CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Cambodia Academy of Digital Technology

Institute of Digital Technology

Department Of Computer Science

Specialized in Data Science

Year 2 | Term 2

Project: *Matching With Your Soulmate*

Description: *Matching with someone based on your personality and preferences*

Group 1 Team 2 members:

1, Chum Phalla No.04	| Algorithm of matching
2, Khy Pichsereyvathanak No.08	| Control flow and user data
3, Soeun Sokchetra No.17	| Clean data and User interface

Course: Object-Oriented Programming (python)

Term 2, Academic Year 2024 - 2025

Submitted to: Lect. Han Leangsiv

Submission date: Sunday 23rd March 2024

==========

Acknowledgments

We would like to express our sincere gratitude to **Lect. Han Leangsiv** for approving our proposal and allowing us to work on this project. Your support and guidance have been invaluable throughout this process.

We also want to extend our thanks to everyone who participated in our survey. Your time and honest response helped us collect meaningful data, which played a crucial role in shaping our project. We truly appreciate the insightful comments and positive responses we received, which motivated us to refine and improve our work.

Special thanks to our team members for their dedication, hard work, and constant support. Throughout this journey, we have understood and helped each other, overcoming challenges together as a team. The collaboration and shared passion for this project truly made a difference.

Thank you all for your support and efforts!

Group 1 | Team 2

Sunday 23rd March 2024

Table of Contents.....Page

I. Introduction

1. Problem Statement

In modern days, technology has greatly impacted the lifestyle of people. We rely on technology pretty much most of the time without us realizing. In Cambodia, people use technology to search for *places* or *food* online. However, not many people have tried searching for a **match online**. The traditional way for people to meet someone new is through blind dates, but it consumes a lot of time. This is the main problem that our project is about to solve, to help people find their **soulmates** online and save time for them.

2. Objective

Our project is to match people based on their zodiac signs, age, personalities, height, and common hobbies using data science principles and object-oriented programming (OOP).

3. Brief Project Description

The system will analyze two people's compatibility on a scale of 0 to 100 according to the above criteria, and a score of above 50 will be considered as a match. Then, the users will have the ability to either reject or accept the people matched with them.

II. Methodology

1. Usage

Tool for communication: We use Telegram to communicate about the project progress as well as sharing important files like code and references for our project.

Tool for Coding: Visual Studio Code

Programing language: Python Version 3.13.1

Gathering data: we use form to do survey and get a data for our analysis as a result we got total 265 anonymous response with a useful data 263 include male: 116, female:147

Txt file: we use 2 txt file to store our data separately for men and women data

2. Process/Workflow

Details about what we did to complete this project:

Week	Details
Week1	- Gathered data using Microsoft Forms survey. - Collected 265 responses (147 from women) - Cleaned data manually: 263 responses were usable - Enter data into file text (txt) using the following format: id|username|gender|age|height|occupation|zodiac|personality|lovelang|h1|h2|h3|h4|h5|h6|p_age|p_height|p_occupation|p_personality|p_lovelang m001|vathanak|Male|19|170|Student|Libra|Extrovert|Gifting|Running|Coding|Movies/Documentaries|Cooking|Traveling|Photography|20-22|170 - 175|Student|Extrover m002|Kikawo|Male|24|170|Worker|Capricorn|Ambivert|Physical Touch|Fishing|Coding|Chess|Coffee brewing|Traveling|Dancing|18-20|160 - 165|Worker|Ambivert|Physi m003|Mariee|Male|18|168|Student|Taurus|Ambivert|Physical Touch|Running|Gaming|Movies/Documentaries|Food photography|Volunteering|Photography|18-20|165 - 170 m004|Meng|Male|22|173|Worker|Virgo|Introvert|Physical Touch|Running|Video editing|Movies/Documentaries|Cooking|Traveling|Photography|20-22|170 - 175|Worker| m005|Nam Jung|Male|23|180|Student|Virgo|Ambivert|Physical Touch|Gym & weightlifting|Video editing|Reading|Food photography|Vlogging|Photography|22-24|175 - m006|Ngeat Ngork|Male|21|175|Student|Pisces|Ambivert|Physical Touch|Gym & weightlifting|Gaming|Movies/Documentaries|Coffee brewing|Traveling|Singing|18-20|1 m007|Pheak|Male|19|168|Student|Libra|Ambivert|Words of Affirmation|Gym & weightlifting|Video editing|Reading|Food photography|Volunteering|Painting|20-22|17 m008|I'm cook|Male|20|165|Student|Cancer|Ambivert|Words of Affirmation|Gym & weightlifting|Gaming|Movies/Documentaries|Coffee brewing|Volunteering|Writing|2 m009|Puthi|Male|18|165|Student|Taurus|Ambivert|Quality Time|Cycling|Coding|Reading|Coffee brewing|Traveling|Dancing|18-20|165 - 170|Student|Ambivert|Physica m010|Louis Partridge|Male|21|183|Worker|Gemini|Introvert|Physical Touch|Swimming|Gaming|Movies/Documentaries|Coffee brewing|Traveling|Playing a musical inst m011|Meng Pheng|Male|20|175|Student|Libra|Ambivert|Physical Touch|Running|UI/UX design|Movies/Documentaries|Coffee brewing|Traveling|Painting|18-20|160 - 16 m012|Nha zin 2|Male|19|170|Student|Leo|Ambivert|Physical Touch|Gym & weightlifting|Gaming|Movies/Documentaries|Coffee brewing|Volunteering|Singing|18-20|165 m013|Lampus Oman|Male|32|175|Worker|Virgo|Introvert|Quality Time|Gym & weightlifting|Robotics|Movies/Documentaries|Coffee brewing|Traveling|Painting|20-22|1 m014|Fake|Male|21|173|Student|Pisces|Ambivert|Physical Touch|Camping|Gaming|Sudoku & puzzles|Cooking|Traveling|Photography|20-22|160 - 165|Student|Ambivert| m015|Panha|Male|20|168|Student|Pisces|Introvert|Acts of Service|Cycling|Coding|Movies/Documentaries|Food photography|Volunteering|Singing|18-20|160 - 165|St m016|tsn|Male|19|165|Student|Aries|Ambivert|Quality Time|Gym & weightlifting|Coding|Movies/Documentaries|Coffee brewing|Volunteering|Dancing|18-20|160 - 165 m017|BCZIN|Male|19|165|Student|Scorpio|Introvert|Quality Time|Running|Coding|Chess|Cooking|Volunteering|Graphic design|18-20|160 - 165|Student|Extrovert|Phy
Week 2	- Implemented a match-making algorithm based on user preferences and matching hobbies - Applied OOP concepts (encapsulation, polymorphism, inheritance) for code efficiency and reusability. And also errors handle as well (try, except...)
Week 3	- Finalized the UI for better user experience. - Fixed minor issues (e.g., ensuring users can exit while inputting information). - Find bug and fix it to ensure that user complete input data according to what we want. - Tested the matchmaking algorithm to ensure reasonable calculations.

3. OOP Concepts

3.1. Data Abstraction

The abstract class *Person* uses Python's *ABC* module. The abstract methods used are *get_profile()* and *match_criteria()*, so that the subclasses must implement these methods.

```python
@abstractmethod
def get_profile(self):
    pass

@abstractmethod
def match_criteria(self):
    pass
```

The class *MatchmakingSystem* loads but abstracts the complexity of finding matches by defining *_calculated_match_score()* and *zodiac_match()* internally. The user interacts with only *suggest_match()* without needing to know the logic behind it.

```python
class MatchmakingSystem:
    def _zodiac_match(self, user_zodiac, partner_zodiac):...

    def _calculate_match_score(self, user, partner):...

    def suggest_match(self, user):
        potential_matches = self._women if isinstance(user, Man) else self._men
        matches = []
        for partner in potential_matches:
            score = self._calculate_match_score(user, partner)
            if score >= 50:
                matches.append((partner, score))
        #sort from highest to lowest percentage
        matches.sort(key=lambda x: x[1], reverse=True)
        return matches
```

3.2. Encapsulation

The attributes in class *Person* are encapsulated using underscore (_) to make sure the attributes are private.

```python
class Person(ABC):
    def __init__(self, id, username, gender, age, height, occupation, zodiac, personality,
                 lovelang, h1, h2, h3, h4, h5, h6, p_age, p_height, p_occupation, p_personality,
                 p_lovelang):
        self._id = id
        self._username = username
        self._gender = gender
        self._age = int(age)
        self._zodiac = zodiac
        self._height = height
        self._occupation = occupation
        self._personality = personality
        self._lovelang = lovelang
        self._hobbies = [h1, h2, h3, h4, h5, h6]
        self._preference = {
            "age": p_age,
            "height": p_height,
            "occupation": p_occupation,
            "personality": p_personality,
            "lovelang": p_lovelang
        }
```

There are also getter methods for each attribute to allow controlled access to the attributes without modifying them directly.

Some examples:

```python
# Getter methods
def get_id(self):
    return self._id

def get_username(self):
    return self._username

def get_gender(self):
    return self._gender

def get_age(self):
    return self._age

def get_zodiac(self):
    return self._zodiac

def get_height(self):
    return self._height
```

3.3. Polymorphism

The subclasses *Man* and *Woman* inherit from the base class *Person*. These subclasses have specific behaviors for the methods *get_profile()* and *match_criteria()*.

```
class Man(Person):
    def get_profile(self):
        hobbies = ', '.join(self._hobbies)
        return (f"Man: {self._username}\nAge: {self._age}\nHeight: {self._height}\n"
        f"Zodiac Sign: {self._zodiac}\nOccupation: {self._occupation}\nPersonality: {self._personality}\n"
        f"Love Language: {self._lovelang}\nHobbies: {hobbies}")

    def match_criteria(self):
        return self._preference

class Woman(Person):
    def get_profile(self):
        hobbies = ', '.join(self._hobbies)
        return (f"Woman: {self._username}\nAge: {self._age}\nHeight: {self._height}\n"
        f"Zodiac Sign: {self._zodiac}\nOccupation: {self._occupation}\nPersonality: {self._personality}\n"
        f"Love Language: {self._lovelang}\nHobbies: {hobbies}")

    def match_criteria(self):
        return self._preference
```

3.4. Inheritance

The subclasses *Man* and *Women* inherit the behaviors from the base class *Person*. Both subclasses override the methods *get_profile()* and *match_criteria()* to define how profiles are displayed and how matching preferences are used.

3.5. Magic Methods

- *__init__*: used to initialize attributes.

```
class Person(ABC):
    def __init__(self, id, username, gender, age, height, occupation, zodiac, personality,
                 lovelang, h1, h2, h3, h4, h5, h6, p_age, p_height, p_occupation, p_personality,
                 p_lovelang):
```

- *__str__*: used to define how an object is displayed when *print(object)* is called.

```
def __str__(self):
    return f"{self._username} ({self._gender}, {self._age} years old)"
```

- *__repr__*: provides a more detailed representation of an object, useful for debugging.

```
def __str__(self):
    return f"{self._username} ({self._gender}, {self._age} years old)"
```

- *__eq__*: defines custom equality checking between *Person* objects.

```
def __eq__(self, other):
    if isinstance(other, Person):
        return self._id == other._id and self._username == other._username
    return False
```

4. Libraries Used

4.1. random

For generating random messages when a user accepts or rejects a match.

```
#randomly display message one of it using random
def display_reject_message():
    #when ppl reject the match
    reject_message = ["Thank you for your time! This match is not the right fit for you, but do not worry—we will keep searching for someone m
    ,"Not every match is meant to be, and that is okay! We will find someone who truly clicks with you. Stay tuned for your next potential mat
    ,"We understand that this match was not the right fit. Your preferences matter, and we will continue working to find a better match for yo
    ,"This one was not the one,' but don't worry—your perfect match could be just around the corner! Lets keep the search going!"
    ,"Got it! We'll keep looking for someone who better matches your vibe. Stay patient—the right match is out there!"]
    print(Fore.YELLOW +random.choice(reject_message))
def display_accept_message():
    #when ppl accept the match
    accept_message = ["It is a match!Looks like you both are interested in each other. Start a conversation and see where it goes!"
    ,"Nice choice!You and your match are on the same page—go ahead and break the ice!"
    ,"Great news! Your match is in—now it's time to get to know each other. Who's making the first move?"
    ,"You've accepted the match! We hope this leads to a great connection. Start chatting and see where things go!"
    ,"Matched! Now it's time to chat and see if sparks fly!"]
    print(Fore.GREEN + random.choice(accept_message))
```

4.2. os

Run operating system commands. Used for clearing screen.

```
def clear_screen():
    """Clear the console screen."""
    os.system('cls' if os.name == 'nt' else 'clear')
```

4.3. abc

For data abstraction and applying abstract methods.

```
@abstractmethod
def get_profile(self):
    pass
```

```
class Person(ABC):
```

4.4. colorama

For styling and color to style the terminal.

```
print(Fore.CYAN + "=" * 50)
print(Fore.LIGHTRED_EX + "WELCOME TO MATCHING WITH YOUR SOULMATE".center(50))
print(Fore.CYAN +"=" * 50)
print(Fore.MAGENTA + "Finding your perfect match has never been easier!")
print(Fore.GREEN + "Let's get started.... \n"+ Style.RESET_ALL)
```

III. Implementation

1. Key Features

1.1. Welcome User:

When first running our code, we have a function to welcome users also provided an option either they want to know more about us and algorithms of matching or start a program directly.

1.2. Data collection

When a program starts, we will ask if they are a newUser or oldUser:

newUser: Users will input their details including name, age, occupation, love language, zodiac sign, height, weight, hobbies, and personality. After they input details according to what we want and, in a condition, we will read a file according to their gender to provide them with an

ID which is unique to identify each of them. The data collected is stored in two separate files called *men.txt* and *women.txt* for data

OldUser: They just need to input Username and ID to run a program. We also have a condition to check if that user exists in our data or not if so, they could process to find a match

1.3. Matching Algorithm

We scored the compatibility from 0 to 100% according to a user's preference. Here are the criteria:

- Age: +12%
- Height: +10%
- Occupation: +10%
- Love Language: +16%
- Personality: +18%
- Zodiac Match: +10%
  - Default: +3%
  - Best: +10%
  - Good: +7%
  - Average: +5%
- Hobbies: +4% for each hobby, a total of +24%

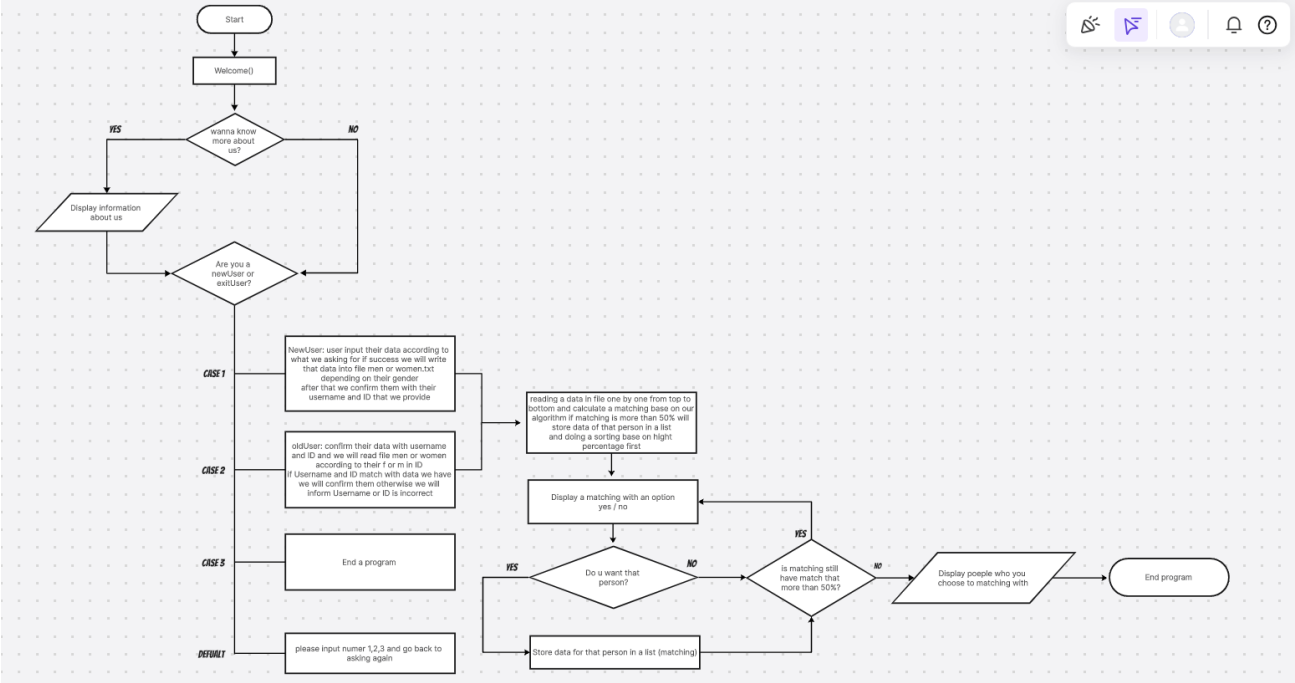
1.4. Process

When user start to find a match we have a function that read data each line in file men or women and doing a calculate of matching percentage base on criteria that we list above and store it in to a list if they match more than 50% and we sorting that data we store in list from Highest to lowest [50%] (Descending order)

1.5. Outcome

The users will be given a list of matches after the scores are calculated, and they will accept or reject the matches.

Flow of program



Flowchart [link]

2. Project Structures

```
G1_T2_Project/
|-- documents/
| ├── G1_T2_Report.pdf # Our report for this project
| ├── G1_T2_Slide.pptx # Slide presentation (PowerPoint format)
| ├── G1_T2_Slide.pdf # Slide presentation (PDF format)
|
|-- G1_T2_Code/
| ├── main.py # Main program logic
| ├── test.py # Checks if each line in text files has 20 columns
| ├── text_files/
| | ├── men.txt # Male user data
| | ├── women.txt # Female user data
|
|-- README.md # Project documentation
```

3. We store the collected data in two separate files called *men.txt* and *women.txt* to easily compare the data across files and find a match for them.
    - The main code is stored in *main.py*, where all the functions take place.
    - As for *test.py* we use it to check whether all the data input has the appropriate number of columns and formatted correctly to ensure that there is no missing data.

4. GitHub Link

   Here our repository Link in GitHub: [Link]

IV. Results and Testing

1. Testing for newUser

What the user sees first

```
==================================================
      WELCOME TO MATCHING WITH YOUR SOULMATE
==================================================
Finding your perfect match has never been easier!
Let's get started....

Wanna learn more about US before starting the program? (yes/no):
```

Asking whether the user is a new or existing user

```
 Alright! Let's continue.

Are you a new user or an existing user?
1. New User (Register)
2. Existing User (confirm Username and ID)
3. Exit
Please enter your choice (1 | 2 | 3): 1
```

If new user:

```
To start our program we need some of your information and prefernce.
Some question you need to write and some are option so just choose number 1, 2, 3, or so on.
If you want to exit, please press 0.
Your Information:
Name: Sokha
```

```
Gender:
0: Exit
1: Male
2: Female
Enter: 2
```

```
Age (only number and must be >= 18): 20
```

```
Height in cm (e.g.: 150): 165
```

```
occupation:
0: Exit
1: Student
2: Worker
Enter: 1
```

```
Zodiac Sign:
0: Exit
1: Aries (March 21 - April 19)
2: Taurus (April 20 - May 20)
3: Gemini (May 21 - June 20)
4: Cancer (June 21 - July 22)
5: Leo (July 23 - August 22)
6: Virgo (August 23 - September 22)
7: Libra (September 23 - October 22)
8: Scorpio (October 23 - November 21)
9: Sagittarius (November 22 - December 21)
10: Capricorn (December 22 - January 19)
11: Aquarius (January 20 - February 18)
12: Pisces (February 19 - March 20)
Enter: 7
```

```
Personality:
0: Exit
1: Introvert
2: Extrovert
3: Ambivert
Enter: 2
```

```
Love Language (Choose one that best matches you):
0: Exit
1: Physical Touch
2: Words of Affirmation
3: Gifting
4: Acts of Service
5: Quality Time
Enter: 3
```

```
Sports & Outdoor Activities (Choose one that best matches you):
1: Gym & weightlifting
2: Swimming
3: Running
4: Cycling
5: Fishing
6: Camping
Enter: 6
```

…continues to ask about 5 other hobbies…

Then asks about the user's preferences:

```
Your Preference in Finding a Partner
Age:
1: 18-20
2: 20-22
3: 22-24
4: 24-26
5: 26-28
6: 28-30
Enter: 3
```

```
Height:
1: below 150
2: 150 - 155
3: 155 - 160
4: 160 - 165
5: 165 - 170
6: 170 - 175
7: 175 - 180
8: 180 - 185
9: 185 - 190
10: 190 +
Enter: 6
```

```
Occupation:
1: Student
2: Worker
Enter: 1
```

```
Personality:
1: Introvert
2: Extrovert
3: Ambivert
Enter: 1
```

```
Love Language (Choose one that best matches you):
1: Physical Touch
2: Words of Affirmation
3: Gifting
4: Acts of Service
5: Quality Time
Enter: 5
```

After all info is input, it will show the user's username and id and potential matches one by one:

```
==================================================
You successfully check in our program.
Username: Sokha
ID: f147
YOU need Username and ID for using our program next-time.
==================================================
Are you ready to find your soulmate?
yes/no: yes
```

```
Here is a potential match:
Man: Rika
Age: 22
Height: 170
Zodiac Sign: Leo
Occupation: Worker
Personality: Ambivert
Love Language: Quality Time
Hobbies: Gym & weightlifting, UI/UX design, Reading, Cooking, Traveling, Painting
Match Score: 61%
Do you accept or reject this person? (yes/no): yes
```

If the user rejects, it will keep running until a user exits and then it displays the matches:

```
You've accepted the match! We hope this leads to a great connection. Start chatting and see where things go!

Here are your accepted matches:
Username: Rika, Score: 61%
Username: Shan, Score: 50%
Username: Yong Bin, Score: 50%
```

2. Testing for oldUser

```
Please enter your Username: Phalla
ID: f147
```

Then, it will display the potential matches just like newUser.

3. Error Handle

   When a user inputs an invalid number, they will be prompted to input again and again until they get it right.

   For example, a user's age must be 18 or above, so here is how we handle it.

```
Age (only number and must be >= 18): 15
Please input a number greater than 18.
Age (only number and must be >= 18): 15
Please input a number greater than 18.
Age (only number and must be >= 18): 17
Please input a number greater than 18.
Age (only number and must be >= 18):
```

We use try-except for ValueError handling.

```python
#Ask for age
while True:
    try:
        age = int(input("Age (only number and must be >= 18): ").strip())
        if age == 0:
            print("Exiting...")
            return
        if age >= 18 and age < 50:
            data.append(age)
            clear_screen()
            break
        else:
            print("Please input a number greater than 18.")
    except ValueError:
        print("Invalid Input! Please enter a valid number! \n")
        clear_screen()
```

V. Conclusion

1. Lesson Learned

Throughout this project, we realized the importance of networking and connections in data collection. Without strong relationships with departments, seniors, juniors, and friends, achieving 265 anonymous responses, especially meeting our target of at least 100 responses per gender, would have been challenging.

Additionally, the feedback from our survey provided valuable insights into the complexities of data collection. We learned that gathering sensitive or credential-based information is not easy, as privacy concerns make individuals hesitant to share personal details.

This project also allowed us to apply our classroom knowledge to a real-world scenario. We conducted research and implemented logical, fair calculations to optimize our matchmaking algorithm, reinforcing both theoretical concepts and practical problem-solving skills.

2. Challenges faced

One of the main challenges we encountered was the lack of existing references on how a matchmaking algorithm should be structured. Without a standardized model to follow, we had to rely on our own experience and logical reasoning to determine the weight of different factors in defining a strong match.

Despite this, we faced minimal difficulties overall. Our passion for the project made the entire process engaging and enjoyable, allowing us to approach problem-solving with enthusiasm and creativity.

3. Future Work

To further enhance our system, we plan to implement several improvements, including:

1. **Administrative Dashboard:** A feature for visualizing data and generating reports.
2. **Enhanced Matching Criteria:** Refining key factors to make the algorithm fairer and more accurate.
3. **Web-Based Platform:** Developing a website for easier user access and increased data collection.
4. **GUI Improvements:** Enhancing the user interface for a smoother and more intuitive experience.

By implementing these improvements, we aim to make our matchmaking system more effective, accessible, and scalable.

VI. References

1. *Astrology Zodiac Signs*. 12 Zodiac Signs: Dates, Traits, Meanings & More. https://www.horoscope.com/zodiac-signs
2. *Zodiac Compatibility*. YourZodiacSign.com. https://www.yourzodiacsign.com/compatibility/
3. Flowchart. https://boardmix.com/app/editor/gW724egdPSdecr7YNdDI0g?inviteCode=qnsorD
4. Tinder Dating App