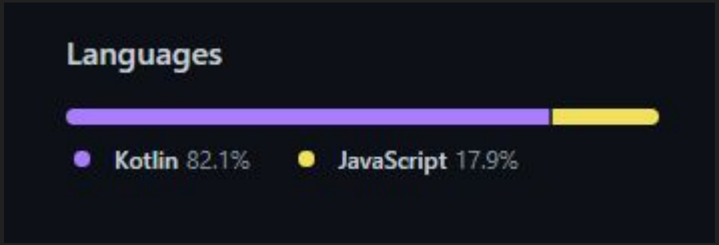
August 07, 2022 - 04:06 PM

Im doing good

August 07, 2022 - 04:06 PM

Demo

Challenges



- Learning new technologies
  - Serverless architecture
  - New technologies and languages(Node.js, Javascript)
    - Ended up with nearly 18% of the entire codebase written in Javascript! 🤯
- Various level of technical amplitude for each individual
  - Require lots of time to do knowledge transfer to bring each other up to speed
- Was too ambitious during planning phase
  - Was planning to create 5 games in total but only completed 3
- UI and user experience enhancements were difficult
- Linking back-end with front-end took more time than expected
  - Resulted in poor progress in terms of front-end for show and tell 2
- Tasks Delegations

Assigned work(Non-exhaustive list)

- Swipe/Match users workflow with audio playback**(Andrew, Feng)**
- Real time chat messaging**(Ben)**
- Mini game integration**(Ben)**
- Search algorithm**(Andrew, Feng)**
- View match history list**(Feng)**
- Navigation bar(**Andrew**)
- Login/Register workflow(**Feng**)
- Add/Remove matched users and friends workflow**(Feng)**
- Icon/Loading screen asset(**Martin**)
- Color theming(**Ben)**
- Mini games against matched users and friends
  - Rock paper scissor**(Andrew)**
  - Card matching**(Martin)**
  - Photo guessing game**(Feng)**
- Global leaderboard for games**(Andrew, Feng)**
- Creating Cloud functions(
- Profile Setup/Edit workflow**(Sterling)**
- Website Management**(Sterling)**
- Data/error validation handling(**Sterling)**
- UI/Feature Enhancement**(Everyone)**
- Bug fixing**(Everyone)**
- Testing**(Everyone)**
- Linking back-end with front-end**(Everyone)**

Lessons Learnt

- Positives
  - Learning new technologies
  - Using version control in a group setting
  - Learning and designing a serverless architecture mobile app

- Negatives
  - Hard to come up with a system design for technologies that we have not worked with before
  - Encountering unexpected bugs as we build and integrate features
  - Perform more initial outlining of the work required in order to better distribute equal tasks amongst team members

Thank you