

CS7IS5 - Adaptive Applications Project Report



The Adaptive Blind-date App

Team Niobrara

Mohan Nishant
Ghosh Soumen
Ishpuniani Dhruv
Srivastava Sneha
Goel Aashima
Kapoor Shuchita

Table of Contents

1	Introduction	3
2	Roles Assignment	3
3	Project Schedule	3
4	Process Flow/Use Cases	4
5	Functional Architecture	4
6	User Characteristics	5
7	Model Generator	6
7.1	User Model	6
7.2	Activity Model	7
8	Compatibility Service	7
8.1	Recommender Engine	7
8.2	Feedback Algorithm	8
8.3	Activity Recommendation Algorithm	8
9	UI/UX	9
9.1	DB	9
9.2	Frontend	9
10	Challenges faced	15
11	Key Features	15
12	Future Scope	16
13	References	17

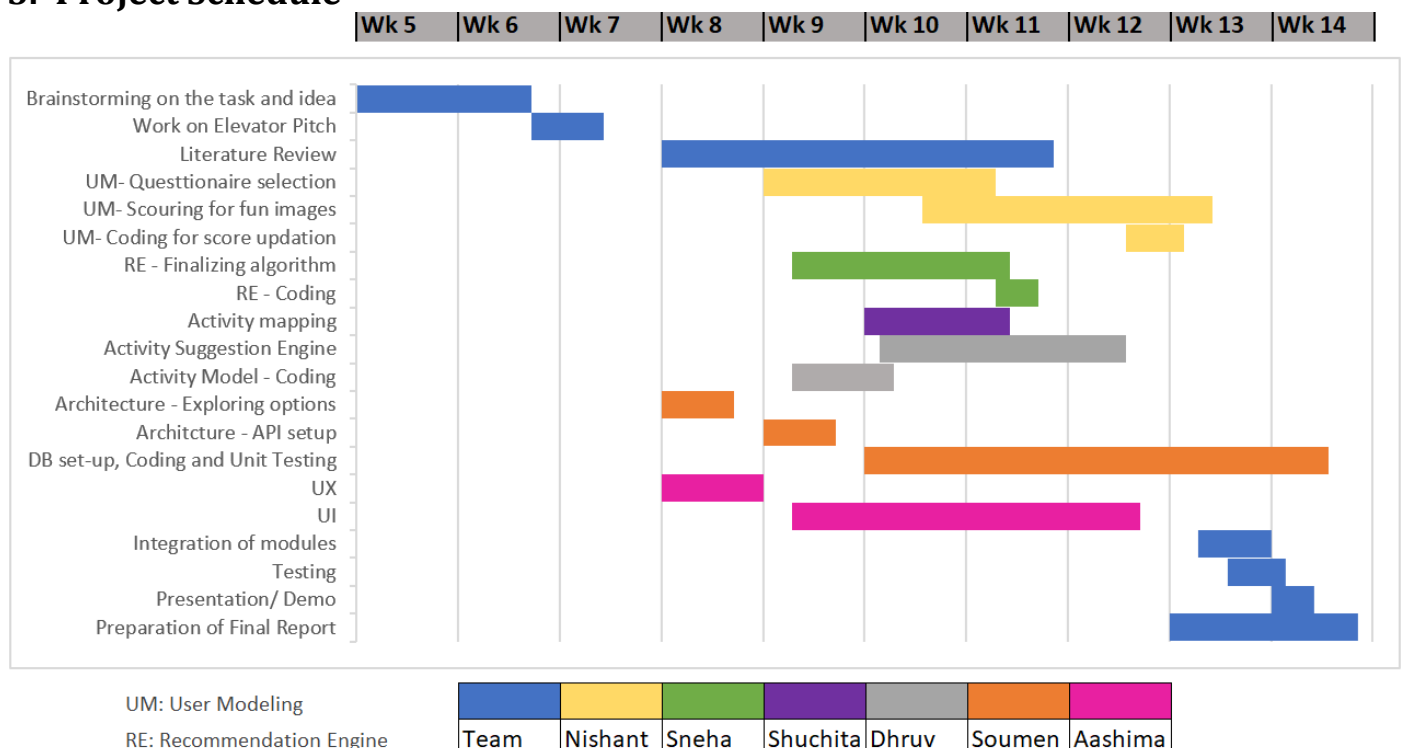
1. Introduction

There are many dating apps in the present and past which assisted users to choose a partner based on their profile and picture. However, such apps have been widely criticized and labelled as being cosmetic and depthless as they do not provide the user of any real information about the other person other than looks. We propose to build an application which suggests users a partner based on their personality traits and preferences instead of their appearance. In fact, the appearance of the person is completely hidden until they meet in-person. Should the user like back our suggested partner, they are encouraged to go out for an activity rather than a date. This, in essence, provides the experience of a blind-date, the only knowledge known apriori being that the person has a personality the user already prefers. The suggested activity is based on the category of user's interest; for example, if the user and his/her match prefer nature trails, the activity suggested would be hiking or trekking. The app is highly adaptive and gives the user the control on what they want, exactly what the theme of this group project is supposed to be.

2. Roles Assignment

	Task/Module	Lead Contributor
1	Activity Crawler	Shuchita
2	User Model	Nishant
3	Compatibility Service	Sneha
4	Activity Model	Dhruv
5	Database setup, API integration	Soumen
6	UI / UX	Aashima

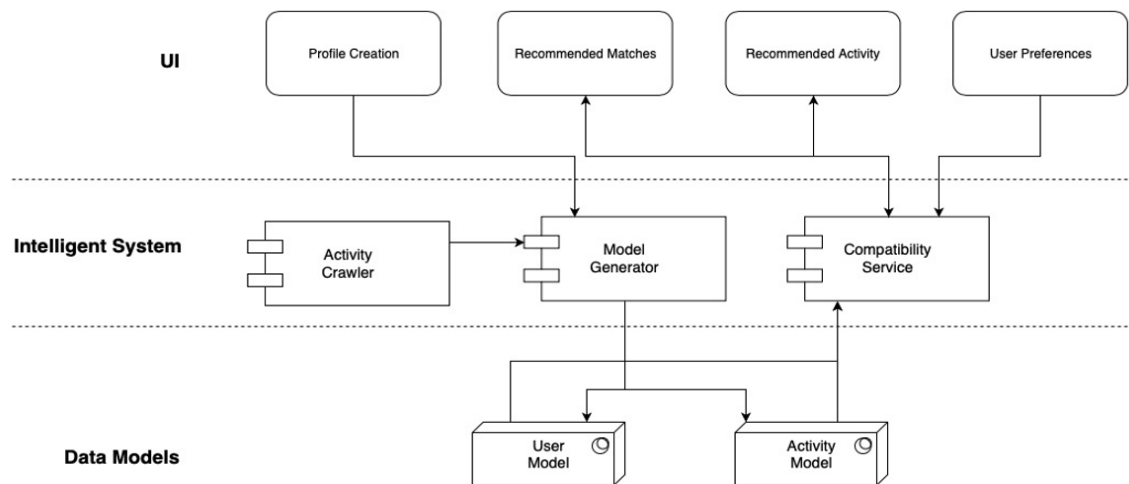
3. Project Schedule



4. Process Flow/ Use Cases

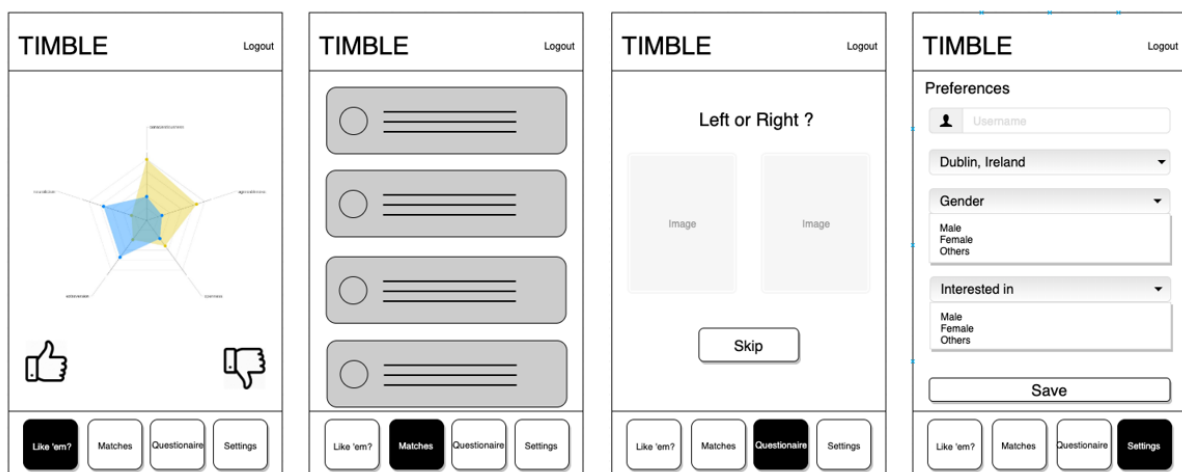
- 4.1. User Signs Up and Logs in
- 4.2. User gives response to a questionnaire; this builds up initial user model to avoid cold start
- 4.3. User takes a look at the recommended suggestions
 - suggestions to come from recommendation engine
 - he/she gives a thumbs up or down
 - Swipes lead to feedback to recommendation engine
- 4.4. As soon as a match occurs, the user goes to next screen- activity suggestions
 - User sees matches
 - User sees the activity suggestion for each match, which comes from the recommendation engine

5. Functional Architecture



Functional Architecture

- 5.1. UI layer
 - 5.1.1. Profile Suggestions
 - 5.1.2. Matches/Recommendations
 - 5.1.3. Questionnaire (User Model Updation)
 - 5.1.4. Search/app Preferences



UI Wireframes

5.2. User Model Generator

5.2.1. Generates user model from the answered questions and preferences

5.2.2. Applies question-mapping technique to attribute the user to a category

5.3. Activity Crawler

5.3.1. scrapes travel websites/API to get data on recommendable activities

5.3.2. Generates activity model according to features and raw data extracted by the Activity crawler and user model

5.4. Activity Model - A model (database entry) of the activity and their features (Eg: activity name, location, category)

5.5. Compatibility Service

5.5.1. Recommends people to match with the user - takes the user model and user ideal match model as input to find out the most compatible match on the basis of euclidean distance algorithms

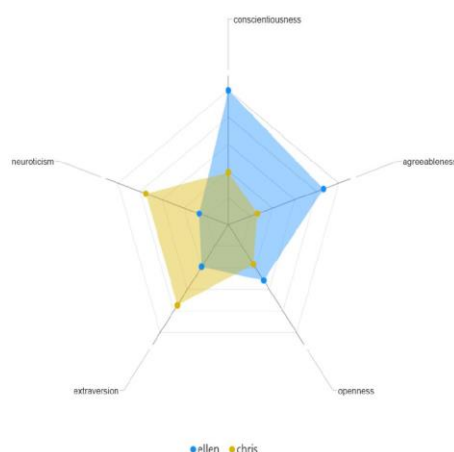
5.5.2. Recommends activity for matched users- takes user model of both the users matched and recommends the SAME activity to them. This is done by matching the characteristics of user models and activity model.

5.5.3. Refines recommendations based on user's response to them.

6. User Characteristics

The aim of our app is to provide user knowledge about the personality of his match. We researched and found out that a widely acceptable method of classifying and quantifying a person's personality is given by OCEAN- Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism [3,4].

Characteristic	Markers
Openness	positive- loves novelty, is generally creative negative- conventional in thinking, prefers routines
Conscientiousness	positive- motivated, disciplined and trustworthy negative- irresponsible and easily distracted
Extraversion	positive- cheerful, takes initiative and is communicative negative- reserved and submissive
Agreeableness	positive- friendly, empathetic and warm negative- shy, suspicious and egocentric
Neuroticism	positive- anxious, inhibited, moody and less self-assured negative- calm, confident and contented?









Radar Chart Visual with personality characteristics of two users

7. Model Generator

7.1. User Model

In order to map and quantify a user to a personality characteristic, we needed an actual questionnaire which the user sees not as an interview, but a fun activity that they wish to do willingly. For this purpose, we decided to build a bank of questions where the answers will be amusing images that the user can relate to. The pool of questions hints was taken from [5], and were rephrased creatively. A bank of images was developed that had two images per answer. A mapping was created to map each question to a personality characteristic and each answer to being positive or negative in that characteristic.

Following are some examples:

Characteristic being quantified	Question	Positive Response	Negative Response
Extraversion	On Fridays, I would rather		
Conscientiousness	People rely on me like		
Neuroticism	My emotions during the day		

After getting a set of response from the user, we calculate a score for each characteristic as:

$$\text{Characteristic score} = \text{previous score} + (\pm \text{response}) / (N_{\text{question_characteristic}})$$

where ' $\pm \text{response}$ ' takes +1 or -1 value depending on the response being positive or negative for that characteristic, and ' $N_{\text{question_characteristic}}$ ' is the number of questions the user has answered for this characteristic. The $N_{\text{question_characteristic}}$ term in denominator ensures that the user's model becomes more robust and less sensitive to change when he has answered higher number of questions of this characteristic.

The final model looks like this:

userid	n_con	conscientiousness	n_neu	neuroticism	n_agr	agreeableness	n_ope	openness	n_ext	extraversion
123	1	3	1	3	1	3	1	3	1	3
234	1	3	1	3	1	3	1	3	1	3
345	1	3	1	3	1	3	1	3	1	3
456	1	3	1	3	1	3	1	3	1	3
567	1	3	1	3	1	3	1	3	1	3
678	1	3	1	3	1	3	1	3	1	3
789	1	3	1	3	1	3	1	3	1	3
8910	1	3	1	3	1	3	1	3	1	3

7.2. Activity Model

We have modelled different activities as a “Domain Model”. The domain here is the user. We’re using the “OCEAN” characteristics for the activity model, same as the User Model.

We used the Yelp! API to get various activities in Dublin and on the basis of their categories, we have manually given them scores for how well they fit in each of the user characteristics. Basically, how probable it is for a user who has a high characteristic score, to do that activity.

For example, a user who has a high score in “Extraversion” is very likely to participate in a “Hiking” activity. Hence, the activity, “Hiking” is given a higher score for “Extraversion” in its activity model.

On the basis of user characteristics and activities, an activity trait map was created

Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Escape Games	Fishing	Hiking	Arcades	Sailing
Do-It-Yourself Food	Yoga	Amusement Parks	Diners	Yoga
Shopping Centres	Museums	Climbing	Aquariums	Art Galleries
Bike Rentals	Tea Rooms	Go Karts	Beaches	Mini Golf
Bowling	Golf	Horseback Riding	Cinema	Lakes

This map was created manually due to time constraints and a boolean value is assigned to these activities so as to calculate activity score for matched users.

8. Compatibility Service

8.1. Recommender Engine

For giving recommendations’ of potential dates to users, we fetch user characteristics’ scores from user model mentioned above as well as user’s ideal characteristic scores and calculate similarity between them with every other user in the database using the concept of euclidean distance.

We then find an aggregate match score for each user with every other user based on the formula:

Match score = euclidean(user1, user(n-1)) + k* euclidean(user(n-1),user1_ideal_match)

where the first term stands for euclidean distance between user1 and all other users taken one at a time (n is the total number of users using our app) . We use (n-1) here as we want to avoid calculating a user's match score with themselves. In the second term, k = weightage given to user and his ideal match score in the calculation of aforementioned match score. k is multiplied by euclidean distance between user and ideal scores of every other user.

These match scores are then sorted in ascending order and the results of top 'x' matched users, with the current user, are sent to UI to be displayed on matches page.

8.2. Feedback Algorithm

Based on user's responses to our recommended matches (thumbs up/ thumbs down), their ideal scores are updated in the database. Each user characteristic of the "ideal" match is updated based on corresponding features of accepted/rejected user based on the formula -

$$\text{ideal_score} = \text{ideal_score} (+/-) \text{other_user}/n$$

where n= number of swipes, + is for thumbs up, - is for thumbs down

This changes their aggregate match scores (formula mentioned above) and in turn helps our app in giving better recommendations to the users. This process is continuous.

Base model for Ideal Match Scores look like this:

userid	conscientiousness	neuroticism	agreeableness	openness	extraversion	swipes
123	5	5	5	5	5	5
234	5	5	5	5	5	5
345	5	5	5	5	5	5
456	5	5	5	5	5	5
567	5	5	5	5	5	5
678	5	5	5	5	5	5
789	5	5	5	5	5	5
8910	5	5	5	5	5	5

8.3. Activity Recommendation Algorithm

- The activity matching engine gets the two matched users as input
- We compare all the activities in our database to the matched users and compute an activity match score as follows:-

$$\text{activity_match_score} = \text{euclidean}(a * \text{User1}, a * \text{User2})$$

Here

- a is the activity model.
- * is an element-wise multiplication.
- User1 is the first user in a match pair.
- User2 is the second user in a match pair.
- An instance of the activity with the lowest match score gets recommended to both the users.

9. UI/UX

9.1. DB

Our design was to create APIs that would return data in JSON format. This motivated us to use MongoDB as the database. A free cloud database was created which would hold 500 Mb of data which was quite sufficient for our app to function.

We chose MongoDB over any other relational databases because of many advantages it has, some of which are mentioned below.

- Schema less – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

9.2. Frontend

‘Timble’ is an adaptive application from UI perspective. User can access the application from various devices like laptops, tablets and mobile.

Frontend included two important tasks:

- UX Design
- User Interface

Before implementing user interface, application design was thought through. Designing phase included:

1. Theme identification
2. Application content
3. Collection of resources like images for questionnaire, icons, etc.

Implementation of UI layer for application is done using Angular 8 as it provides features like component-based architecture, reusability, two-way data binding, typescript etc.

Development

While development, key points that were taken into consideration.

- Cross platform
- Mobile first development
- Modular approach followed

Application consists of:

1. Login/Sign Up

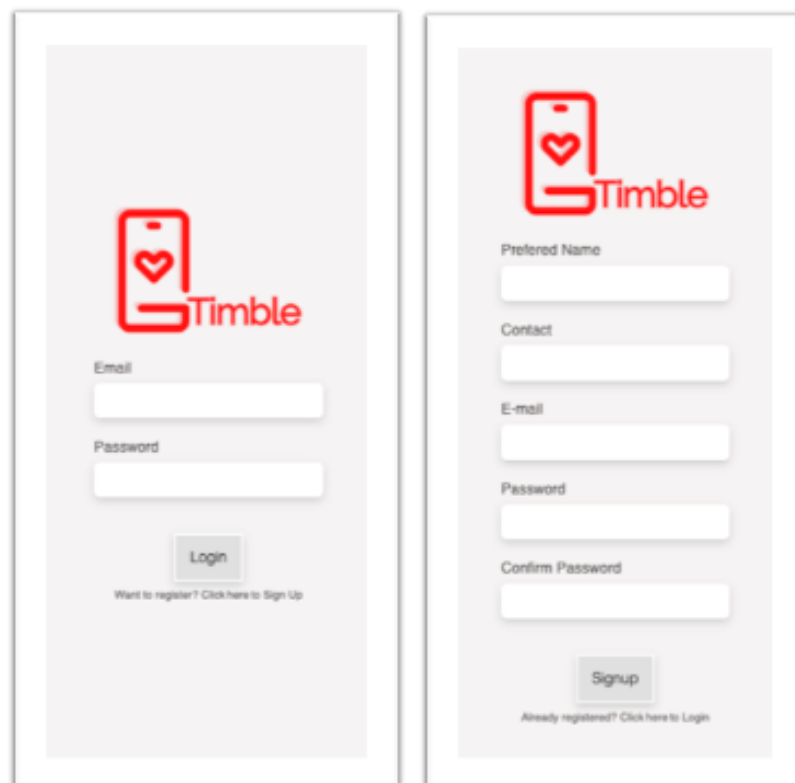
This is default landing page for application. User can login or sign up here.

Desktop



The desktop form is centered on a light gray background. At the top is the Timble logo, which consists of a red smartphone icon with a heart inside and the word "Timble" in red. Below the logo are three white input fields: "Email", "Password", and a "Login" button. At the bottom, there is a small link that says "Want to register? Click here to Sign Up".

Mobile

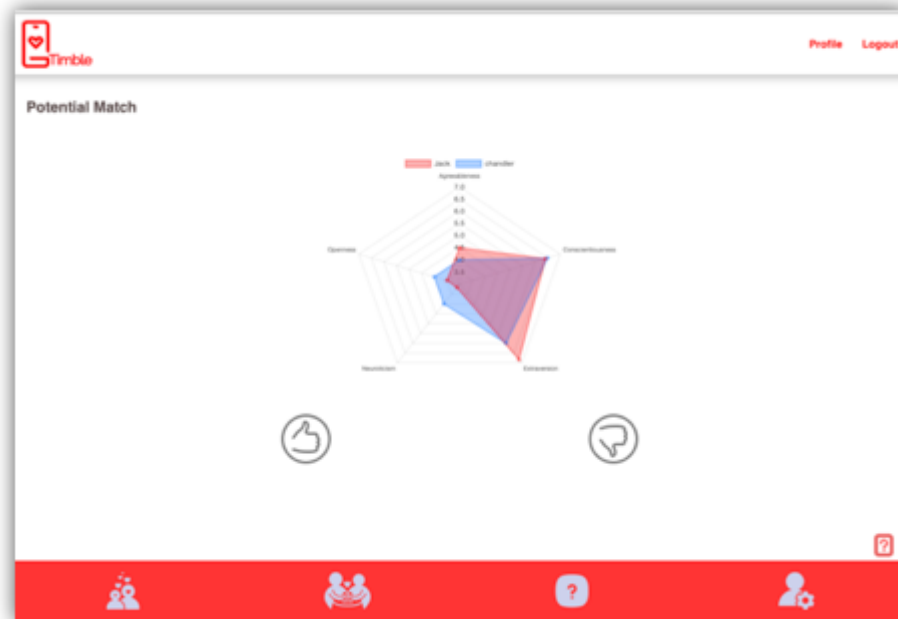


The mobile forms are shown as two side-by-side vertical panels. The left panel is the login form, featuring the Timble logo at the top, followed by "Email" and "Password" input fields, a "Login" button, and a link "Want to register? Click here to Sign Up". The right panel is the sign up form, featuring the Timble logo at the top, followed by "Preferred Name", "Contact", "E-mail", "Password", and "Confirm Password" input fields, a "Signup" button, and a link "Already registered? Click here to Login".

2. Potential Matches

This screen shows potential matches for the user. User sees commonality with the potential match in the form of a radar chart in five dimensions which are, openness, conscientiousness, extraversion, agreeableness and neuroticism. User can 👍👎.

Desktop



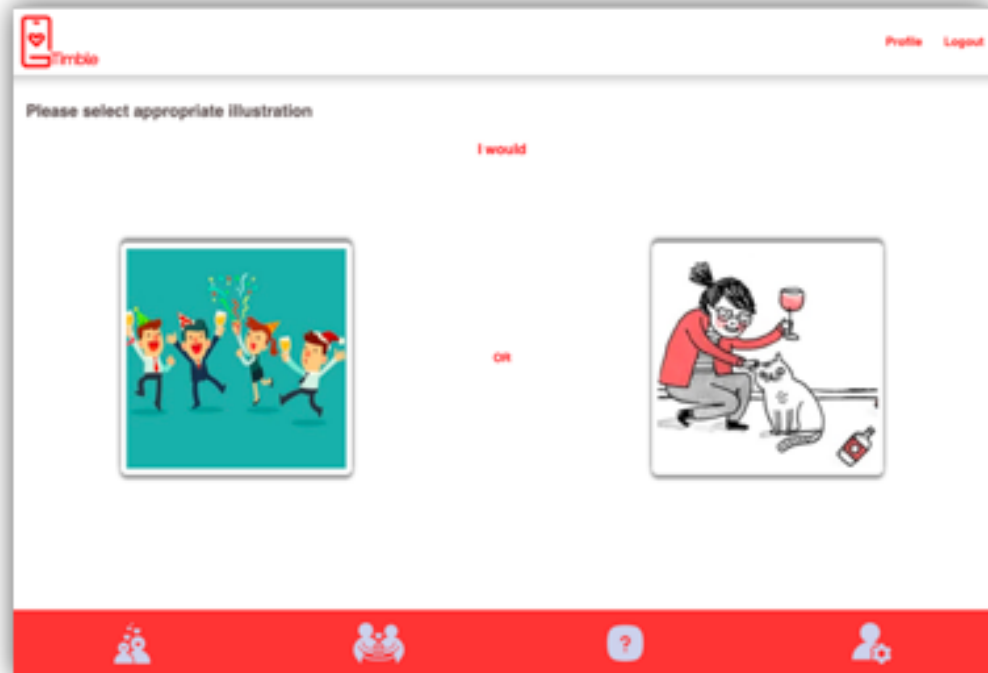
Mobile



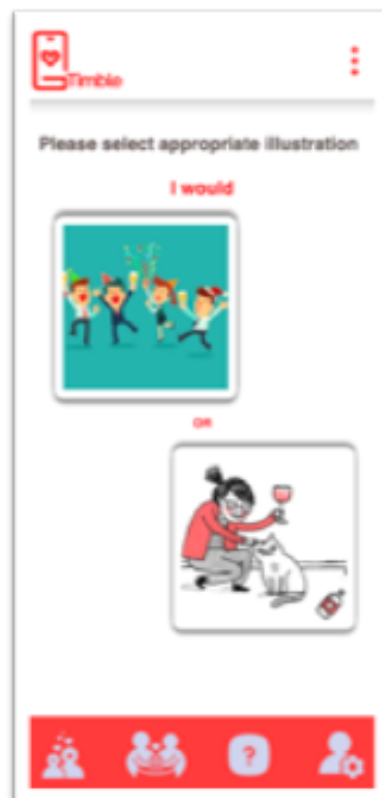
3. Questionnaire

This screen gives user control to modify his/her model by answering questions.

Desktop



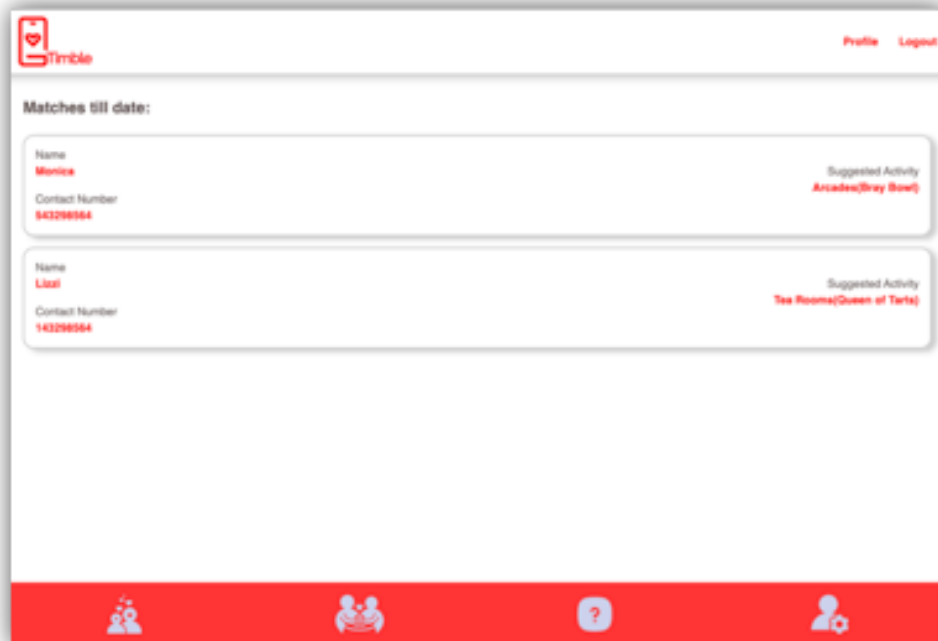
Mobile



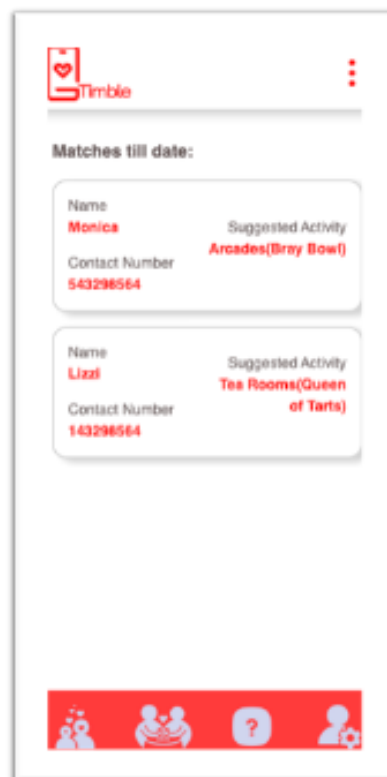
4. Matches till date

User can see all the matches till date on this screen. If two potential matches give 'thumbs up', they are matched, and both receives notification about the same along with a suggested activity for their blind date based on their interests.

Desktop



Mobile



5. Preferences

This screen gives additional control to user to specify/modify preferences like city, age, gender. These preferences help in narrowing down potential matches for user.

Desktop



The desktop view of the Timble Preferences screen features a clean, white background with a red header bar at the top. The header bar contains the Timble logo on the left and 'Profile' and 'Logout' links on the right. Below the header, the 'Preferences' section is displayed. It includes a 'Preferred City' field with 'Dublin' entered, a 'Gender' dropdown menu with 'Select' as the current option, and an 'Interested in' dropdown menu with 'Select' as the current option. Below these, there are two input fields for 'Age' (minimum and maximum) separated by a 'to' label, and an 'Update' button. At the bottom of the screen is a red navigation bar with four white icons: a heart, a couple, a question mark, and a person with a gear.

Mobile



The mobile view of the Timble Preferences screen is designed for a vertical layout. It features a red header bar at the top with the Timble logo on the left and a 'Profile' and 'Logout' menu on the right. Below the header, the 'Preferences' section is displayed. It includes a 'Preferred City' field with 'Dublin' entered, a 'Gender' dropdown menu with 'Female' as the current option, and an 'Interested in' dropdown menu with 'Male' as the current option. Below these, there are two input fields for 'Age' (minimum and maximum) separated by a 'to' label, and an 'Update' button. At the bottom of the screen is a red navigation bar with four white icons: a heart, a couple, a question mark, and a person with a gear.

10. Challenges faced

10.1. User Characteristics

Accurate metric was needed so that we could calculate a user model which is robust and becomes less sensitive to change when user answers questions of similar type. We went ahead with 'Euclidean distance' as it aptly measured distance between users based on multiple axes, each axis being one characteristic.

A good and comprehensible visualization chart was needed for user understanding. Radar/Spider chart suited our needs as it gives information on multiple axes and multiple users can be plotted on the same chart.

10.2. Building a questionnaire

We had to make sure that the questionnaire does not feel like an interview to the user. Therefore, the questions were phrased to be more friendly and amusing. The answer images were made to be funny. The question-characteristic mapping needed to be accurate in order to present the user with reliable charts. Therefore mapping was done manually, taking cues from [5].

10.3. Recommendation

10.3.1. Which metric would be best for user match score?

10.3.2. How to create an "ideal" match?

10.3.3. Can we improve recommendations through feedback?

10.4. Activity Modeling

10.4.1. The biggest challenge was modeling the activities in such a way that we could recommend them to the users.

10.4.2. Activity modeling was done manually and might have included the developer's bias.

10.5. Architecture

10.5.1. Choosing a correct architecture to support our app was quite crucial keeping in mind couple of aspects like flexibility and scalability .

10.5.2. Created restful APIs in Python took some research as there are many ways to achieve the same thing and each one of them has its pros and cons

11. Key Features

11.1. User Modelling

With the approach followed, User Model is less prone to alter as more questions are answered. This makes the model reliable and robust. Also, User Model is updated dynamically after each answer given by the user. This ensures accurate mapping if user answers many questions in one go.

11.2. Recommendation/Activity Engine

- Feedback taken on each positive or negative swipe
- Match score dynamically calculated for every user with every other user

11.3. User Cognition

- Visualization to help user understand what is common and what is not

- Help button provided to explain the meaning of each trait term

11.4. **Dual View:** App can be accessed on both mobile devices as well as desktops

12. Future Scope

- **User Scrutability-** Show user why a match/activity was suggested
- **User Control-** Using other characteristics rather than just personality traits to match users, such as distance, age, height
- **More Adaptivity-** Using feedback on Activity Model to improve activity suggestions
- **App aesthetics-** There is always room for improvement in design!
- **Robust modelling-** Use of statistically backed machine learning model for User Model Generation, Recommender Engine and Activity Engine
- **Research-** More focus on Question-User trait mapping and Activity-User trait mapping

References

- [1] Andrea Zanda, Ernestina Menasalvas, Santiago Eibe, A social network activity recommender system for ubiquitous devices, 2011 11th International Conference on Intelligent Systems Design and Applications, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6121704&tag=1>
- [2] <https://medium.com/@betterhalfai/how-betterhalf-ai-uses-ai-to-match-users-to-their-compatible-partners-7176a10b5960>
- [3] <https://pages.uoregon.edu/sanjay/pubs/bigfive.pdf>
- [4] https://www.researchgate.net/profile/Frederick_Oswald/publication/7014171_The_Mini-IPIP_Scales_Tiny-yet-Effective_Measures_of_the_Big_Five_Factors_of_Personality/links/00b49529d3205d5c29000000.pdf
- [5] Administering IPIP Measures, with a 50-item Sample Questionnaire, https://ipip.ori.org/new_ipip-50-item-scale.htm
- [6] <https://medium.com/@betterhalfai/how-betterhalf-ai-uses-ai-to-match-users-to-their-compatible-partners-7176a10b5960>
- [7] <http://www.personal.psu.edu/%7Eej5i/IPIP/>
- [8] <https://ipip.ori.org>
- [9] https://openpsychometrics.org/_rawdata/
- [10] <https://research.peoplematching.org>
- [11] <https://patentimages.storage.googleapis.com/7c/8d/29/e0bb37b52baec9/US20020040310A1.pdf>
- [12] <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6121704&tag=1>