DATA-DRIVEN SIMILARITY MEASURES FOR MATRIMONIAL APPLICATION

CHIA YONG FANG

UNIVERSITI TUNKU ABDUL RAHMAN

DATA-DRIVEN SIMILARITY MEASURES FOR MATRIMONIAL APPLICATION

CHIA YONG FANG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2020

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Chia Yong Fang
ID No.	:	1700288
Date	:	10/9/2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DATA-DRIVEN SIMILARITY MEASURES FOR MATRIMONIAL APPLICATION"** was prepared by **CHIA YONG FANG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Too Chian Wen
Date	:	10/9/2020

Signature	:	kckhor
Co-Supervisor	:	Khor Kok Chin
Date	:	10/9/2020

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Too Chian Wen and co-supervisor, Dr Khor Kok Chin for their invaluable advice, guidance and their enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to everyone who had helped me and contributed to the development of this project.

ABSTRACT

Marriage is a life-long commitment that completes our life. It will widen our horizons and the meaning of life on this Earth. However, the marriage rate in Malaysia continues to decline. Late marriage is one of the reasons that led to the decline of the marriage rate. Many people tend to get married later because of the difficulties in seeking a suitable spouse. Offline dating is time-consuming and limited by geographic proximity. Due to the convenience provided by the Internet, online dating has become a new trend in seeking potential partners. Hence, this project aims to develop a web-based matrimonial application that enables people to find their potential partner for marriage. Five similarity measures were proposed in this project to overcome the limitations of rule-based approach and Standard Query Language (SQL). The five similarity measures included Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance. The application used the similarity measures to perform matching based on user preferences.

In this project, the adopted software development methodology was phased development, which divided the development process into several phases. After the completion of system implementation, remote usability testing was conducted to evaluate which similarity measure is effective in finding matches that suit user preferences. The sample user data used for testing were collected from 85 people through a questionnaire. The test results showed that the match result obtained by Manhattan Distance was better then the other similarity measures. At the end of the project, all the objectives had been achieved. People can use the application to find their potential matches for marriage as per their priorities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

API	Application Programming Interface
HTTP	HyperText Transfer Protocol
SDLC	Software Development Life Cycle
SQL	Structured Query Language
XP	Extreme Programming

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter provides an overview of this project such as project background, problem statements, project objectives, proposed solution, proposed approach and project scope.

Marriage is one of the most important events in one's life. It is a bond that unites two souls together into one. Back in the days, without the use of the Internet, people used to find their life partner through third persons or relatives. However, traditional offline dating is very time-consuming and based on spatial proximity. People hard to find the perfect match due to limited choices. They might only explore small pools of potential partners within the same community.

As the world becomes technologically advanced, many aspects of our society have transformed. Nearly 60% of the seven billion people in the world can access the Internet (InternetWorldStats.com, 2019). With the Internet, distance and language barriers are broken. Our society has constantly exposed to communication technologies that keep evolving. People can interact with anyone from different corners of the world almost instantaneously via multiple channels (Finkel, et al., 2012). Hence, individuals tend to change the way to meet each other and establish a relationship. A new phenomenon has emerged, which people are becoming more interested in finding the life partner via online that suits their views and ideologies. According to Rosenfeld, Thomas and Hausen (2019), the traditional ways of meeting partners through offline have all been declining sharply since 2000 (Appendix A). The introduction of graphical web around 1995 and the introduction of the smartphone after 2007 had caused the rapid rise of meeting online (Rosenfeld, Thomas and Hausen, 2019).

The new phenomenon has led to the growth of online matrimonial application development. With the help of the online matrimonial application, finding a life partner become easier. Compared to traditional dating, online dating has some significant benefits for daters. Before deciding to meet a potential partner in person, an individual may collect an initial understanding of their compatibility with potential partners. It can be obtained through online instead of relying on family members or any third person to select an unacquainted single (Finkel et al., 2012). Online dating sites also enable users' access to many potential partners who vary in demography and lifestyle characteristics.

1.2 Problem Statement

1. Late Marriage in Malaysia

Late marriage is a recent trend in Malaysia, particularly in the non-muslim community (Stevenson and Wolfers, 2017; Department of Statistics Malaysia, 2019a). The women are facing difficulties in finding suitable partners according to the Malaysia Population Research Hub (2019). Good opportunities for education and employment had led women to postpone marriage (Yuen, 2019). As women are getting better qualification for jobs and more economically independent than before, they tend to have higher expectations for choosing a marriage partner. Women might demand greater equality and shared responsibilities in a marital relationship. According to the research done by Campbell, Chin and Stanton (2016), people tend to enter a new relationship with others with characteristics that suit their ideal preferences more closely.

There are several effects of late marriage. For example, it is not easy to find a partner that meet the expectations as the circle of suitable candidate might become smaller due to age increase. Besides, late marriage is one of the reasons that led to low fertility rate as women will delay childbearing. The infertility rate will also increase as females and males get older (NICHD, 2020). According to the data from Department of Statistics Malaysia (2019b), the total fertility rate per woman in age group (15 to 49) was decreased from 4.9 babies in 1970 to 1.8 babies in 2018 (Appendix B). The total fertility rate is known as the average number of children to be born per woman in her lifetime.

2. Limitation in Rule-based System and SQL Query

The rule-based system uses a series of rules that are usually expressed as "if-then" clauses to derive actions (Kwasny and Faisal, 1990). According to Ross (2004), the if-then rules can be expressed as "IF cause (antecedent) THEN effect (consequent)". The rule-based system has some weaknesses. Using the rule-based approach may cause a combinatorial explosion as the classification of data often contains a huge number of rules (Liu, Ma, & Wong, 2001). Generating all the rules and conditions for a complex system is quite difficult and time-consuming, while those rules and conditions might be important for accurate classification. Hence, it is not suitable for the online matrimonial application as users can set a lot of searching preferences to find for potential matches. It will generate a huge number of rules for the system.

Furthermore, almost all applications that work with databases analysis and manipulate relational data through SQL (Bourgeois and Bourgeois, 2014). The SQL query is excellent at finding exact matches. For example, when the user specifies conditions to construct the queries, the system will use the conditions to refine the database records and retrieve the results. However, it might lead to no result being returned when no record in the database meets the SQL conditions.

1.3 Project Objectives

1. To develop a web-based application by providing a solution that enables an individual to find their potential matches for marriage as per their priorities.
2. To perform matching through similarity measures based on the requirements and priorities set by users.

1.4 Proposed Solution

To solve the problems stated above, a similarity measures based matrimonial application is proposed. The target user will be individuals who are single and wish to explore the opportunities and resources to find for a suitable life partner. The application combines "data" and "calculation" to help an individual finds the perfect soulmate. It will implement five algorithms to calculate the similarity of the potential partners based on the selections.

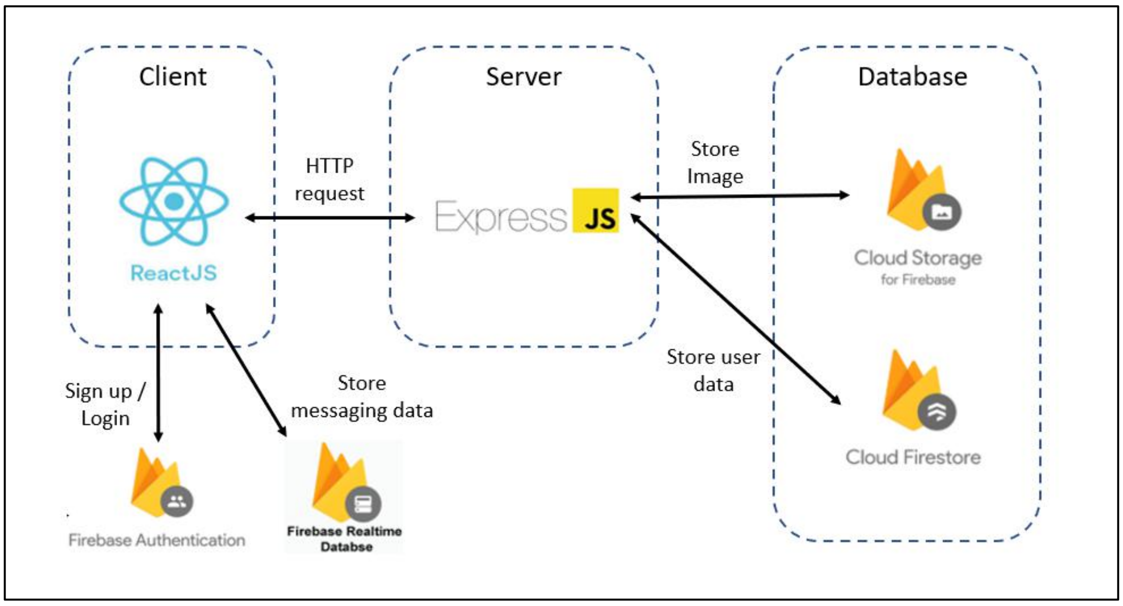


Figure 1.1 The High-Level Architecture of the System

In this project, React is used to develop the front-end of the application. It is a Javascript library for creating reactive and iterative user interface (Reactjs, 2020). React can handle the logic flow of the system, update and render the right component. It helps to increase development productivity by allowing code reuse.

Furthermore, express.js and Firebase platform are implemented to provide some Backend-as-a-Service solutions for web-based applications. When users sign up for an account, Firebase Authentication will create a new user record (email address and password). Firebase Authentication also provides convenience for developers as it provides added security and helps to prevent abuse during sign-up and authentication (Firebase, 2020).

The express.js will act as the server that integrates with Firebase to handle all HTTP request received from the web-based application. After sign up, users will be prompted to enter their personal information such as demographics and lifestyle characteristics along with an option to upload their photo. The application will send HTTP requests to the express.js to do all the validations for storing or retrieving data from Firestore and uploading or downloading files from Firebase Cloud Storage.

Besides, users can search for profiles that match their requirement and priorities. After the users have set the specifications, the server will retrieve user data from Firestore and perform the matching algorithms. The result of matching will be returned to the application and shown to the user.

In this project, we use similarity measures to find the perfect match for an individual. The application will calculate the similarity between an individual with the requirements by using five algorithms which are Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance.

1.5 Proposed Approach

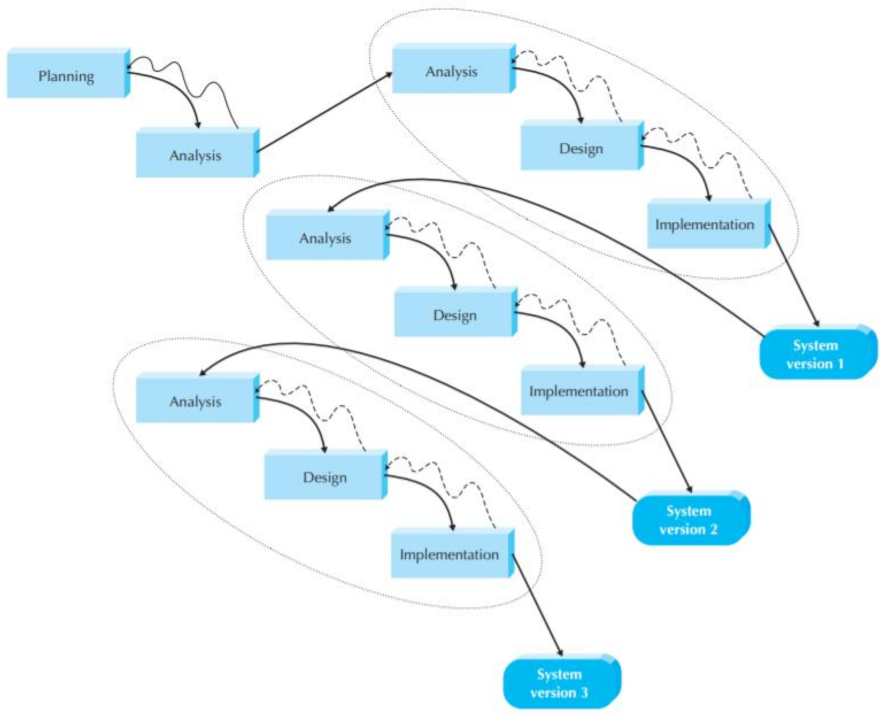


Figure 1.2 Phased Development–based Methodology (Tegarden, Dennis and Wixom, 2009)

The development methodology implemented in this project is Phased Development. By applying this methodology, the overall system will be divided into several versions and each version is developed logically and sequentially. The most fundamental function will go into the first version (Tegarden, Dennis, and Wixom, 2009). Once the preceding version is completed, the next version will start to build.

i. **Version 1: Basic functionality and user interface**
   In version 1, the front-end of the system was developed. All the basic functionality and user interface was designed and implemented.

ii. **Version 2: Firebase Authentication Implementation**
   In version 2, a user authentication function was implemented. Firebase Authentication was used to handle the authentication event. Firebase Authentication was designed to allow users authenticate with email address and password.

iii. **Version 3: Server-side and Firebase Implementation**
   The version 3 started to set up the express.js server and Firebase that would integrate with front-end functionality. The express.js server was designed to handle the HTTP requests and integrate with Firestore and Firebase Cloud Storage. Besides, Real-time database was also set up to handle all messaging data in the application. Database of the system was designed to build a good and simple database model.

iv. **Version 4: Algorithm implementation**
   This version started to implement the algorithms for data mining. The express.js integrated with Firestore to retrieve the required data for the data mining process. The five algorithms which were Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance were designed and implemented in express.js.

1.6 Project Scope

1.6.1 Target Users

In this project, the target users will be the individuals who are single with age between 22 and 26 and wish to find for a suitable life partner for marriage. According to the data from the Department of Statistics Malaysia (2019a), the average marriage age in Malaysia is between 26 and 28. An individual has to establish a stable relationship with the partner before getting married. Hence, this project aims to help the individuals find for their potential matches based on their priorities and encourage early marriage. The system will match two individuals through similarity measures.

1.6.2 Features Covered

The features below would be included in the application to achieve the project's objectives.

i. Collect user data for data mining

This system must provide the user interface for the users to input their personal information. For example, users need to provide their name, gender, age, religion, height, posture, marital status, mother tongue, education level, education field, occupation, smoking habits, child wish, hobby, etc. along with an option to upload their photo. All the user data will be stored in the database. This feature is important as the system needs user data to perform matching. Besides, the users can get a basic understanding of their potential matches through these details before deciding to approach them.

ii. User can set searching preference with priority

This system must enable users to set their searching preferences. The user can set their priority from highest to lowest. The priority included age, religion, height, posture, marital status, mother tongue, education level, education field, occupation, smoking habits, child wish, hobby etc. The system will perform matching and find for potential matches based on user's requirement and priorities.

iii. Matching based on user preferences by applying five algorithms

The system must be able to capture those matching requirements set by the user and sent as an input to the server for data analysis. The server will calculate the similarity percentage of a profile with the requirements set by the user by using similarity measures. There are five algorithms will be implemented for similarity measures, which are Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance.

iv. **Display profile that matches the user's requirements and priorities after the matching algorithm performed.**

Once the server completes the matching through similarity measures, the system must be able to display a list of profiles that match the user's preferences. The system will display the profiles in the descending order of similarity. The profile with the highest percentage will be displayed on the top. It will make the profile selection process simpler as the user can easily recognize which profile is closer to their ideal partner.

v. **Personal chat function**

The system must provide the chat function for the user and his/her potential match. If the user is interested in one of the profiles that suit his/her requirements, the user can choose to start a conversation with the match and get to know each other.

vi. **Like function**

The system must provide a "Like" function for the user. This feature is similar to the "Like" function on Instagram. When viewing a profile, the system must allow the user to "like" the profile. The number of likes will indicate the popularity of a particular profile. This is a quantitative measurement for the outlook of the members.

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Online Related System

There are five similar systems being studied in this section. Five websites were studied, analysed and evaluated based on their features and functionalities.

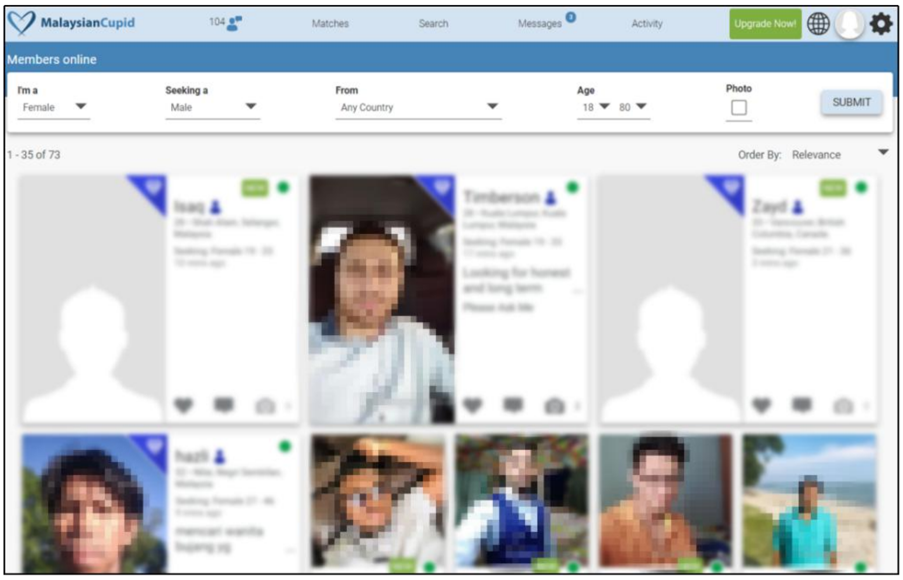
2.1.1 MalaysianCupid.com

(Available at https://www.malaysiancupid.com/)

MalaysianCupid.com is an online dating and matchmaking sites operated by Cupid Media Pty Ltd. Cupid Media Pty Ltd is a company specializing in developing database-driven dating sites. It provides a platform for Malaysian singles to find their perfect and true love match through sophisticated search and messaging facilities.

(a) Members Online

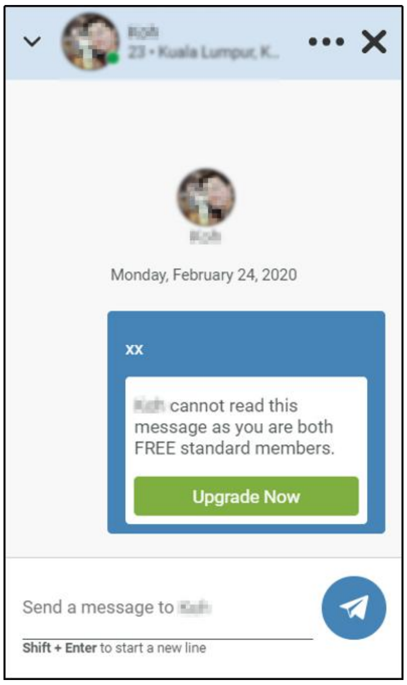
Upon logging in, the user will be redirected to the home page of MalaysianCupid.com. The home page will display the members who are currently online. User can browse the page, send interest or add the profile to favourites. This "members online" feature provides a simple filtering function that enables the user to browse through the profiles that match their requirements.



I'm a: Female
Seeking a: Male
From: Any Country
Age: 18 – 80
Photo: ☐
1 - 35 of 73

Figure 2.1 Home Page of MalaysianCupid.com Site

There is a chat button below each photo for instant messaging. User can choose to chat with the interested profile. However, the instant messaging function is not available for free standard members.


Monday, February 24, 2020

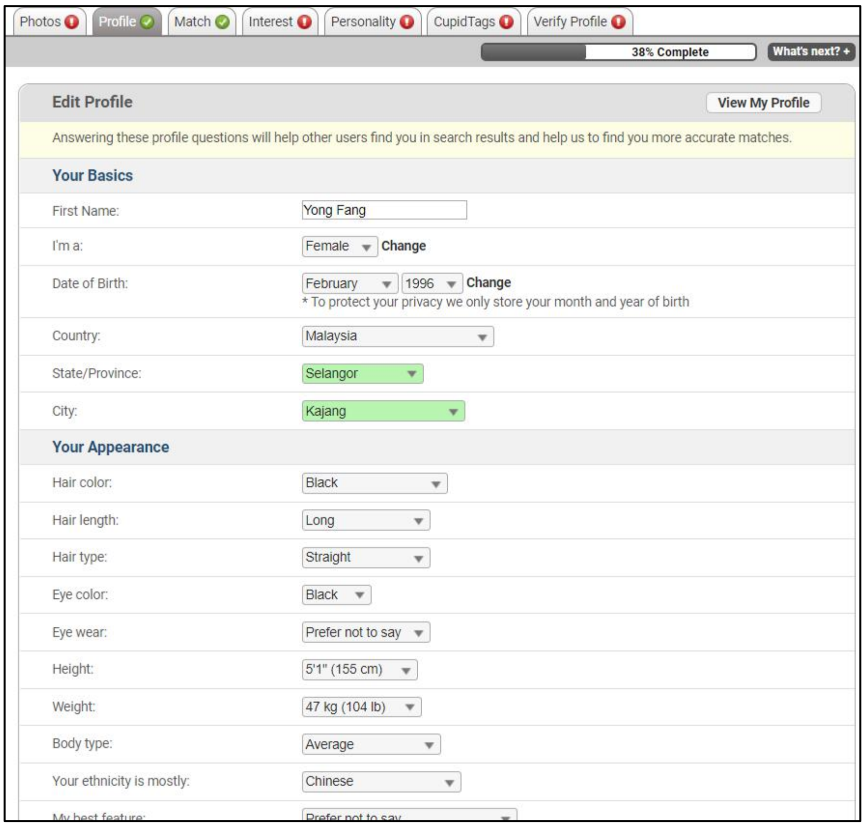
xx

cannot read this message as you are both FREE standard members.

Figure 2.2 Instant Messaging Function

(b) Matching Profiles

Before searching for matches, the user needs to update his/her personal info in the profile section. The profile section includes the user's basic info, appearance, lifestyle, cultural values, interests and personality profile. The system will also use the information to help the user find more accurate matches.



Edit Profile

Answering these profile questions will help other users find you in search results and help us to find you more accurate matches.

Your Basics

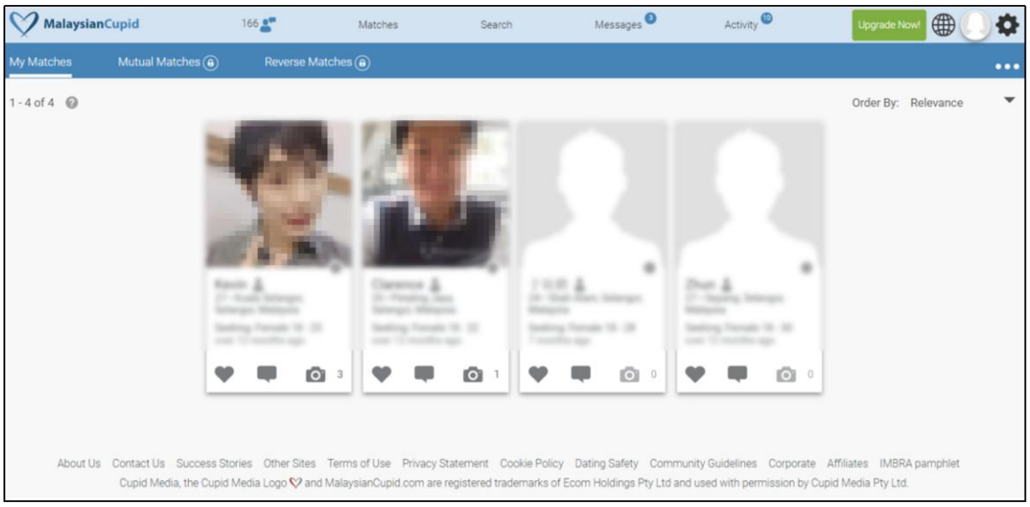
First Name:	Yong Fang
I'm a:	Female Change
Date of Birth:	February 1996 Change * To protect your privacy we only store your month and year of birth
Country:	Malaysia
State/Province:	Selangor
City:	Kajang

Your Appearance

Hair color:	Black
Hair length:	Long
Hair type:	Straight
Eye color:	Black
Eye wear:	Prefer not to say
Height:	5'1" (155 cm)
Weight:	47 kg (104 lb)
Body type:	Average
Your ethnicity is mostly:	Chinese
My best feature:	Prefer not to say

Figure 2.3 Sample Input List for User Profile Information

After updating the profile, the user can set their match criteria to search for the perfect match. The search engine will use the match criteria to filter the huge selection of profiles in the database and find for exact matches. A list of exact match profiles will be displayed for user viewing.

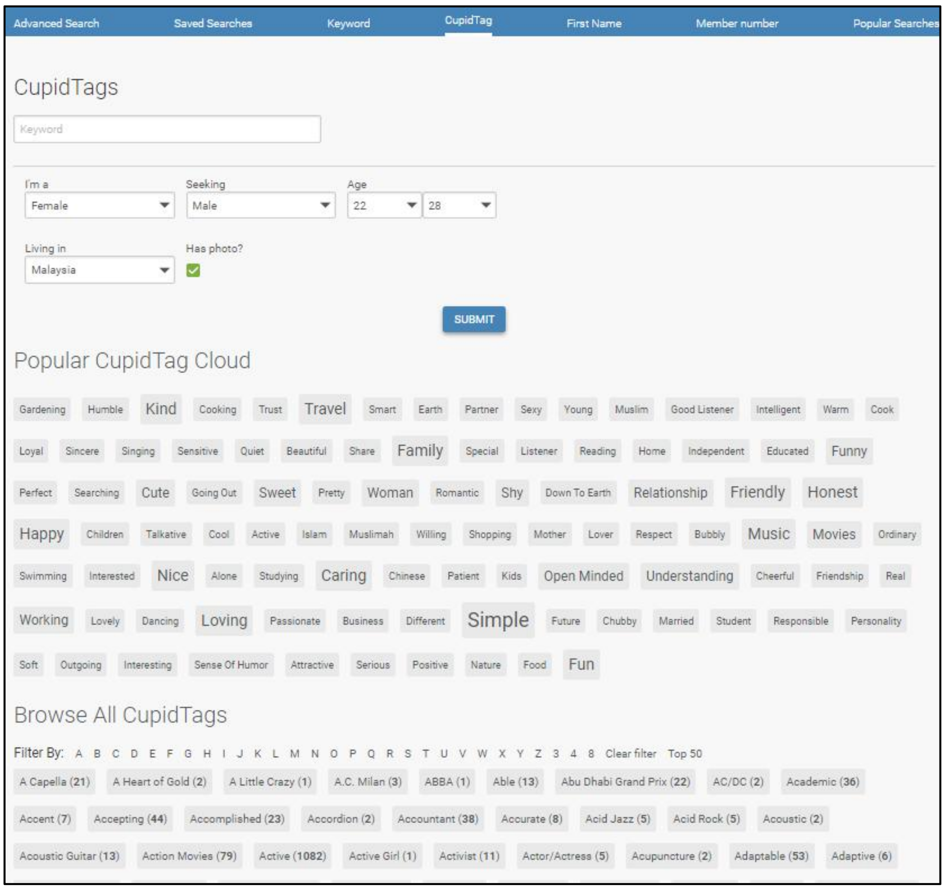


Cupid Media, the Cupid Media Logo and MalaysianCupid.com are registered trademarks of Ecom Holdings Pty Ltd and used with permission by Cupid Media Pty Ltd.

Figure 2.4 Page Showing the Relevant Profiles After Matching

(c) Searching Profiles

The system provides an advanced searching feature. Similar to the matching features, user can fill in the criteria such as appearance, lifestyle and cultural values for searching. There is also the CupidTag search option that enables the user to search for the members' tags that he/she wishes to meet.

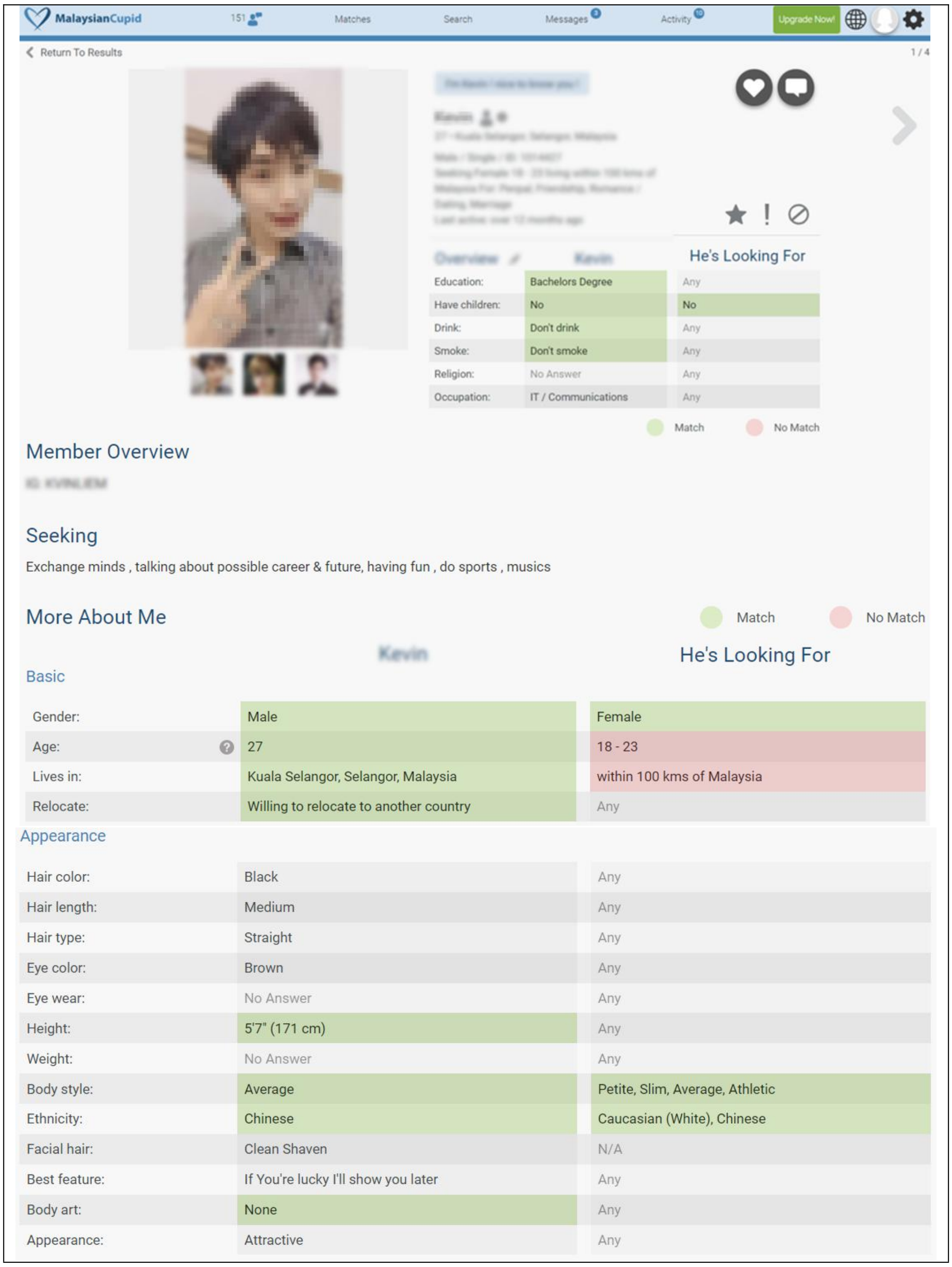

Advanced Search
Saved Searches
Keyword
CupidTag
First Name
Member number
Popular Searches
CupidTags
Keyword
SUBMIT
Popular CupidTag Cloud
Gardening Humble Kind Cooking Trust Travel Smart Earth Partner Sexy Young Muslim Good Listener Intelligent Warm Cook
Loyal Sincere Singing Sensitive Quiet Beautiful Share Family Special Listener Reading Home Independent Educated Funny
Perfect Searching Cute Going Out Sweet Pretty Woman Romantic Shy Down To Earth Relationship Friendly Honest
Happy Children Talkative Cool Active Islam Muslimah Willing Shopping Mother Lover Respect Bubbly Music Movies Ordinary
Swimming Interested Nice Alone Studying Caring Chinese Patient Kids Open Minded Understanding Cheerful Friendship Real
Working Lovely Dancing Loving Passionate Business Different Simple Future Chubby Married Student Responsible Personality
Soft Outgoing Interesting Sense Of Humor Attractive Serious Positive Nature Food Fun
Browse All CupidTags
Filter By: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 3 4 8 Clear filter Top 50
A Capella (21) A Heart of Gold (2) A Little Crazy (1) A.C. Milan (3) ABBA (1) Able (13) Abu Dhabi Grand Prix (22) AC/DC (2) Academic (36)
Accent (7) Accepting (44) Accomplished (23) Accordion (2) Accountant (38) Accurate (8) Acid Jazz (5) Acid Rock (5) Acoustic (2)
Acoustic Guitar (13) Action Movies (79) Active (1082) Active Girl (1) Activist (11) Actor/Actress (5) Acupuncture (2) Adaptable (53) Adaptive (6)

I'm a: Female
Seeking: Male
Age: 22 – 28
Living in: Malaysia
Has photo? ✓

Figure 2.5 Advanced Search Feature of MalaysianCupid.com

(d) Profiles Viewing

By clicking into one of the profiles displayed by the system, the user can view the complete information of the profile, including the specific details of the profile's match preferences. User can send interest, add the profile to favourites or send a message.



[redacted]

[redacted]

[redacted]	[redacted]	He's Looking For
Education:	Bachelors Degree	Any
Have children:	No	No
Drink:	Don't drink	Any
Smoke:	Don't smoke	Any
Religion:	No Answer	Any
Occupation:	IT / Communications	Any

🟩 Match 🟥 No Match

Member Overview

[redacted]

Seeking

Exchange minds , talking about possible career & future, having fun , do sports , musics

More About Me

🟩 Match 🟥 No Match

	[redacted]	He's Looking For
Basic		
Gender:	Male	Female
Age:	27	18 - 23
Lives in:	Kuala Selangor, Selangor, Malaysia	within 100 kms of Malaysia
Relocate:	Willing to relocate to another country	Any
Appearance		
Hair color:	Black	Any
Hair length:	Medium	Any
Hair type:	Straight	Any
Eye color:	Brown	Any
Eye wear:	No Answer	Any
Height:	5'7" (171 cm)	Any
Weight:	No Answer	Any
Body style:	Average	Petite, Slim, Average, Athletic
Ethnicity:	Chinese	Caucasian (White), Chinese
Facial hair:	Clean Shaven	N/A
Best feature:	If You're lucky I'll show you later	Any
Body art:	None	Any
Appearance:	Attractive	Any

Figure 2.6 Profile Information of Matched Member

2.1.2 Match.com

(Available at https://my.match.com/home)

Match.com is one of the oldest dating site founded by IAC in the year 1993. It aims to prove the feasibility of an online classified ads platform for dating. It provides a platform for singles to express themselves through various writing sections.

(a) Top Pick / Recommended

In the home page, there is a special feature that displays the recommended profiles to the user. Recommended are members of the user's area that are selected based on user preferences. The user can choose to skip or like the profiles.

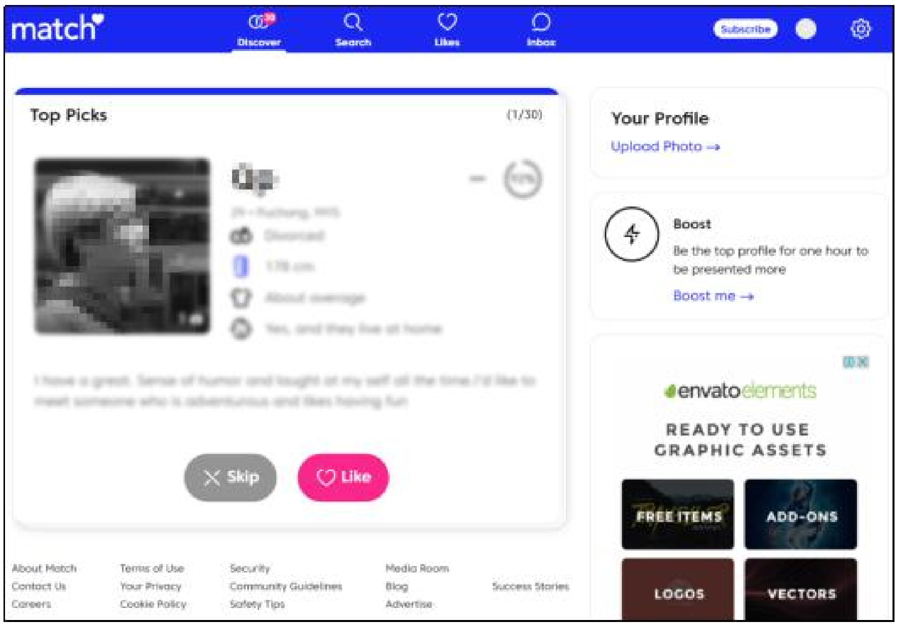


Figure 2.7 Home Page of Match.com

(b) Search for Profiles

The system provides a searching feature called 'Discover'. Similar to the matching features, user can fill in the criteria such as gender, age, location, interests, personal information and lifestyle for searching. The system also provides the option to search by keyword.

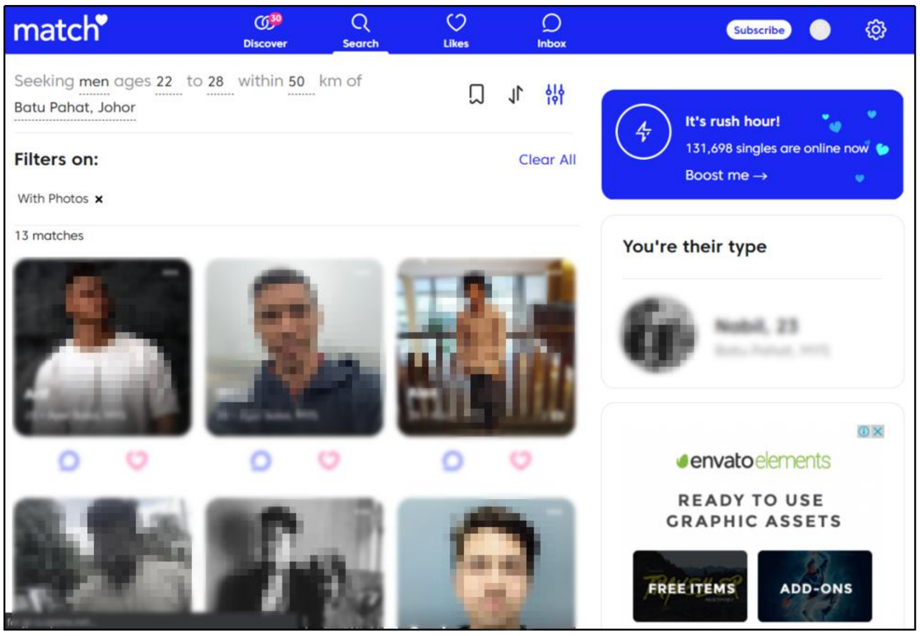


Figure 2.8 Search Feature of Match.com

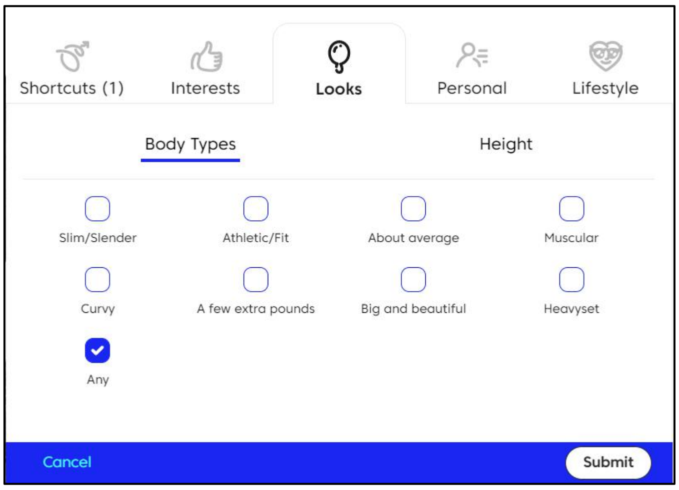
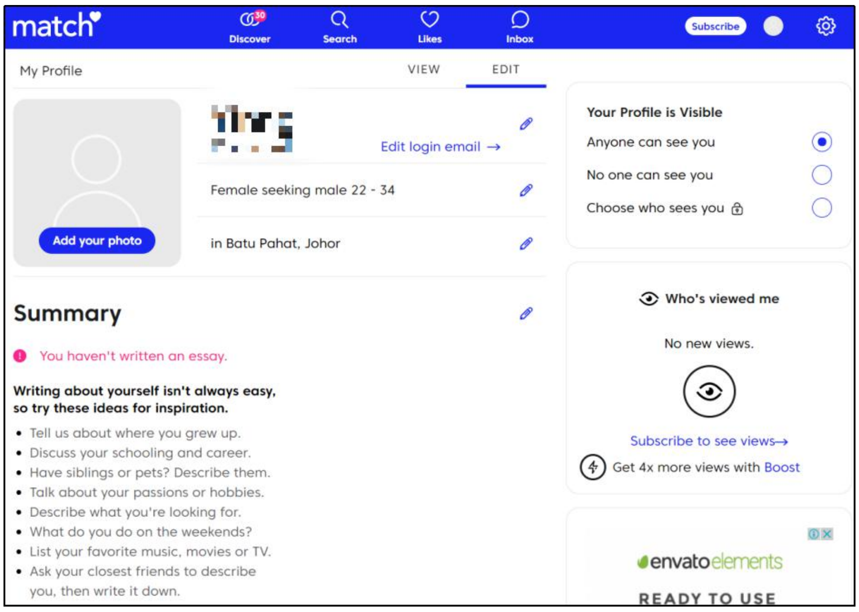


Figure 2.9 Search Filter Option

(c) Matching Profile

In Match.com, user can provide their detailed personal information before finding for suitable matches. With the information provided, the system will match the user with the right one. The profile consists of the self-written profile summary, a list of personal info, interests and a photo gallery.


match Discover Search Likes Inbox Subscribe

My Profile VIEW EDIT

Add your photo

Edit login email →

Female seeking male 22 - 34

in Batu Pahat, Johor

Your Profile is Visible

Anyone can see you [✓]

No one can see you

Choose who sees you

Summary

You haven't written an essay.

Writing about yourself isn't always easy, so try these ideas for inspiration.

- Tell us about where you grew up.
- Discuss your schooling and career.
- Have siblings or pets? Describe them.
- Talk about your passions or hobbies.
- Describe what you're looking for.
- What do you do on the weekends?
- List your favorite music, movies or TV.
- Ask your closest friends to describe you, then write it down.

Who's viewed me

No new views.

Subscribe to see views→

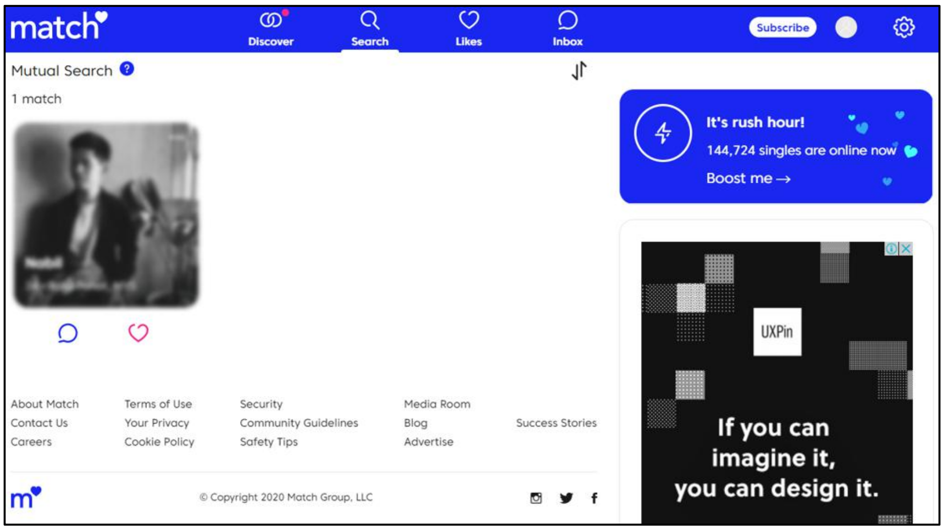
Get 4x more views with Boost

envatoelements

READY TO USE

Figure 2.10 Page for Editing User Information

There are two matching features in Match.com, which are mutual search and reverse search. Mutual search will display the profiles when both the user and the member meet the criteria set by each other. Reverse search will display the profile if the user matches the particular profile's preferences.


match Discover Search Likes Inbox Subscribe

Mutual Search

1 match

It's rush hour!

144,724 singles are online now

Boost me →

About Match | Terms of Use | Security | Media Room |
Contact Us | Your Privacy | Community Guidelines | Blog | Success Stories
Careers | Cookie Policy | Safety Tips | Advertise |

© Copyright 2020 Match Group, LLC

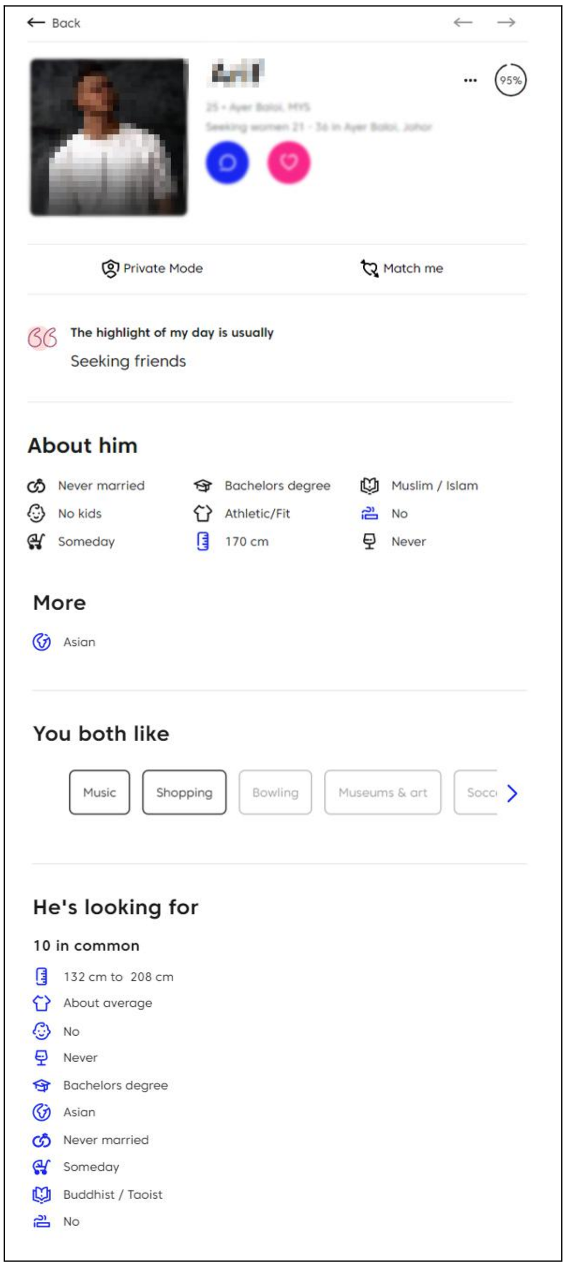
UXPin

If you can imagine it, you can design it.

Figure 2.11 Page Showing the Matched Profiles

(d) Profiles Viewing

The user can view the profile's summary, personal information and the specific details of the profile's match preferences. Besides, the percentage of the user and the particular profile's compatibility will be displayed. It is calculated based on the user and the particular member’s interests and what they are looking for. User can send interest or send a message.



Back

[redacted]

[redacted]

95%

Private Mode   Match me

The highlight of my day is usually

Seeking friends

About him

Never married	Bachelors degree	Muslim / Islam
No kids	Athletic/Fit	No
Someday	170 cm	Never

More

Asian

You both like

Music | Shopping | Bowling | Museums & art | Socc

He's looking for

10 in common

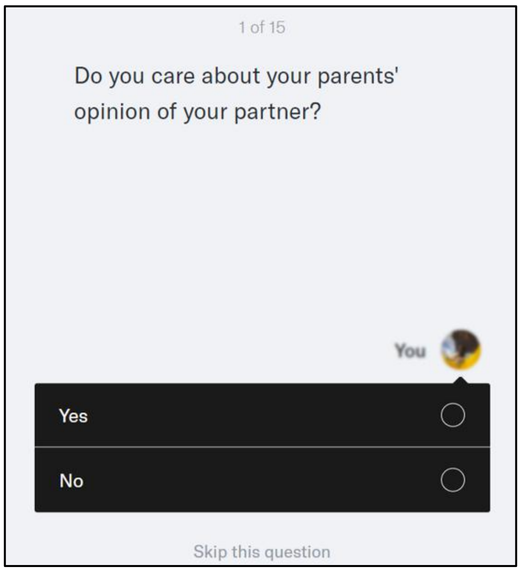
- 132 cm to 208 cm
- About average
- No
- Never
- Bachelors degree
- Asian
- Never married
- Someday
- Buddhist / Taoist
- No

Figure 2.12 Sample User Profile Information Page of Match.com

2.1.3 OkCupid.com

(Available at https://www.okcupid.com)

OkCupid is a dating website that launched in the year 2004. According to the proof from the website, it has over 91 million connections made every year and 50 thousand dates made every week. OkCupid combines user-generated questions and mathematics to determine members' compatibility. Users will need to answer several questions when signing up for an account.


1 of 15

Do you care about your parents' opinion of your partner?

You

Yes

No

Skip this question

Figure 2.13 Sample Question to be Answered During Sign Up

(a) Double Take

After logging, users will be able to see the Double Take feature in the home page of OkCupid.com. Double Take feature will display the potential profiles based on the criteria set by users and the overall match percentage with the users based on their answers. Users can choose to “Pass” the profile if they are not interested, or “Like” if they want to.

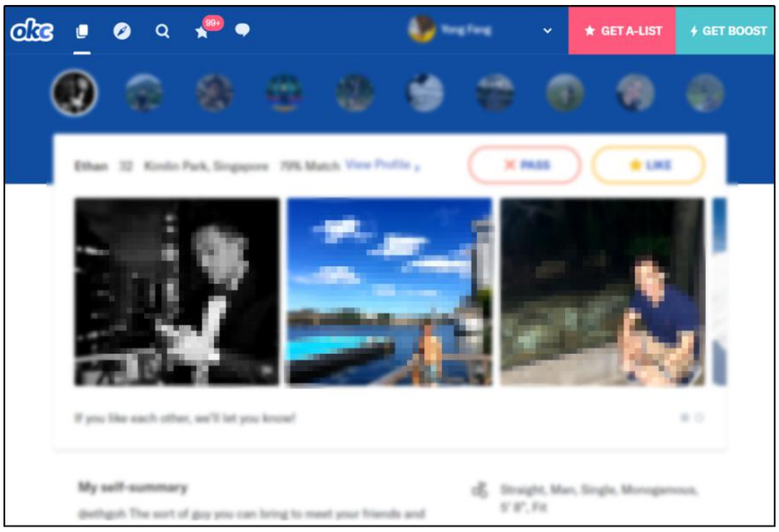


Figure 2.14 Home Page of OkCupid.com

(b) Discovery

The Discovery feature enables users to search for people who share similar interest. Users can type in general topics like hiking or more specific keywords like lonely. The system will scan the members' profiles to find for the keyword. Besides, the Discovery feature will display the comments of the users' potential matches on snapshot and questions. Users can search for people who care about the same questions.

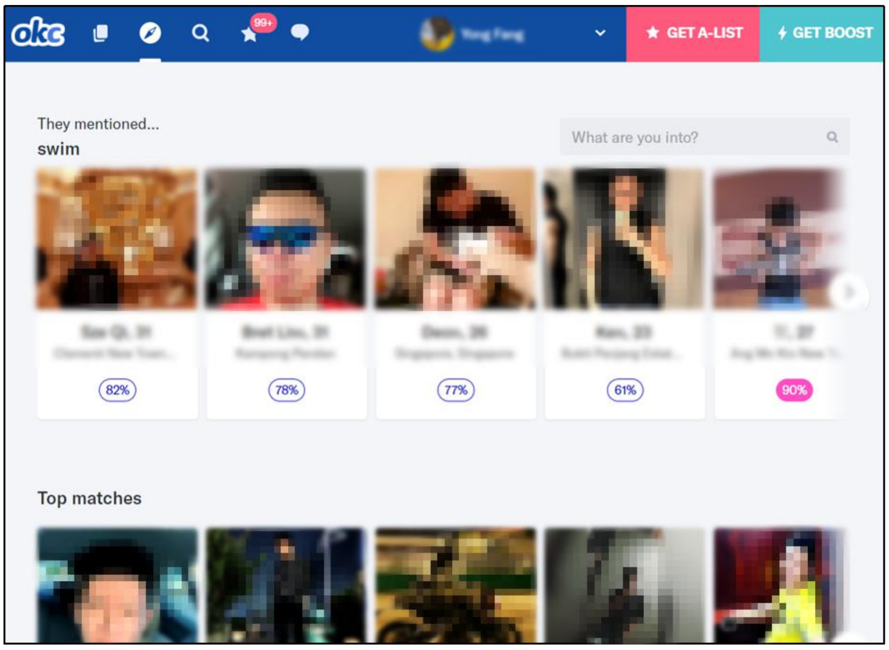
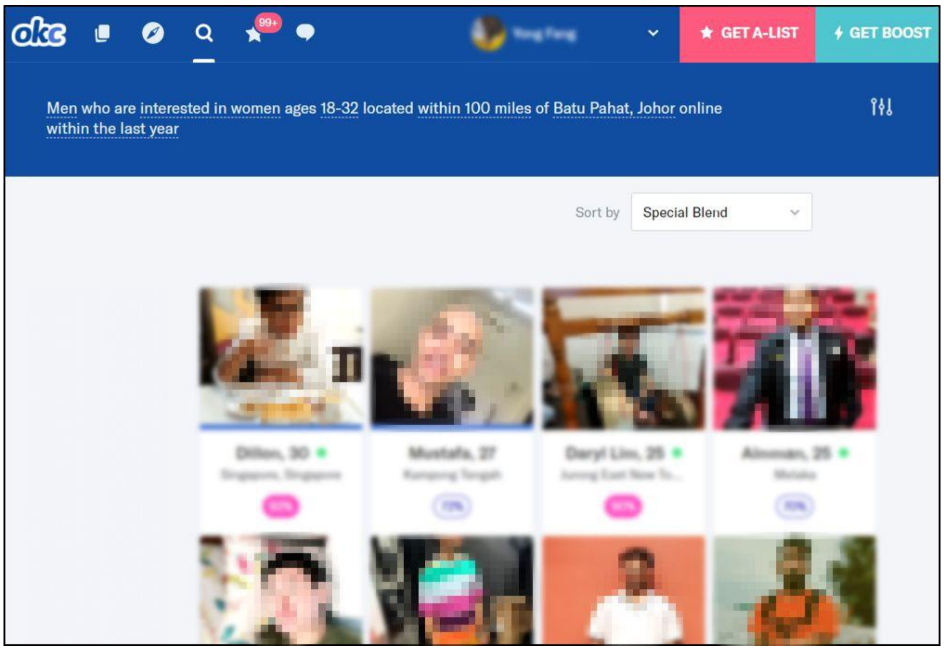


Figure 2.15 Discovery Feature of OkCupid.com

(c) Search for Profiles

The website provides a searching feature. Users can set the search criteria such as gender, age, location, looks, background, availability, vices and the members' answer to a question. The system displays a profiles' list that exactly match the criteria.



Men who are interested in women ages 18-32 located within 100 miles of Batu Pahat, Johor online within the last year

Figure 2.16 Search Feature of OkCupid.com

(d) Profiles Viewing

The user can view the profile's summary, personal information and the specific details of the profile's match preferences. Besides, the percentage of compatibility between the user and the profile will be displayed. It is calculated based on the user and the member's answers in the personality quiz. The user can view the questions that they agree or disagree with. The user can also send a message to the members that he/she interested. However, the members will only see the message if they liked the user back.

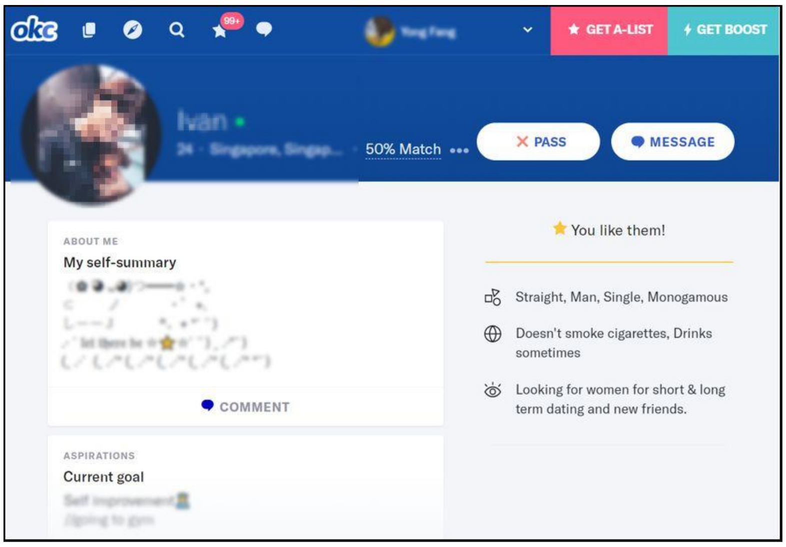

GET A-LIST
GET BOOST
50% Match
PASS
MESSAGE
ABOUT ME
My self-summary
COMMENT
ASPIRATIONS
Current goal
You like them!
Straight, Man, Single, Monogamous
Doesn't smoke cigarettes, Drinks sometimes
Looking for women for short & long term dating and new friends.

Figure 2.17 Sample User Profile Information Page of OkCupid.com

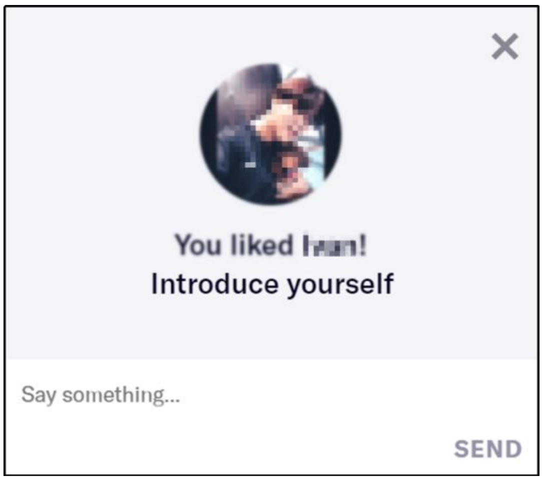


Figure 2.18 Messaging Function

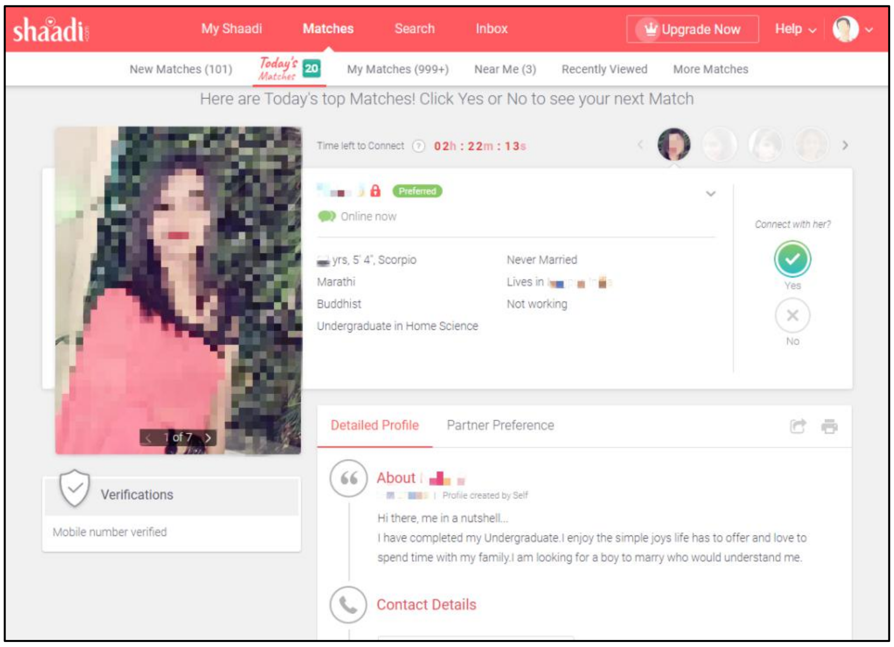
2.1.4 Shaadi.com

(Available at https://www.shaadi.com/)

Shaadi.com is the oldest matrimonial service in the world that caters to India, Canada, UK, Australia, Singapore, and the USA. According to Anupam Mittal, the founder of Shaadi.com, Shaadi.com was founded to increase the chances to meet potential life partners through a superior matchmaking experience.

(a) Today's Matches

After logging, users will be able to see the Today Match feature on the home page of Shaadi.com. Today Match feature will display the daily matches recommended by Shaadi.com based around user's partner preference. Users can choose to click "Yes" for the profile they interested, or "No" in the other way.



Here are Today's top Matches! Click Yes or No to see your next Match

yrs, 5' 4", Scorpio
Marathi
Buddhist
Undergraduate in Home Science
Never Married
Lives in
Not working

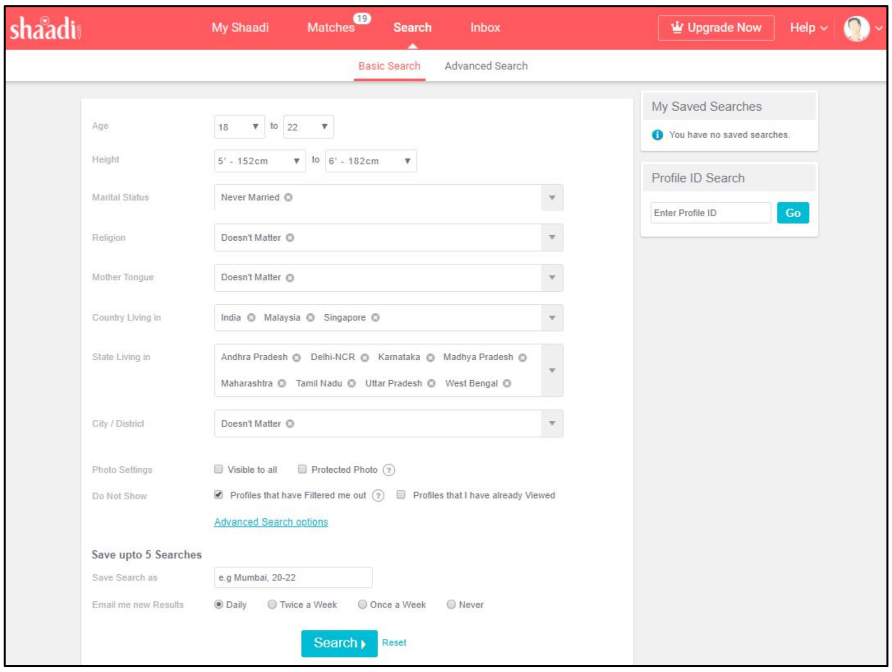
Mobile number verified

Hi there, me in a nutshell...
I have completed my Undergraduate.I enjoy the simple joys life has to offer and love to spend time with my family.I am looking for a boy to marry who would understand me.

Figure 2.19 Home Page of Shaadi.com

(b) Search for Profiles

The system provides two searching features, which are basic search and advanced search. The advanced search enables the users to search profiles based on their preferences, including sensitive details such as income, educational achievement, and lifestyle. The system also provides the option to search by keyword.



Age	18 to 22
Height	5' - 152cm to 6' - 182cm
Marital Status	Never Married
Religion	Doesn't Matter
Mother Tongue	Doesn't Matter
Country Living in	India, Malaysia, Singapore
State Living in	Andhra Pradesh, Delhi-NCR, Karnataka, Madhya Pradesh, Maharashtra, Tamil Nadu, Uttar Pradesh, West Bengal
City / District	Doesn't Matter
Photo Settings	☐ Visible to all ☐ Protected Photo
Do Not Show	☑ Profiles that have Filtered me out ☐ Profiles that I have already Viewed

Advanced Search options

Save upto 5 Searches

Save Search as	e.g Mumbai, 20-22
Email me new Results	[✓ Daily] Twice a Week Once a Week Never

My Saved Searches

You have no saved searches.

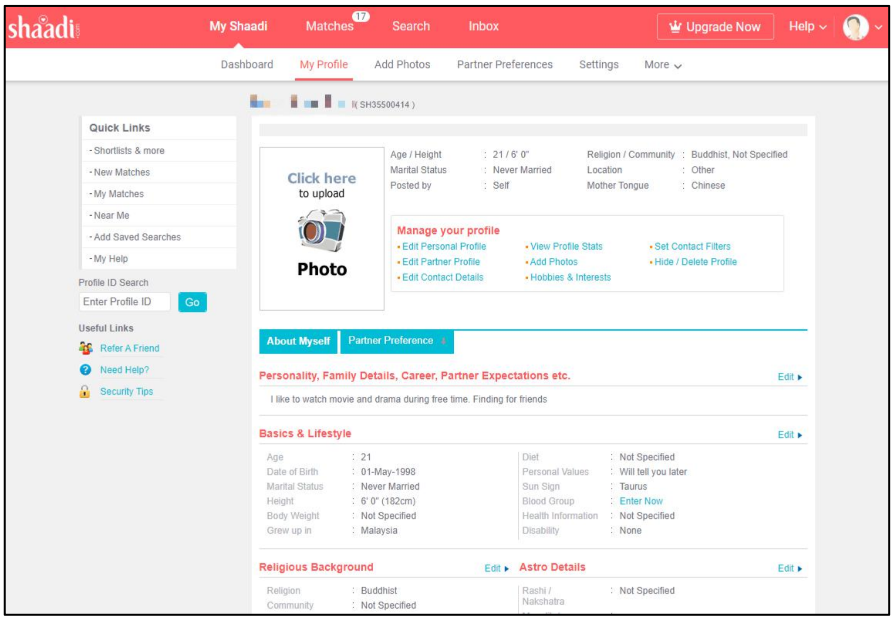
Profile ID Search

Enter Profile ID

Figure 2.20 Search Feature of Shaadi.com

(c) Matching Profile

User can provide complete biodata to increase the probability of matching with other members. The biodata includes lifestyle, location, religious background, astro details, family details, education, career and interests. Through the information provided, if the members are interested in the user, they can have a better understanding of the user before approaching.



(SH35500414)

Age / Height	21 / 6' 0"	Religion / Community	Buddhist, Not Specified
Marital Status	Never Married	Location	Other
Posted by	Self	Mother Tongue	Chinese

Personality, Family Details, Career, Partner Expectations etc.

I like to watch movie and drama during free time. Finding for friends

Basics & Lifestyle

Age	21	Diet	Not Specified
Date of Birth	01-May-1998	Personal Values	Will tell you later
Marital Status	Never Married	Sun Sign	Taurus
Height	6' 0" (182cm)	Blood Group	Enter Now
Body Weight	Not Specified	Health Information	Not Specified
Grew up in	Malaysia	Disability	None

Religious Background

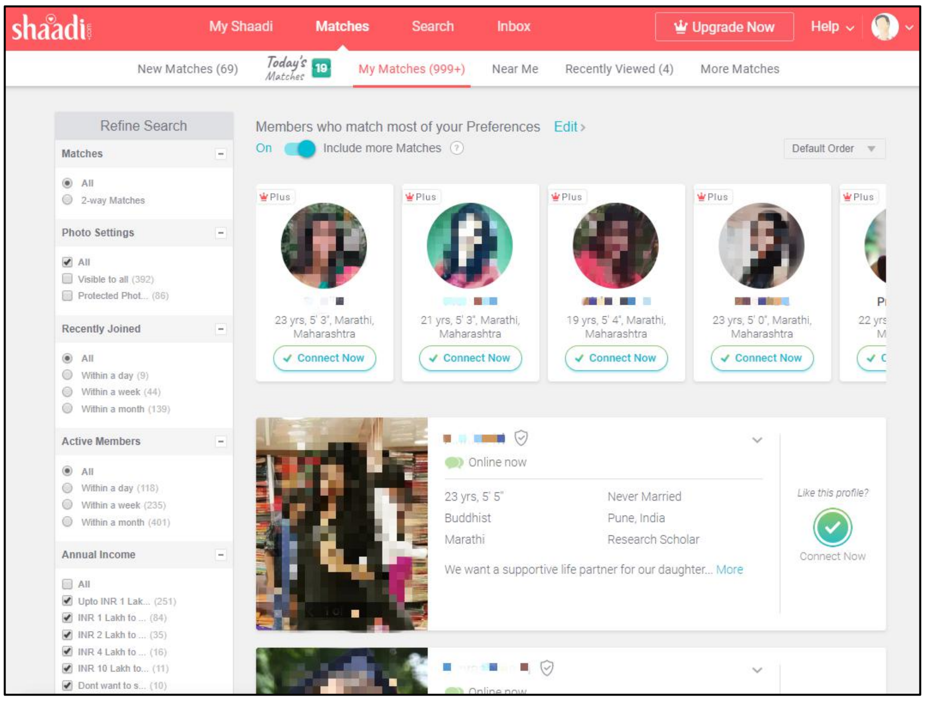
Religion	Buddhist
Community	Not Specified

Astro Details

Rashi / Nakshatra	Not Specified

Figure 2.21 Sample Input List for User Profile Information

Besides, the user can set their partner preferences. The system will find potential matches based on user preferences. A list of exact match profiles will be displayed for user viewing. Users can also use the filter option provided to refine the matched profiles.



Members who match most of your Preferences Edit ›

On Include more Matches

Matches: ● All ○ 2-way Matches

Photo Settings: ☑ All ☐ Visible to all (392) ☐ Protected Phot... (86)

Recently Joined: ● All ○ Within a day (9) ○ Within a week (44) ○ Within a month (139)

Active Members: ● All ○ Within a day (118) ○ Within a week (235) ○ Within a month (401)

Annual Income: ☐ All ☑ Upto INR 1 Lak... (251) ☑ INR 1 Lakh to ... (84) ☑ INR 2 Lakh to ... (35) ☑ INR 4 Lakh to ... (16) ☑ INR 10 Lakh to... (11) ☑ Dont want to s... (10)

23 yrs, 5' 3", Marathi, Maharashtra

21 yrs, 5' 3", Marathi, Maharashtra

19 yrs, 5' 4", Marathi, Maharashtra

23 yrs, 5' 0", Marathi, Maharashtra

22 yrs

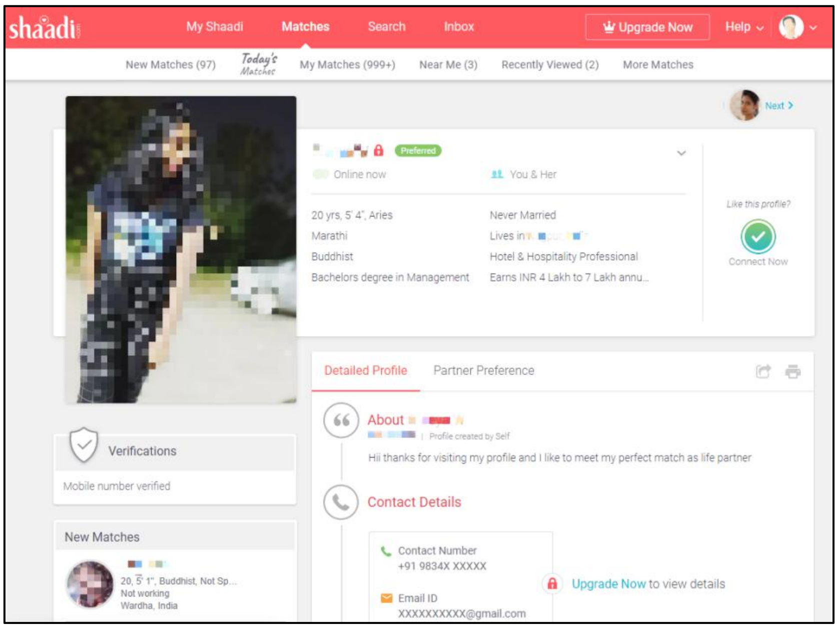
23 yrs, 5' 5"	Never Married
Buddhist	Pune, India
Marathi	Research Scholar

We want a supportive life partner for our daughter... More

Figure 2.22 Page Showing the Matched Profiles

(d) Profiles Viewing

The user can access the profile's personal information and the specific details of the profile's match preferences. However, only paid members can view the contact information. The system will display the number of the profile's partner preference that the user matched.



[redacted] Preferred

Online now   You & Her

20 yrs, 5' 4", Aries   Never Married
Marathi   Lives in [redacted]
Buddhist   Hotel & Hospitality Professional
Bachelors degree in Management   Earns INR 4 Lakh to 7 Lakh annu...

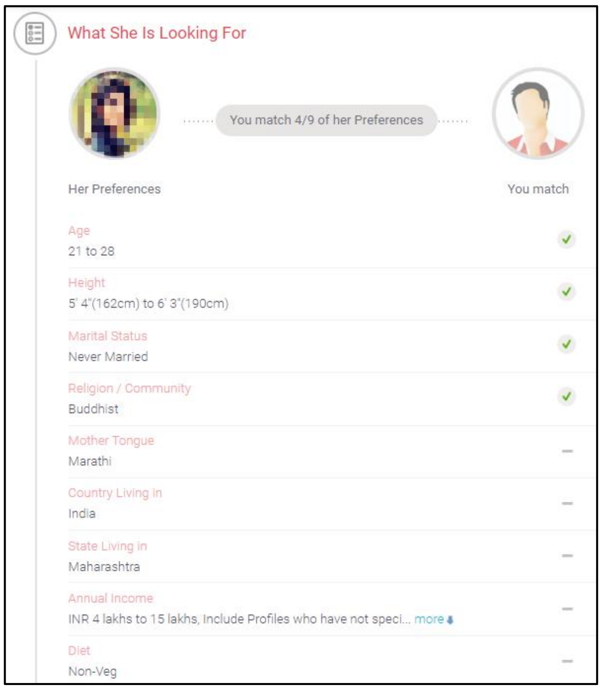
Verifications
Mobile number verified

New Matches
20, 5' 1", Buddhist, Not Sp...
Not working
Wardha, India

About [redacted]
[redacted] | Profile created by Self
Hii thanks for visiting my profile and I like to meet my perfect match as life partner

Contact Details
Contact Number
+91 9834X XXXXX
Email ID
XXXXXXXXXX@gmail.com
Upgrade Now to view details

Figure 2.23 Sample User Profile Information Page of Shaadi.com


What She Is Looking For

You match 4/9 of her Preferences

Her Preferences	You match
Age 21 to 28	✓
Height 5' 4"(162cm) to 6' 3"(190cm)	✓
Marital Status Never Married	✓
Religion / Community Buddhist	✓
Mother Tongue Marathi	–
Country Living in India	–
State Living in Maharashtra	–
Annual Income INR 4 lakhs to 15 lakhs, Include Profiles who have not speci... more	–
Diet Non-Veg	–

Figure 2.24 Matching Status of Preferences

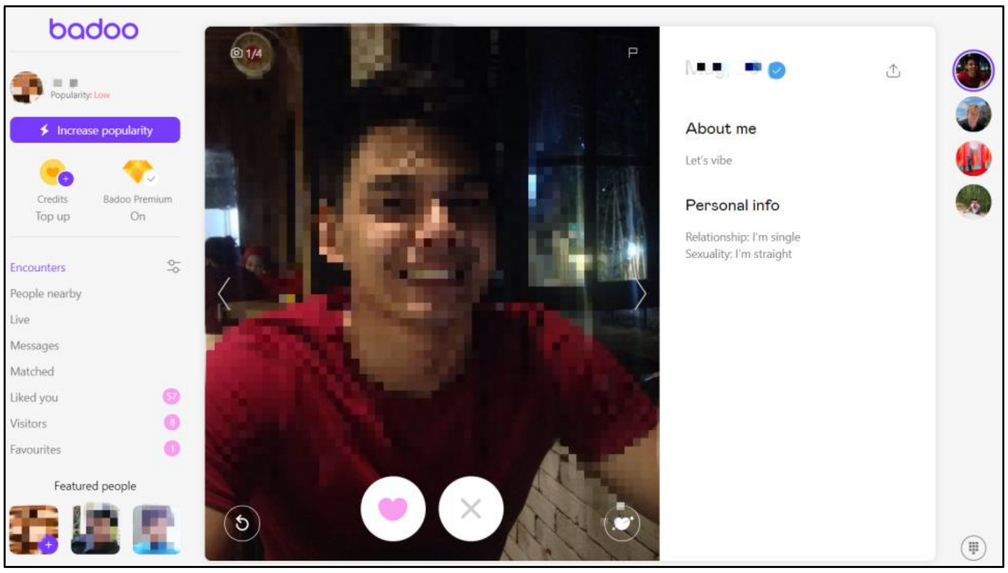
2.1.5 Badoo

(Available at https://badoo.com/encounters)

Badoo is the largest dating-focused social discovery network in the world. It is good for people looking for friendship or casual dating.

(a) Encounters

After logging, users will be able to see the Encounter feature on the home page. This feature enables users to search for matches quickly. It will show the member's picture with some personal information such as name and age. Users can choose to click "Like" for the profile they interested, or "Skip".



About me

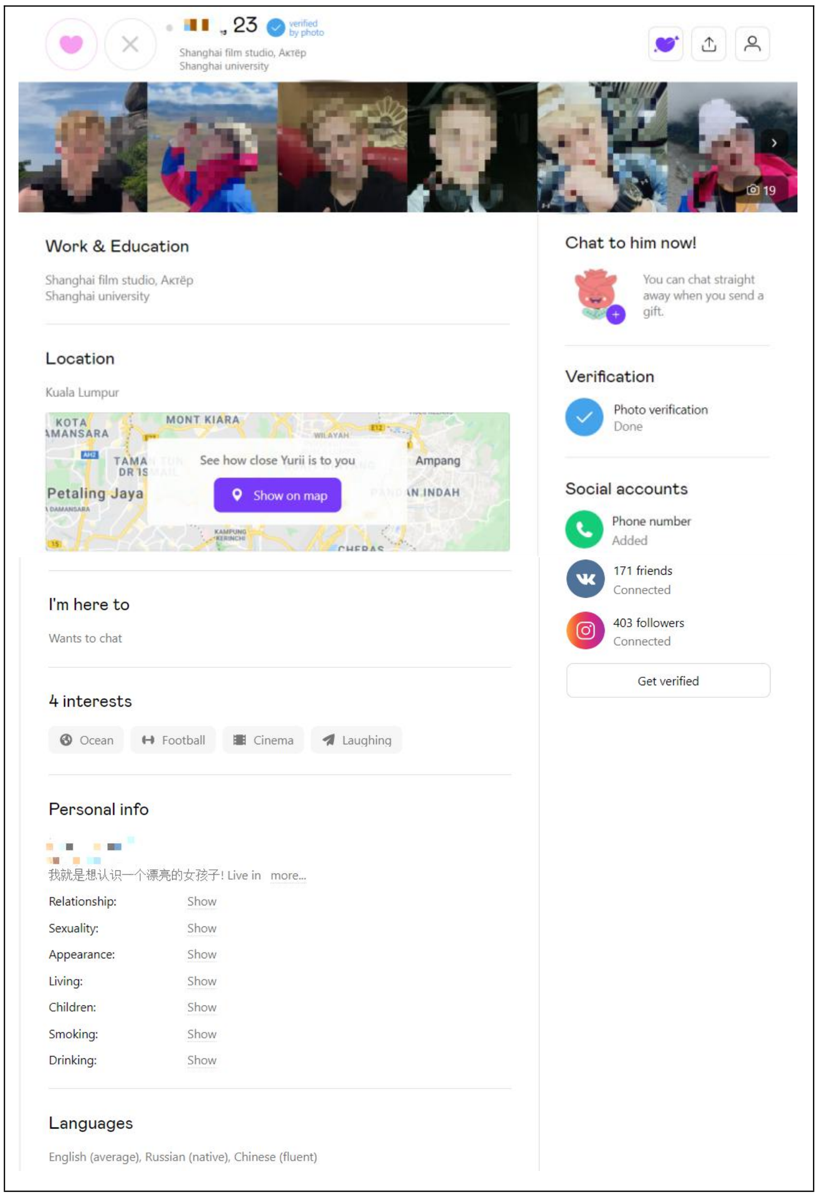
Let's vibe

Personal info

Relationship: I'm single
Sexuality: I'm straight

Figure 2.25 Home Page of Badoo

By clicking the name on the profile, users can view the detailed information.



, 23 verified by photo

Shanghai film studio, Актёр
Shanghai university

19

Work & Education

Shanghai film studio, Актёр
Shanghai university

Location

Kuala Lumpur

See how close Yurii is to you

Show on map

I'm here to

Wants to chat

4 interests

Ocean | Football | Cinema | Laughing

Personal info

我就是想认识一个漂亮的女孩子! Live in more...

Relationship:	Show
Sexuality:	Show
Appearance:	Show
Living:	Show
Children:	Show
Smoking:	Show
Drinking:	Show

Languages

English (average), Russian (native), Chinese (fluent)

Chat to him now!

You can chat straight away when you send a gift.

Verification

Photo verification
Done

Social accounts

Phone number
Added

171 friends
Connected

403 followers
Connected

Get verified

Figure 2.26 Sample User Profile Information Page of Badoo

(b) Matching

Two people are matched if both of them choose to “Like” each other. After matched, they are able to chat with each other.

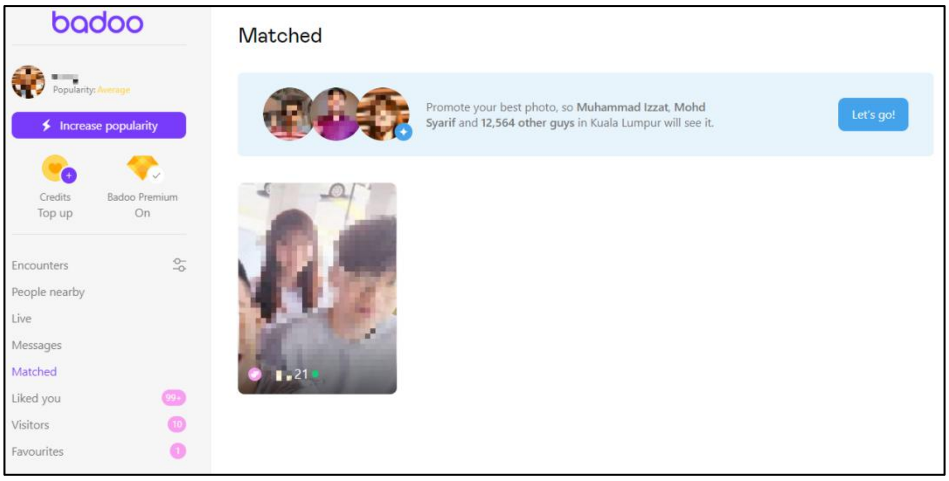


Figure 2.27 Page Showing the Matched Profiles

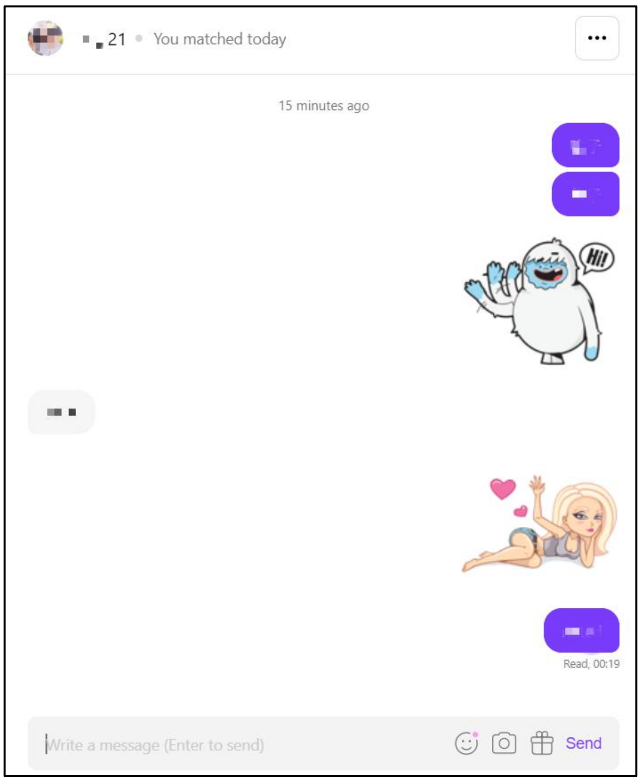


Figure 2.28 Chat Function

2.1.6 Comparisons on Existing Online Related System

Table 2.1 Comparison Matrix of Various Applications Similar to the Matrimonial System

Features Application Name	**Malaysian Cupid.com**	**Match. com**	**OkCupid. com**	**Shaadi. com**	**Badoo**
Recommended/Potential Profile	No	Yes	Yes	No	No
View Online Members	Yes	No	No	No	No
Matchmaking	Yes	Yes	No	Yes	Yes
Profile Searching	Yes	Yes	Yes	Yes	No
Viewing Profile	Yes	Yes	Yes	Yes	Yes
Add to Favourite	Yes	Yes	No	Yes	Yes
Chat	Yes	Yes	Yes	Yes	Yes
Compatibility Percentage	No	Yes	Yes	No	No

In conclusion, each of the systems has its speciality and unique features. Those systems have provided some interface design concept for the project. Besides, the existing online related systems also provide a variety of features that are appropriate for my project. This project will include all of the features mentioned below. These features are selected as they will contribute to the project's objective.

i. Profile Matching

This feature is important because it can help the user to find for their potential life partner. After the user set his/her match criteria, the system will perform similarity measures to find for the profiles that match the user's preferences. The system will display the result list obtained. The user can select the suitable profile to approach.

ii. Viewing Profile

This feature is chosen because the user can know about the members' information. The user can have a better understanding of that particular member before deciding to approach. It can assist the user in decision making for choosing the desired partner.

iii. Add to Favourite

The reason for choosing this feature is due to the convenience it provides. If the user is interested in one of the profiles, the user can add the profile to favourite. This feature enables the user to compare a list of profiles that are added to the favourite easily. The user can choose the most suitable profile after making a comparison.

iv. Chat Function

The user can interact with the members that he/she interested. This feature is useful to help the user in establishing a relationship with the potential life partner. The user may be able to know the potential partner more thoroughly through the conversation.

v. Compatibility Percentage

The compatibility percentage will be replaced with the similarity percentage. For each profile that matches the user's requirements, the application will display the similarity percentage of the profile with the user's preferences. Implementing this feature will make the user easier to select the desired partner. The higher percentage indicates that the member is closer to their partner preference. Hence, this feature can help the user to make the decision more correctly.

2.2 Matching Algorithm

2.2.1 Rule-based Approach

The rule-based system has a knowledge base represented as a collection of “rules” that are typically known as “if-then” clauses. According to Ross (2004), the if-then rules can be expressed as:

“IF cause (antecedent) THEN effect (consequent)”

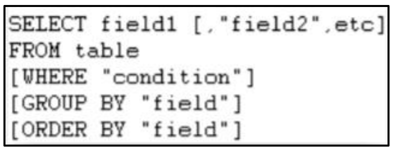
If the inputs like the premise, antecedent and condition are given, the output as a consequent can be derived. A rule can hold multiple inputs on the left-hand side (antecedent), but only one output on the right-hand side (consequent). Liu, Gegov and Cocea (2016) stated that the rules would be conjunctive if all the rules are joined by ‘and’ connector, or disjunctive if rules are linked by ‘or’ connectives. In general, the use of expert knowledge or learning from real data can help in designing the rules.

Table 2.2 Limitations of Rule-based Approach

Limitation	● Combinatorial explosion may occur due to large number of rules. ● Difficult and time-consuming to generate all the rules and conditions for a complex system. ● Not suitable for continuous variable. ● The existence of inconsistent rules may lead to uncertainty in classification.

2.2.2 Database Query / Exact Matching

Structured Query Language (SQL) is a standard language used to interact with a relational database (Almeida, 2016). SQL statements are used to perform tasks in databases like store, manipulate or retrieve data. The basic structure of a SQL query consists of the following elements:



```
SELECT field1 [,"field2",etc]
FROM table
[WHERE "condition"]
[GROUP BY "field"]
[ORDER BY "field"]
```

Figure 2.29 Basic Structure of SQL (Almeida, 2016)

The initial clauses, “Select” and “From” are mandatory whereas other elements are optional. One of the most important features of SQL query is the ability to filter data in databases, such as to pick only those records that fulfil certain requirements. The "Where" clause can be used to restrict the elements of a table that will be shown. For example, in the program that implemented the SQL query, if the user sets the search criteria, the criteria as the condition of a “Where” clause will be passed directly to the underlying database for processing. A list of exact matches will be returned and displayed to the user. However, if the leading column of an index on the table does not match with any columns in that “Where” clause, a full table scan will be performed. It will lead to slow performance.

Table 2.3 Limitations of Database Query

Limitation	● No probabilistic matching. ● Exact matching might cause no result to be returned if all the records in the database do not meet the constraints. ● Ordering is not well determined. ● Slow performance if a full table scan occurs.

2.2.3 Similarity Measure

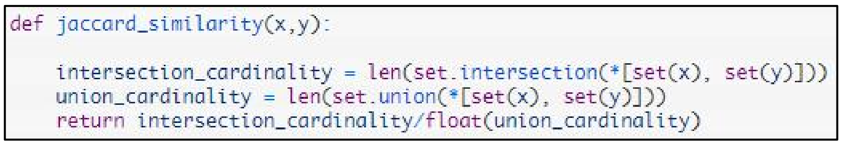
The similarity measure is the measure of the relation between a pair of objects (Polamuri, 2015). It determines how much identical two data items are. Similarity measures can also be defined as the distance with dimensions representing features of the objects. Two objects that have a high degree of similarity normally will have small distance among them. Similarity is usually measured in the range between 0 and 1, which 0 represents no similarity and 1 indicates the complete similarity. There are five most popular similarity measures that will be discussed in the following subsections.

1. Jaccard Coefficient

Jaccard Coefficient measures the similarity of two sets of data (Polamuri, 2015). To measure the similarity between two data sets through Jaccard Coefficient, the division between the size of the intersection and the size of the union of two data sets is calculated. The Jaccard coefficient takes a value between [0, 1] with 1 indicating the two data sets are completely similar and 0 indicating otherwise. When Jaccard coefficient between two sets of data is one, the number of elements in the intersection is the same as the union, which A∩B = A∪B. The mathematical representation of Jaccard Coefficient:

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|} \tag{2.1}$$

where ∩ = intersect, ∪ = union



```
def jaccard_similarity(x,y):

    intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
    union_cardinality = len(set.union(*[set(x), set(y)]))
    return intersection_cardinality/float(union_cardinality)
```

Figure 2.30 Jaccard Coefficient Implementation in Python (Polamuri, 2015)

Table 2.4 Strengths and Limitations of Jaccard Coefficient

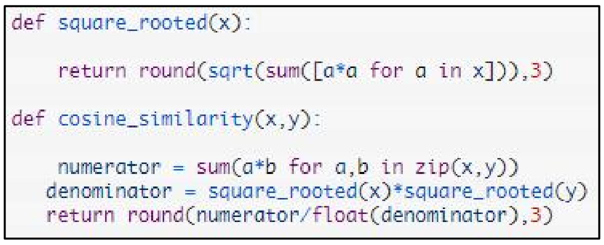
Strength	● It is good for measuring the similarity of binary data ● It is invariant to rotation
Limitation	● It is strongly oriented to weight common elements. ● It may give incorrect results when data sets contain missing observations.

2. Cosine Similarity

Cosine similarity measures the normalised dot product of the two attributes by finding the cosine angle between the two vectors (Polamuri, 2015). The outcome of the cosine similarity is between zero and one. If the angle between two vectors is 0^o, the two vectors will have a similarity of 1. In the other hand, the two vectors at 90^o have the similarity of 0, independent of the magnitude. The larger the angle between two vectors, the smaller their similarity. The mathematical representation of Cosine Similarity:

$$similarity(A,B)=\cos(\theta)=\frac{A\bullet B}{\|A\|\times\|B\|}=\frac{\sum_{i=1}^{d}A_i\times B_i}{\sqrt{\sum_{i=1}^{d}A_i^2}\sqrt{\sum_{i=1}^{d}B_i^2}} \tag{2.2}$$

where θ = angle between two vectors



```
def square_rooted(x):

    return round(sqrt(sum([a*a for a in x])),3)

def cosine_similarity(x,y):

    numerator = sum(a*b for a,b in zip(x,y))
   denominator = square_rooted(x)*square_rooted(y)
   return round(numerator/float(denominator),3)
```

Figure 2.31 Cosine Similarity Implementation in Python (Polamuri, 2015)

Table 2.5 Strengths and Limitations of Cosine Similarity (Shirkhorshidi, Aghabozorgi and Wah, 2015)

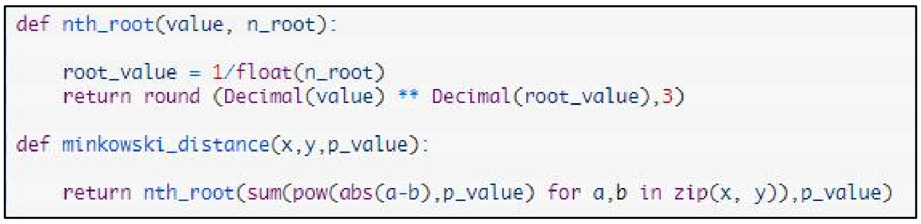
Strength	● It is invariant to rotation ● It is independent of vector length
Limitation	● It is variant to the linear transformation

3. Minkowski Distance

Minkowski distance is a generalisation of the Euclidean and Manhattan distances (Polamuri, 2015). It is a similarity measurement between two points in the normed vector space. The order of Minkowski metric, λ can be manipulated to calculate the distance in three different ways. When λ=1, it can be defined as Manhattan Distance. When λ=2, it is same as Euclidean Distance. When λ=∞ , it is Chebyshev Distance. Minkowski distance can represent the absolute distance between objects independently of their distance from the origin. The mathematical representation of Minkowski Distance:

$$d(A,B)=\sqrt[\lambda]{\sum_{i=1}^{d}|A_i-B_i|^{\lambda}} \tag{2.3}$$

where λ = the order of Minkowski metric



```
def nth_root(value, n_root):

    root_value = 1/float(n_root)
    return round (Decimal(value) ** Decimal(root_value),3)

def minkowski_distance(x,y,p_value):

    return nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_value)
```

Figure 2.32 Minkowski Distance Implementation in Python (Polamuri, 2015)

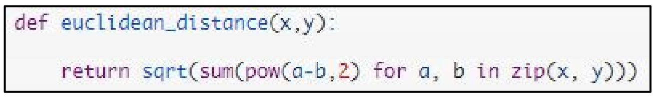
Table 2.6 Strengths and Limitations of Minkowski Distance (Shirkhorshidi, Ghabozorgi and Wah, 2015)

Strength	● It can perform well with the dataset clusters that are isolated or compacted
Limitation	● The large-scale features may dominate the others

4. Euclidean Distance

Euclidean Distance is the most used distance function in many applications (Polamuri, 2015). The Euclidean distance measures the straight-line distance between two points by following the Pythagorean rule. The result of Euclidean Distance is usually greater than or equal to zero where zero indicates that two points are identical and the higher value shows less similarity. The mathematical representation of Euclidean Distance:

$$d(A,B) = \sqrt{\sum_{i=1}^{d} (A_i - B_i)^2} \tag{2.4}$$



```
def euclidean_distance(x,y):

    return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))
```

Figure 2.33 Euclidean Distance Implementation in Python (Polamuri, 2015)

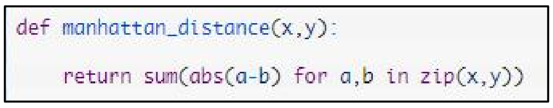
Table 2.7 Strengths and Limitations of Euclidean Distance (Shirkhorshidi, Aghabozorgi and Wah, 2015)

Strength	● It can perform well when applying to datasets with isolated or compact clusters.
Limitation	● The two data vectors that do not have shared attribute values tend to have a smaller distance, comparing to other pair of data vectors that contain the same attribute values. ● The large-scaled feature tends to dominate over others

5. Manhattan Distance

Manhattan Distance is a measure to obtain the difference between two points along axes at right angles (Polamuri, 2015). It calculates the absolute sum of the difference between their Cartesian coordinates which are the x-coordinates and y-coordinates. Manhattan distance metric is also recognized as the taxicab metric, rectilinear distance, or city block distance. The mathematical representation of Manhattan Distance:

$$d(A,B)=\sum_{i=1}^{d}|A_i-B_i| \qquad (2.5)$$



```
def manhattan_distance(x,y):

    return sum(abs(a-b) for a,b in zip(x,y))
```

Figure 2.34 Manhattan Distance Implementation in Python (Polamuri, 2015)

Table 2.8 Strengths and Limitations of Manhattan Distance (Shirkhorshidi, Aghabozorgi and Wah, 2015)

Strength	● Similar to Minkowski Distance, it can perform well when applying to datasets clusters that are compact or isolated.
Limitation	● It is sensitive to outliers.

2.2.4 Comparison of Matching Algorithms

Based on the reviews on the rule-based approach, database query and similarity measure, similarity measures are proposed to apply in this project. The rule-based approach is not suitable because each match preference will be a rule. As the application provides numerous match preferences, a large number of rules might be generated and causing a combinatorial explosion. Furthermore, it is not suitable for continuous variable like height. For the database query, exact matching will be performed instead of probabilistic matching. Exact matching might cause no result to be returned if all the records in the database do not meet the constraints. Besides, the ordering of the result returned is not well determined. Full table scan might occur and lead to slow performance.

The matrimonial application provides several match preferences for the users to select, such as age, religion, height, posture, marital status, mother tongue, education level, education field, occupation, smoking habits, child wish, hobby etc. Through similarity measures, the users can receive a list of results, sorted by similarity. Similarity measures can eliminate the drawback of exact matching. Even if no record in the database can exactly match the preferences set by the user, the system will still be able to return a list of low similarity results.

2.3 Software Development Methodology

Software Development Life Cycle (SDLC) is typically known as the process of developing software in a systematic manner and maintaining the quality of the product as per the standard. The SDLC framework involves several activities from the pre-development planning to post-development software testing and evaluation. All software project will undergo the phases of planning, analysis, designing, implementation and testing.

A methodology is a formalized approach that implements SDLC (Tegarden, Dennis and Wixom, 2009). The software development methodologies provide the basis for planning and controlling the entire process of development. There are a wide variety of software development methodologies in the current market. In the case of this project, four software development methodologies will be considered which are the waterfall, prototyping, phased development and agile.

2.3.1 Waterfall Methodology

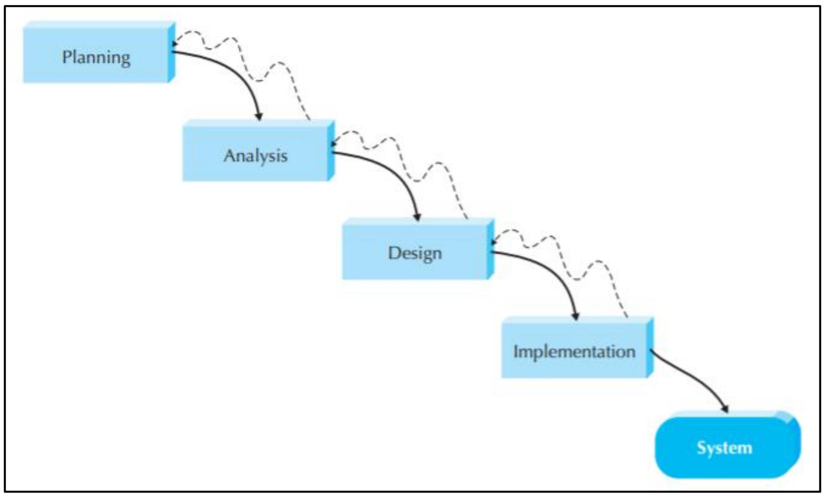


Figure 2.35 Waterfall Model Life Cycle (Tegarden, Dennis and Wixom, 2009)

The waterfall is the most well-known traditional software development methodology (Alshamrani and Bahattab, 2015). It is a linear and sequential approach in developing software. Before moving to the next phase, the current phase must be completed. In a waterfall process, the output of each phase will be the input of the subsequent phase. The key deliverables for each phase need verification and validation from the project sponsor (Tegarden, Dennis and Wixom, 2009). Once the key deliverables are approved, the phase end and the following phase begins. It is difficult to go backwards in the SDLC as the iterations can be costly. Significant rework will be required by repeating the previous phase for any adjustment. Furthermore, this methodology involves an extensive amount of written or electronic documentation. The following shows some pros and cons of the Waterfall methodology.

Table 2.9 Pros and Cons of Waterfall Methodology

Pros	System development progress can be monitored and controlled easily as each step has a clearly documentation.
	Faults in one phase can be detected before starting the following phase (Kannan, Jhajharia and Verma, 2014).
	Requirements are clearly understood before proceeding to the development.
Cons	Not suitable for projects that have ambiguous objectives. New requirements that arise during the development phase will not be considered.
	Inflexible and high amounts of risk and uncertainty (Alshamrani and Bahattab, 2015).
	It is time-consuming and costly as each phase may take a long time to process (Kannan, Jhajharia and Verma, 2014).
	Stakeholders are only involved at the beginning and end of project development.

2.3.2 Prototyping Methodology

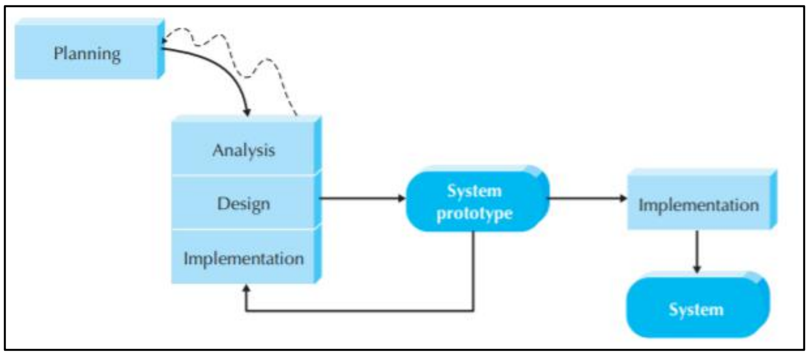


Figure 2.36 Evolutionary Prototyping Methodology (Tegarden, Dennis and Wixom, 2009)

Prototyping is an approach that supports the development of the system's initial version with a minimal number of features in a quickly way before actual implementation of the system (Rodríguez-Martínez, Mora and Alvarez, 2009). The prototypes can help developers to have better understand the user requirements as the users can view the overall design of the proposed system and provide feedback. Through the prototyping approach, a product can be built and refined subsequently to meet user expectation. The comments from users or project sponsor will be analysed to implement more features in the next version prototype (Tegarden, Dennis and Wixom, 2009). Once all the requirements have been clarified, the actual system will be implemented based on the approved prototype. The following shows some pros and cons of the Prototyping methodology.

Table 2.10 Pros and Cons of Prototyping Methodology

Pros	Higher likelihood of user acceptance of the final system due to user engagement throughout the development process
	Useful in understanding the user requirement. Prototype can help developers to resolve any unclear requirements.
	Reduce the time and cost as the errors can be found in the early development stage.
Cons	Difficult to be managed and controlled due to frequently changed requirements.
	The scope might be expanded significantly and increase the complexity of the prototypes (Beynon-Davies, Tudhope and Mackay, 1999).
	Users might have false expectations that the prototype is the complete system

2.3.3 Phased Development Methodology

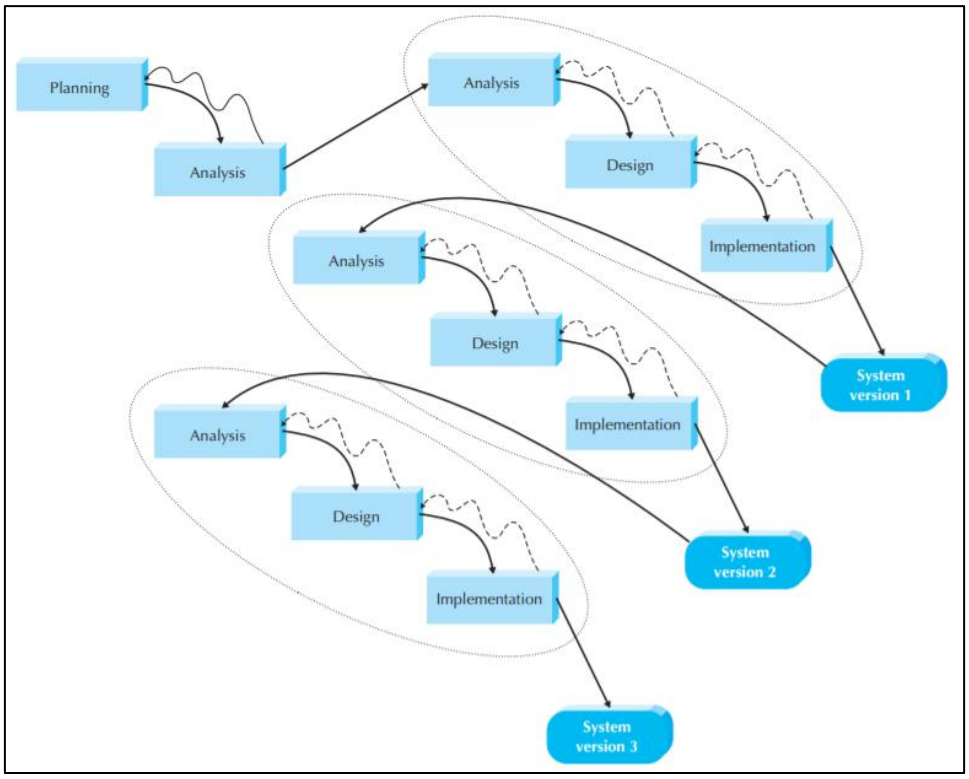


Figure 2.37 Phased Development Methodology (Tegarden, Dennis and Wixom, 2009)

The phased development methodology is a sequential approach which breaks the whole system into several phases or versions (Tegarden, Dennis and Wixom, 2009). In the phased development methodology, the overall system concept will be defined in the analysis phase. Then, the requirements will be categorized into a variety of versions. After the analysis phase, the design and implementation phase will begins with the requirements defined for the first version only. The first version of the system consists of the most important and fundamental requirements. Each version has its unique process of analysis, design and implementation. Once the implementation of the first version is completed, additional analysis will be conducted on the previously defined requirements, which will be coupled with the user feedback on their experience with the first version. Then, the second version will start to be designed and implemented. The iteration process will continue until the system is completed and accepted by users. The following shows some pros and cons of the phased development methodology.

Table 2.11 Pros and Cons of Phased Development Methodology

Pros	Defects are easy to be identified and handled during each iteration.
	Quickly getting a workable system to users can create business value early (Tegarden, Dennis and Wixom, 2009).
	Risk of failure and changing the requirement can be reduced. (Alshamrani and Bahattab, 2015)
Cons	Require good planning and design to identify the important and fundamental features for the first version (Alshamrani and Bahattab, 2015)
	User works with the intentionally incomplete system (Tegarden, Dennis and Wixom, 2009)

2.3.4 Extreme Programming

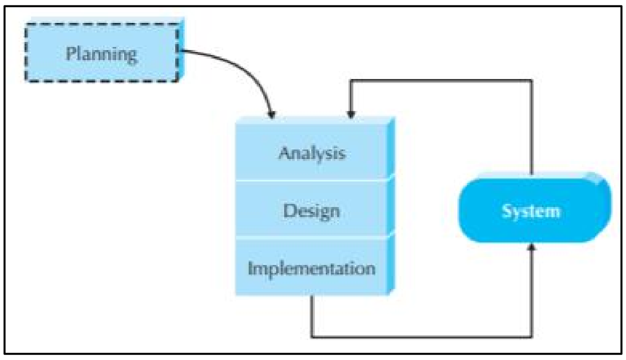


Figure 2.38 Extreme Programming (Tegarden, Dennis and Wixom, 2009)

According to Geambasu, et al. (2011), extreme programming (XP) is a technique for developing software based on the principles of simplicity, communication, feedback, respect and courage. Extreme Programming allows the requirements to be modified at any stage throughout the project life. In XP, user stories are written to describe the user requirements (Tegarden, Dennis and Wixom, 2009). Then, the system will be organized into smaller incremental parts to implement the stories. Simple analysis, design and implementation phases will be performed iteratively after the planning phase. Small releases of iterative versions of the system will be released frequently to customers. Testing and efficient coding practices are important. Sharma, Sakar and Gupta (2012) suggested that the bugs that have been detected during the testing will be removed in the next iteration. Besides collecting user feedback through functional tests, the development team can also collect system feedback through unit tests (Geambasu, et al., 2011). Those feedback will be used to evaluate the system and ensure the user requirements are correctly met. The following shows some pros and cons of the extreme programming methodology.

Table 2.12 Pros and Cons of Extreme Programming

Pros	Flexible schedule and enable the changing of requirements throughout the project life (Sharma, Sarkar and Gupta, 2012).
	Ensure user satisfaction due to user involvement throughout the development (Sharma, Sarkar and Gupta, 2012).
	The bugs can be identified quickly and easily due to frequent software testing (Yadav, Yasvi and Shubhika, 2019).
Cons	Unable to maintain a large and complex system that built with XP due to lack of analysis and design documentation (Tegarden, Dennis and Wixom, 2009).
	Not suitable for a large and complex system as the communications between large groups might not be effective.
	Required skill programmers to incorporate frequent changes in the project (Yadav, Yasvi and Shubhika, 2019).
	Twice of development cost due to the practice of pair programming. One work will require two people to do instead of one.

2.3.5 Comparison of Software Development Methodologies

Table 2.13 Comparison matrix of various methodologies (Tegarden, Dennis and Wixom, 2009)

Factors for comparison	**Waterfall**	**Prototyping**	**Phased Development**	**Extreme Programming**
Unclear user requirement	Poor	Good	Good	Good
System complexity	Good	Poor	Good	Poor
Short Time Schedule	Poor	Good	Good	Good
With Unfamiliar Technology	Poor	Poor	Good	Poor

After reviewing and comparing on several development methodologies, phased development methodology is selected to be adopted for the development of the matrimonial application. This methodology is chosen as it is suitable for the project with unfamiliar technology. In this project, Firebase will be implemented in the back-end of the system. The phased development methodology provides an opportunity to investigate and understand the Firebase in depth before completing the system design (Tegarden, Dennis and Wixom, 2009). It allows the developer to adapt to the system in smaller incremental steps instead of leaping towards a major new product. Furthermore, this methodology supports the project with a short-time schedule. It is appropriated for this project as the working product is required within a short period of approximately two to three months.

Waterfall is more suitable for large and complex projects. The strict controls and clearly defined development steps of the Waterfall methodology may cause the application development take a long time to process. Furthermore, waterfall methodology is more appropriate for the projects that have clearly defined user requirement. Any changing requirement requires a rework by repeating the previous process stages. It will significantly affect the project schedule. Hence, waterfall methodology will not be employed in this project that has a short-time schedule.

For prototyping methodology, it is not suitable for this project that uses unfamiliar technology. The prototypes in the early phases usually only touch on the surface of the new technology. It has a high possibility that the weakness or problems in the new technology are recognized after a few prototypes were developed. Furthermore, a complex system needs to be analysed and designed in details. However, the prototyping methodology only performs basic analysis and design, then immediately start on the prototype development. Consequently, prototyping methodology will not be adopted in this project.

As Extreme Programming practices pair programming, it is only suitable for the development team that has a minimum of two members and a maximum of 10 members. However, this project has only one developer. Besides, this methodology lacks detailed documentation that may increase the risk of scope creep. Problems may arise when implementing the unfamiliar technology in the later stage of the project. However, there is no proper documentation can be referred to solve the problem. Therefore, the XP methodology will not be considered in this project even though this methodology is responsive to changing requirement and supporting short-time schedule.

2.4 Usability Testing

According to Usability.gov (2020a), usability testing is the evaluation of a product or service by testing it with real users. It measures how easy a product is to use and how easy it is for the users to achieve their goal. Usability testing is different from traditional testing, such as bug testing. Usability testing is carried out with actual end users of the product, while traditional testing might only involve the developer, designer or project manager. Participants are required to complete typical tasks during usability testing. Then, the observers will record the participants' performance for evaluation. Usability testing aims to recognize any usability problems, collect qualitative and quantitative data and evaluate user satisfaction toward the product.

Nielsen (1993) classified usability into five sub-attributes, which are learnability, efficiency, memorability, errors, and satisfaction.

- Learnability describes the ability of users to perform fundamental tasks when they first experience the design.
- Efficiency measures the time spent by the users to accomplish the tasks.
- Memorability refers to the ability of the users to recollect how to use the system after a period of not using it.
- Errors measure the error rate of the system and the user ability to recover from mistakes.
- Satisfaction measures the comfort and acceptability of users when using the system.

Learnability, efficiency, and memorability can be evaluated by selecting users from different categories for the test, such as novices and experts. Satisfaction is typically measured by the users' rating with the system.

Table 2.14 Pros and Cons of Usability Testing

Pros (Usability.gov, 2020a)	It helps to identify the ease of use of the system.
	User feedback can be used to evaluate the application performance and make improvement.
	Issues and potential problems can be identified before launching the application
	It offers insight into how satisfied users are with the product.
Cons (Dicks, 2002)	It is hard to predict success in long-term usage
	Testing environment is different from the real work environment
	It is only possible to undertake with a small sample of potential users and the test participants might not fully represent the target population.

Furthermore, the System Usability Scale (SUS) can be used to evaluate the usability of a system (Usability.gov, 2020b). It consists of ten questions with five response choices. Among the ten questions, odd-numbered items worded positively, whereas even-numbered items worded negatively.

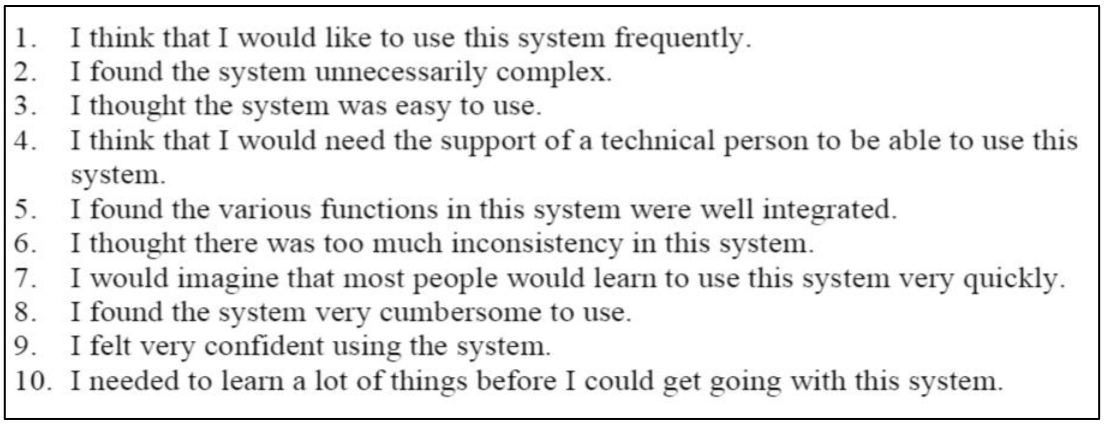

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Figure 2.39 Items in SUS (Lewis and Sauro, 2009)

According to Lewis and Sauro (2009), participants will rate each question from 1 to 5 based on their degree of agreement, which ranges from strongly agree to strongly disagree. Before the SUS score is calculated, the odd items' score is subtracted by 1, whereas the score of even items is subtracted by 5. Then, the sum of the all item score will be multiplied by 2.5. Hence, the SUS score will be range from 0 to 100. The average SUS score is 68 (Usability.gov, 2020b). If the score is below

68, it indicates that there would be some problems with the system usability. SUS is comparatively quick, simple and inexpensive but a reliable way to gauge the system usability. It can help to differentiate between usable and unusable systems effectively.

2.4.1 Lab Usability Testing

Lab usability testing is a testing that runs in a controlled environment and supervised by a moderator. Participants are required to complete a number of tasks by following the pre-defined scenarios in the usability laboratory. Testing sessions are conducted individually and the performance of participants will be recorded. The data collected will be used to calculate the performance times, identify and explain errors. User opinions' about the system can be evaluated through the user satisfaction questionnaires and interviews. Example of steps for conducting usability testing in the lab:

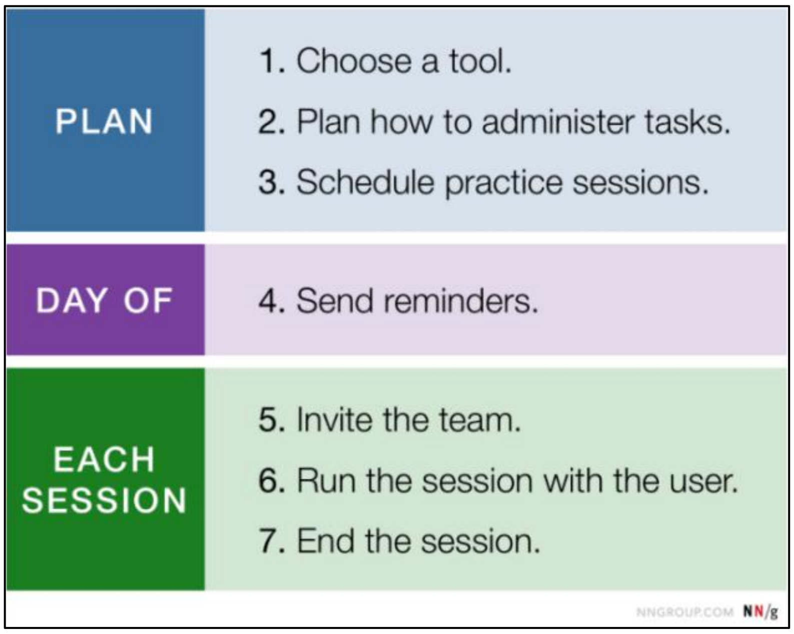
1. A representative group of users are selected to participate
2. The facilitator explains the test session to the participants.
3. Participants read the test scenario aloud and begin to perform tasks according to the scenario.
4. The participants' behaviours, comments, errors, and completion on each task will be observed and recorded.
5. Participants need to complete follow-up questions after complete the testing.
6. After the test session, the data collected will be evaluated.

Table 2.15 Pros and Cons of Lab Usability Testing

Pros	It provides extra insight into user behaviour through non-verbal cues such as facial expression and body language
	Controlled environment for testing enables participants to be more concentrated on the tasks
Cons	Higher cost compared to other types of testing.
	Scheduling issues might affect the timeline of testing.

2.4.2 Remote Usability Testing

Remote usability testing is a way to conduct usability testing in participants' natural environment (Usability.gov, 2020c). Unlike traditional usability testing, the remote usability testing session can be carried out even though the researcher and the participants located in different geographical locations. There are two types of remote usability testing, which are moderated and unmoderated (Usability.gov, 2020c). During moderated remote testing, the moderator will observe and communicate with the participants when they perform the testing. Moderated tests can be performed through real-time screen-sharing. Unmoderated remote testing is conducted with participants complete the tasks independently without interaction with the moderator. However, unmoderated remote testing is not recommended as it is harder to control as no moderator involved.



PLAN	1. Choose a tool. 2. Plan how to administer tasks. 3. Schedule practice sessions.
DAY OF	4. Send reminders.
EACH SESSION	5. Invite the team. 6. Run the session with the user. 7. End the session.

NNGROUP.COM NN/g

Figure 2.40 Summary of the Steps Involved in Remote Moderated Usability Testing (Moran and Pernice, 2018)

Table 2.16 Pros and Cons of Moderated Remote Usability Testing

Pros	It helps to save time and cost as the cost of hiring a professional usability lab can be eliminated.
	Testing in a natural environment can provide a better insight into true user behaviour.
	It provides an opportunity to connect with participants from various geographical regions.
Cons	The environment is hard to control.
	Non-verbal cues such as facial expression and body language will be missed out.
	Internet connectivity and internet speed are important as the remote testing relies on third-party screen-sharing tools.

2.4.3 Guerrilla Usability Testing

Traditional usability testing uses a specific place to conduct the test. However, guerrilla usability testing sets up a minimalist system that imitates the functionality of traditional labs in public spaces (Collado, Mora and Parham, 2013). Participants are approached in public areas instead of formal recruitment. Cafe, offices, classroom or almost any place can be used to host the usability testing sessions. The deliverables of guerrilla usability testing are typically qualitative rather than quantitative. The design and functionality can be validated quickly. Steps for conducting guerrilla usability testing (Babich, 2017) are as follow:

1. Pick the right location and approach people
2. Explains the test session to the participants
3. Start the testing session.
4. The participants' behaviours, comments, errors, and completion of each task will be observed and recorded.
5. Participants have to complete follow-up questions after complete the testing.
6. After the test session, the data collected will be evaluated.

Table 2.17 Pros and Cons of Guerrilla Usability Testing

Pros	It is low cost compared to formal testing
	Testing in a natural environment can provide a better insight into true user behaviour.
	A quick way to collect a large number of data
Cons	The participants may not be relevant to the target audience.
	Can lead to bias and over-dependency if all the participants are drawn in the same place.

2.4.4 Comparison of Usability Testing Methods

Table 2.18 Comparison on Attributes of Different Usability Testing Types

Attributes	**Lab Usability Testing**	**Remote Usability Testing (Moderated)**	**Guerrilla Usability Testing**
Geographic Diversity	Poor Limited to a single location	Good Not limited to a single location	Good No geographic limitations
Recruiting	Difficult Geographic pool is limited to the testing location.	Easier No geographic limitation for recruitment	Easier Approaching participants in public areas instead of formal recruitment
Qualitative Insights	Good Direct observation of user reactions. Participants' facial expressions and body language can be captured.	Poor Direct observation of user reactions. However, participants' non-verbal cues will be missed out.	Good Direct observation of user reactions. Non-verbal cues from the participants can be captured.
Cost	High High compensation costs for users and facilitator time.	Low No facility costs	Low Inexpensive to set up and run testing in public areas.

After reviewing and comparing several usability testing types, remote usability testing is selected to be adopted in this project. The most appropriate type of usability testing that will be applied in this project is typically dependent on project budget and time constraints. Remote usability testing is a low-cost method as it is simple to set up and no need much equipment. Hence, it can help to save more project budget. Besides, due to the outbreak of COVID-19 in Malaysia, it is more suitable to perform remote usability testing, rather than lab usability testing and guerrilla usability testing. Although remote usability testing is unable to provide insight into user's non-verbal behaviour, it can avoid direct contact with the participants and help in preventing the spread of COVID-19.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter explained the details of the methodology applied in this project, which is the phased development methodology. The work plan along a timeline was also discussed in the sub-section, including the work breakdown structure and the Gantt chart. The development tools will also be stated in this chapter.

3.2 Software Development Methodology

As the phased development methodology was implemented for this project, the development of similarity measures based matrimonial application was separated into several phases. The most important and fundamental part of the system was included in the first version.

3.2.1 Planning Phase

The problem statement of this matrimonial application was determined through some researches during the planning phase. It was found that there is an inclining trend in the number of marriages in Malaysia. Some people tend to late marriage as finding a suitable partner that meet the expectations is not easy. Besides, the limitations of the rule-based approach and SQL query were also identified after conducting in-depth analysis and interpretation.

Based on the defined problems, the project objectives were declared. The objective provided the direction for the development of application so that it can solve the issues faced by Malaysian nowadays. A clear objective can help to increase the chance leading to a successful outcome. The objectives defined for this project were as follow:

1. To develop a web-based application by providing a solution that enables an individual to find their potential matches for marriage as per their priorities.
2. To perform matching through similarity measures based on the requirements and priorities set by users.

After clarified the problem statements and objectives, the proposed approach, proposed solution as well as the project scope were identified. For the proposed approach, various types of software development methodologies were compared to find out the most suitable methodology for this project. The system architecture was demonstrated as an overview of the application's process flow. In this project, the target users were the individuals aged between 22 and 26 and wish to find for a suitable life partner for marriage. The necessary features to be delivered in the application were also defined in the project scope.

According to the project's objective and scope, a work plan had been determined, including the work breakdown structure and Gantt chart. By following the work plan during the project life cycle, the project can be completed within the timeline. It is a guideline to monitor and control the project.

3.2.2 Analysis and Design Phase

During the analysis phase, the overall system concept was identified. Some existing related systems were reviewed and analysed to determine the project requirements. A survey was also conducted to understand user requirements.

A total of 5 systems were being reviewed. Matrimonial application has the similar goal with the dating application, which is finding a partner. Hence, some dating applications were selected to be reviewed. Through observing and analysing the existing related system, the potential features that were useful for this project were identified. Those features were incorporated into the proposed system to create a comprehensive application. The existing systems also provided a better idea on the UI design of the matrimonial application.

A list of match preferences was collected from the existing related system. Then, an online survey was conducted with the distribution of a questionnaire that consisted of the list obtained. The purpose of the survey was to investigate user preferences during the selection of a life partner. There were a total of 25 respondents attended the survey. After analysing the responses from the respondents, the first 12 match preferences selected by most of the respondents were chosen to be implemented in the application.

After recognizing the features needed by the application, the development process was divided into four sequentially developed versions. The versions included:

1. Basic functionality and user interface
2. Firebase Authentication implementation

3. Server-side and Firebase implementation
4. Algorithm implementation

A prototype was prepared after the initial requirements had been collected. The prototype was used to demonstrate how users use the application to achieve their goals. It also showed the initial design of the application's user interface.

3.2.3 Phased Implementation

The system implementation was divided into three phases. Each phase has its unique process of analysis, design and implementation. When the current phase ended and the next phase began, additional analysis was conducted on requirements that were defined in the previous phase. Furthermore, testing was conducted before entering the next phase. It was to ensure the current implementation can run properly and will not break the previous implementation.

i. Phase 1: Basic functionality and user interface

In this phase, the front-end of the system was developed. The first version of the application included the most fundamental requirements, which implemented all the basic functionality and user interface.

ii. Phase 2: Firebase Authentication Implementation

Firebase Authentication was designed to enforce the authentication mechanism that can prevent unauthorized access to the application. In this project, only email/password sign-in method was enabled.

iii. Phase 3: Server-side and Firebase Implementation

The express.js and Firebase were set up to integrate with front-end functionality in phase 3. Database of the system was designed to set up the entity tables and define the relationship between each entity. Firestore, Real-time Database and Firebase Cloud Storage were responsible for storing data required in the application. Lastly, express.js was implemented as the server side to work with the HTTP requests and integrate with Firestore and Firebase Cloud Storage.

iii. Phase 4: Algorithm implementation

For phase 4, the matching algorithms for the application were implemented on the server-side (express.js). The server integrated with Firestore to retrieve the required user data for matching. The results of the matching algorithms will be returned to the application and displayed to users.

3.2.4 Testing Phase

After the final version had been finalized, unit testing, API testing and end-to-end testing were performed to ensure that the application will work as expected. Besides, usability testing was conducted to find out which matching algorithm is most suitable to help users find their life partner.

Before conducting the usability testing, dataset for testing was collected from 40 males and 45 females through questionnaires. They provided their personal information, such as demographic info, background and lifestyle. Before storing their data into the application, the data was transformed from strings to numerical values. For example, “Male” in the gender field was represented by 1, while “Female” was represented by 2. Such transformation was necessary so that similarity measures can be applied to the data later.

Then, eight participants (six males and two females) who aged between 22 and 26 were recruited for the usability testing. The testing sessions were conducted through one-to-one meetings in Microsoft Team. After all the tests completed, user acceptance testing was conducted to ensure that the system’s functionality fulfils the users’ expectations.

3.2.5 Deployment

Before deployment, the final version of the system was strictly tested to ensure the whole system can work properly. Once all the tests had been passed, the documentation for the application was finalized and the application was deployed.

3.3 Development Tools

3.3.1 ReactJs

React.js is a JavaScript library that used to develop the user interfaces, specifically for the single-page application. The reason for using React in this project is because it is easy to learn and has a wide variety of documentation, tutorials and training resources. Besides, it allows code reuse which can help to increase the development productivity. All components in React are isolated. Each component has its logic and controls its rendering. Hence, components are reusable and change in one component will not affect others. Code re-usability can help to boost productivity and facilitates further maintenance.

3.3.2 Firebase

This project implemented Firebase as the back-end service for application development. Firebase is a comprehensive app development platform that provides a lot of infrastructures for developing an application. The infrastructures provided include Cloud Firestore, Firebase Authentication, Cloud Storage and Real-time Database.

1. **Firebase Authentication**

   Firebase Authentication offers an easy and secure sign-in process. It offers an end-to-end identity solution, including email, password and popular federated identity providers such as Google and Facebook. It is also easy to implement and flexible for customization.

2. **Cloud Firestore**

   Cloud Firestore is a NoSQL document that enables for quickly storing, synchronizing and querying data for applications at the global scale. All data are stored as documents and collections. NoSQL is suitable for storing a large amount of data that are required for the data mining process in this project. Furthermore, Cloud Firestore enables synchronization data across devices, either online or offline.

3. **Cloud Storage**

   Cloud Storage was implemented in this project for storing and serving user-generated content such as user profile photos. It provides a simple and durable object storage service that scales to exabytes of data. The Firebase Cloud Storage can be used to upload and download files regardless of network quality. It will help save users' time and bandwidth as the users can retry the operation right where they stopped.

4. **Real-time Database**

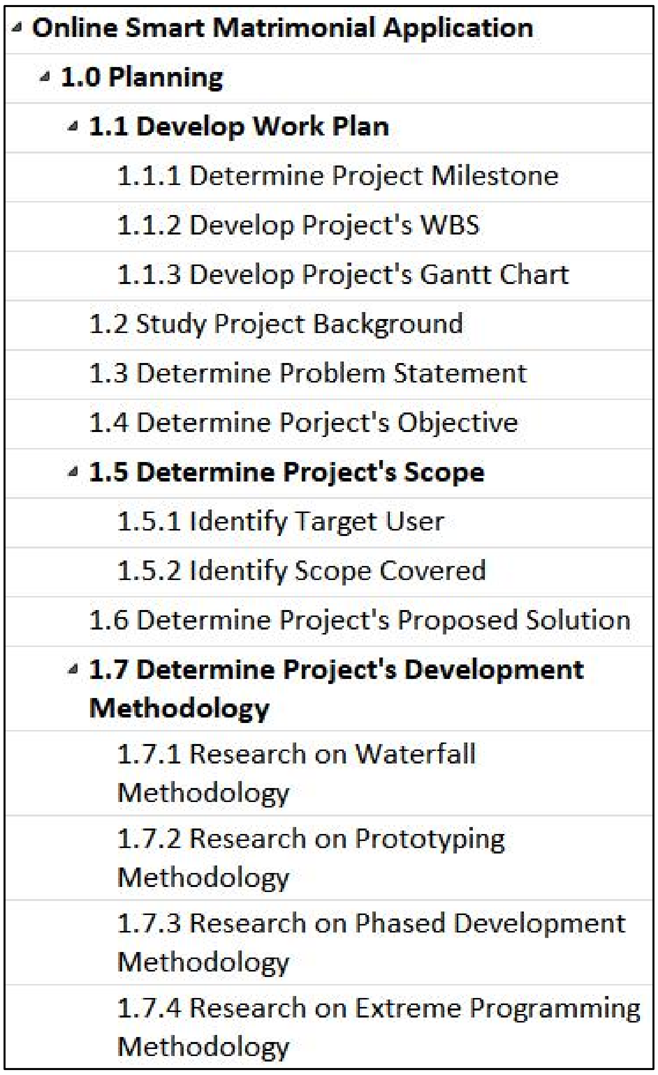
   By implementing Real-time Database, data are synchronized for users in real-time, even if the application goes offline. It is very useful for chat functionality in the application. All the users will be able to instantly receive updates with the latest data as they share the same instance of Real-time Database.

3.3.3 Express.js

Express.js is a Node.js web application framework with route support that can be used to build and handle API. In this project, express.js acted as the server that helps the application to interact with Firestore and Firebase Cloud Storage. The application will send the HTTP request to the server by calling the specific API. The server will then process the request and send the response back to the application. Express.js enables the code easier to maintain as the code will run in a managed environment. Besides, this project used express.js for data mining process. It was used to process and query user data based on the similarity measures.

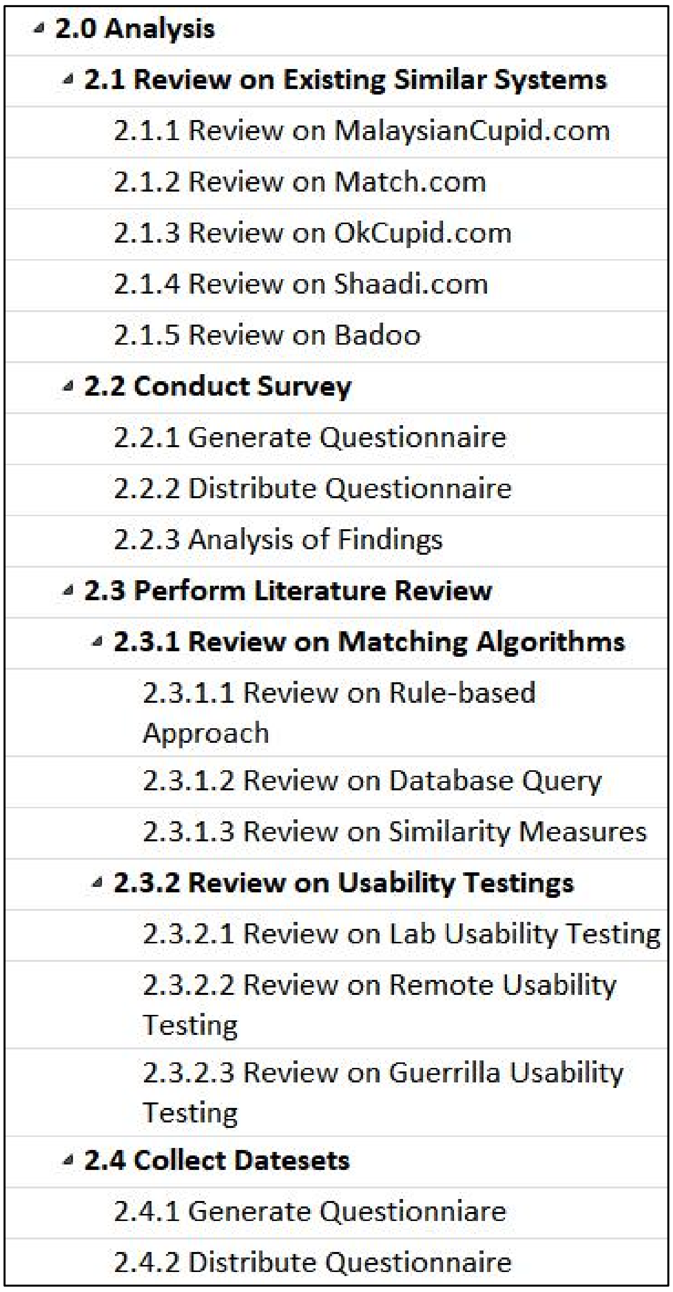
3.4 Project Planning

3.4.1 Work Breakdown Structure



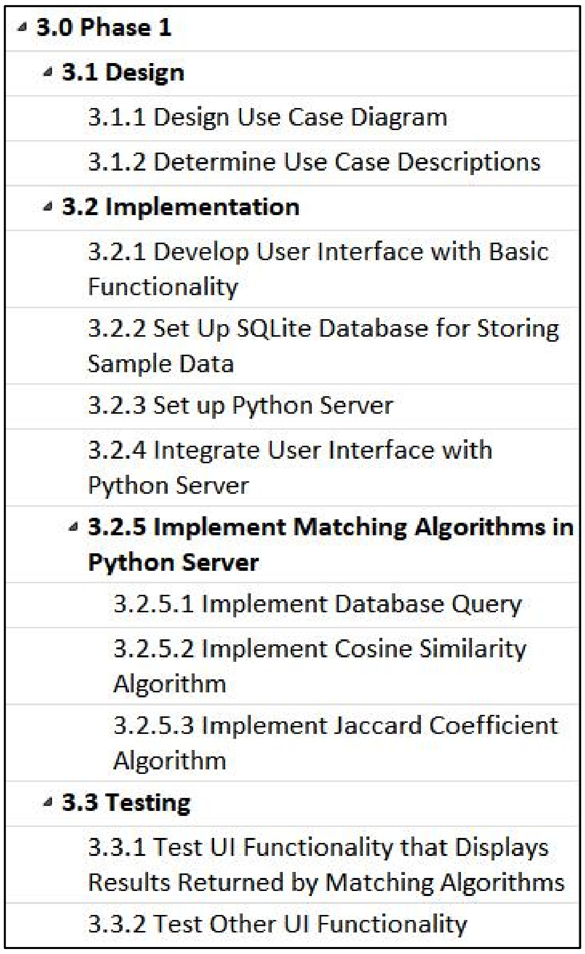
Online Smart Matrimonial Application
- **1.0 Planning**
  - **1.1 Develop Work Plan**
    - 1.1.1 Determine Project Milestone
    - 1.1.2 Develop Project's WBS
    - 1.1.3 Develop Project's Gantt Chart
  - 1.2 Study Project Background
  - 1.3 Determine Problem Statement
  - 1.4 Determine Porject's Objective
  - **1.5 Determine Project's Scope**
    - 1.5.1 Identify Target User
    - 1.5.2 Identify Scope Covered
  - 1.6 Determine Project's Proposed Solution
  - **1.7 Determine Project's Development Methodology**
    - 1.7.1 Research on Waterfall Methodology
    - 1.7.2 Research on Prototyping Methodology
    - 1.7.3 Research on Phased Development Methodology
    - 1.7.4 Research on Extreme Programming Methodology

Figure 3.1 Work Breakdown Structure Part 1



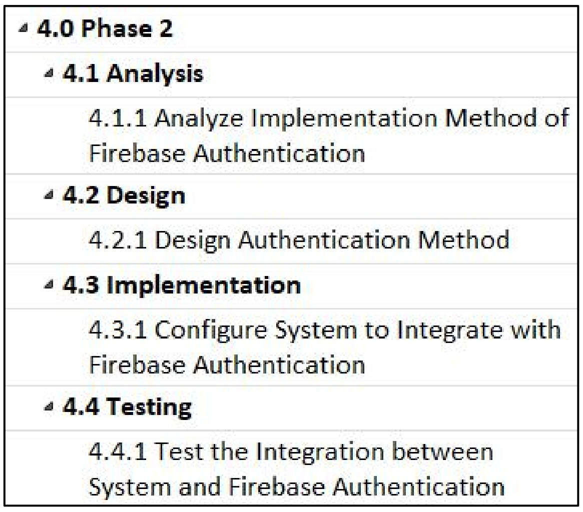
- **2.0 Analysis**
  - **2.1 Review on Existing Similar Systems**
    - 2.1.1 Review on MalaysianCupid.com
    - 2.1.2 Review on Match.com
    - 2.1.3 Review on OkCupid.com
    - 2.1.4 Review on Shaadi.com
    - 2.1.5 Review on Badoo
  - **2.2 Conduct Survey**
    - 2.2.1 Generate Questionnaire
    - 2.2.2 Distribute Questionnaire
    - 2.2.3 Analysis of Findings
  - **2.3 Perform Literature Review**
    - **2.3.1 Review on Matching Algorithms**
      - 2.3.1.1 Review on Rule-based Approach
      - 2.3.1.2 Review on Database Query
      - 2.3.1.3 Review on Similarity Measures
    - **2.3.2 Review on Usability Testings**
      - 2.3.2.1 Review on Lab Usability Testing
      - 2.3.2.2 Review on Remote Usability Testing
      - 2.3.2.3 Review on Guerrilla Usability Testing
  - **2.4 Collect Datesets**
    - 2.4.1 Generate Questionniare
    - 2.4.2 Distribute Questionnaire

Figure 3.2 Work Breakdown Structure Part 2



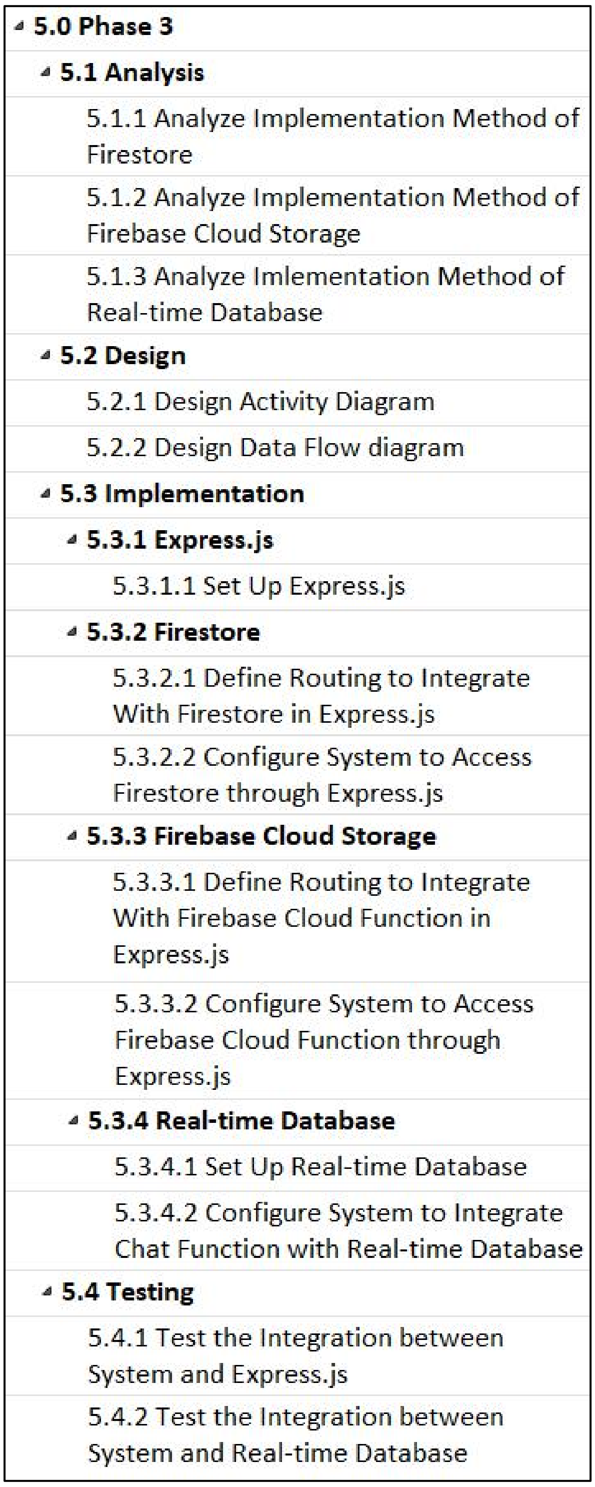
◢ **3.0 Phase 1**
- ◢ **3.1 Design**
  - 3.1.1 Design Use Case Diagram
  - 3.1.2 Determine Use Case Descriptions
- ◢ **3.2 Implementation**
  - 3.2.1 Develop User Interface with Basic Functionality
  - 3.2.2 Set Up SQLite Database for Storing Sample Data
  - 3.2.3 Set up Python Server
  - 3.2.4 Integrate User Interface with Python Server
  - ◢ **3.2.5 Implement Matching Algorithms in Python Server**
    - 3.2.5.1 Implement Database Query
    - 3.2.5.2 Implement Cosine Similarity Algorithm
    - 3.2.5.3 Implement Jaccard Coefficient Algorithm
- ◢ **3.3 Testing**
  - 3.3.1 Test UI Functionality that Displays Results Returned by Matching Algorithms
  - 3.3.2 Test Other UI Functionality

Figure 3.3 Work Breakdown Structure Part 3



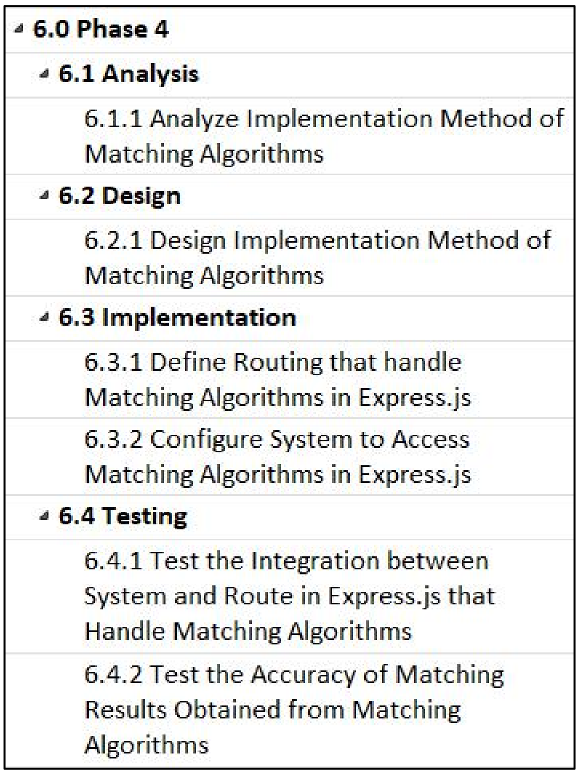
◢ **4.0 Phase 2**
- ◢ **4.1 Analysis**
  - 4.1.1 Analyze Implementation Method of Firebase Authentication
- ◢ **4.2 Design**
  - 4.2.1 Design Authentication Method
- ◢ **4.3 Implementation**
  - 4.3.1 Configure System to Integrate with Firebase Authentication
- ◢ **4.4 Testing**
  - 4.4.1 Test the Integration between System and Firebase Authentication

Figure 3.4 Work Breakdown Structure Part 4



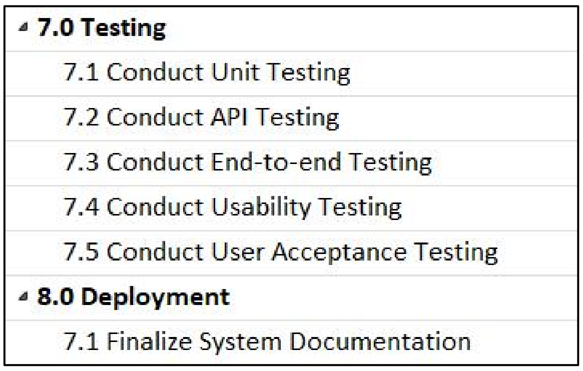
◢ **5.0 Phase 3**
- ◢ **5.1 Analysis**
  - 5.1.1 Analyze Implementation Method of Firestore
  - 5.1.2 Analyze Implementation Method of Firebase Cloud Storage
  - 5.1.3 Analyze Imlementation Method of Real-time Database
- ◢ **5.2 Design**
  - 5.2.1 Design Activity Diagram
  - 5.2.2 Design Data Flow diagram
- ◢ **5.3 Implementation**
  - ◢ **5.3.1 Express.js**
    - 5.3.1.1 Set Up Express.js
  - ◢ **5.3.2 Firestore**
    - 5.3.2.1 Define Routing to Integrate With Firestore in Express.js
    - 5.3.2.2 Configure System to Access Firestore through Express.js
  - ◢ **5.3.3 Firebase Cloud Storage**
    - 5.3.3.1 Define Routing to Integrate With Firebase Cloud Function in Express.js
    - 5.3.3.2 Configure System to Access Firebase Cloud Function through Express.js
  - ◢ **5.3.4 Real-time Database**
    - 5.3.4.1 Set Up Real-time Database
    - 5.3.4.2 Configure System to Integrate Chat Function with Real-time Database
- ◢ **5.4 Testing**
  - 5.4.1 Test the Integration between System and Express.js
  - 5.4.2 Test the Integration between System and Real-time Database

Figure 3.5 Work Breakdown Structure Part 5



- **6.0 Phase 4**
  - **6.1 Analysis**
    - 6.1.1 Analyze Implementation Method of Matching Algorithms
  - **6.2 Design**
    - 6.2.1 Design Implementation Method of Matching Algorithms
  - **6.3 Implementation**
    - 6.3.1 Define Routing that handle Matching Algorithms in Express.js
    - 6.3.2 Configure System to Access Matching Algorithms in Express.js
  - **6.4 Testing**
    - 6.4.1 Test the Integration between System and Route in Express.js that Handle Matching Algorithms
    - 6.4.2 Test the Accuracy of Matching Results Obtained from Matching Algorithms

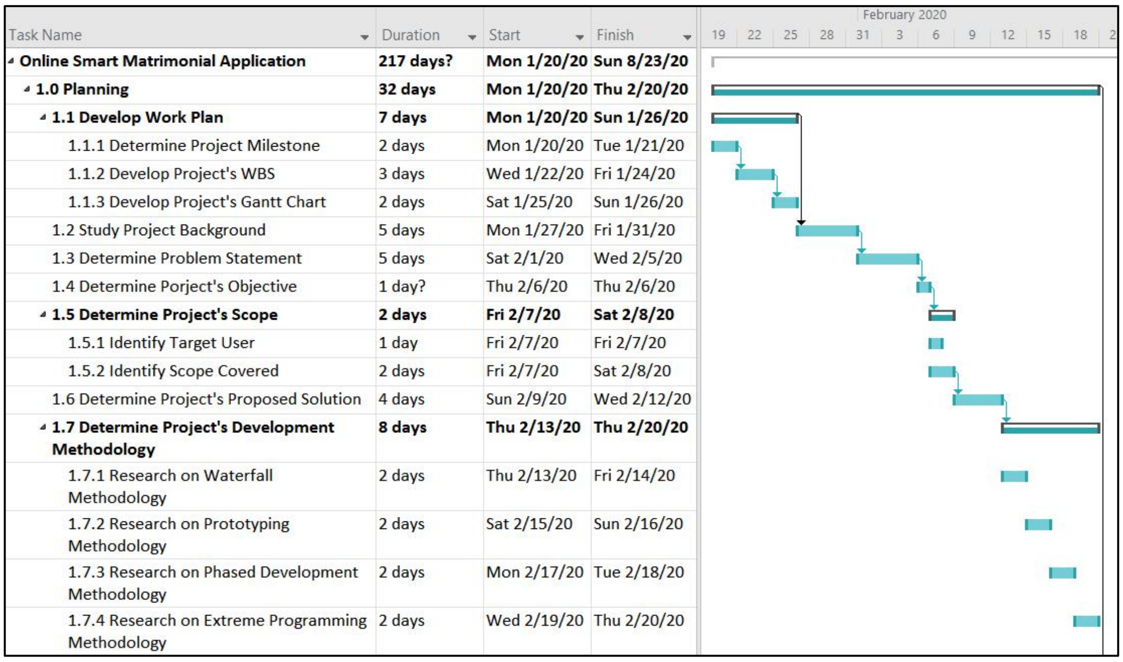
Figure 3.6 Work Breakdown Structure Part 6



- **7.0 Testing**
  - 7.1 Conduct Unit Testing
  - 7.2 Conduct API Testing
  - 7.3 Conduct End-to-end Testing
  - 7.4 Conduct Usability Testing
  - 7.5 Conduct User Acceptance Testing
- **8.0 Deployment**
  - 7.1 Finalize System Documentation

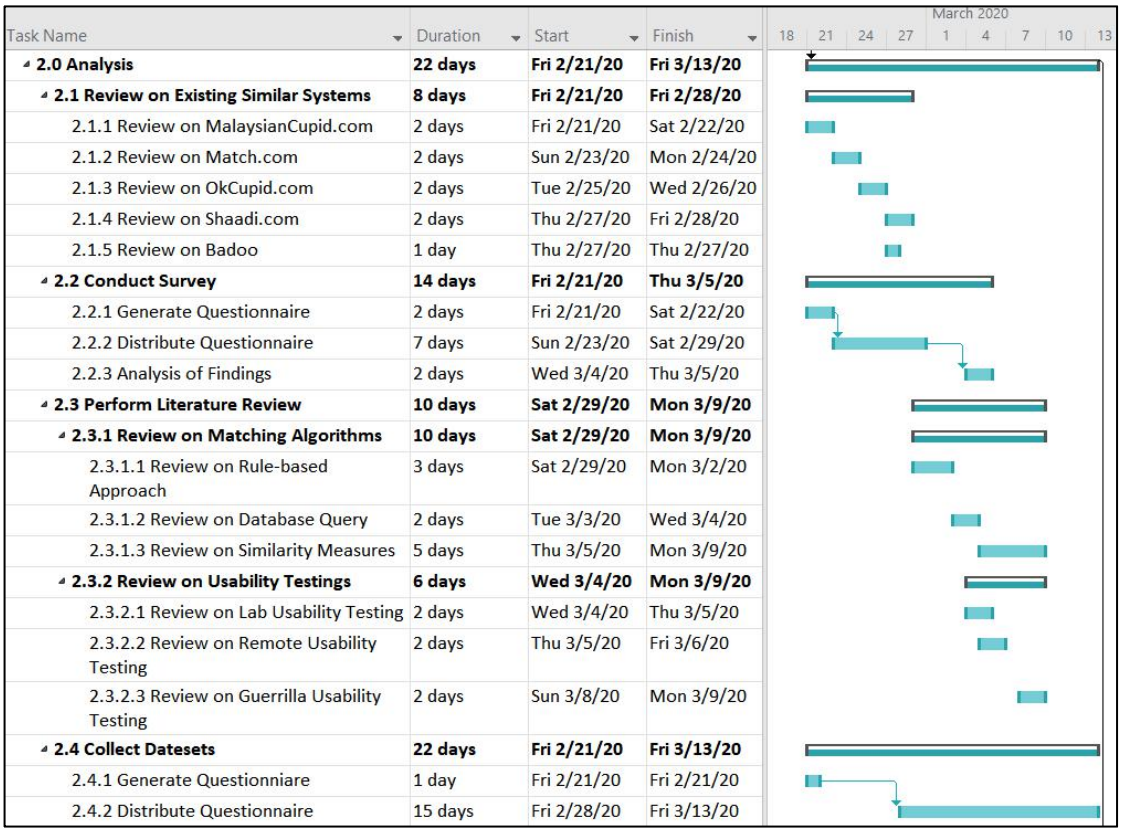
Figure 3.7 Work Breakdown Structure Part 7

3.4.2 Gantt Chart



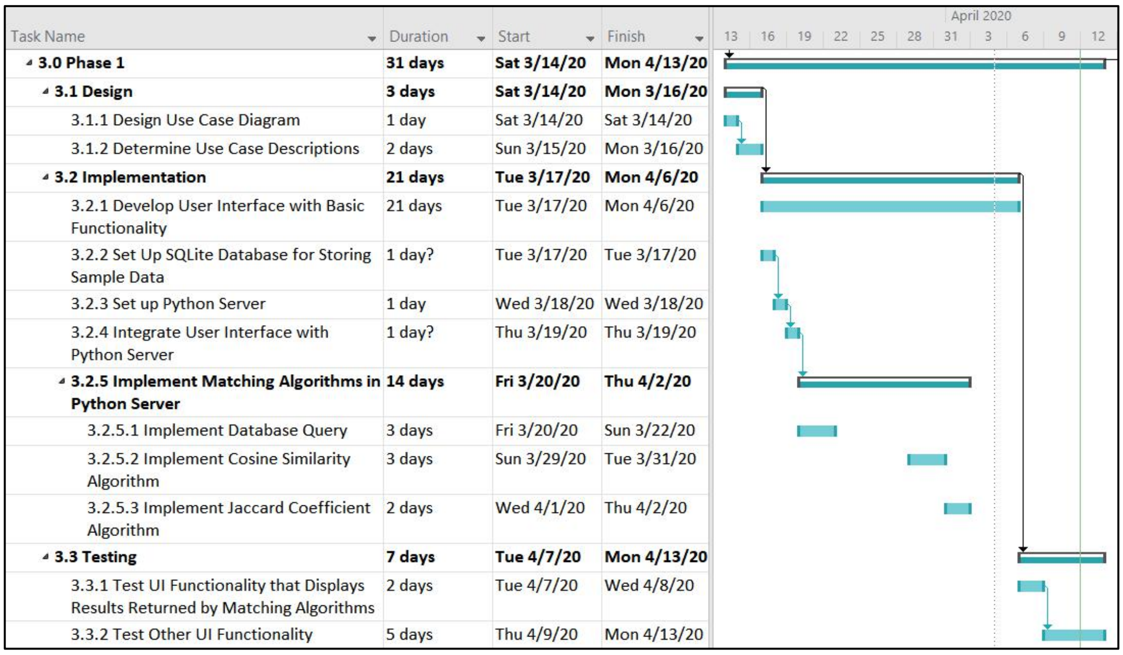
Task Name	Duration	Start	Finish
Online Smart Matrimonial Application	**217 days?**	**Mon 1/20/20**	**Sun 8/23/20**
1.0 Planning	**32 days**	**Mon 1/20/20**	**Thu 2/20/20**
1.1 Develop Work Plan	**7 days**	**Mon 1/20/20**	**Sun 1/26/20**
1.1.1 Determine Project Milestone	2 days	Mon 1/20/20	Tue 1/21/20
1.1.2 Develop Project's WBS	3 days	Wed 1/22/20	Fri 1/24/20
1.1.3 Develop Project's Gantt Chart	2 days	Sat 1/25/20	Sun 1/26/20
1.2 Study Project Background	5 days	Mon 1/27/20	Fri 1/31/20
1.3 Determine Problem Statement	5 days	Sat 2/1/20	Wed 2/5/20
1.4 Determine Porject's Objective	1 day?	Thu 2/6/20	Thu 2/6/20
1.5 Determine Project's Scope	**2 days**	**Fri 2/7/20**	**Sat 2/8/20**
1.5.1 Identify Target User	1 day	Fri 2/7/20	Fri 2/7/20
1.5.2 Identify Scope Covered	2 days	Fri 2/7/20	Sat 2/8/20
1.6 Determine Project's Proposed Solution	4 days	Sun 2/9/20	Wed 2/12/20
1.7 Determine Project's Development Methodology	**8 days**	**Thu 2/13/20**	**Thu 2/20/20**
1.7.1 Research on Waterfall Methodology	2 days	Thu 2/13/20	Fri 2/14/20
1.7.2 Research on Prototyping Methodology	2 days	Sat 2/15/20	Sun 2/16/20
1.7.3 Research on Phased Development Methodology	2 days	Mon 2/17/20	Tue 2/18/20
1.7.4 Research on Extreme Programming Methodology	2 days	Wed 2/19/20	Thu 2/20/20

Figure 3.8 Gantt Chart of Planning Phase



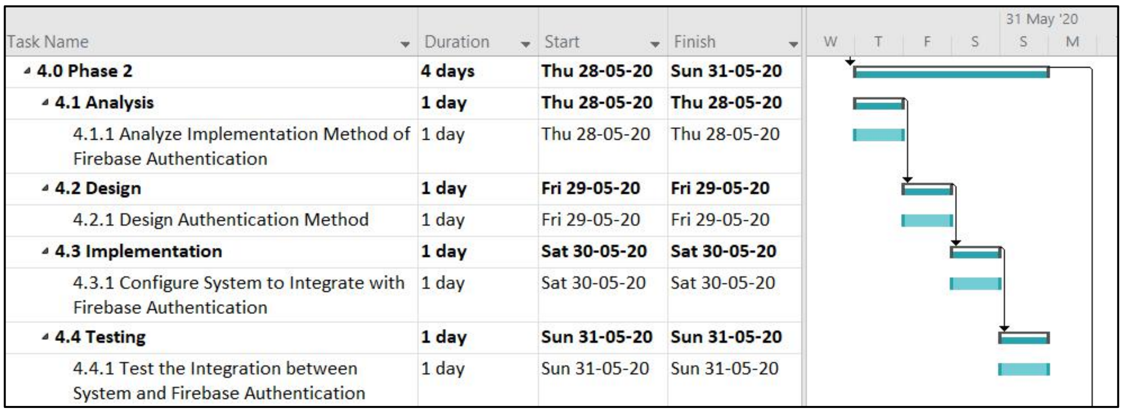
Task Name	Duration	Start	Finish
2.0 Analysis	**22 days**	**Fri 2/21/20**	**Fri 3/13/20**
2.1 Review on Existing Similar Systems	**8 days**	**Fri 2/21/20**	**Fri 2/28/20**
2.1.1 Review on MalaysianCupid.com	2 days	Fri 2/21/20	Sat 2/22/20
2.1.2 Review on Match.com	2 days	Sun 2/23/20	Mon 2/24/20
2.1.3 Review on OkCupid.com	2 days	Tue 2/25/20	Wed 2/26/20
2.1.4 Review on Shaadi.com	2 days	Thu 2/27/20	Fri 2/28/20
2.1.5 Review on Badoo	1 day	Thu 2/27/20	Thu 2/27/20
2.2 Conduct Survey	**14 days**	**Fri 2/21/20**	**Thu 3/5/20**
2.2.1 Generate Questionnaire	2 days	Fri 2/21/20	Sat 2/22/20
2.2.2 Distribute Questionnaire	7 days	Sun 2/23/20	Sat 2/29/20
2.2.3 Analysis of Findings	2 days	Wed 3/4/20	Thu 3/5/20
2.3 Perform Literature Review	**10 days**	**Sat 2/29/20**	**Mon 3/9/20**
2.3.1 Review on Matching Algorithms	**10 days**	**Sat 2/29/20**	**Mon 3/9/20**
2.3.1.1 Review on Rule-based Approach	3 days	Sat 2/29/20	Mon 3/2/20
2.3.1.2 Review on Database Query	2 days	Tue 3/3/20	Wed 3/4/20
2.3.1.3 Review on Similarity Measures	5 days	Thu 3/5/20	Mon 3/9/20
2.3.2 Review on Usability Testings	**6 days**	**Wed 3/4/20**	**Mon 3/9/20**
2.3.2.1 Review on Lab Usability Testing	2 days	Wed 3/4/20	Thu 3/5/20
2.3.2.2 Review on Remote Usability Testing	2 days	Thu 3/5/20	Fri 3/6/20
2.3.2.3 Review on Guerrilla Usability Testing	2 days	Sun 3/8/20	Mon 3/9/20
2.4 Collect Datesets	**22 days**	**Fri 2/21/20**	**Fri 3/13/20**
2.4.1 Generate Questionniare	1 day	Fri 2/21/20	Fri 2/21/20
2.4.2 Distribute Questionnaire	15 days	Fri 2/28/20	Fri 3/13/20

Figure 3.9 Gantt Chart of Analysis Phase



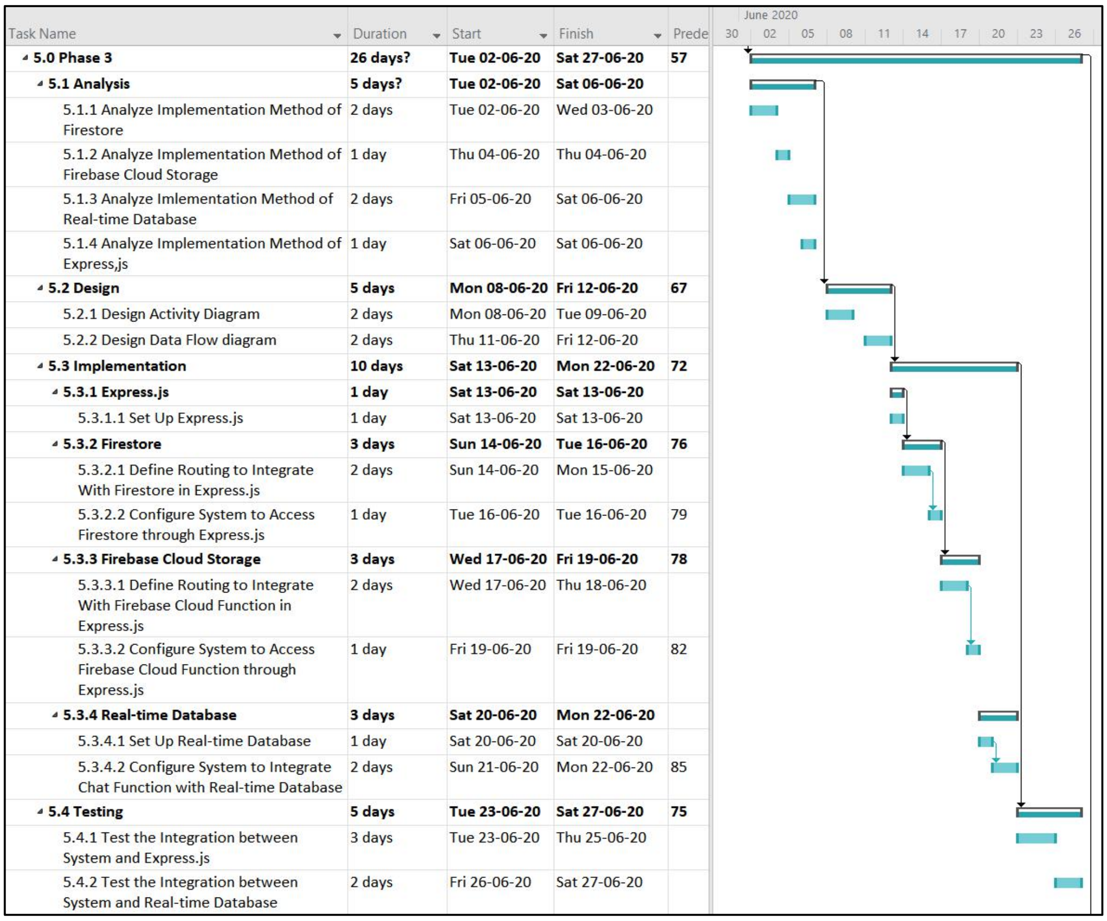
Task Name	Duration	Start	Finish
3.0 Phase 1	**31 days**	**Sat 3/14/20**	**Mon 4/13/20**
3.1 Design	**3 days**	**Sat 3/14/20**	**Mon 3/16/20**
3.1.1 Design Use Case Diagram	1 day	Sat 3/14/20	Sat 3/14/20
3.1.2 Determine Use Case Descriptions	2 days	Sun 3/15/20	Mon 3/16/20
3.2 Implementation	**21 days**	**Tue 3/17/20**	**Mon 4/6/20**
3.2.1 Develop User Interface with Basic Functionality	21 days	Tue 3/17/20	Mon 4/6/20
3.2.2 Set Up SQLite Database for Storing Sample Data	1 day?	Tue 3/17/20	Tue 3/17/20
3.2.3 Set up Python Server	1 day	Wed 3/18/20	Wed 3/18/20
3.2.4 Integrate User Interface with Python Server	1 day?	Thu 3/19/20	Thu 3/19/20
3.2.5 Implement Matching Algorithms in Python Server	**14 days**	**Fri 3/20/20**	**Thu 4/2/20**
3.2.5.1 Implement Database Query	3 days	Fri 3/20/20	Sun 3/22/20
3.2.5.2 Implement Cosine Similarity Algorithm	3 days	Sun 3/29/20	Tue 3/31/20
3.2.5.3 Implement Jaccard Coefficient Algorithm	2 days	Wed 4/1/20	Thu 4/2/20
3.3 Testing	**7 days**	**Tue 4/7/20**	**Mon 4/13/20**
3.3.1 Test UI Functionality that Displays Results Returned by Matching Algorithms	2 days	Tue 4/7/20	Wed 4/8/20
3.3.2 Test Other UI Functionality	5 days	Thu 4/9/20	Mon 4/13/20

Figure 3.10 Gantt Chart of Phase 1



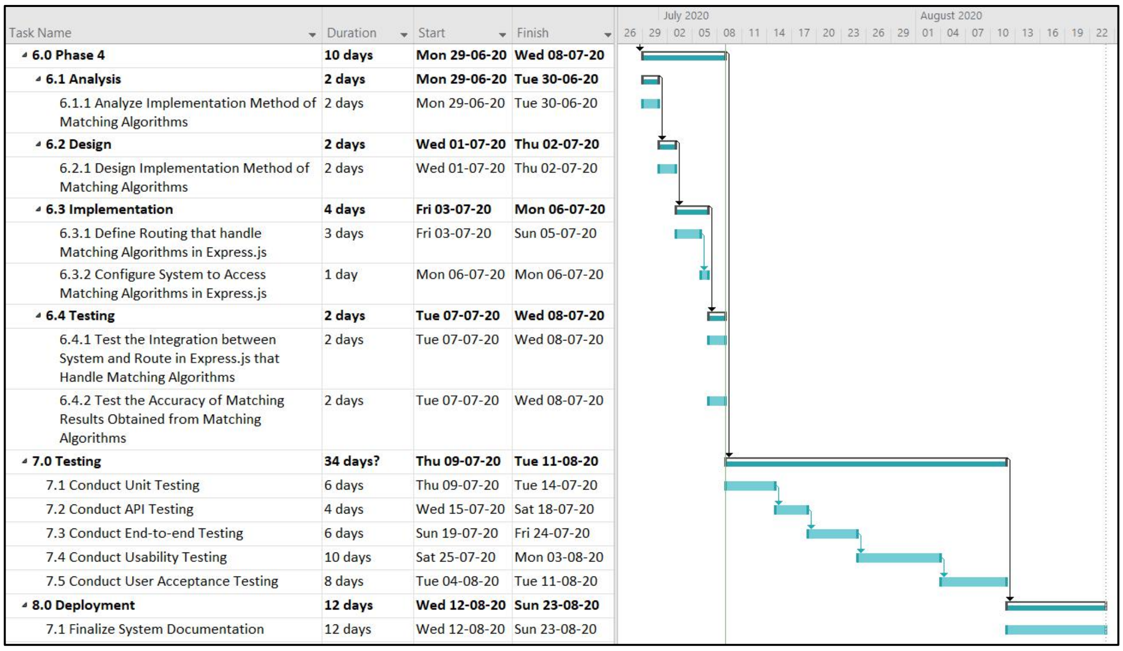
Task Name	Duration	Start	Finish
4.0 Phase 2	**4 days**	**Thu 28-05-20**	**Sun 31-05-20**
4.1 Analysis	**1 day**	**Thu 28-05-20**	**Thu 28-05-20**
4.1.1 Analyze Implementation Method of Firebase Authentication	1 day	Thu 28-05-20	Thu 28-05-20
4.2 Design	**1 day**	**Fri 29-05-20**	**Fri 29-05-20**
4.2.1 Design Authentication Method	1 day	Fri 29-05-20	Fri 29-05-20
4.3 Implementation	**1 day**	**Sat 30-05-20**	**Sat 30-05-20**
4.3.1 Configure System to Integrate with Firebase Authentication	1 day	Sat 30-05-20	Sat 30-05-20
4.4 Testing	**1 day**	**Sun 31-05-20**	**Sun 31-05-20**
4.4.1 Test the Integration between System and Firebase Authentication	1 day	Sun 31-05-20	Sun 31-05-20

Figure 3.11 Gantt Chart of Phase 2



Task Name	Duration	Start	Finish	Prede
5.0 Phase 3	**26 days?**	**Tue 02-06-20**	**Sat 27-06-20**	**57**
5.1 Analysis	**5 days?**	**Tue 02-06-20**	**Sat 06-06-20**	
5.1.1 Analyze Implementation Method of Firestore	2 days	Tue 02-06-20	Wed 03-06-20	
5.1.2 Analyze Implementation Method of Firebase Cloud Storage	1 day	Thu 04-06-20	Thu 04-06-20	
5.1.3 Analyze Imlementation Method of Real-time Database	2 days	Fri 05-06-20	Sat 06-06-20	
5.1.4 Analyze Implementation Method of Express,js	1 day	Sat 06-06-20	Sat 06-06-20	
5.2 Design	**5 days**	**Mon 08-06-20**	**Fri 12-06-20**	**67**
5.2.1 Design Activity Diagram	2 days	Mon 08-06-20	Tue 09-06-20	
5.2.2 Design Data Flow diagram	2 days	Thu 11-06-20	Fri 12-06-20	
5.3 Implementation	**10 days**	**Sat 13-06-20**	**Mon 22-06-20**	**72**
5.3.1 Express.js	**1 day**	**Sat 13-06-20**	**Sat 13-06-20**	
5.3.1.1 Set Up Express.js	1 day	Sat 13-06-20	Sat 13-06-20	
5.3.2 Firestore	**3 days**	**Sun 14-06-20**	**Tue 16-06-20**	**76**
5.3.2.1 Define Routing to Integrate With Firestore in Express.js	2 days	Sun 14-06-20	Mon 15-06-20	
5.3.2.2 Configure System to Access Firestore through Express.js	1 day	Tue 16-06-20	Tue 16-06-20	79
5.3.3 Firebase Cloud Storage	**3 days**	**Wed 17-06-20**	**Fri 19-06-20**	**78**
5.3.3.1 Define Routing to Integrate With Firebase Cloud Function in Express.js	2 days	Wed 17-06-20	Thu 18-06-20	
5.3.3.2 Configure System to Access Firebase Cloud Function through Express.js	1 day	Fri 19-06-20	Fri 19-06-20	82
5.3.4 Real-time Database	**3 days**	**Sat 20-06-20**	**Mon 22-06-20**	
5.3.4.1 Set Up Real-time Database	1 day	Sat 20-06-20	Sat 20-06-20	
5.3.4.2 Configure System to Integrate Chat Function with Real-time Database	2 days	Sun 21-06-20	Mon 22-06-20	85
5.4 Testing	**5 days**	**Tue 23-06-20**	**Sat 27-06-20**	**75**
5.4.1 Test the Integration between System and Express.js	3 days	Tue 23-06-20	Thu 25-06-20	
5.4.2 Test the Integration between System and Real-time Database	2 days	Fri 26-06-20	Sat 27-06-20	

Figure 3.12 Gantt Chart of Phase 3



Task Name	Duration	Start	Finish
6.0 Phase 4	**10 days**	**Mon 29-06-20**	**Wed 08-07-20**
6.1 Analysis	**2 days**	**Mon 29-06-20**	**Tue 30-06-20**
6.1.1 Analyze Implementation Method of Matching Algorithms	2 days	Mon 29-06-20	Tue 30-06-20
6.2 Design	**2 days**	**Wed 01-07-20**	**Thu 02-07-20**
6.2.1 Design Implementation Method of Matching Algorithms	2 days	Wed 01-07-20	Thu 02-07-20
6.3 Implementation	**4 days**	**Fri 03-07-20**	**Mon 06-07-20**
6.3.1 Define Routing that handle Matching Algorithms in Express.js	3 days	Fri 03-07-20	Sun 05-07-20
6.3.2 Configure System to Access Matching Algorithms in Express.js	1 day	Mon 06-07-20	Mon 06-07-20
6.4 Testing	**2 days**	**Tue 07-07-20**	**Wed 08-07-20**
6.4.1 Test the Integration between System and Route in Express.js that Handle Matching Algorithms	2 days	Tue 07-07-20	Wed 08-07-20
6.4.2 Test the Accuracy of Matching Results Obtained from Matching Algorithms	2 days	Tue 07-07-20	Wed 08-07-20
7.0 Testing	**34 days?**	**Thu 09-07-20**	**Tue 11-08-20**
7.1 Conduct Unit Testing	6 days	Thu 09-07-20	Tue 14-07-20
7.2 Conduct API Testing	4 days	Wed 15-07-20	Sat 18-07-20
7.3 Conduct End-to-end Testing	6 days	Sun 19-07-20	Fri 24-07-20
7.4 Conduct Usability Testing	10 days	Sat 25-07-20	Mon 03-08-20
7.5 Conduct User Acceptance Testing	8 days	Tue 04-08-20	Tue 11-08-20
8.0 Deployment	**12 days**	**Wed 12-08-20**	**Sun 23-08-20**
7.1 Finalize System Documentation	12 days	Wed 12-08-20	Sun 23-08-20

Figure 3.13 Gantt Chart of Phase 4, Testing Phase and Deployment Phase

CHAPTER 4

PROJECT INITIAL SPECIFICATION

4.1 Introduction

This chapter discussed the requirements of the proposed system. Use case diagram was introduced to demonstrate the overall flow of the application. The detailed information of the flows was defined in the use case descriptions. This chapter also covered some preliminary system design.

4.2 Functional Requirements

1. The system must enable users to sign-in with their email and password.
2. The system must enable first-time users to register for a new account.
3. The system must authenticate users after the users enter their email and password.
4. The system must allow users to select the desired search criteria before searching for potential matches.
5. The system must be able to perform similarity measures based on the search preference set by users.
6. The system must allow users to view a list of potential matches that are sorted in descending order of similarity after performing the similarity measures.
7. The system must allow users to view the details of the desired profiles such as basic information, background and lifestyle.
8. The system must allow users to add the desired profiles to the favourite list.
9. The system must allow users to send a message to the interested profiles.
10. The system must allow users to view and modify their personal profile details.

4.3 Non-functional Requirements

1. **Usability**
   a) The system shall be built in a user-friendly way by presenting a simple and consistent graphical user interface.
   b) The system shall be able to operate easily by users to achieve their objectives.

2. **Performance**
    a) The system shall respond to user requests in less than 1 second.
    b) The system should ensure the results of similarity measure are correct.
    c) The system shall be able to manage concurrent request from multiple users with a failure rate below 0.5%.
3. **Security**
    a) The system shall protect the personal information of the users and restrict unauthorized users for accessing the system.
    b) The system shall allow users to use their email address for verification if they forget their password.
4. **Development**
    a) Each user record shall be stored on a well-built and efficient database schema.
    b) The methodology adopted in this project is phased development.
5. **Operational**
    a) The system shall function well and accessible at any time as long as users are connected to the internet.

4.4 Use Case Diagram

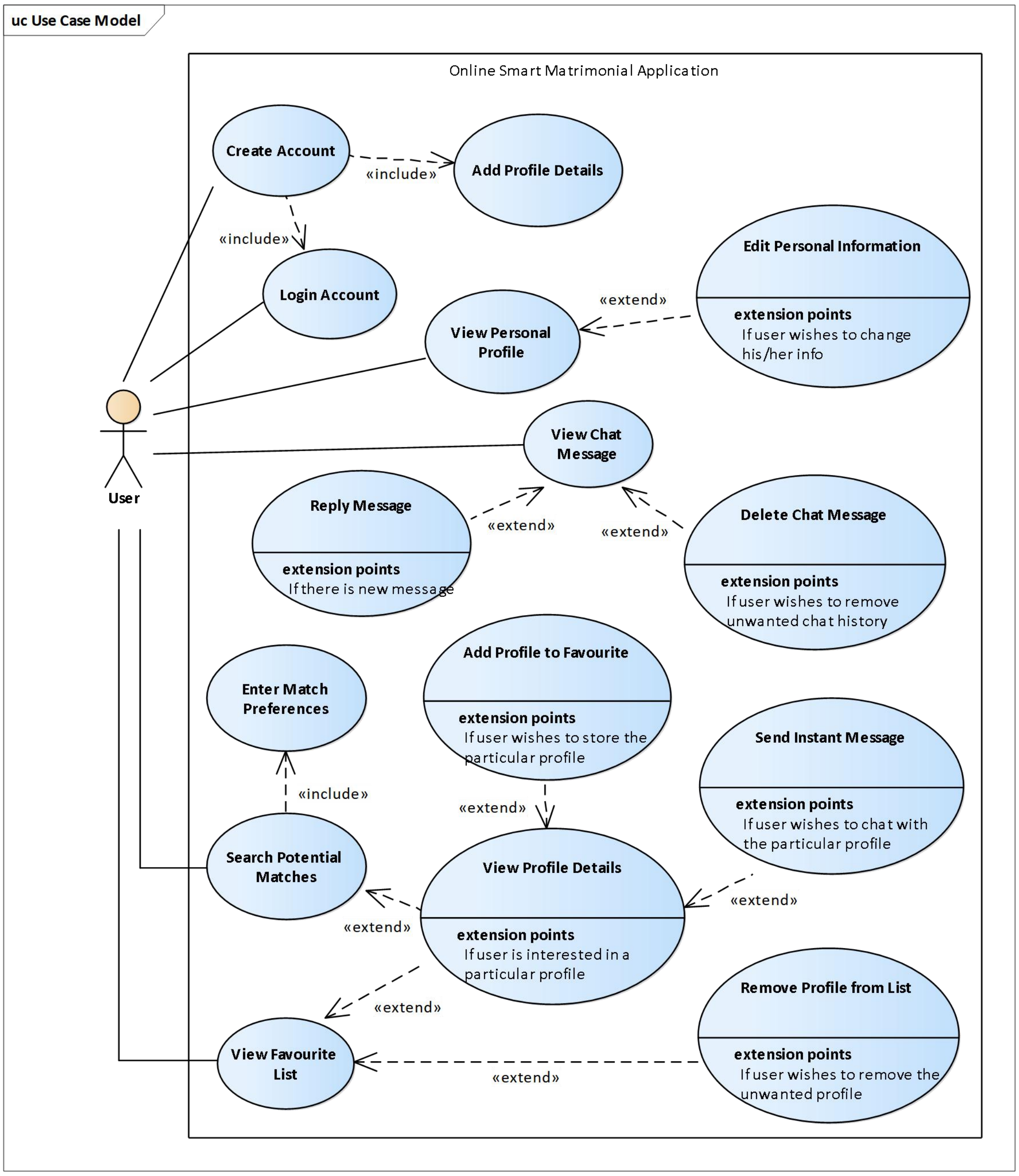


Figure 4.1 Use Case Diagram

4.5 Use Case Description

Table 4.1 Use Case - Create Account

Use Case: Create Account	ID: 1
Stakeholder: User – a person who is new to the system and want to use the system for finding the potential life partner.	
Description: Use case that describes how a user creates an account.	
Triggering event: User wants to access the system.	
Related Use Cases: Association: User Include: Login Account, Add Profile Details	
Basic Paths: 1. User will enter an interface to create an account. 2. User enters his/her email and password. 3. An account is successfully created and the system redirects the user to a page that prompt user input for his/her profile details. 4. User enters his/her personal details, including demographic info, background and lifestyle. 5. User uploads his/her profile picture. 6. The system stores user's profile details and photo into the database and redirect the user to the home page of the system.	
Exceptional Paths: 2.1 User re-enters the email if the email is being used	

Table 4.2 Use Case - Login Account

Use Case: Login Account	ID: 2
Stakeholder: User – a person who wants to access the system and already has an account.	
Description: Use case that describes how a user sign-in into the system.	
Triggering event: User who wants to login the system.	
Related Use Cases: Association: User Extend: Add Profile Details	
Basic Paths: 1. User will enter an interface to login. 2. User enters his/her email and password. 3. The system verifies the user's identity. 4. The system redirects the user to the home page of the system	
Exceptional Paths: 2.1 User enters invalid email or incorrect password 2.1.1 The system prompts the user to input the required information again.	

Table 4.3 Use Case - Search Potential Matches

Use Case: Search Potential Matches	ID: 3
Actor's Information: User – a person who wants to find the potential life partner.	
Description: Use case that describes how a user searches for the potential matches by inputting some match preferences.	
Triggering event: User wants to find for potential matches.	
Related Use Cases: Association: User Include: Enter Match Preferences Extend: View Profile Details	
Basic Paths: 1. User selects his/her desired match preferences. 2. The system processes the match preferences and redirect the user to the page that consists of a list of matched profiles. 3. User can edit his/her desired match preferences if not satisfies with the results. 4. The system displays a new matched profile list to the user.	
Exception Paths: 4.1 User clicks on one of the profiles that he/she interested in. 4.1.1 The system redirects the user to the page that consists of the particular profile's details (refer to use case ID: 4).	

Table 4.4 Use Case - View Profile Details

Use Case: View Profile Details	ID: 4
Stakeholder: User – a person who wants to view a particular profile.	
Description: Use case that describes how a user view the personal information of a profile.	
Triggering event: User wants to view an interested profile.	
Related Use Cases: Association: User Extend: Send Message & Add to Favourite	
Basic Paths: 1. The system displays the detailed information of the profile, including demographic info, background and lifestyle. 2. User views the details. 2.1 If the user wishes to start a conversation with the particular profile, sub-path 2.1 will be performed. 2.2 If the user wishes to store a particular profile, sub-path 2.2 will be performed.	
Sub-paths: 2.1 User click on the "Chat" icon. 2.1.1 The system prompts the user for the message he/she wants to send. 2.1.2 The system sends the user's message to the particular profile. 2.2 User click on the "Favourite" icon. 2.2.1 The system saves the particular profile into the user's favourite list.	

Table 4.5 Use Case - View Favourite List

Use Case: View Favourite List	ID: 5
Actor's Information: User – a person who wants to access the profiles that he/she added into the favourite list.	
Description: Use case that describes how a user view the profiles that have been added into the favourite list.	
Triggering event: User wants to view the favourite list.	
Related Use Cases: Association: User Extend: View Profile Details & Remove Profile from List	
Basic Paths: 1. User clicks on the "Favourite List" option on the navigation bar. 2. The system directs the user to the pages that consists of a list of profiles that added to favourite. 3. User views the list of profiles. 3.1 If the user wishes to view the profile details, sub-path 3.1 will be performed. 3.2 if the user wishes to remove unwanted profiles, sub-path 3.2 will be performed.	
Sub-paths: 3.1 User clicks on the desired profile. 3.1.1 The system redirects the user to the page that consists of the particular profile's details (refer to use case ID: 4). 3.2 User clicks on the "Trash" icon on the profile. 3.2.1 The system removes the profile from the favourite list.	

Table 4.6 Use Case - View Chat Message

Use Case: View Chat Message	ID: 6
Actor's Information: User – a person who wants to view the inbox message.	
Description: Use case that describes how a user accesses the chat message.	
Triggering event: User wants to view the chat message.	
Related Use Cases: Association: User Extend: Send Message & Reply Message	
Basic Paths: 1. User clicks on the "Message" on the navigation bar. 2. The system redirects the user to the pages that consists of a chat list. 3. User clicks on one of the chats. 4. The system displays the chat history between the user and the particular profile. 5. User views the conversation with a particular profile. 5.1 If there is a new message, sub-path5.1 will be performed. 5.2 If the user wants to delete the whole chat history, sub-path 5.2 will be performed.	
Sub-paths: 5.1 User enters the message he/she wants to reply and click "Send" icon. 5.1.1 The system sends the reply message to the particular profile. 5.2 User clicks on the "Trash" icon. 5.2.1 The system clears the chat history and remove the conversation from the chat list.	

Table 4.7 Use Case - View Personal Profile

Use Case: View Personal Profile	ID: 7
Actor's Information: User – a person who wants to view his/her own profile information.	
Description: Use case that describes how a user views the personal profile information.	
Triggering event: User wishes to view his/her personal profile information.	
Related Use Cases: Association: User Extend: Edit Profile Information	
Basic Paths: 1. User clicks on the "My Profile" option on the navigation bar. 2. The system redirects the user to the page that shows his/her personal details, including demographic info, background and lifestyle. 3. User view for his/her personal information 3.1 If the user wishes to modify his/her personal info, sub-path 3.1 will be performed.	
Sub-paths: 3.1 User clicks on the "Edit" icon. 3.1.1 The system redirects the user to the page that showing a list of user's personal information. 3.1.2 User edits one of the details. 3.1.3 The system updates the new details to the database and redirects the user back to the personal profile page.	

4.6 Fact Findings

An online survey had been conducted with 25 respondents to investigate user preferences during the selection of a life partner. In the questionnaire, there are a total of 23 criteria for the respondents to select. The respondents were requested to select 10 partner criteria that they will consider when selecting the life partner.

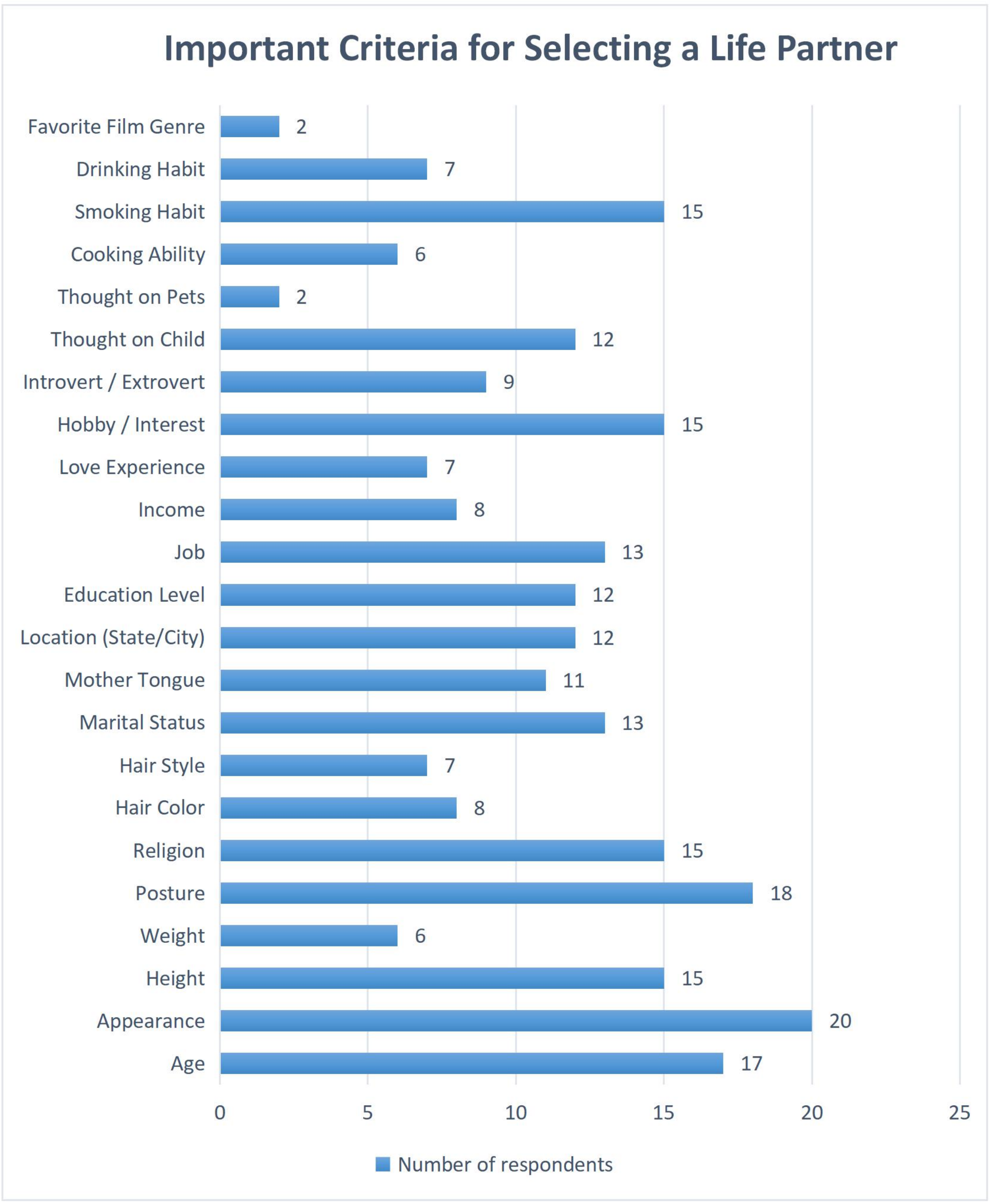


Figure 4.2 Important Criteria for Selecting a Life Partner

Based on Figure 4.2, about 80% of the respondents thought that appearance is the most important criteria in selecting the life partner. Hence, appearance criteria will be implemented in the application by displaying the user's profile picture. Besides, except the appearance, the top 12 match preferences selected by most of the respondents will be selected to be implemented in the application as user profile information. According to Figure 4.2, age, height, posture, religion, marital status, mother tongue, location, education level, job, hobby/interest, thought on child and smoking habit was selected by most of the respondents. Hence, when users sign up in the application, they will be prompted to enter their information for creating the user profiles.

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture Design

Three-tier architecture (client-server architecture) is implemented in this project. A standard three-tier architecture consists of 3 layers, which are presentation tier, application tier and data tier. According to Chen, et al. (2003), the presentation tier is the graphical user interface that manages data input and output from end-user. The middle tier is usually responsible for business logic, such as data queries, and transaction. The data tier consists of the database server that is responsible for storing and retrieving information needed for the application.

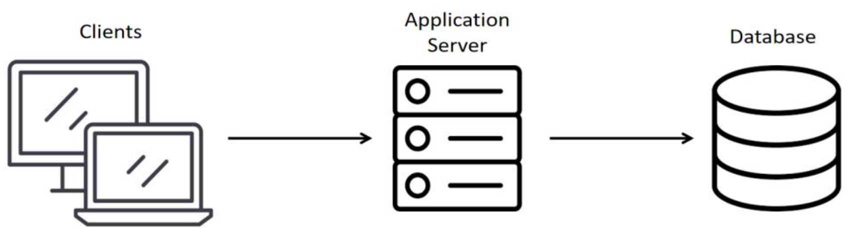


Figure 5.1 Three-tier Architecture

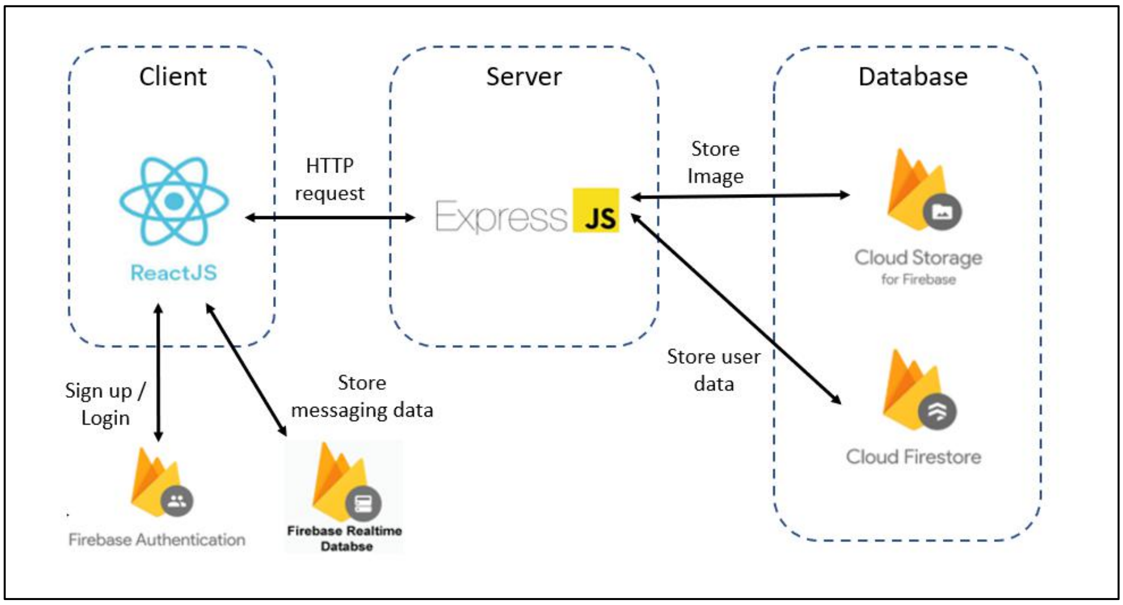


Figure 5.2 System Architecture Design

In this project, ReactJs was used to develop the presentation tier of the application. React is a simple and easy to learn library that can help in collecting user's input and displaying the data received from the application tier (server). React can communicate with the server by sending the HTTP request.

Express.js was implemented as the application logic layer. It was responsible for the application's core functionality, such as data processing for similarity measures. By implementing the application logic layer, the technical details will be hiding from the end-users. Express.js was designed to integrate with Firebase and handle all HTTP request received from the web-based application.

Firebase provides some services that can be used to develop the data tier. For example, Firestore can be used to store user data while Firebase Cloud Storage can be used to store image files. Hence, if the server receives HTTP requests from the presentation tier, it can process the request by storing/retrieving data from Firestore and Firebase Cloud Storage.

Besides, ReactJs was designed to integrate with Firebase Authentication for the authentication mechanism. Firebase Authentication was responsible for handling users' sign in or sign up event. ReactJs was also designed for directly accessing the Firebase Real-time Database. Firebase Real-time Database was not implemented in server-side due to its data synchronization. By integrating with ReactJs, the client was able to receive the real-time update within milliseconds as Firebase Real-time Database will listen for data changes.

5.1.1 Data Flow Diagram

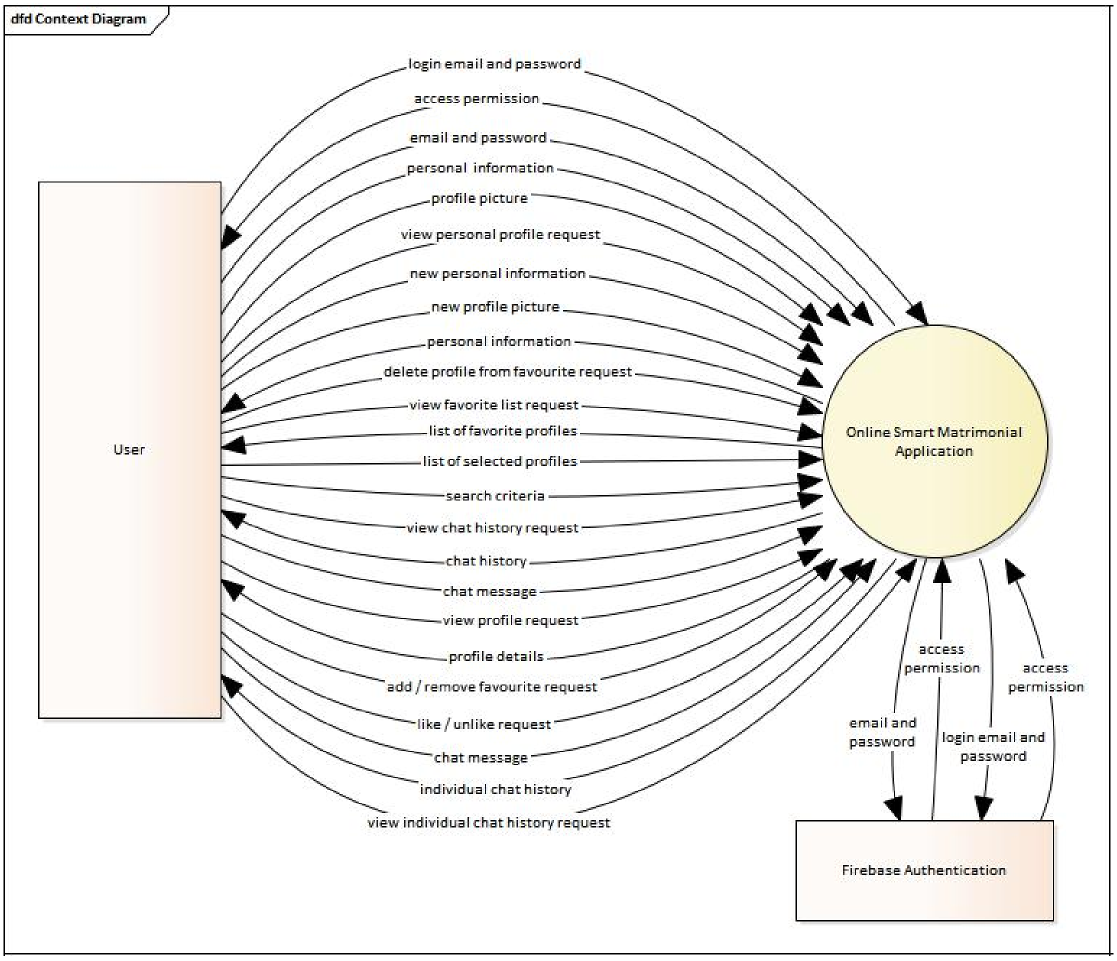


Figure 5.3 Context Diagram

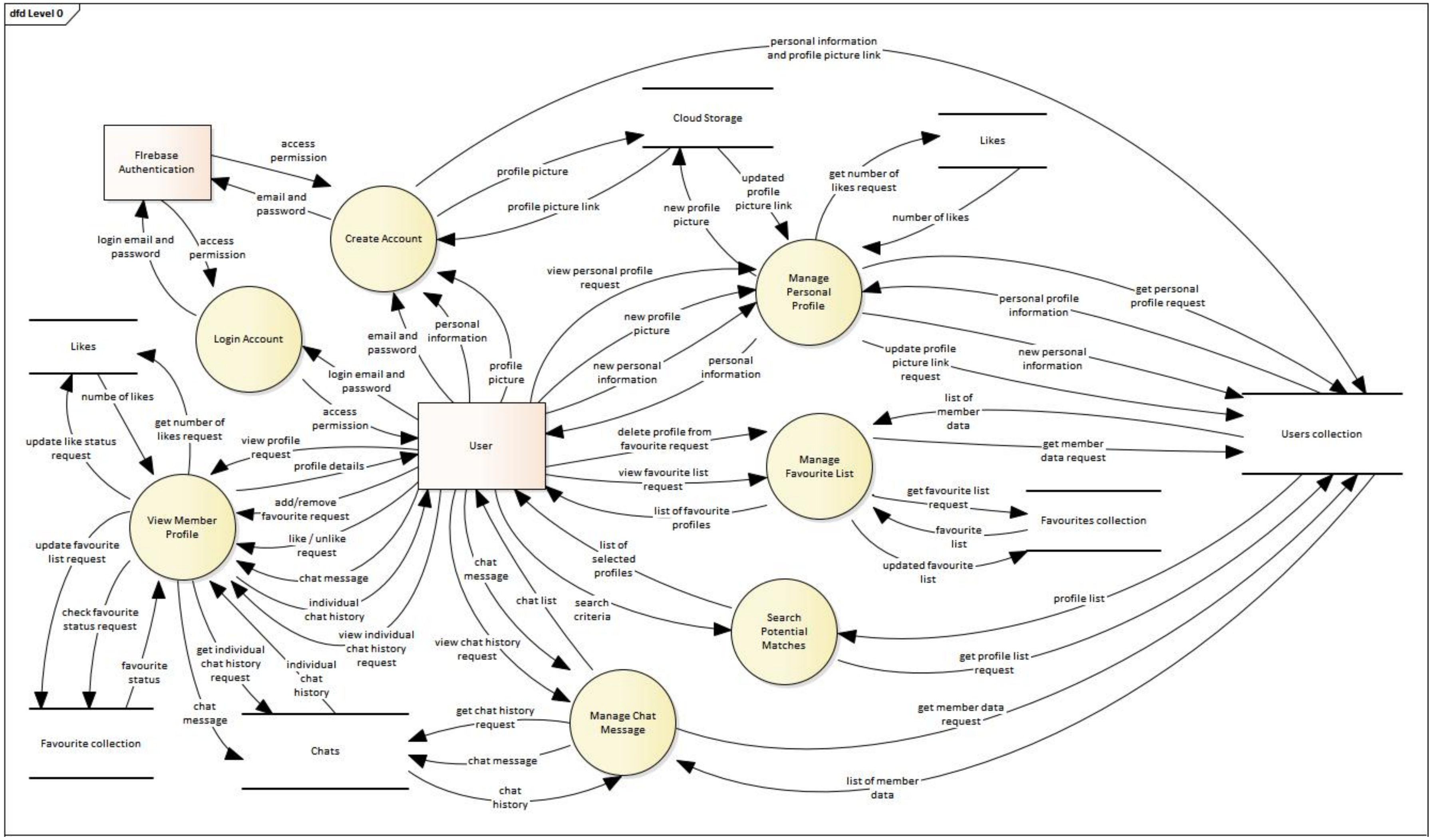


Figure 5.4 Level 0 DFD

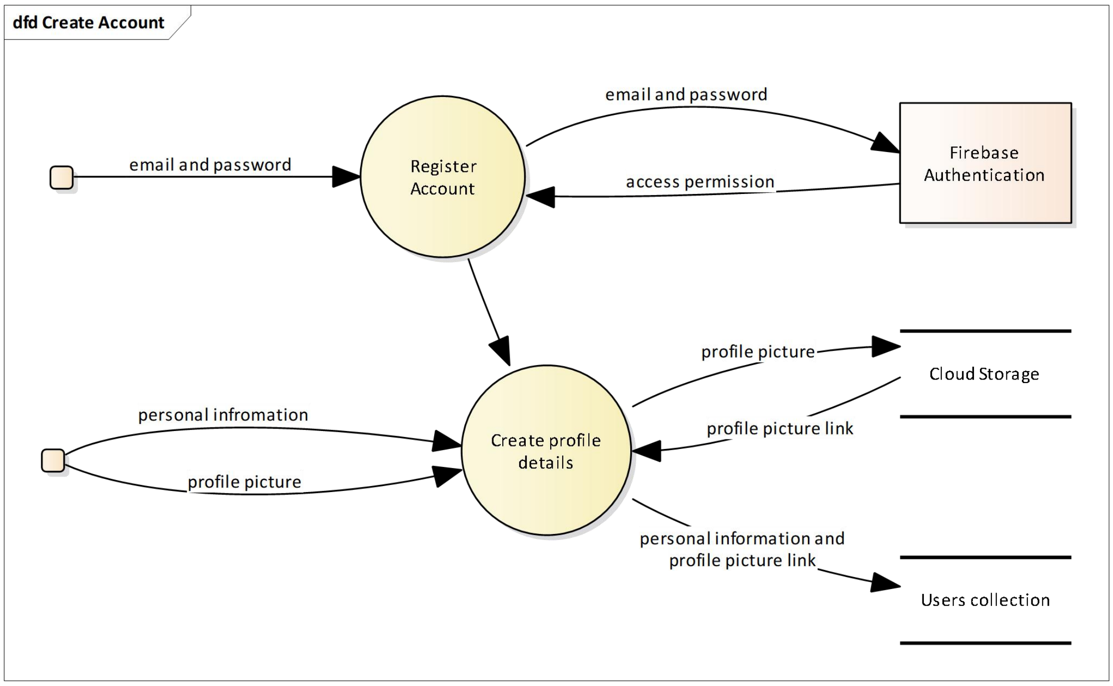


Figure 5.5 Level 1 DFD for Create Account

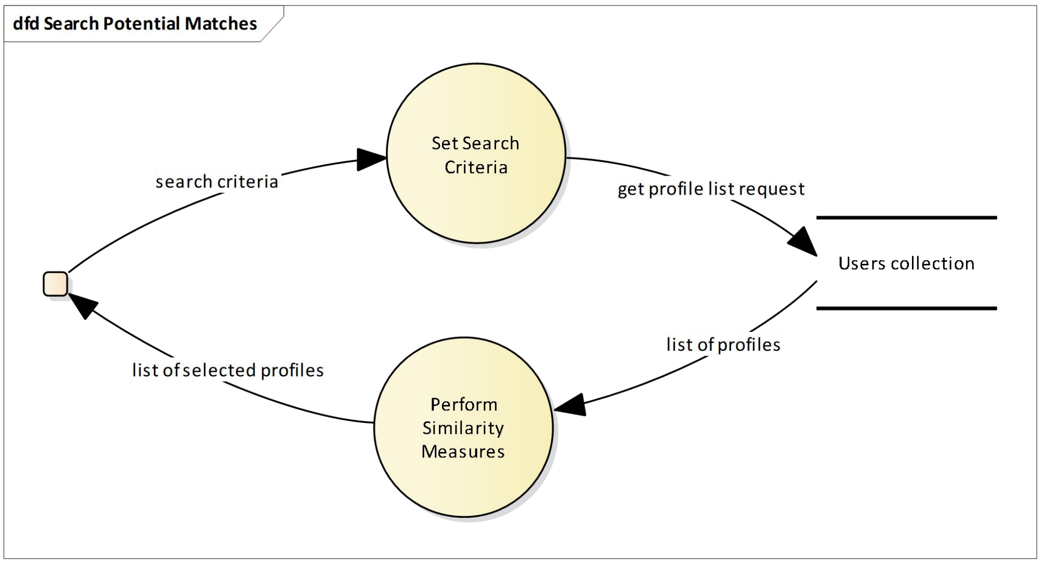


Figure 5.6 Level 1 DFD for Search Potential Matches

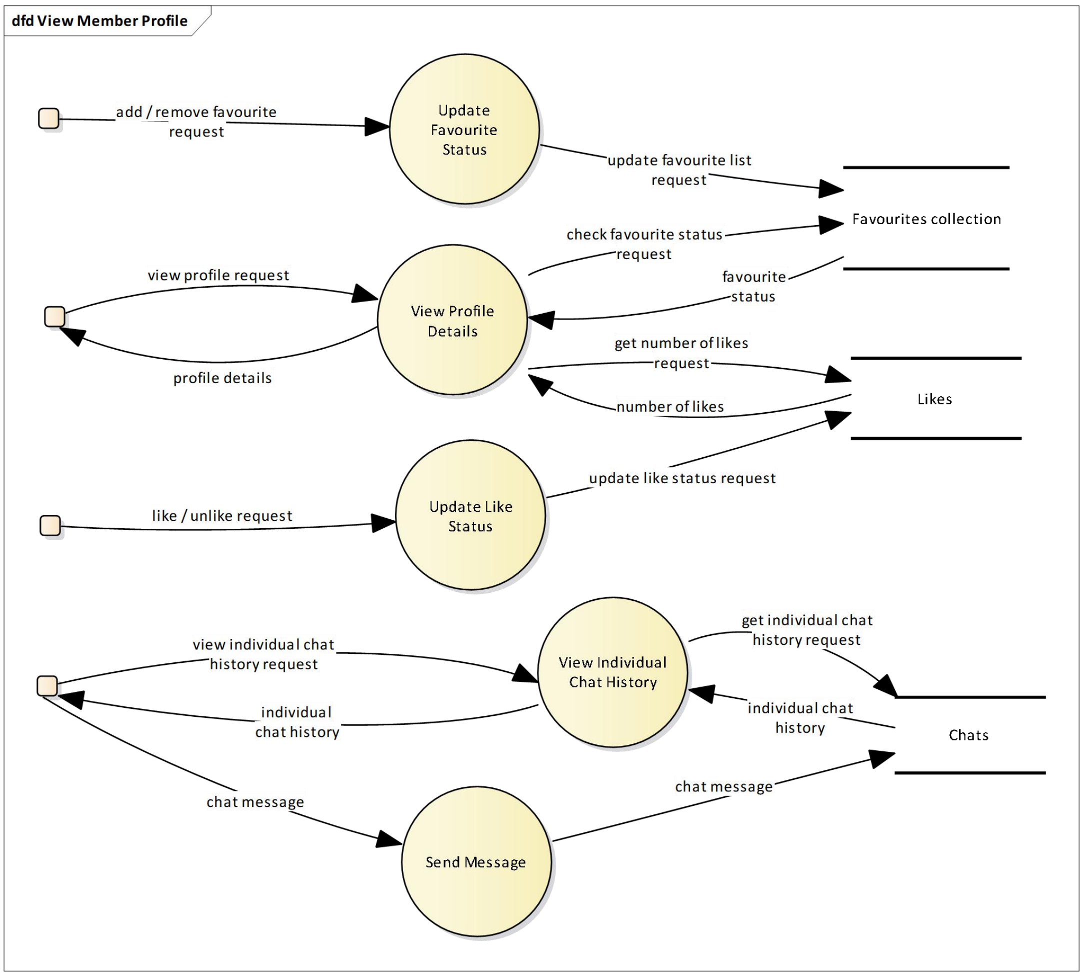


Figure 5.7 Level 1 DFD for View Member Profile

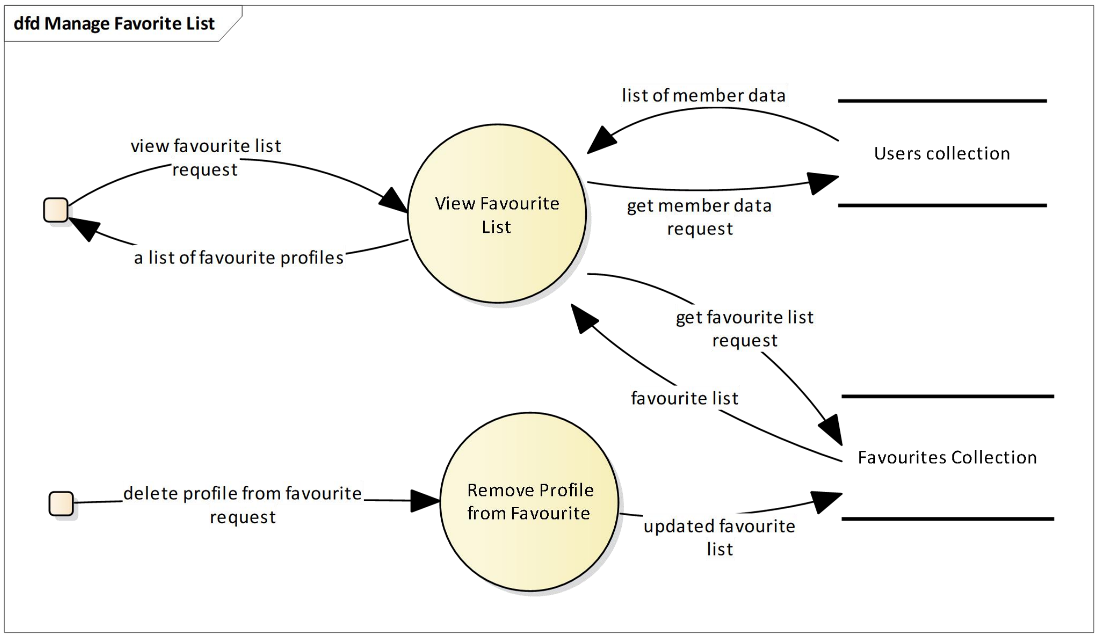


Figure 5.8 Level 1 DFD for Manage Favourite List

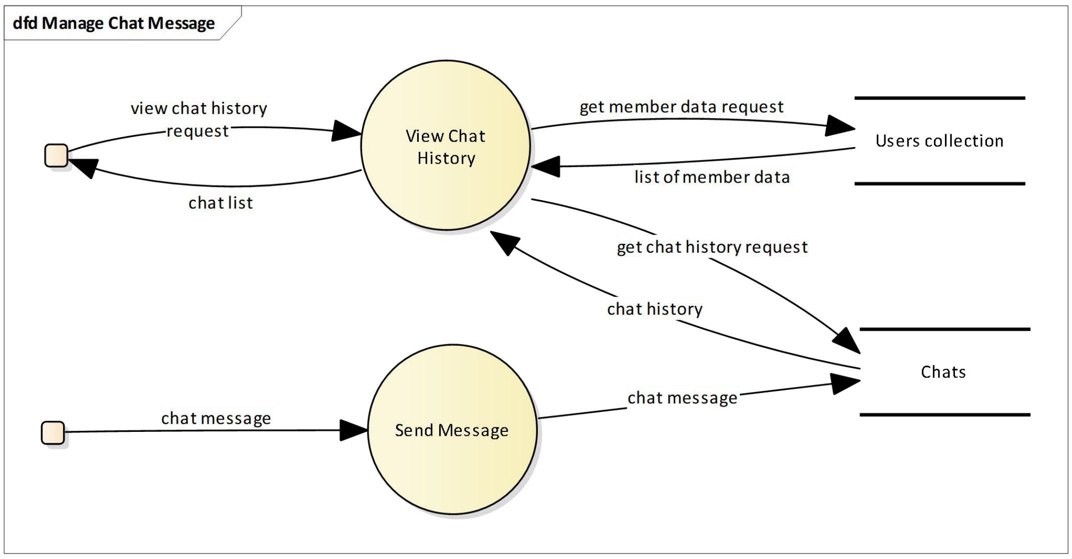


Figure 5.9 Level 1 DFD for Manage Chat Message

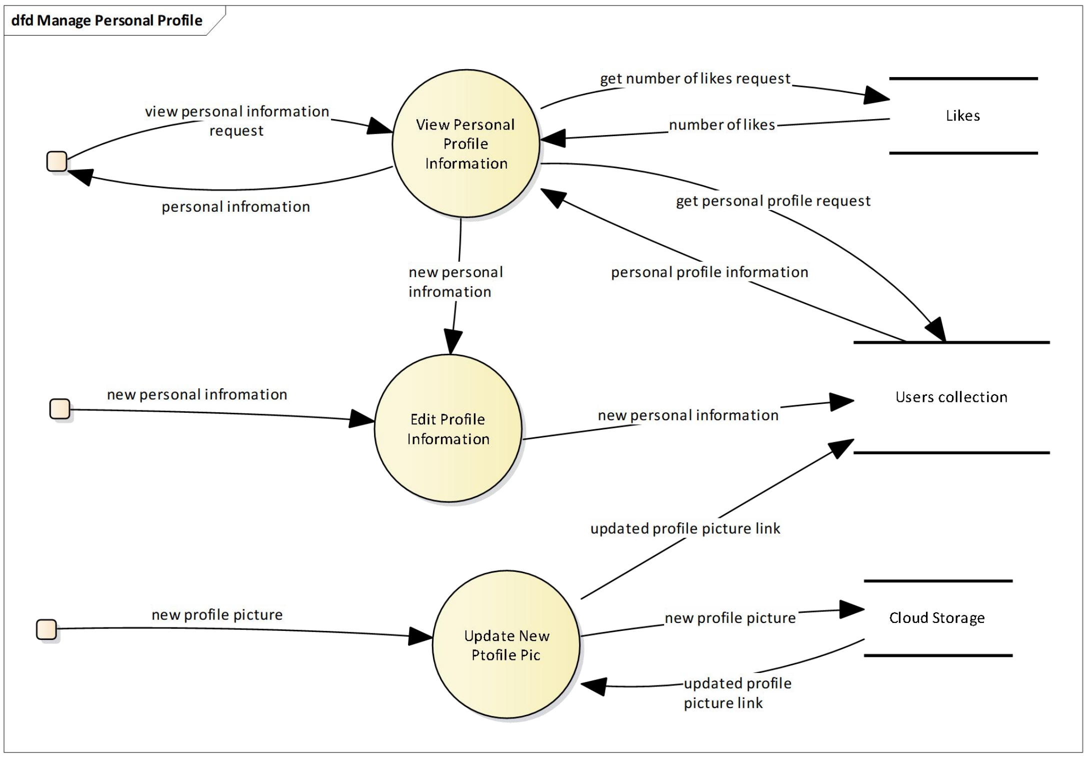

dfd Manage Personal Profile
view personal information request
personal infromation
View Personal Profile Information
get number of likes request
number of likes
Likes
get personal profile request
personal profile information
new personal infromation
new personal infromation
Edit Profile Information
new personal information
Users collection
updated profile picture link
new profile picture
Update New Ptofile Pic
new profile picture
Cloud Storage
updated profile picture link

Figure 5.10 Level 1 DFD for Manage Personal Profile

5.1.2 Activity Diagram

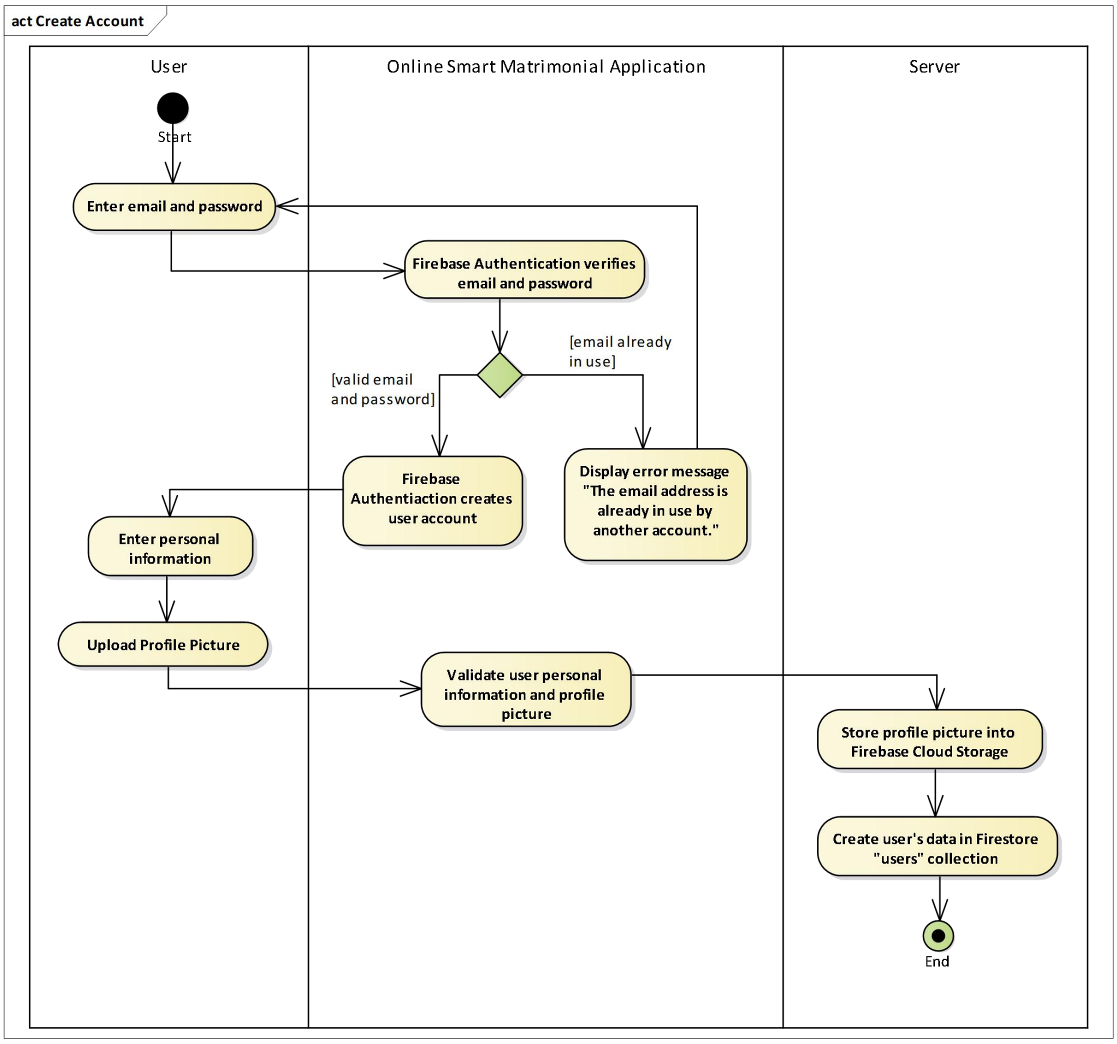

act Create Account
User
Online Smart Matrimonial Application
Server
Start
Enter email and password
Firebase Authentication verifies email and password
[valid email and password]
[email already in use]
Firebase Authentiaction creates user account
Display error message "The email address is already in use by another account."
Enter personal information
Upload Profile Picture
Validate user personal information and profile picture
Store profile picture into Firebase Cloud Storage
Create user's data in Firestore "users" collection
End

Figure 5.11 “Create Account” Activity Diagram

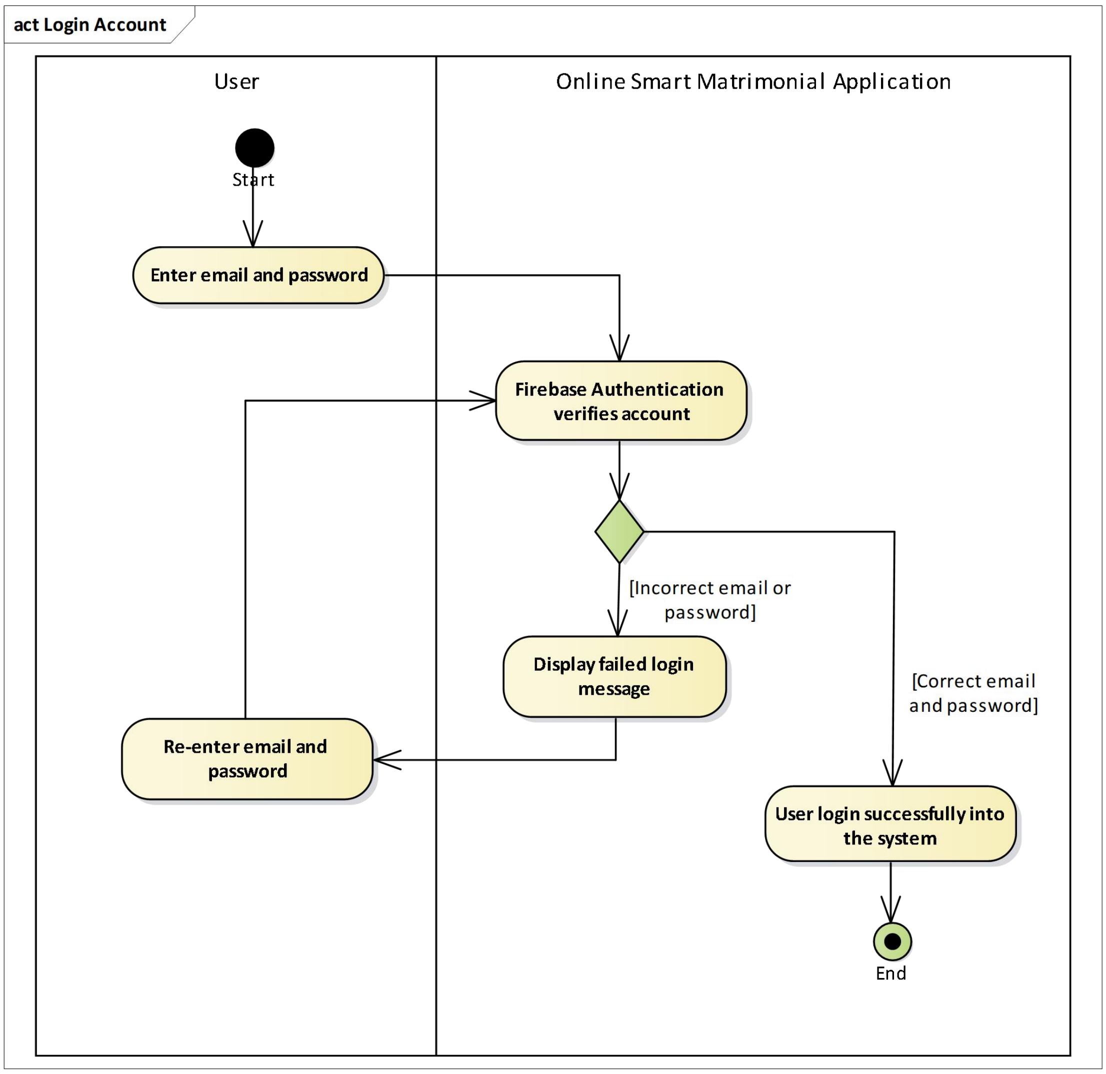

act Login Account
User
Online Smart Matrimonial Application
Start
Enter email and password
Firebase Authentication verifies account
[Incorrect email or password]
Display failed login message
Re-enter email and password
[Correct email and password]
User login successfully into the system
End

Figure 5.12 “Login Account” Activity Diagram

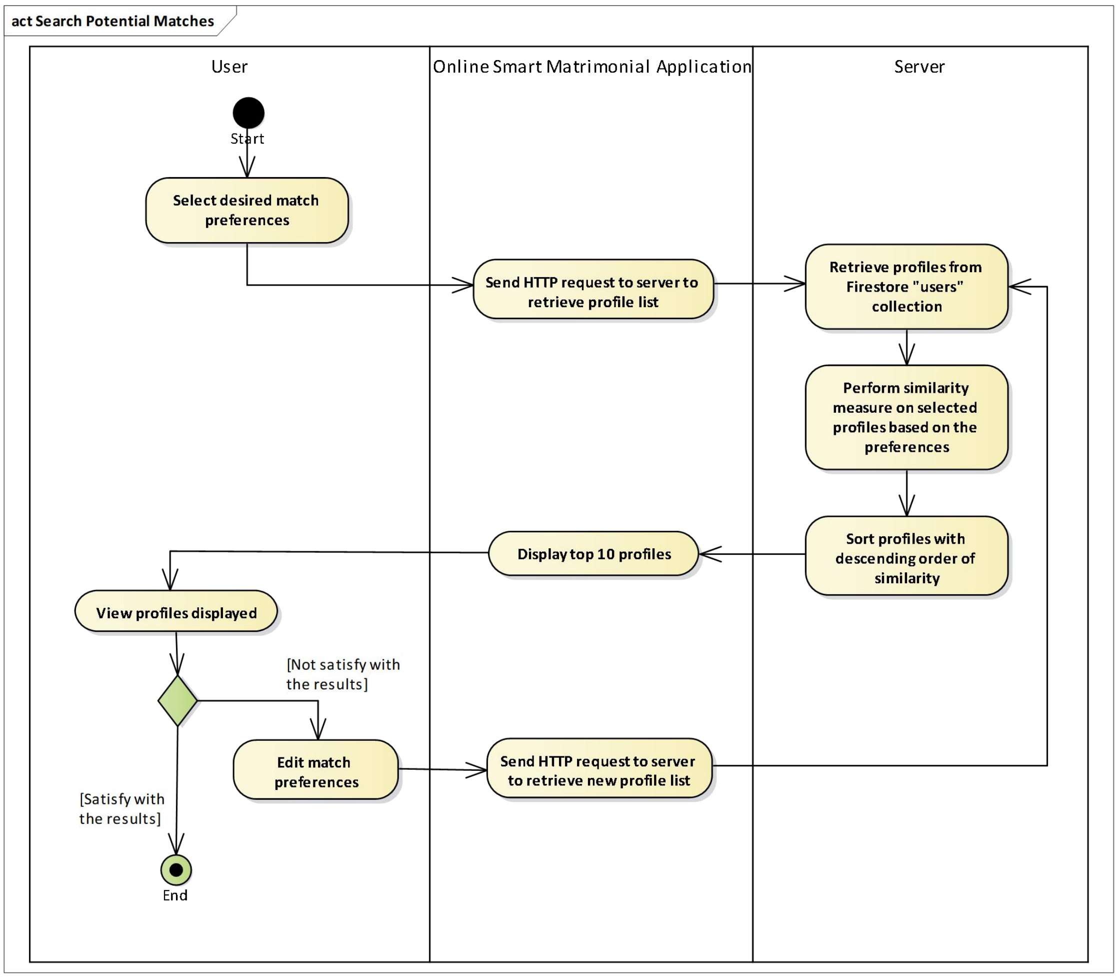

act Search Potential Matches
User
Online Smart Matrimonial Application
Server
Start
Select desired match preferences
Send HTTP request to server to retrieve profile list
Retrieve profiles from Firestore "users" collection
Perform similarity measure on selected profiles based on the preferences
Sort profiles with descending order of similarity
Display top 10 profiles
View profiles displayed
[Not satisfy with the results]
Edit match preferences
Send HTTP request to server to retrieve new profile list
[Satisfy with the results]
End

Figure 5.13 “Search Potential Matches” Activity Diagram

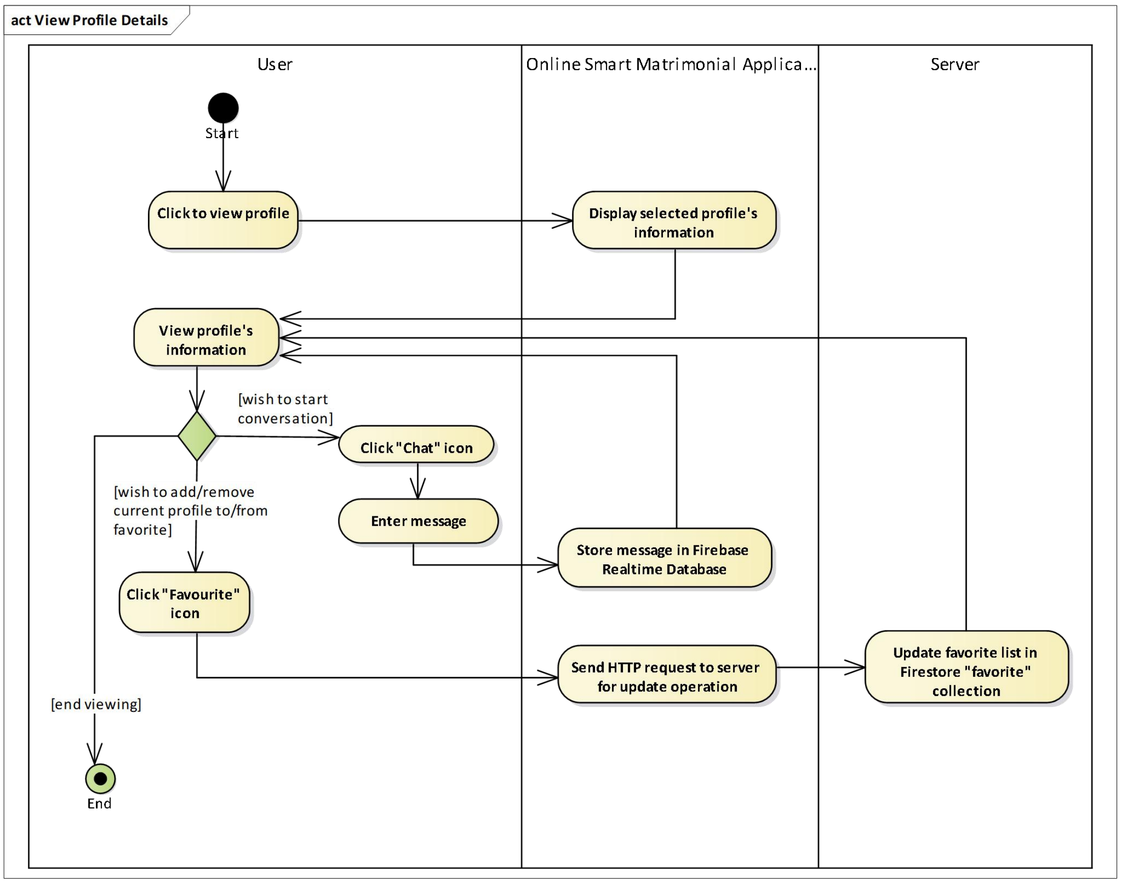

act View Profile Details
User
Online Smart Matrimonial Applica...
Server
Start
Click to view profile
Display selected profile's information
View profile's information
[wish to start conversation]
Click "Chat" icon
Enter message
Store message in Firebase Realtime Database
[wish to add/remove current profile to/from favorite]
Click "Favourite" icon
Send HTTP request to server for update operation
Update favorite list in Firestore "favorite" collection
[end viewing]
End

Figure 5.14 “View Profile Details” Activity Diagram

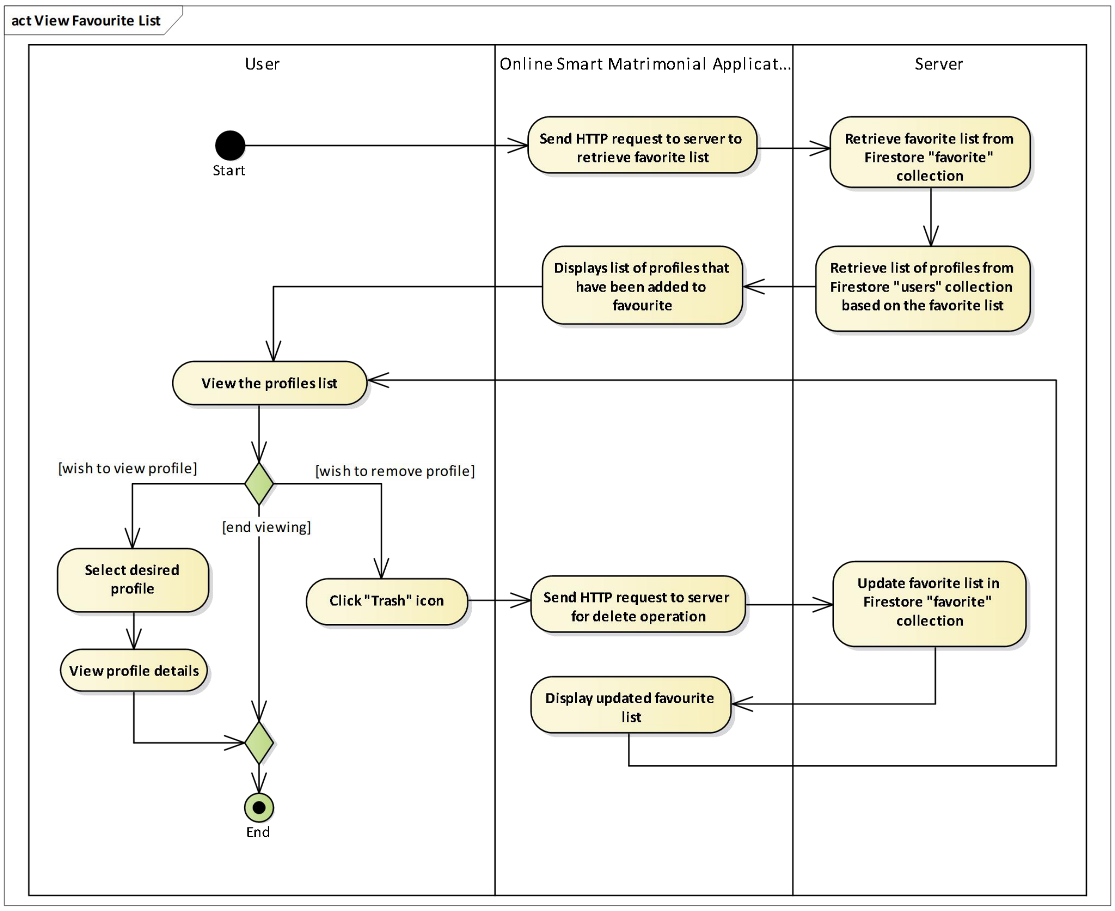

act View Favourite List
User
Online Smart Matrimonial Applicat...
Server
Start
Send HTTP request to server to retrieve favorite list
Retrieve favorite list from Firestore "favorite" collection
Retrieve list of profiles from Firestore "users" collection based on the favorite list
Displays list of profiles that have been added to favourite
View the profiles list
[wish to view profile]
[wish to remove profile]
[end viewing]
Select desired profile
View profile details
Click "Trash" icon
Send HTTP request to server for delete operation
Update favorite list in Firestore "favorite" collection
Display updated favourite list
End

Figure 5.15 “View Favourite List” Activity Diagram

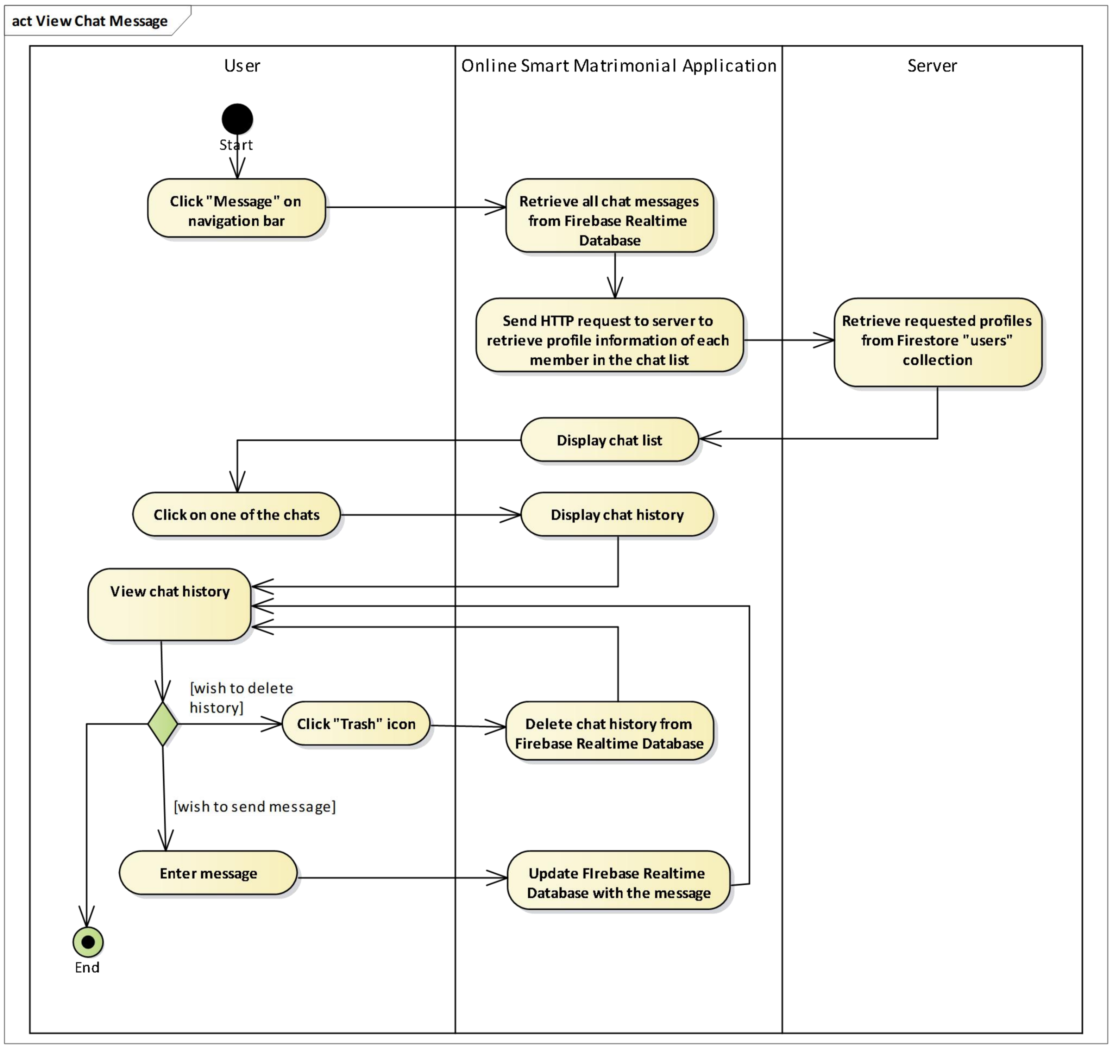

act View Chat Message
User
Online Smart Matrimonial Application
Server
Start
Click "Message" on navigation bar
Retrieve all chat messages from Firebase Realtime Database
Send HTTP request to server to retrieve profile information of each member in the chat list
Retrieve requested profiles from Firestore "users" collection
Display chat list
Click on one of the chats
Display chat history
View chat history
[wish to delete history]
Click "Trash" icon
Delete chat history from Firebase Realtime Database
[wish to send message]
Enter message
Update FIrebase Realtime Database with the message
End

Figure 5.16 “View Chat Message” Activity Diagram

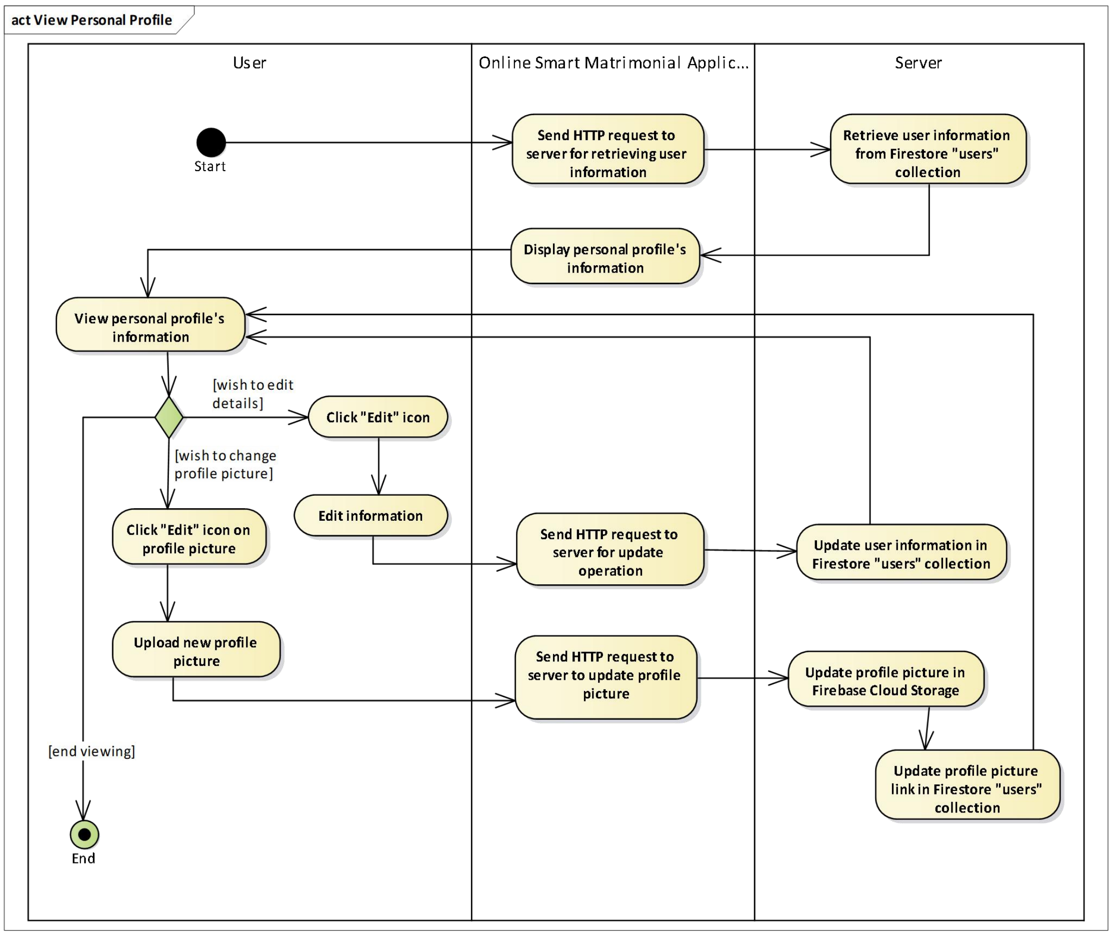

act View Personal Profile
User
Online Smart Matrimonial Applic...
Server
Start
Send HTTP request to server for retrieving user information
Retrieve user information from Firestore "users" collection
Display personal profile's information
View personal profile's information
[wish to edit details]
Click "Edit" icon
[wish to change profile picture]
Edit information
Click "Edit" icon on profile picture
Send HTTP request to server for update operation
Update user information in Firestore "users" collection
Upload new profile picture
Send HTTP request to server to update profile picture
Update profile picture in Firebase Cloud Storage
[end viewing]
Update profile picture link in Firestore "users" collection
End

Figure 5.17 “View Personal Profile” Activity Diagram

5.2 Data Model Diagram

5.2.1 Logical Data Model Diagram

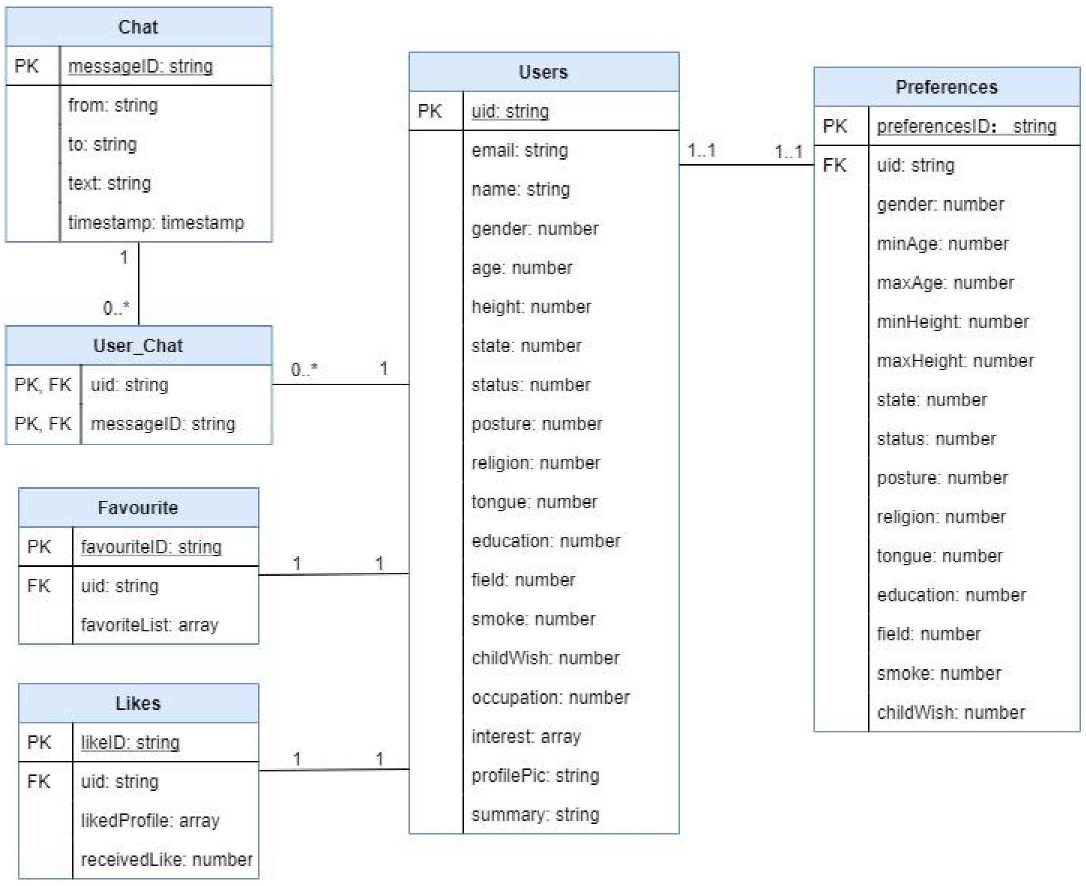
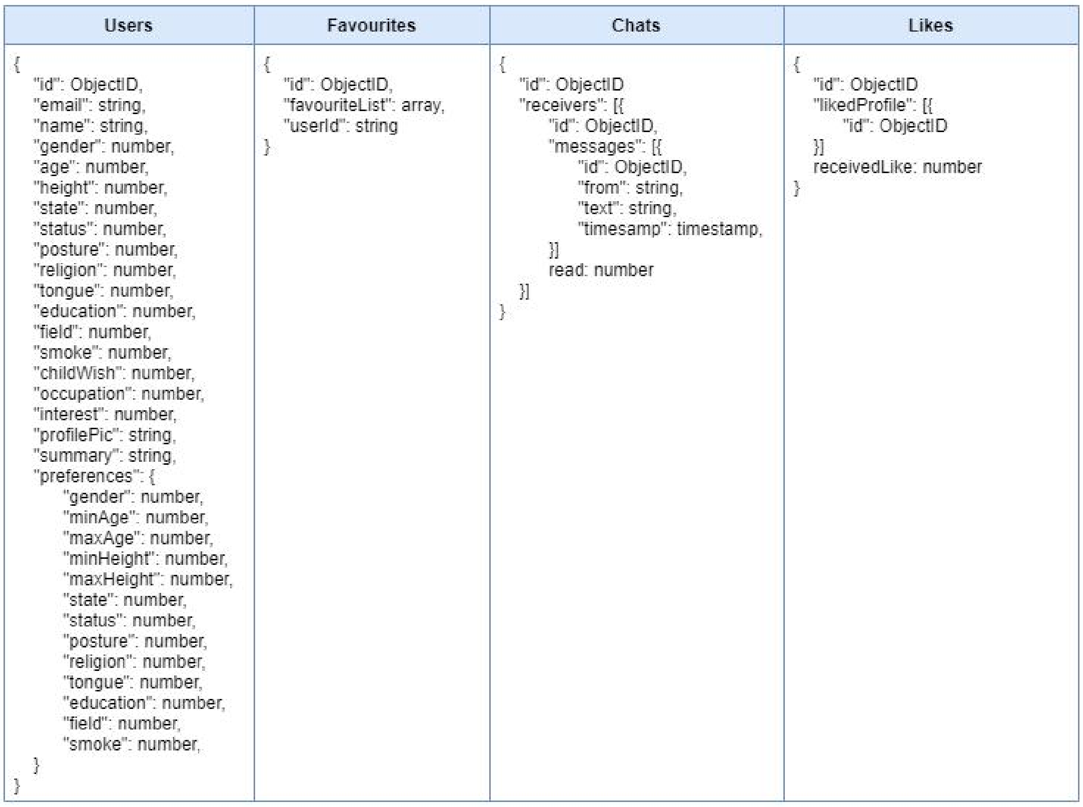


Figure 5.18 Logical Data Model Diagram

5.2.2 Physical Data Model Diagram



Users

```
{
    "id": ObjectID,
    "email": string,
    "name": string,
    "gender": number,
    "age": number,
    "height": number,
    "state": number,
    "status": number,
    "posture": number,
    "religion": number,
    "tongue": number,
    "education": number,
    "field": number,
    "smoke": number,
    "childWish": number,
    "occupation": number,
    "interest": number,
    "profilePic": string,
    "summary": string,
    "preferences": {
        "gender": number,
        "minAge": number,
        "maxAge": number,
        "minHeight": number,
        "maxHeight": number,
        "state": number,
        "status": number,
        "posture": number,
        "religion": number,
        "tongue": number,
        "education": number,
        "field": number,
        "smoke": number,
    }
}
```

Favourites

```
{
    "id": ObjectID,
    "favouriteList": array,
    "userId": string
}
```

Chats

```
{
    "id": ObjectID
    "receivers": [{
        "id": ObjectID,
        "messages": [{
            "id": ObjectID,
            "from": string,
            "text": string,
            "timesamp": timestamp,
        }]
        read: number
    }]
}
```

Likes

```
{
    "id": ObjectID
    "likedProfile": [{
        "id": ObjectID
    }]
    receivedLike: number
}
```

Figure 5.19 Firebase Schema Design

The users table, preferences table and favourite table was implemented in Firestore. The attributes of preferences are embedded in the Users collection. In this project, the preferences information and user information are always retrieved and displayed together as user profile information. Thus, the need to join in query can be reduced by combining both tables and the data retrieving speed will be improved.

The chats table and likes table were implemented in Firebase Real-time Database. However, the data are structured as a JSON tree in Firebase Real-time Database. In this project, one user can chat with many members, while the user can send many messages to a member. Hence, the database structure was implemented as the schema design in Figure 5.4. Besides, the likes table stored the uid of the members that liked by the user. The number of likes received by the user will be shown in the personal profile page.

5.2.3 Data Dictionary

Table 5.1 Data Dictionary of Users Collection

Field Name	**Data Type**	**Caption**	**PK / FK**	**Nullable**
uid	string	Identification for every user	PK	No
email	string	User’s registered email address	-	No
gender	number	User’s gender	-	No
name	string	User’s name	-	No
age	number	User’s age	-	No
height	number	User’s height	-	No
state	number	User’s living state / province	-	No
status	number	User’s current marital status	-	No
posture	number	User’s body type / posture	-	No
religion	number	User’s religion	-	No
tongue	number	User’s native language	-	No
education	number	User’s education level	-	No
field	number	User’s field of study	-	No
smoke	number	User’s smoking habit	-	No
childWish	number	User’s desire for a child	-	No
occupation	number	User’s current occupation	-	No
interest	array	User’s hobbies / interests	-	No
profilePic	string	The link of user profile picture in Firebase Cloud Storage	-	No
summary	string	A short brief to describes user	-	Yes
preferences	object	User’s match preferences	-	No

Table 5.2 Data Dictionary of Preferences Attribute in Users Collection

Field Name	**Data Type**	**Caption**	**PK / FK**	**Nullable**
gender	number	Preferred partner’s gender	PK	No
minAge	number	Preferred partner’s age range (minimum age)	-	No
maxAge	number	Preferred partner’s age range (maximum age)	-	No

minHeight	number	Preferred partner's height range (minimum height)	-	No
maxHeight	number	Preferred partner's height range (maximum height)	-	No
state	number	Preferred partner's living state / province	-	No
status	number	Preferred partner's current marital status	-	No
posture	number	Preferred partner's body type / posture	-	No
religion	number	Preferred partner's religion	-	No
tongue	number	Preferred partner's native language	-	No
education	number	Preferred partner's education level	-	No
field	number	Preferred partner's field of study	-	No
smoke	number	Preferred partner's smoking habit	-	No
childWish	number	Preferred partner's desire for a child	-	No

Table 5.3 Data Dictionary of Favourites Collection

Field Name	**Data Type**	**Caption**	**PK / FK**	**Nullable**
favouriteId	string	Identification for favourite	PK	No
list	array	User's Favourite list	-	No
uid	string	Identification for user that the favourite list belongs to	FK	No

Table 5.4 Data Dictionary of Chats JSON tree

Field Name	Data Type	Caption	PK / FK	Nullable
id	string	Identification for each chat (uid of the user who send the chat)	PK	No
receivers	array of object	Chat message receiver	-	No

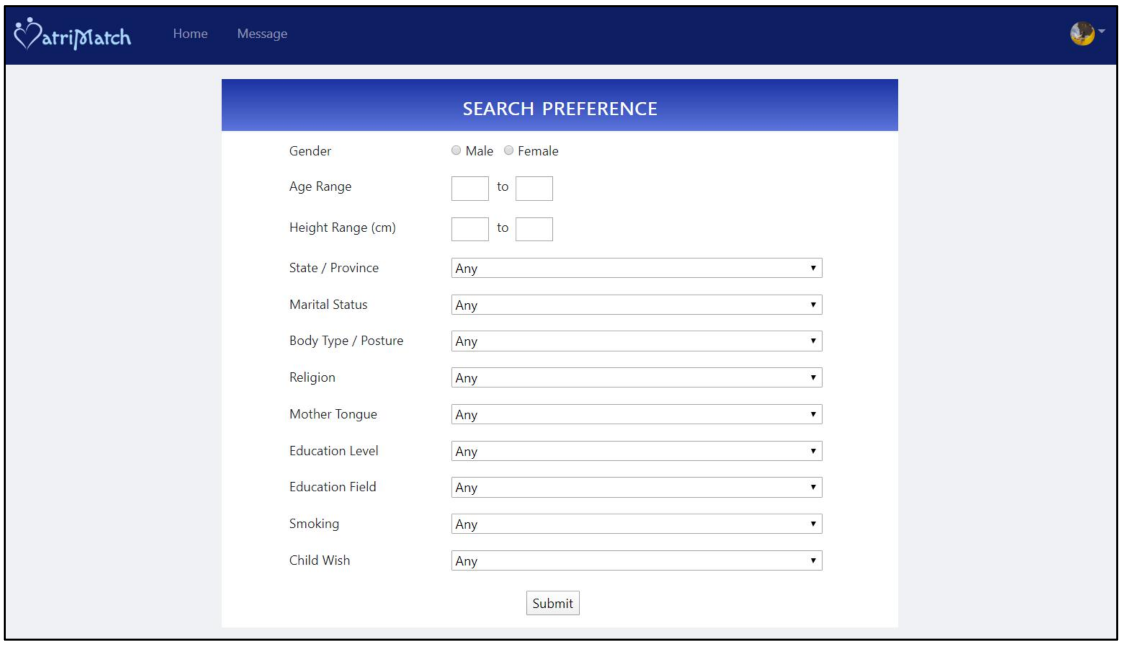
Table 5.5 Data Dictionary of Receivers Attribute in Chats JSON tree

Field Name	Data Type	Caption	PK / FK	Nullable
id	string	Identification for each receiver (uid of the receiver)	PK	No
messages	array of object	Chat message content ● id: identification of message ● from: user who send the message ● text: content of the message ● timestamp: the time when the sender send the message	-	No
read	number	The number of messages that has been read by user	-	Yes

Table 5.6 Data Dictionary of Likes JSON tree

Field Name	Data Type	Caption	PK / FK	Nullable
id	string	Identification for each like (uid of the user who send the chat)	PK	No
likedProfile	array of object	A list of profiles liked by user ● id: identification of liked profile (uid of the liked profile)	-	Yes
receivedLike	number	Number of likes received from other members	-	Yes

5.3 Preliminary User Interface Design

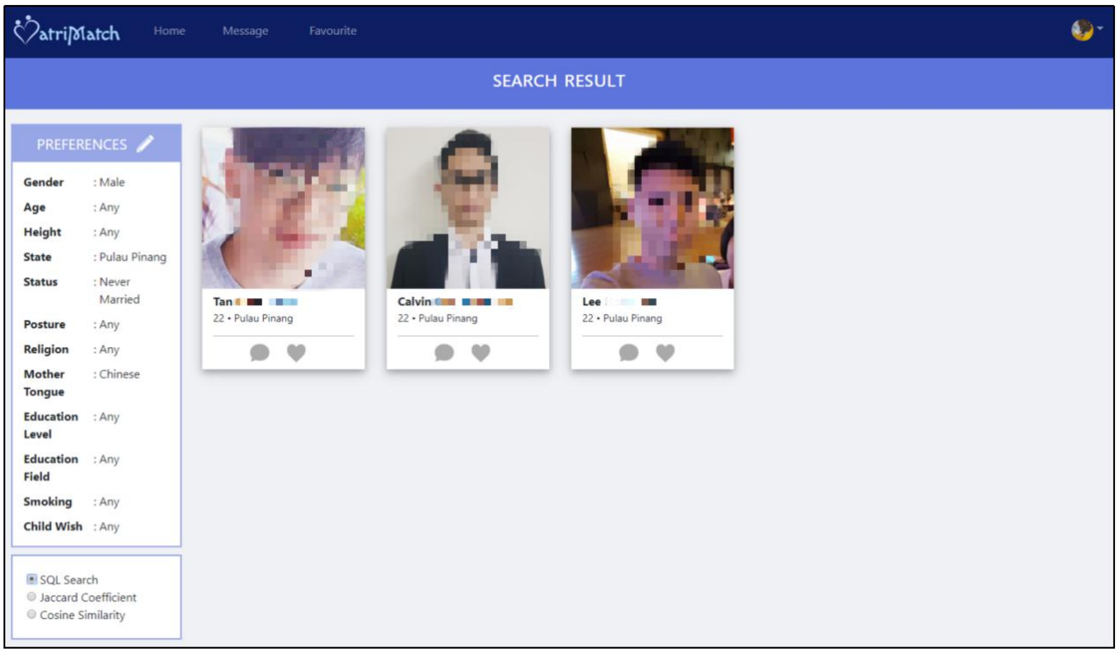


SEARCH PREFERENCE

Gender	○ Male ○ Female
Age Range	to
Height Range (cm)	to
State / Province	Any
Marital Status	Any
Body Type / Posture	Any
Religion	Any
Mother Tongue	Any
Education Level	Any
Education Field	Any
Smoking	Any
Child Wish	Any

Submit

Figure 5.20 UI – Overview Layout of the application



SEARCH RESULT

PREFERENCES

Gender	: Male
Age	: Any
Height	: Any
State	: Pulau Pinang
Status	: Never Married
Posture	: Any
Religion	: Any
Mother Tongue	: Chinese
Education Level	: Any
Education Field	: Any
Smoking	: Any
Child Wish	: Any

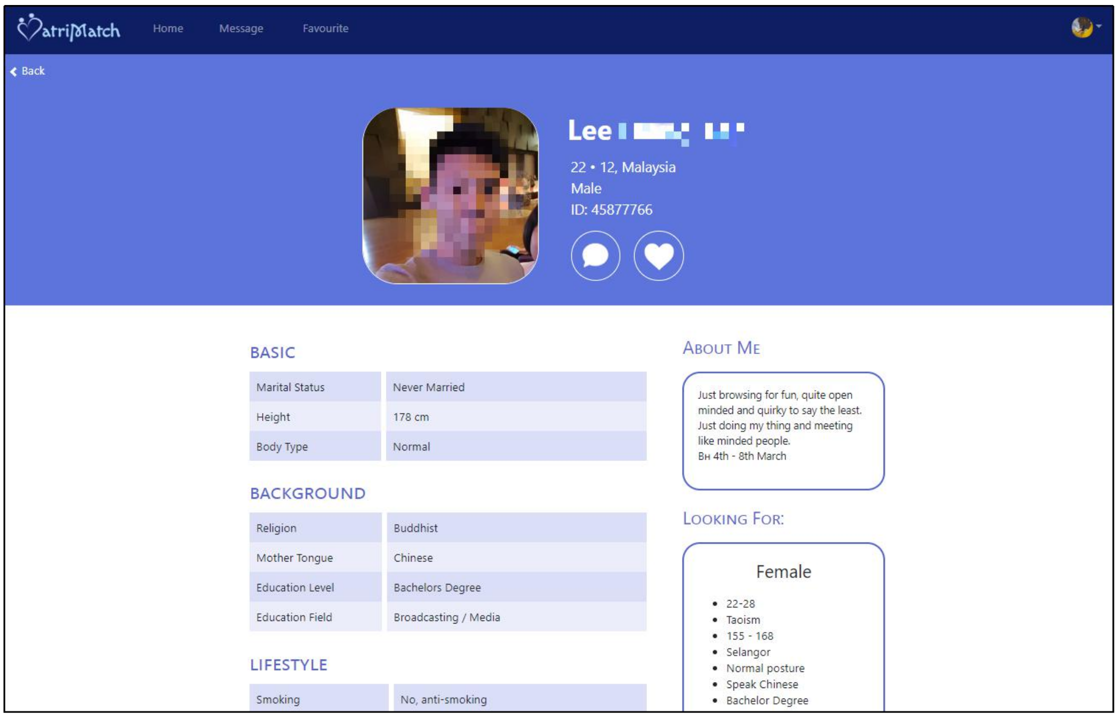
● SQL Search
○ Jaccard Coefficient
○ Cosine Similarity

Tan [redacted]
22 • Pulau Pinang

Calvin [redacted]
22 • Pulau Pinang

Lee [redacted]
22 • Pulau Pinang

Figure 5.21 UI – Search Result Page (Exact Match)



Lee [redacted]

22 • 12, Malaysia
Male
ID: 45877766

BASIC

Marital Status	Never Married
Height	178 cm
Body Type	Normal

BACKGROUND

Religion	Buddhist
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Broadcasting / Media

LIFESTYLE

Smoking	No, anti-smoking

ABOUT ME

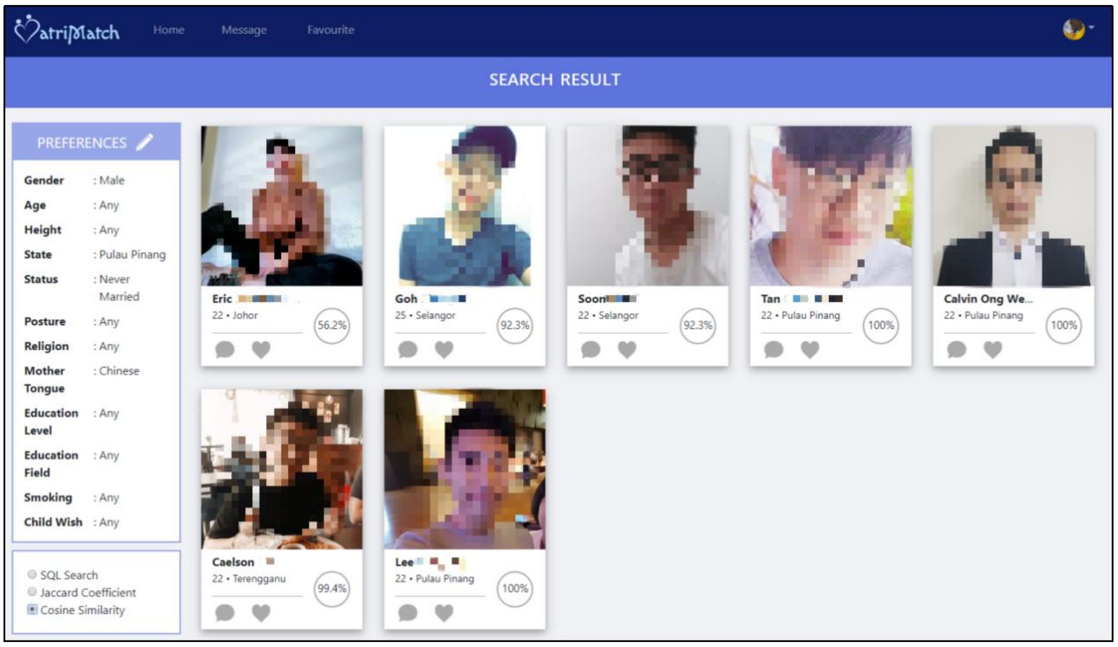
Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded people.
BH 4th - 8th March

LOOKING FOR:

Female

- 22-28
- Taoism
- 155 - 168
- Selangor
- Normal posture
- Speak Chinese
- Bachelor Degree

Figure 5.22 UI – Display Profile Details (Exact Match)



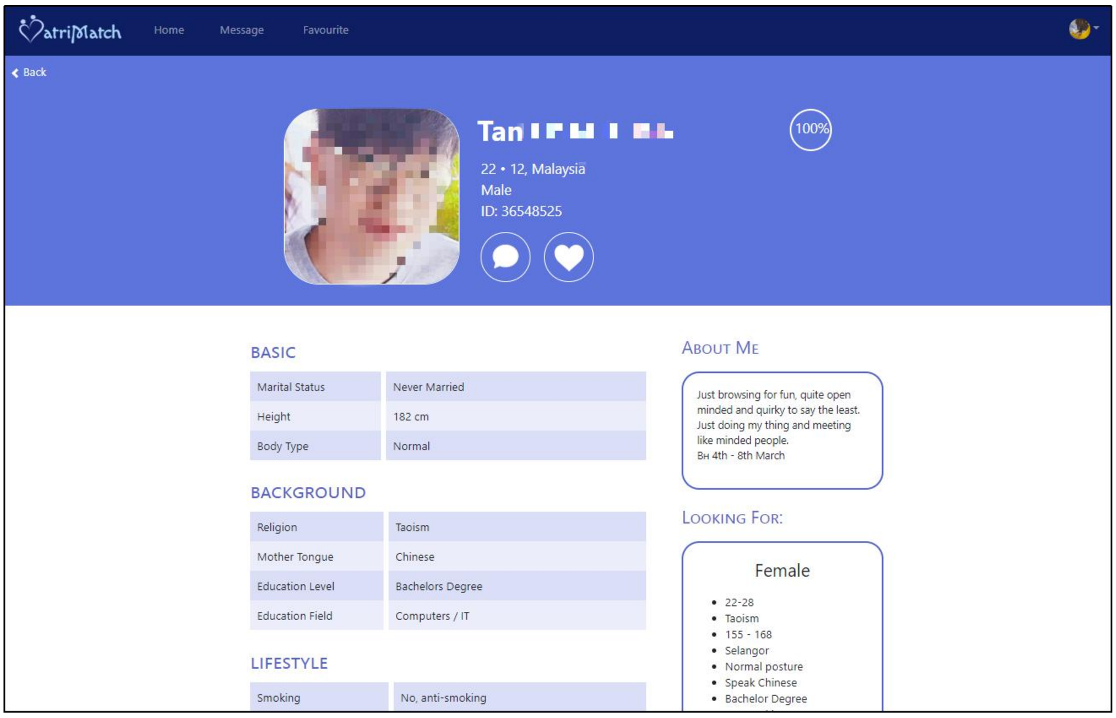
PREFERENCES

Gender	: Male
Age	: Any
Height	: Any
State	: Pulau Pinang
Status	: Never Married
Posture	: Any
Religion	: Any
Mother Tongue	: Chinese
Education Level	: Any
Education Field	: Any
Smoking	: Any
Child Wish	: Any

○ SQL Search
○ Jaccard Coefficient
● Cosine Similarity

Eric [redacted]	22 • Johor	56.2%
Goh [redacted]	25 • Selangor	92.3%
Soon [redacted]	22 • Selangor	92.3%
Tan [redacted]	22 • Pulau Pinang	100%
Calvin Ong We...	22 • Pulau Pinang	100%
Caelson [redacted]	22 • Terengganu	99.4%
Lee [redacted]	22 • Pulau Pinang	100%

Figure 5.23 UI – Search Result Page (Similarity Measures)



Home Message Favourite

Back

Tan [redacted]

100%

22 • 12, Malaysia
Male
ID: 36548525

BASIC

Marital Status	Never Married
Height	182 cm
Body Type	Normal

BACKGROUND

Religion	Taoism
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Computers / IT

LIFESTYLE

Smoking	No, anti-smoking

ABOUT ME

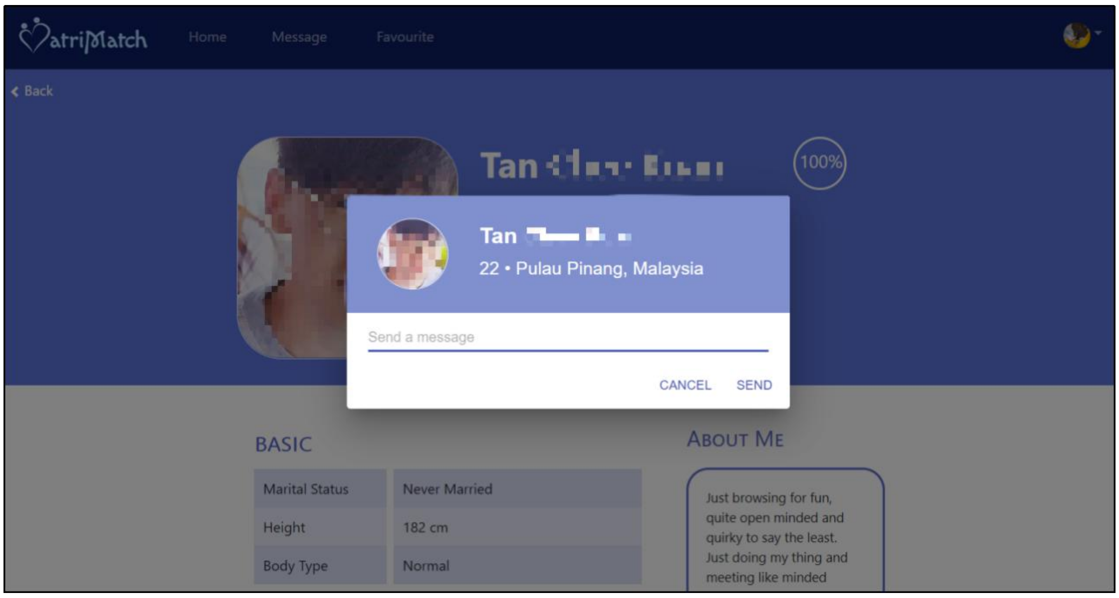
Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded people.
BH 4th - 8th March

LOOKING FOR:

Female

- 22-28
- Taoism
- 155 - 168
- Selangor
- Normal posture
- Speak Chinese
- Bachelor Degree

Figure 5.24 UI – Display Profile Details (Similarity Measures)



Home Message Favourite

Back

Tan [redacted]

100%

Tan [redacted]

22 • Pulau Pinang, Malaysia

Send a message

CANCEL SEND

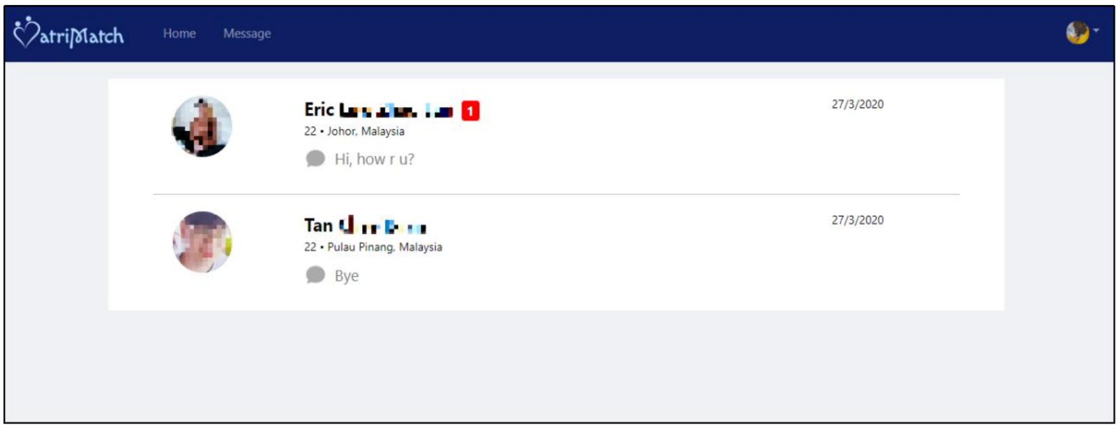
BASIC

Marital Status	Never Married
Height	182 cm
Body Type	Normal

ABOUT ME

Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded

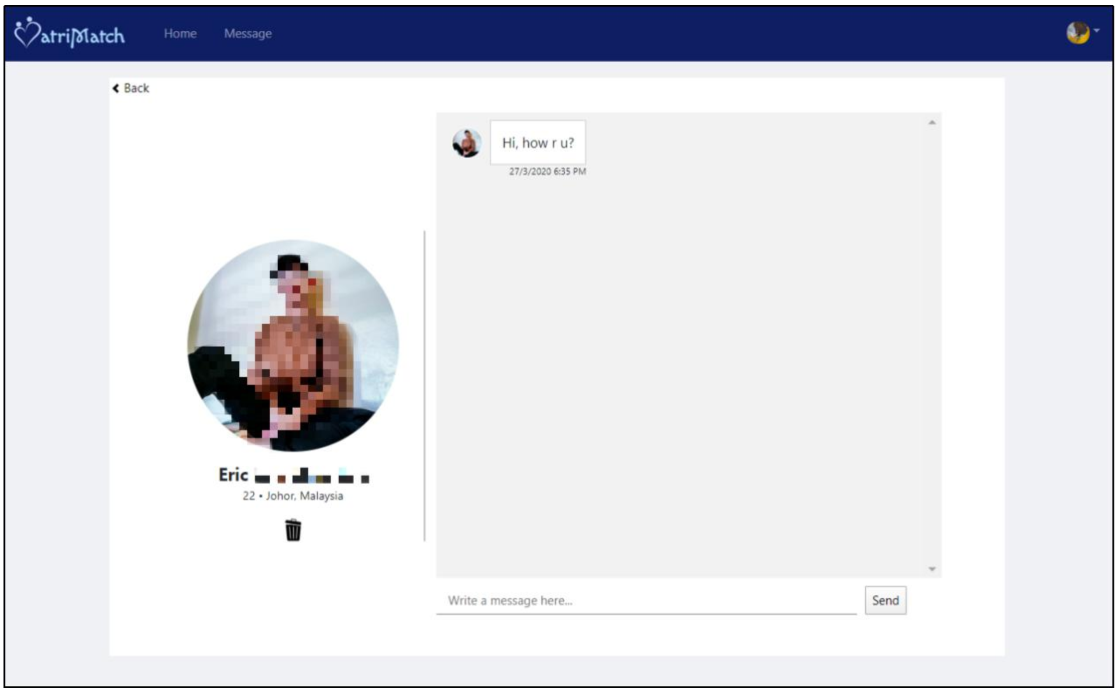
Figure 5.25 UI – Instant Message Feature



Eric [redacted] 1
22 • Johor, Malaysia
Hi, how r u?
27/3/2020

Tan [redacted]
22 • Pulau Pinang, Malaysia
Bye
27/3/2020

Figure 5.26 UI – Chat List Page

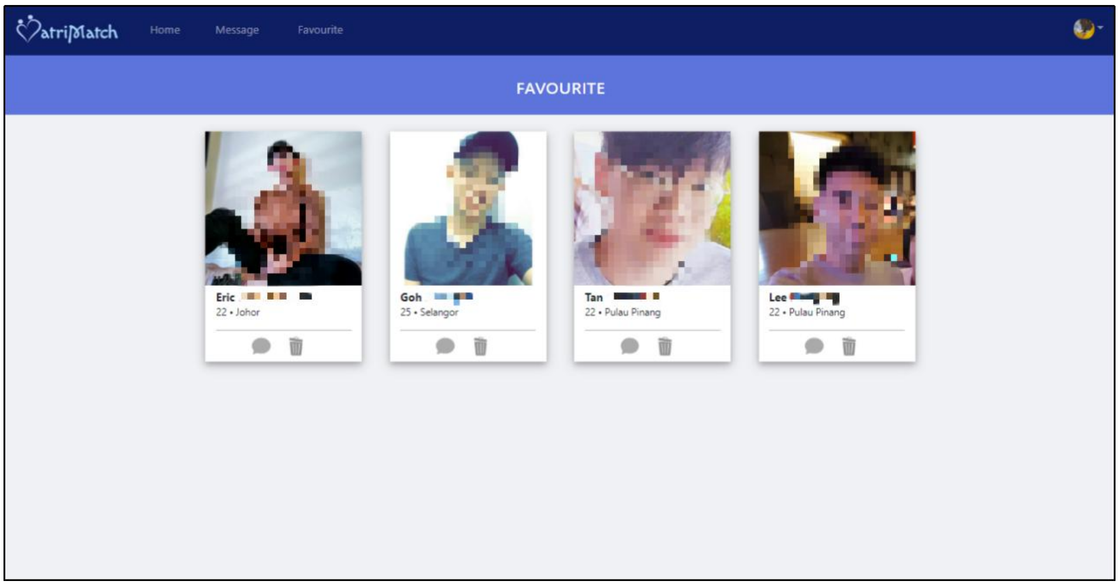


Eric [redacted]
22 • Johor, Malaysia

Hi, how r u?
27/3/2020 6:35 PM

Write a message here...

Figure 5.27 UI – Chat Message Page



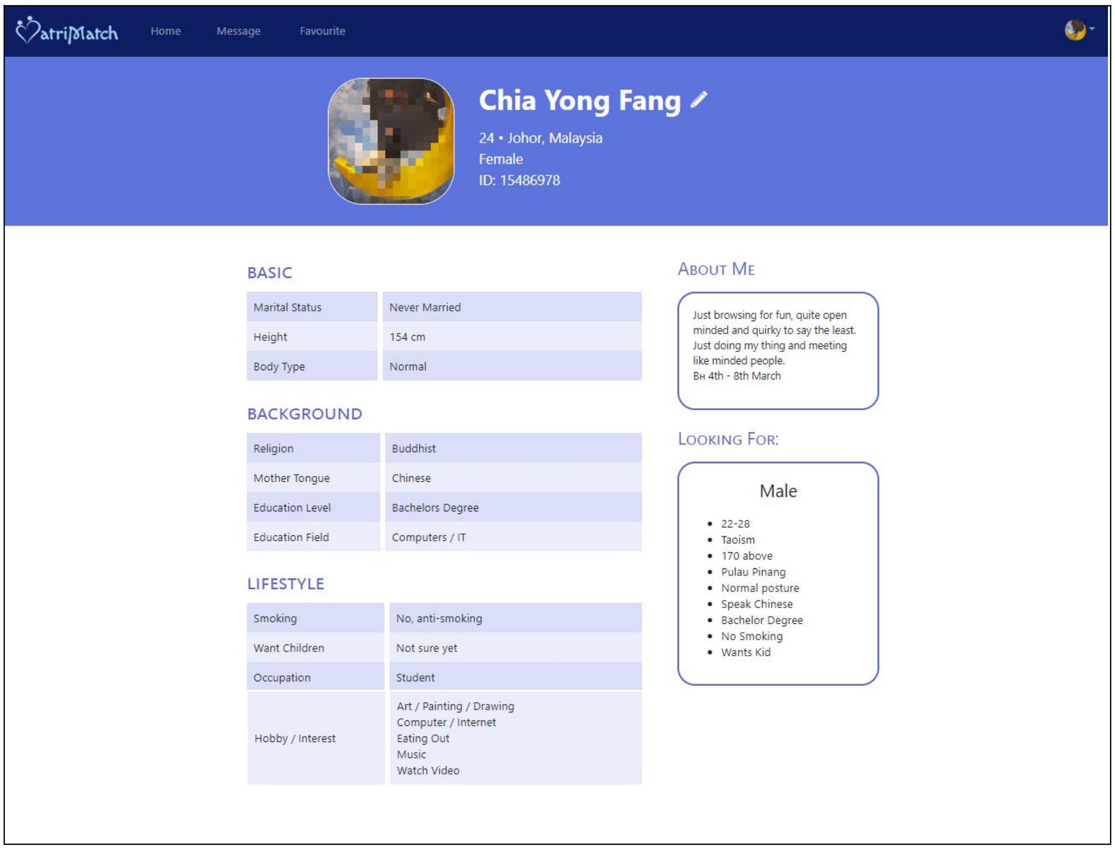
Eric [redacted]
22 • Johor

Goh [redacted]
25 • Selangor

Tan [redacted]
22 • Pulau Pinang

Lee [redacted]
22 • Pulau Pinang

Figure 5.28 UI – Favourite List Page



Chia Yong Fang

24 • Johor, Malaysia
Female
ID: 15486978

BASIC

Marital Status	Never Married
Height	154 cm
Body Type	Normal

BACKGROUND

Religion	Buddhist
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Computers / IT

LIFESTYLE

Smoking	No, anti-smoking
Want Children	Not sure yet
Occupation	Student
Hobby / Interest	Art / Painting / Drawing Computer / Internet Eating Out Music Watch Video

About Me

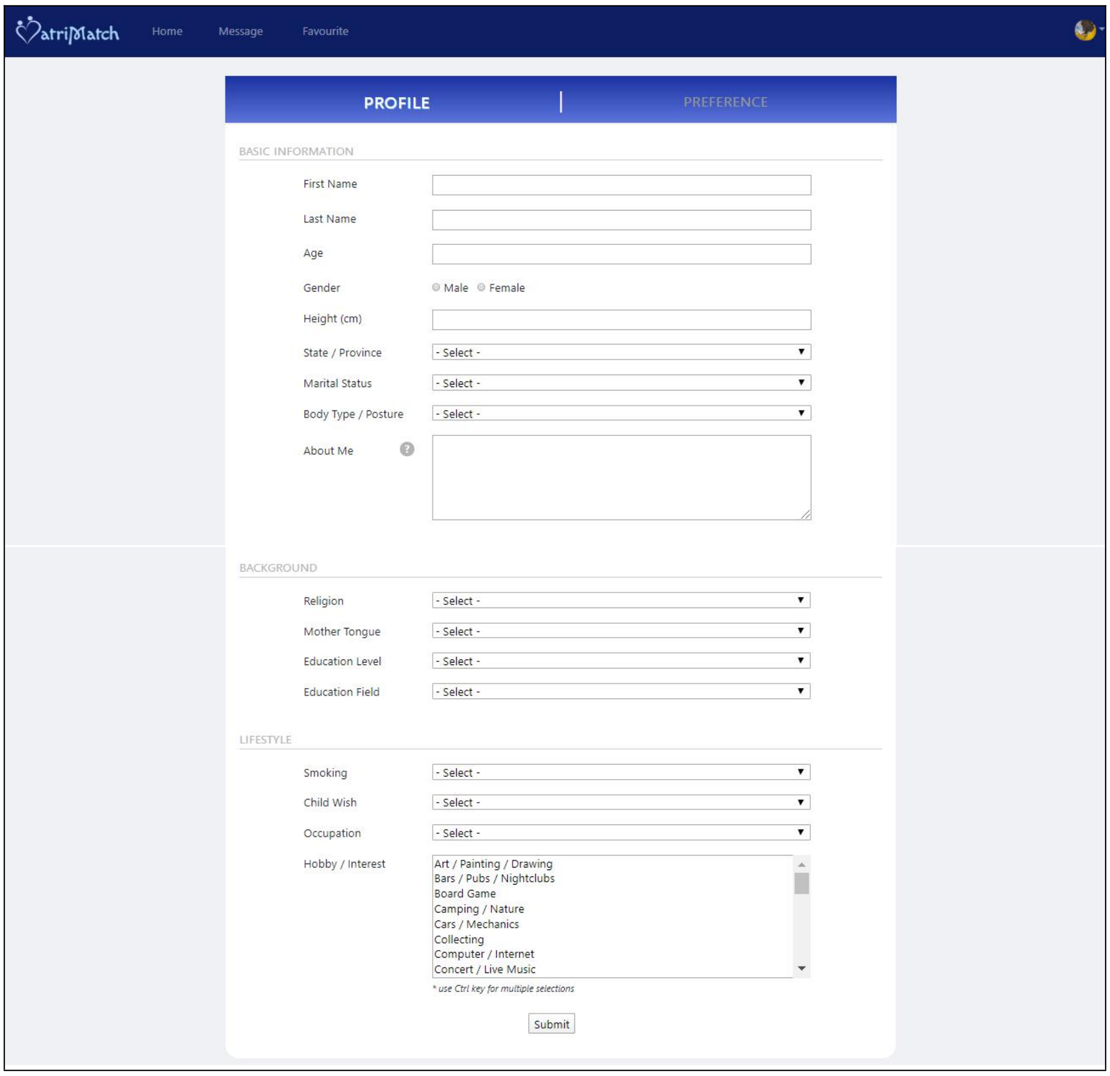
Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded people.
Bн 4th - 8th March

Looking For:

Male

- 22-28
- Taoism
- 170 above
- Pulau Pinang
- Normal posture
- Speak Chinese
- Bachelor Degree
- No Smoking
- Wants Kid

Figure 5.29 UI – Personal Profile Page


PatriMatch Home Message Favourite

PROFILE | PREFERENCE

BASIC INFORMATION

First Name	
Last Name	
Age	
Gender	○ Male ○ Female
Height (cm)	
State / Province	- Select -
Marital Status	- Select -
Body Type / Posture	- Select -
About Me	

BACKGROUND

Religion	- Select -
Mother Tongue	- Select -
Education Level	- Select -
Education Field	- Select -

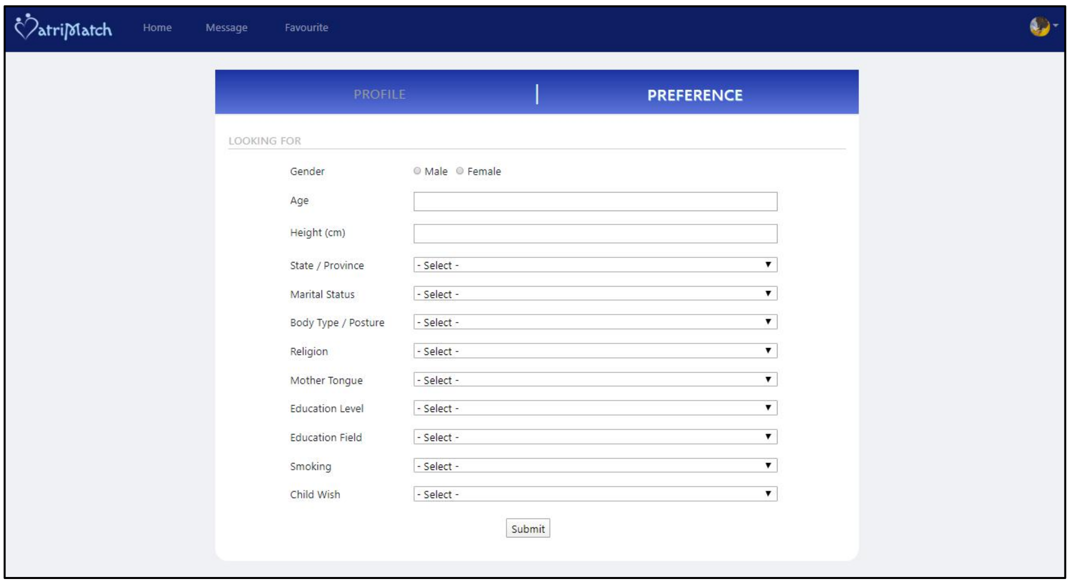
LIFESTYLE

Smoking	- Select -
Child Wish	- Select -
Occupation	- Select -
Hobby / Interest	Art / Painting / Drawing Bars / Pubs / Nightclubs Board Game Camping / Nature Cars / Mechanics Collecting Computer / Internet Concert / Live Music

* use Ctrl key for multiple selections

Submit

Figure 5.30 UI – Edit Profile Page (Personal Information)


PatriMatch Home Message Favourite

PROFILE | PREFERENCE

LOOKING FOR

Gender	○ Male ○ Female
Age	
Height (cm)	
State / Province	- Select -
Marital Status	- Select -
Body Type / Posture	- Select -
Religion	- Select -
Mother Tongue	- Select -
Education Level	- Select -
Education Field	- Select -
Smoking	- Select -
Child Wish	- Select -

Submit

Figure 5.31 UI – Edit Profile Page (Preferences)

CHAPTER 6

IMPLEMENTATION

6.1 Web API Endpoint

There are 13 API endpoints used in this project. All the API endpoints were implemented in the server (Express.js).

Table 6.1: List of Web API Endpoints

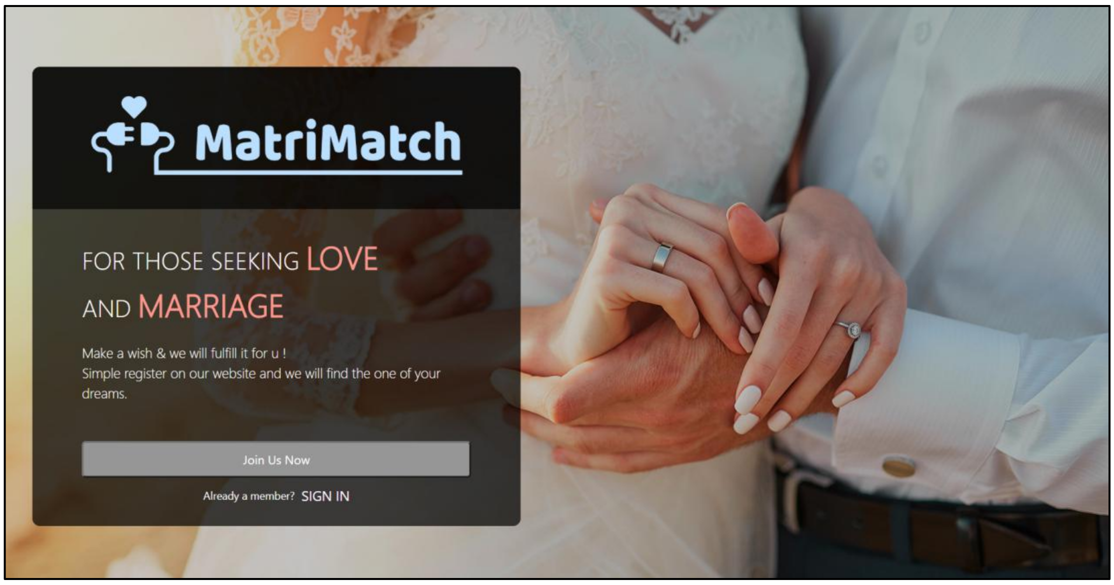
Route	**Type**	**Description**
/api/createProfile	POST	Create a profile for the new user
/api/getPersonalInfo	POST	Get personal profile information
/api/updateProfilePic	POST	Update user's profile picture
/api/updateProfileDetails	POST	Update user's personal information (demographic, background and lifestyle)
/api/updateProfilePreferences	POST	Update user's personal information (match preferences)
/api/getExactMatchResult	POST	Get matching results by exact matching
/api/getSimilarityResult	POST	Get matching results by similarity measures (Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance)
/api/checkFavourite	POST	Get the favourite list to check whether the currently viewed profile was added into favourite

/api/updateFavourite	POST	Update favourite list to remove currently viewed profile
/api/getFavouriteList	POST	Get a list of profiles that added to favourite
/api/deleteFavourite	POST	Update the favourite list to remove the unwanted profile
/api/getMemberData	POST	Get the profile information of the receivers in the chat list
/api/getProfilePicture	POST	Get user profile picture to be displayed in the chat

6.2 Application Implementation

6.2.1 Main Page

This is the first screen the visitors will encounter when they enter the application. If the visitors wish to access the application, they have to register/login an account.


MatriMatch
FOR THOSE SEEKING LOVE AND MARRIAGE
Join Us Now
Already a member? SIGN IN

Make a wish & we will fulfill it for u !
Simple register on our website and we will find the one of your dreams.

Figure 6.1 Visitor’s Main Page

6.2.2 Registration

For the new users, they can choose to click the "Join Us Now" button to register a new account. They will be redirected to the registration page. The users have to enter their email address and password for the application. If the users already had an account, they can press the "Login" button to be redirected to the login page.

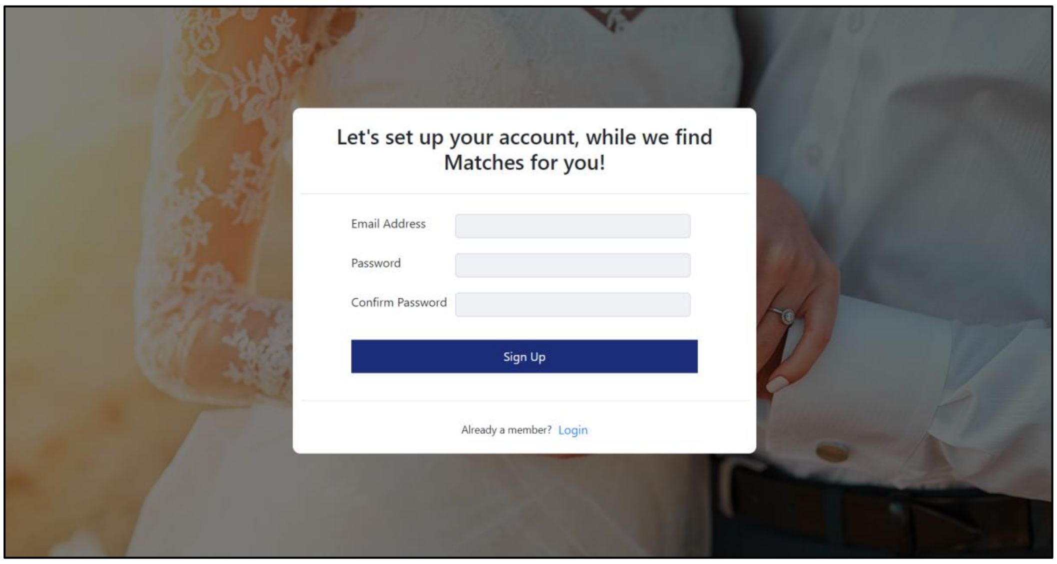


Figure 6.2 Registration Page

After the users successfully register their accounts, they are required to enter their personal information. They have to enter their demographic, background and lifestyle information along with their profile picture. All details are required to fill in, except for the "About You" section. The users may choose whether to fill in or leave it blank.

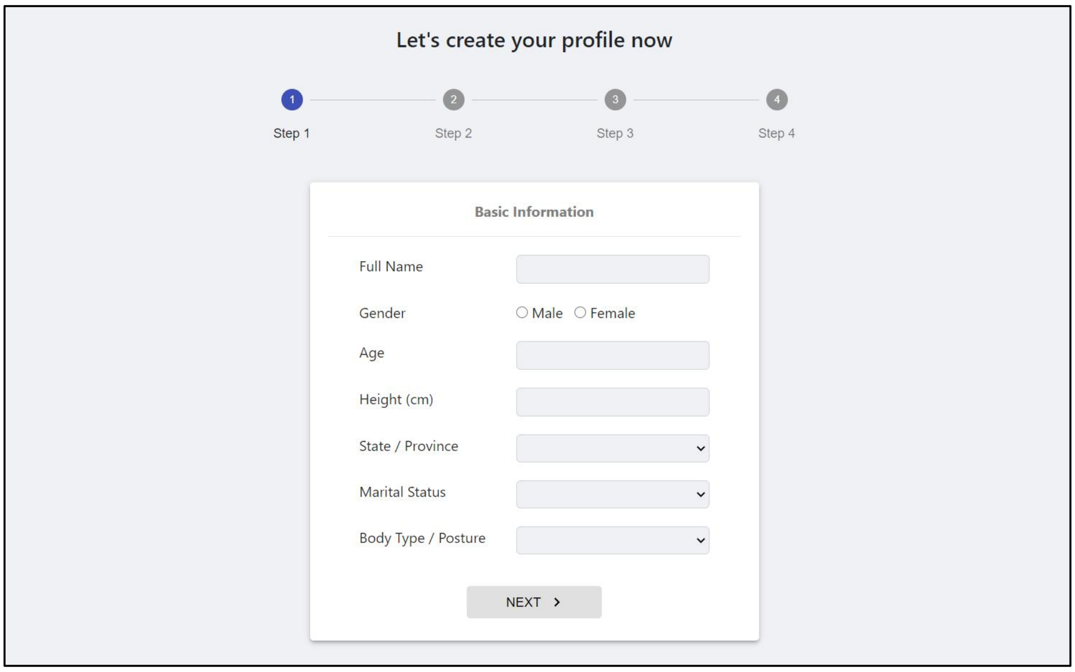


Figure 6.3 Create Profile Page - Part 1

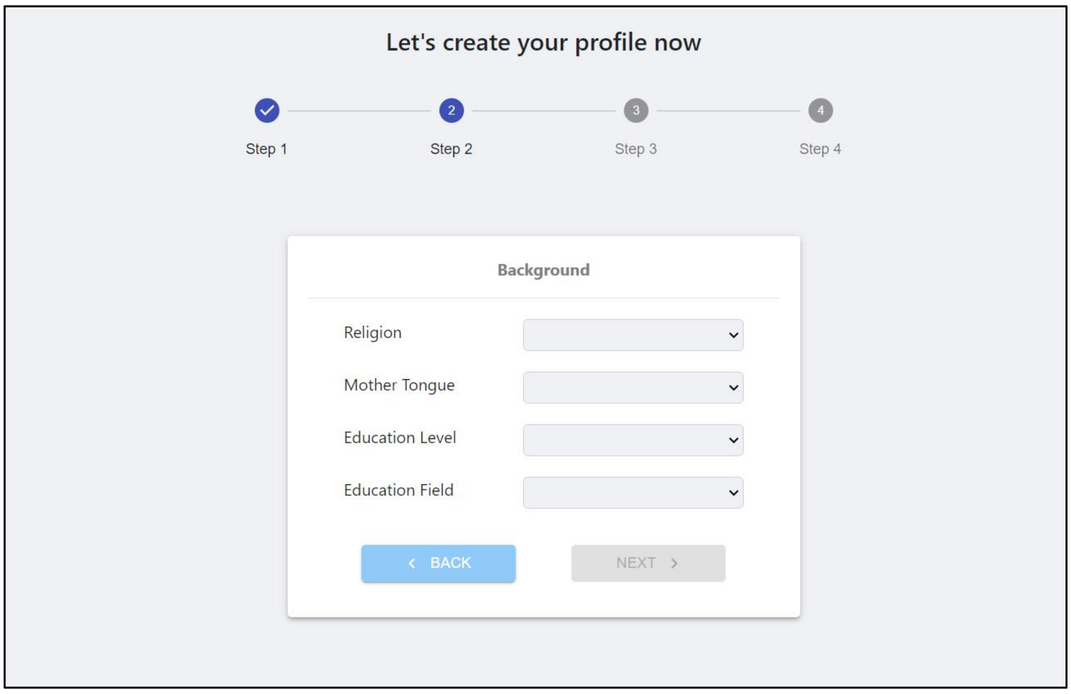


Figure 6.4 Create Profile Page - Part 2

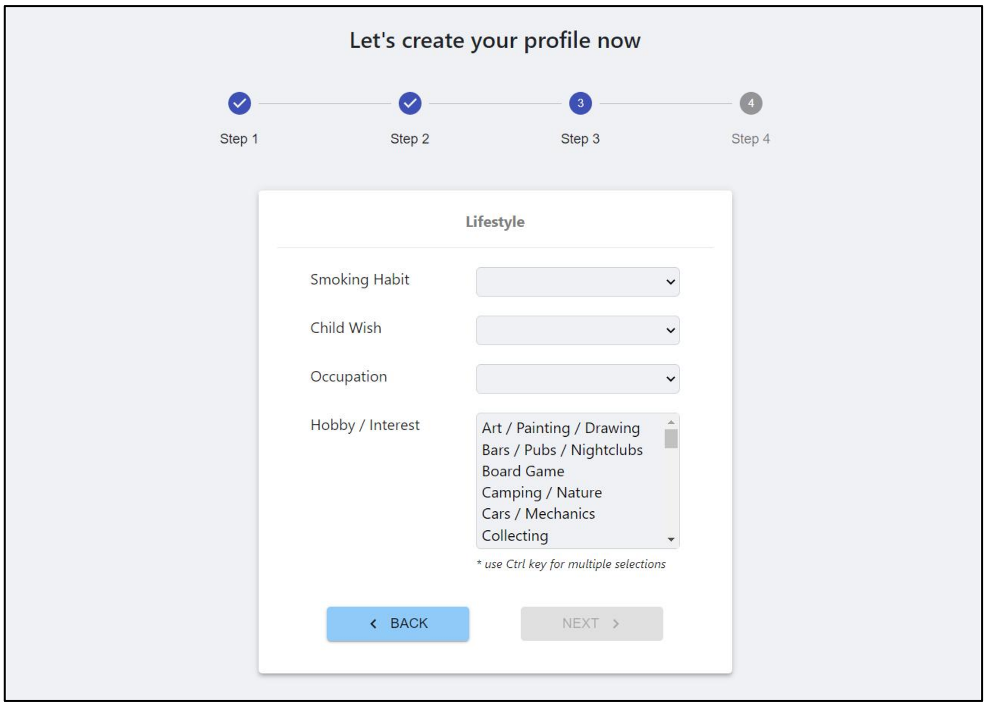


Figure 6.5 Create Profile Page - Part 3

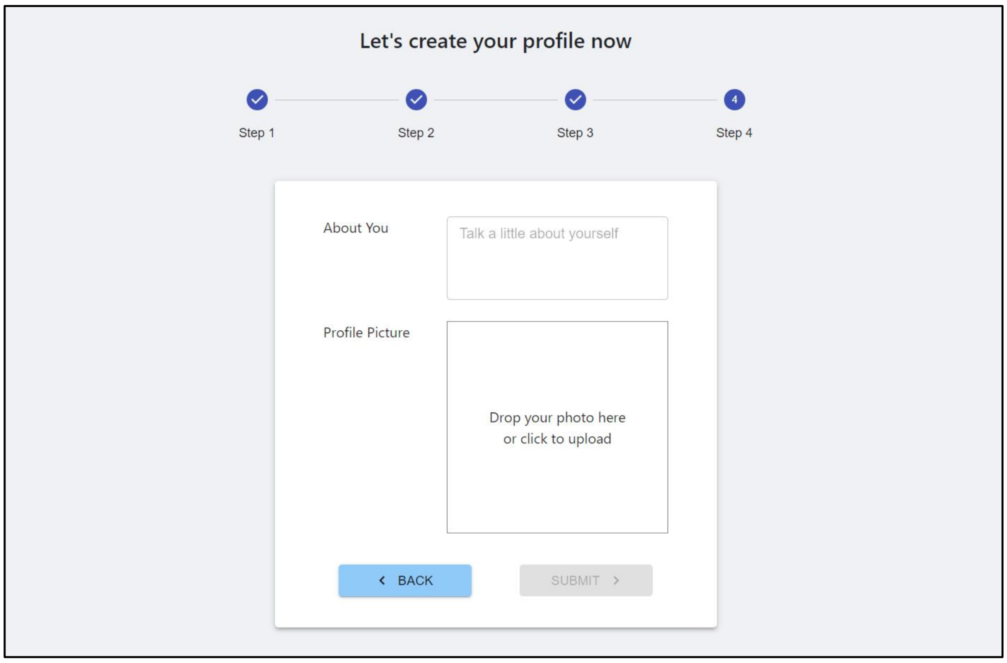
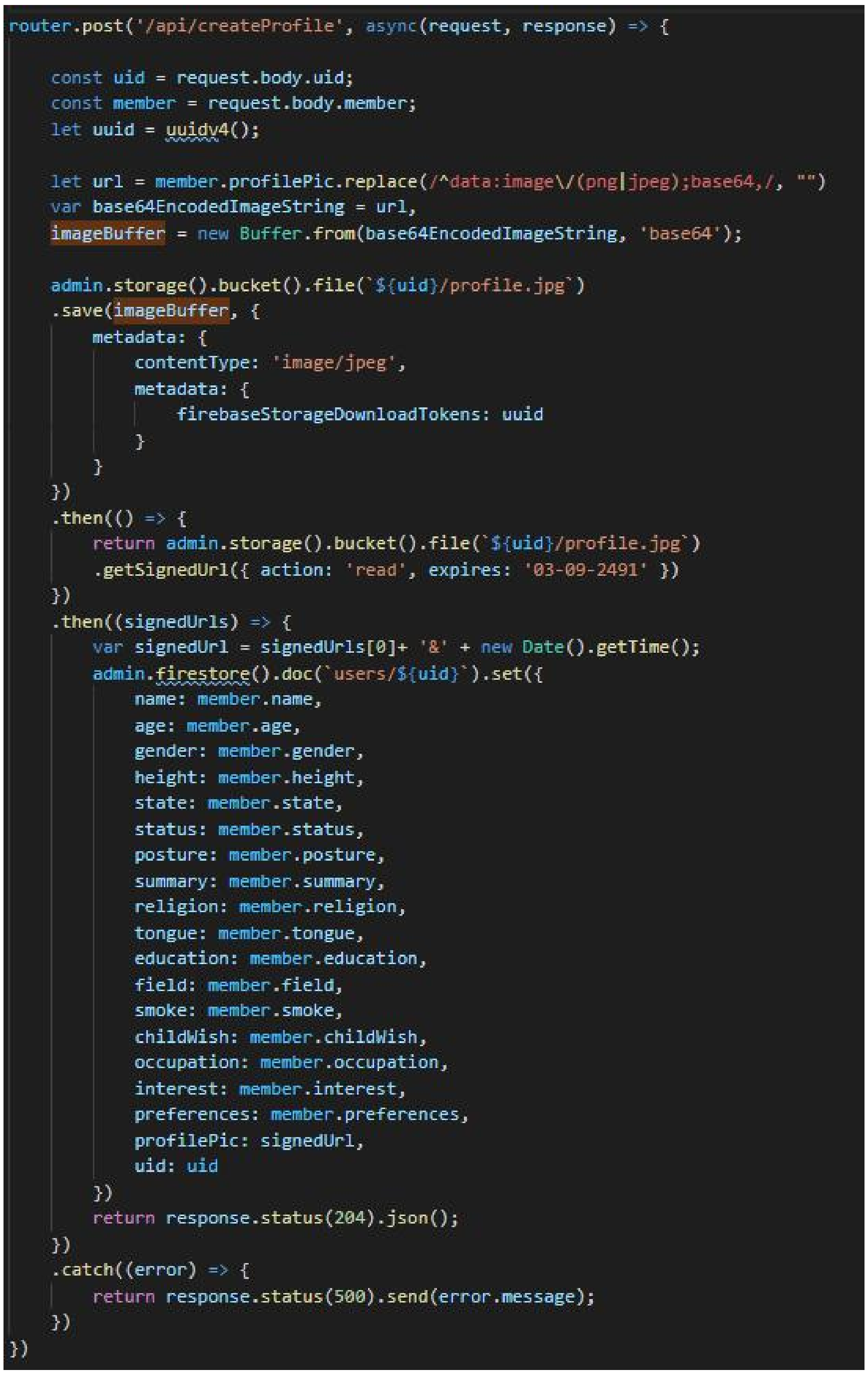


Figure 6.6 Create Profile Page - Part 4

After complete, the users can click the "Submit" button. The application will send a POST request to API endpoint (/api/createProfile) on the server. The server will store the image into Firebase Cloud Storage. Then, the server will add a new document to Firestore users collection to store the user data and the image URL retrieved from Firebase Cloud Storage. After that, the registration process is completed.

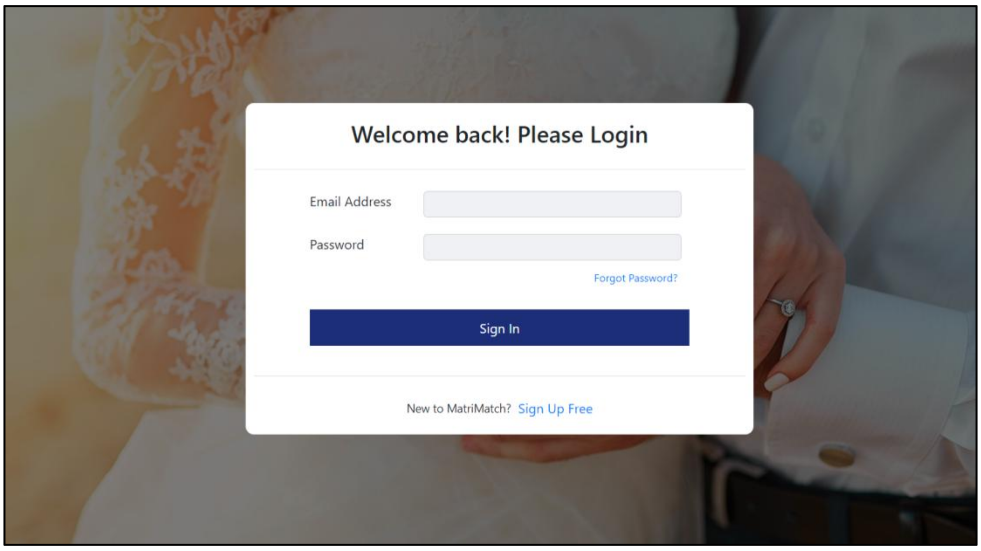


```
router.post('/api/createProfile', async(request, response) => {

    const uid = request.body.uid;
    const member = request.body.member;
    let uuid = uuidv4();

    let url = member.profilePic.replace(/^data:image\/(png|jpeg);base64,/, "")
    var base64EncodedImageString = url,
    imageBuffer = new Buffer.from(base64EncodedImageString, 'base64');

    admin.storage().bucket().file(`${uid}/profile.jpg`)
    .save(imageBuffer, {
        metadata: {
            contentType: 'image/jpeg',
            metadata: {
                firebaseStorageDownloadTokens: uuid
            }
        }
    })
    .then(() => {
        return admin.storage().bucket().file(`${uid}/profile.jpg`)
        .getSignedUrl({ action: 'read', expires: '03-09-2491' })
    })
    .then((signedUrls) => {
        var signedUrl = signedUrls[0]+ '&' + new Date().getTime();
        admin.firestore().doc(`users/${uid}`).set({
            name: member.name,
            age: member.age,
            gender: member.gender,
            height: member.height,
            state: member.state,
            status: member.status,
            posture: member.posture,
            summary: member.summary,
            religion: member.religion,
            tongue: member.tongue,
            education: member.education,
            field: member.field,
            smoke: member.smoke,
            childWish: member.childWish,
            occupation: member.occupation,
            interest: member.interest,
            preferences: member.preferences,
            profilePic: signedUrl,
            uid: uid
        })
        return response.status(204).json();
    })
    .catch((error) => {
        return response.status(500).send(error.message);
    })
})
```

Figure 6.7 Section Code for Create Profile (Server)

6.2.3 Login

For the existing users, they can choose to click “Sign In” button to login their account. They will be redirected to the login page. The users have to enter their email address and password for the application. If the users do not have an account, they can press the “Sign Up Free” button to be redirected to the registration page.


Welcome back! Please Login

Email Address

Password

Forgot Password?

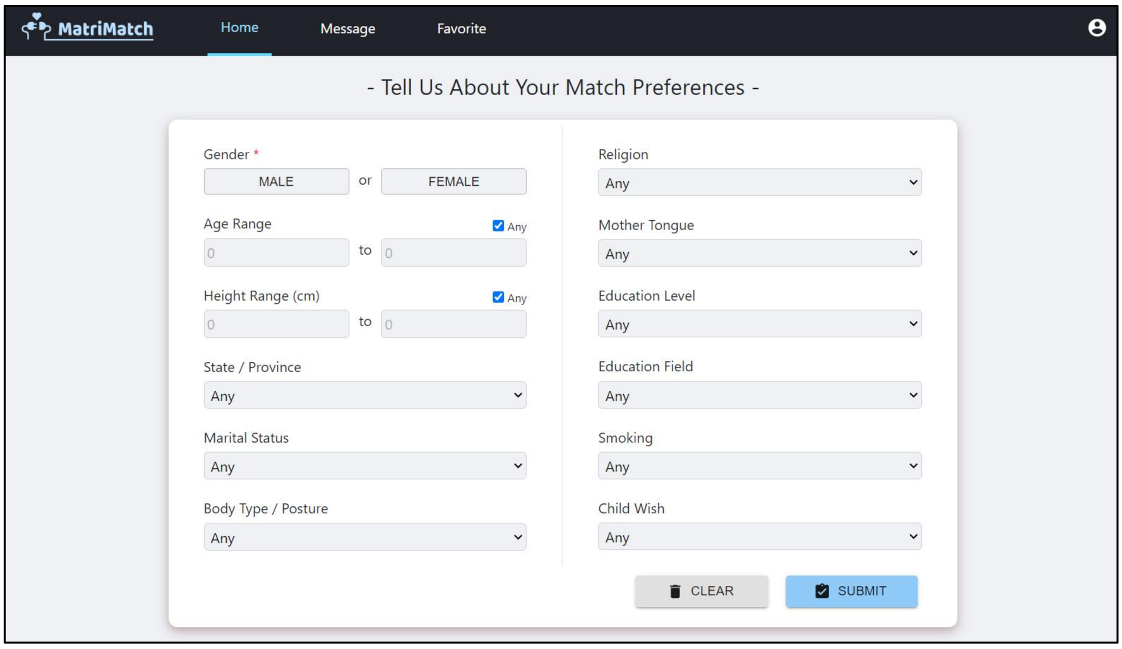
Sign In

New to MatriMatch? Sign Up Free

Figure 6.8 Login Page

6.2.4 Search Potential Matches

After the users register or login successfully, they will be redirected to the home page. In the home page, the users can set their match preferences to search for potential matches. There are 12 options for users to select. After the users set their match preferences, they can click the "Submit" button to view the search results.

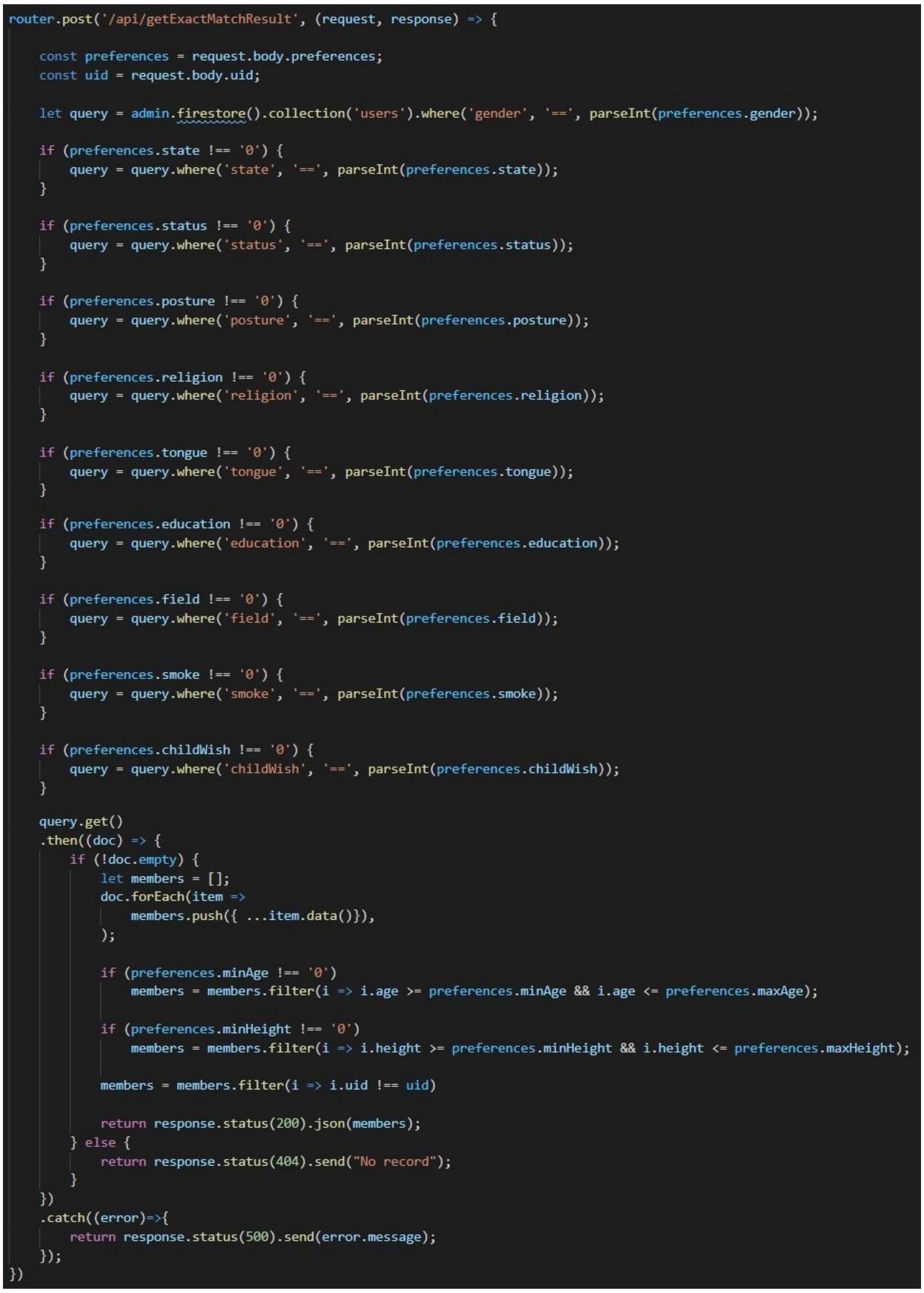


- Tell Us About Your Match Preferences -

Gender * MALE or FEMALE	Religion: Any
Age Range [✓ Any] 0 to 0	Mother Tongue: Any
Height Range (cm) [✓ Any] 0 to 0	Education Level: Any
State / Province: Any	Education Field: Any
Marital Status: Any	Smoking: Any
Body Type / Posture: Any	Child Wish: Any

Figure 6.9 Set Match Preferences Page

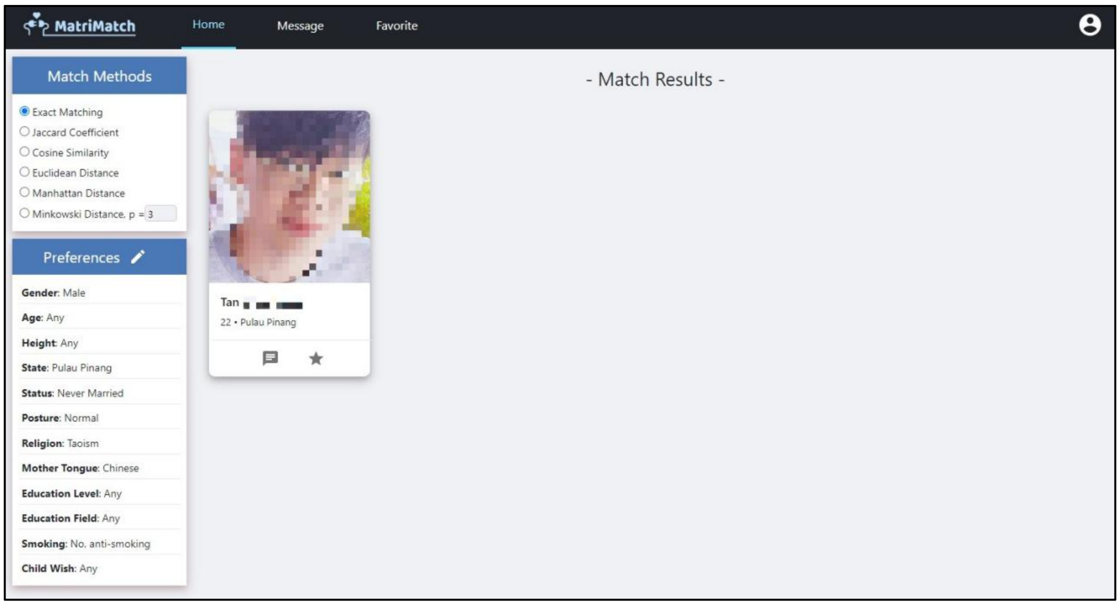
All the match preferences will be transformed form strings into numerical values. Then, the application will send the match preferences through a POST request to API endpoint (/api/getExactMatchResult) on the server. The server will perform exact matching to retrieve the match results from Firestore users collection. It will only return the profiles that exactly match the preferences set by users.



```javascript
router.post('/api/getExactMatchResult', (request, response) => {

    const preferences = request.body.preferences;
    const uid = request.body.uid;

    let query = admin.firestore().collection('users').where('gender', '==', parseInt(preferences.gender));

    if (preferences.state !== '0') {
        query = query.where('state', '==', parseInt(preferences.state));
    }

    if (preferences.status !== '0') {
        query = query.where('status', '==', parseInt(preferences.status));
    }

    if (preferences.posture !== '0') {
        query = query.where('posture', '==', parseInt(preferences.posture));
    }

    if (preferences.religion !== '0') {
        query = query.where('religion', '==', parseInt(preferences.religion));
    }

    if (preferences.tongue !== '0') {
        query = query.where('tongue', '==', parseInt(preferences.tongue));
    }

    if (preferences.education !== '0') {
        query = query.where('education', '==', parseInt(preferences.education));
    }

    if (preferences.field !== '0') {
        query = query.where('field', '==', parseInt(preferences.field));
    }

    if (preferences.smoke !== '0') {
        query = query.where('smoke', '==', parseInt(preferences.smoke));
    }

    if (preferences.childWish !== '0') {
        query = query.where('childWish', '==', parseInt(preferences.childWish));
    }

    query.get()
    .then((doc) => {
        if (!doc.empty) {
            let members = [];
            doc.forEach(item =>
                members.push({ ...item.data()}),
            );

            if (preferences.minAge !== '0')
                members = members.filter(i => i.age >= preferences.minAge && i.age <= preferences.maxAge);

            if (preferences.minHeight !== '0')
                members = members.filter(i => i.height >= preferences.minHeight && i.height <= preferences.maxHeight);

            members = members.filter(i => i.uid !== uid)

            return response.status(200).json(members);
        } else {
            return response.status(404).send("No record");
        }
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.10 Section Code for Get Exact Matching Result (Server)

If the users are not satisfied with the results, they can edit the match preferences by clicking the “Edit” button. The users will be redirected back to the Set Match Preferences Page (Figure 6.9).



Match Methods

- [✓ Exact Matching]
- Jaccard Coefficient
- Cosine Similarity
- Euclidean Distance
- Manhattan Distance
- Minkowski Distance, p = 3

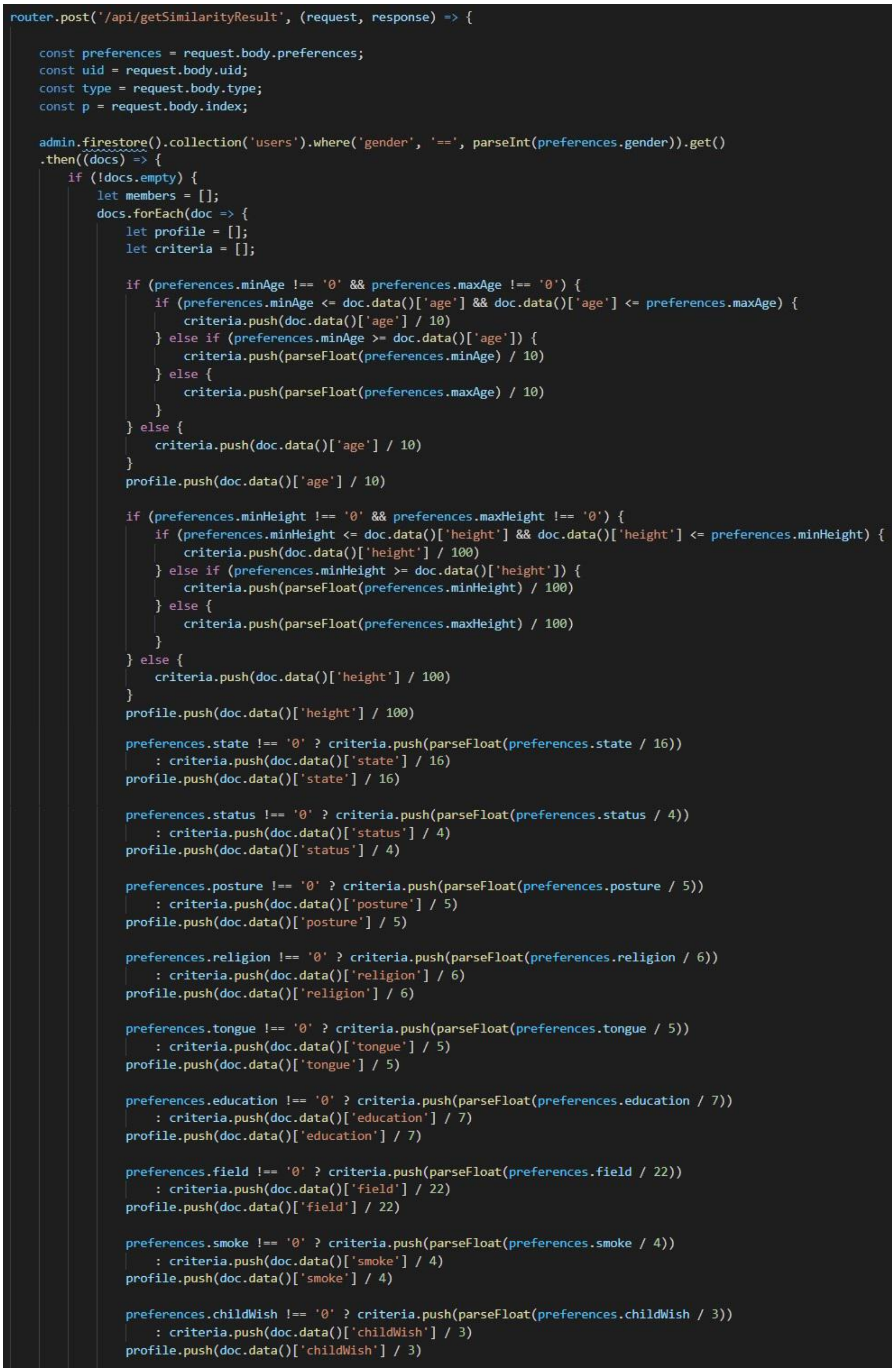
Preferences

Gender: Male
Age: Any
Height: Any
State: Pulau Pinang
Status: Never Married
Posture: Normal
Religion: Taoism
Mother Tongue: Chinese
Education Level: Any
Education Field: Any
Smoking: No, anti-smoking
Child Wish: Any

Tan [redacted]
22 • Pulau Pinang

Figure 6.11 Match Result Page (Exact Matching)

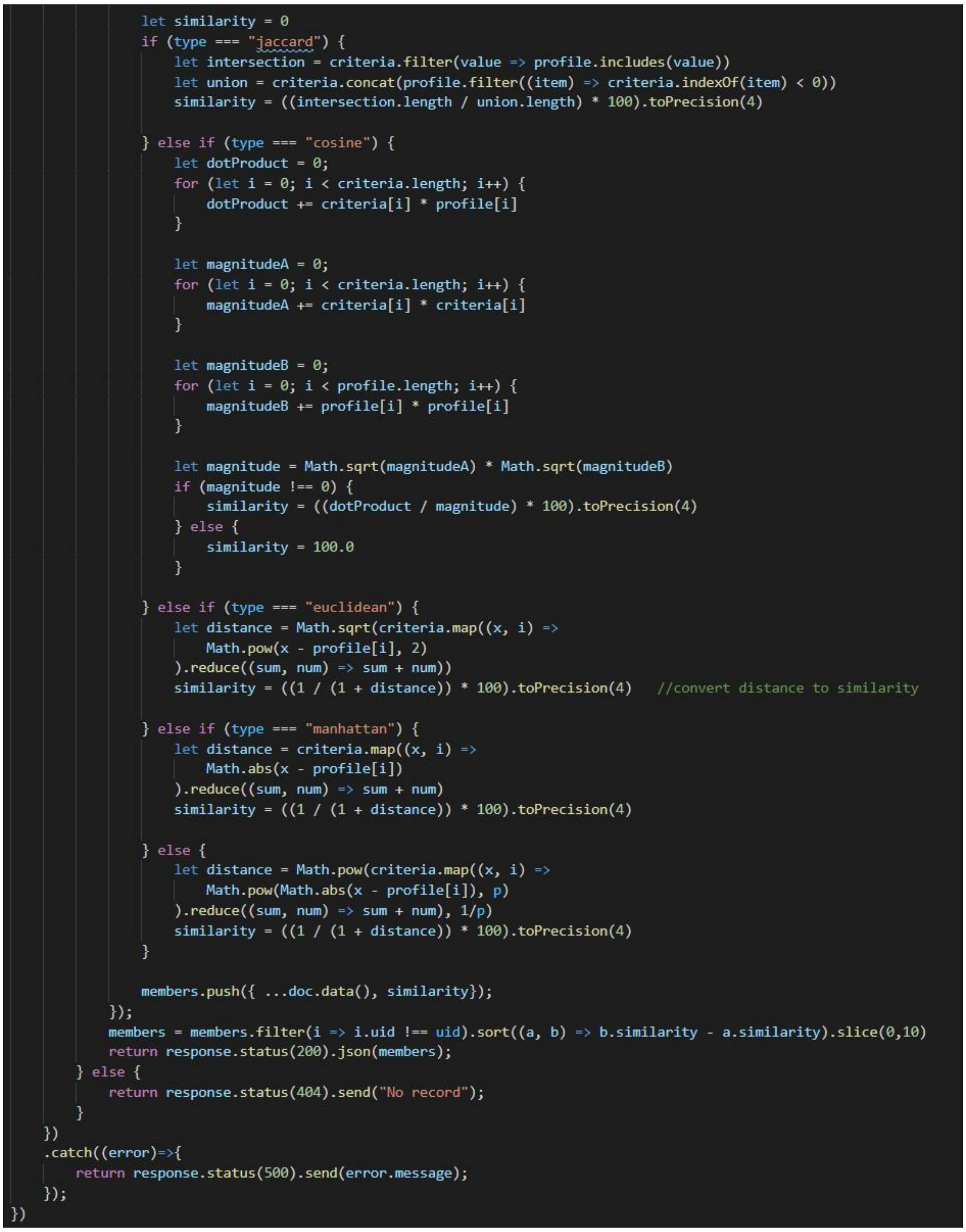
There are 6 match methods can be chosen by the users, which are Exact matching, Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance. Exact Matching is the default match method. When the users select other match methods, the application will send the match type and user preferences through a POST request to API endpoint (/api/getSimilarityResult). The server will retrieve a list of profiles from Firestore users collection for similarity measures. Normalization will be performed to scale the value of user preferences and members’ data between 0 and 1. This is to avoid the large scale variable for dominating the measure.



```
router.post('/api/getSimilarityResult', (request, response) => {

    const preferences = request.body.preferences;
    const uid = request.body.uid;
    const type = request.body.type;
    const p = request.body.index;

    admin.firestore().collection('users').where('gender', '==', parseInt(preferences.gender)).get()
    .then((docs) => {
        if (!docs.empty) {
            let members = [];
            docs.forEach(doc => {
                let profile = [];
                let criteria = [];

                if (preferences.minAge !== '0' && preferences.maxAge !== '0') {
                    if (preferences.minAge <= doc.data()['age'] && doc.data()['age'] <= preferences.maxAge) {
                        criteria.push(doc.data()['age'] / 10)
                    } else if (preferences.minAge >= doc.data()['age']) {
                        criteria.push(parseFloat(preferences.minAge) / 10)
                    } else {
                        criteria.push(parseFloat(preferences.maxAge) / 10)
                    }
                } else {
                    criteria.push(doc.data()['age'] / 10)
                }
                profile.push(doc.data()['age'] / 10)

                if (preferences.minHeight !== '0' && preferences.maxHeight !== '0') {
                    if (preferences.minHeight <= doc.data()['height'] && doc.data()['height'] <= preferences.minHeight) {
                        criteria.push(doc.data()['height'] / 100)
                    } else if (preferences.minHeight >= doc.data()['height']) {
                        criteria.push(parseFloat(preferences.minHeight) / 100)
                    } else {
                        criteria.push(parseFloat(preferences.maxHeight) / 100)
                    }
                } else {
                    criteria.push(doc.data()['height'] / 100)
                }
                profile.push(doc.data()['height'] / 100)

                preferences.state !== '0' ? criteria.push(parseFloat(preferences.state / 16))
                    : criteria.push(doc.data()['state'] / 16)
                profile.push(doc.data()['state'] / 16)

                preferences.status !== '0' ? criteria.push(parseFloat(preferences.status / 4))
                    : criteria.push(doc.data()['status'] / 4)
                profile.push(doc.data()['status'] / 4)

                preferences.posture !== '0' ? criteria.push(parseFloat(preferences.posture / 5))
                    : criteria.push(doc.data()['posture'] / 5)
                profile.push(doc.data()['posture'] / 5)

                preferences.religion !== '0' ? criteria.push(parseFloat(preferences.religion / 6))
                    : criteria.push(doc.data()['religion'] / 6)
                profile.push(doc.data()['religion'] / 6)

                preferences.tongue !== '0' ? criteria.push(parseFloat(preferences.tongue / 5))
                    : criteria.push(doc.data()['tongue'] / 5)
                profile.push(doc.data()['tongue'] / 5)

                preferences.education !== '0' ? criteria.push(parseFloat(preferences.education / 7))
                    : criteria.push(doc.data()['education'] / 7)
                profile.push(doc.data()['education'] / 7)

                preferences.field !== '0' ? criteria.push(parseFloat(preferences.field / 22))
                    : criteria.push(doc.data()['field'] / 22)
                profile.push(doc.data()['field'] / 22)

                preferences.smoke !== '0' ? criteria.push(parseFloat(preferences.smoke / 4))
                    : criteria.push(doc.data()['smoke'] / 4)
                profile.push(doc.data()['smoke'] / 4)

                preferences.childWish !== '0' ? criteria.push(parseFloat(preferences.childWish / 3))
                    : criteria.push(doc.data()['childWish'] / 3)
                profile.push(doc.data()['childWish'] / 3)
```

Figure 6.12 Section Code for Get Similarity Measures Result (Server) - Part 1

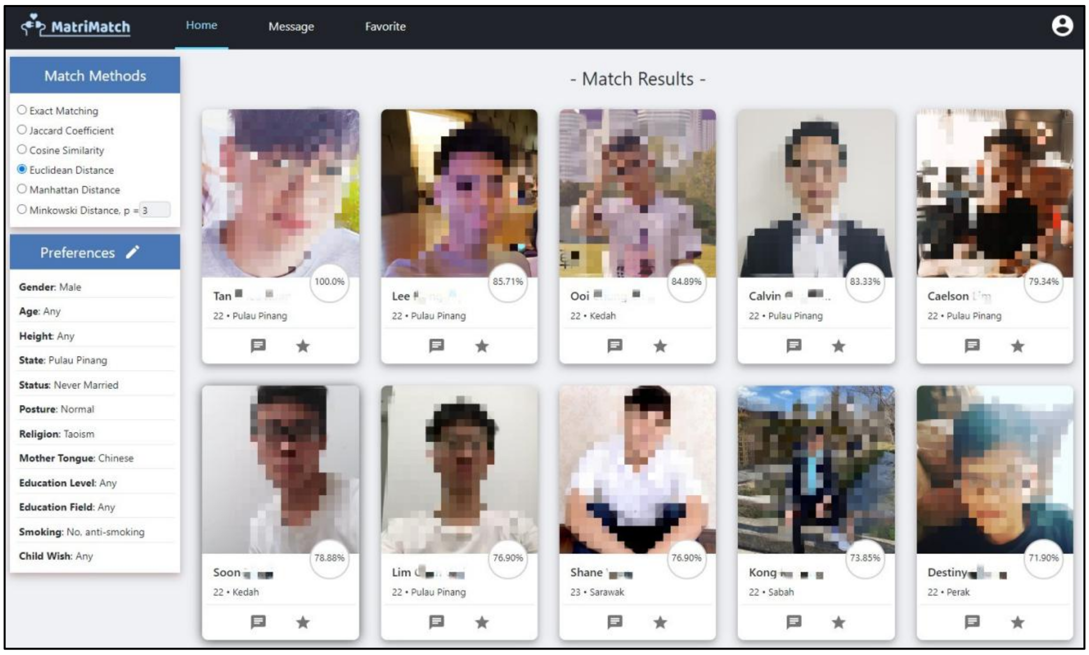
Then, the server will perform the similarity measures based on the match method selected by users. The similarity of each profile with the user's match preferences will be calculated. The server will only return the top 10 profiles with highest similarity percentages as the response to the application.



```
                let similarity = 0
                if (type === "jaccard") {
                    let intersection = criteria.filter(value => profile.includes(value))
                    let union = criteria.concat(profile.filter((item) => criteria.indexOf(item) < 0))
                    similarity = ((intersection.length / union.length) * 100).toPrecision(4)

                } else if (type === "cosine") {
                    let dotProduct = 0;
                    for (let i = 0; i < criteria.length; i++) {
                        dotProduct += criteria[i] * profile[i]
                    }

                    let magnitudeA = 0;
                    for (let i = 0; i < criteria.length; i++) {
                        magnitudeA += criteria[i] * criteria[i]
                    }

                    let magnitudeB = 0;
                    for (let i = 0; i < profile.length; i++) {
                        magnitudeB += profile[i] * profile[i]
                    }

                    let magnitude = Math.sqrt(magnitudeA) * Math.sqrt(magnitudeB)
                    if (magnitude !== 0) {
                        similarity = ((dotProduct / magnitude) * 100).toPrecision(4)
                    } else {
                        similarity = 100.0
                    }

                } else if (type === "euclidean") {
                    let distance = Math.sqrt(criteria.map((x, i) =>
                        Math.pow(x - profile[i], 2)
                    ).reduce((sum, num) => sum + num))
                    similarity = ((1 / (1 + distance)) * 100).toPrecision(4)   //convert distance to similarity

                } else if (type === "manhattan") {
                    let distance = criteria.map((x, i) =>
                        Math.abs(x - profile[i])
                    ).reduce((sum, num) => sum + num)
                    similarity = ((1 / (1 + distance)) * 100).toPrecision(4)

                } else {
                    let distance = Math.pow(criteria.map((x, i) =>
                        Math.pow(Math.abs(x - profile[i]), p)
                    ).reduce((sum, num) => sum + num), 1/p)
                    similarity = ((1 / (1 + distance)) * 100).toPrecision(4)
                }

                members.push({ ...doc.data(), similarity});
            });
            members = members.filter(i => i.uid !== uid).sort((a, b) => b.similarity - a.similarity).slice(0,10)
            return response.status(200).json(members);
        } else {
            return response.status(404).send("No record");
        }
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.13 Section Code for Get Similarity Measures Result (Server) - Part 2

When displaying the match results, the application will also display the similarity percentage beside the member's name. Since the higher percentage indicates that the member is closer to their partner preference, the similarity percentage makes it easier for the users to pick the desired partner.



Match Methods

- Exact Matching
- Jaccard Coefficient
- Cosine Similarity
- [✓ Euclidean Distance]
- Manhattan Distance
- Minkowski Distance, p = 3

Preferences

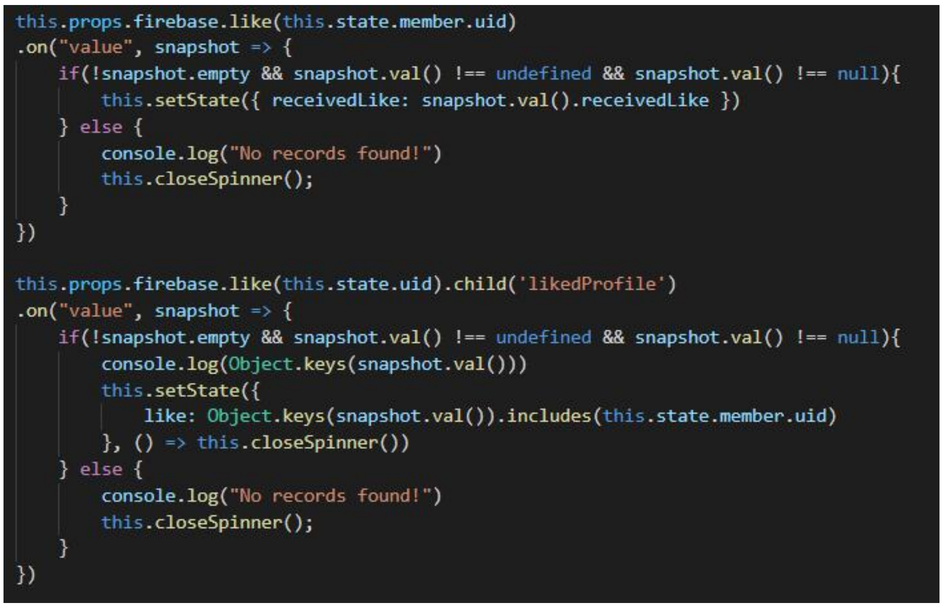
Gender: Male
Age: Any
Height: Any
State: Pulau Pinang
Status: Never Married
Posture: Normal
Religion: Taoism
Mother Tongue: Chinese
Education Level: Any
Education Field: Any
Smoking: No, anti-smoking
Child Wish: Any

Tan [redacted] 100.0% 22 • Pulau Pinang
Lee [redacted] 85.71% 22 • Pulau Pinang
Ooi [redacted] 84.89% 22 • Kedah
Calvin [redacted] 83.33% 22 • Pulau Pinang
Caelson [redacted] 79.34% 22 • Pulau Pinang
Soon [redacted] 78.88% 22 • Kedah
Lim [redacted] 76.90% 22 • Pulau Pinang
Shane [redacted] 76.90% 23 • Sarawak
Kong [redacted] 73.85% 22 • Sabah
Destiny [redacted] 71.90% 22 • Perak

Figure 6.14 Match Result Page (Similarity Measures)

6.2.5 View Member Profile

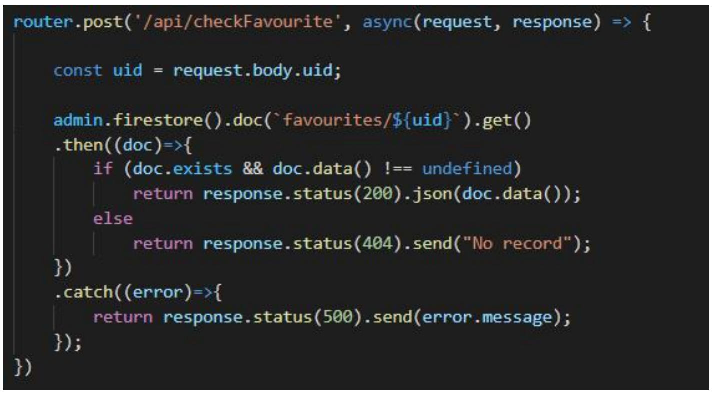
When users click on one of the profiles, they can view the member's detailed information. Before displaying the profile information, the application will retrieve the number of likes received by the profile from Firebase Real-time Database. The users can know how many people liked the profile. The number of likes will indicate the popularity of the profile. The application will also check if the users have liked the current profile by retrieving a list of liked profiles from Firebase Real-time Database. If the list contains the profile, the application will display a filled "Love" icon. Else an outline "Love" icon will be shown.



```
this.props.firebase.like(this.state.member.uid)
.on("value", snapshot => {
    if(!snapshot.empty && snapshot.val() !== undefined && snapshot.val() !== null){
        this.setState({ receivedLike: snapshot.val().receivedLike })
    } else {
        console.log("No records found!")
        this.closeSpinner();
    }
})

this.props.firebase.like(this.state.uid).child('likedProfile')
.on("value", snapshot => {
    if(!snapshot.empty && snapshot.val() !== undefined && snapshot.val() !== null){
        console.log(Object.keys(snapshot.val()))
        this.setState({
            like: Object.keys(snapshot.val()).includes(this.state.member.uid)
        }, () => this.closeSpinner())
    } else {
        console.log("No records found!")
        this.closeSpinner();
    }
})
```

Figure 6.15 Section Code for Check Like Status (Front-end)

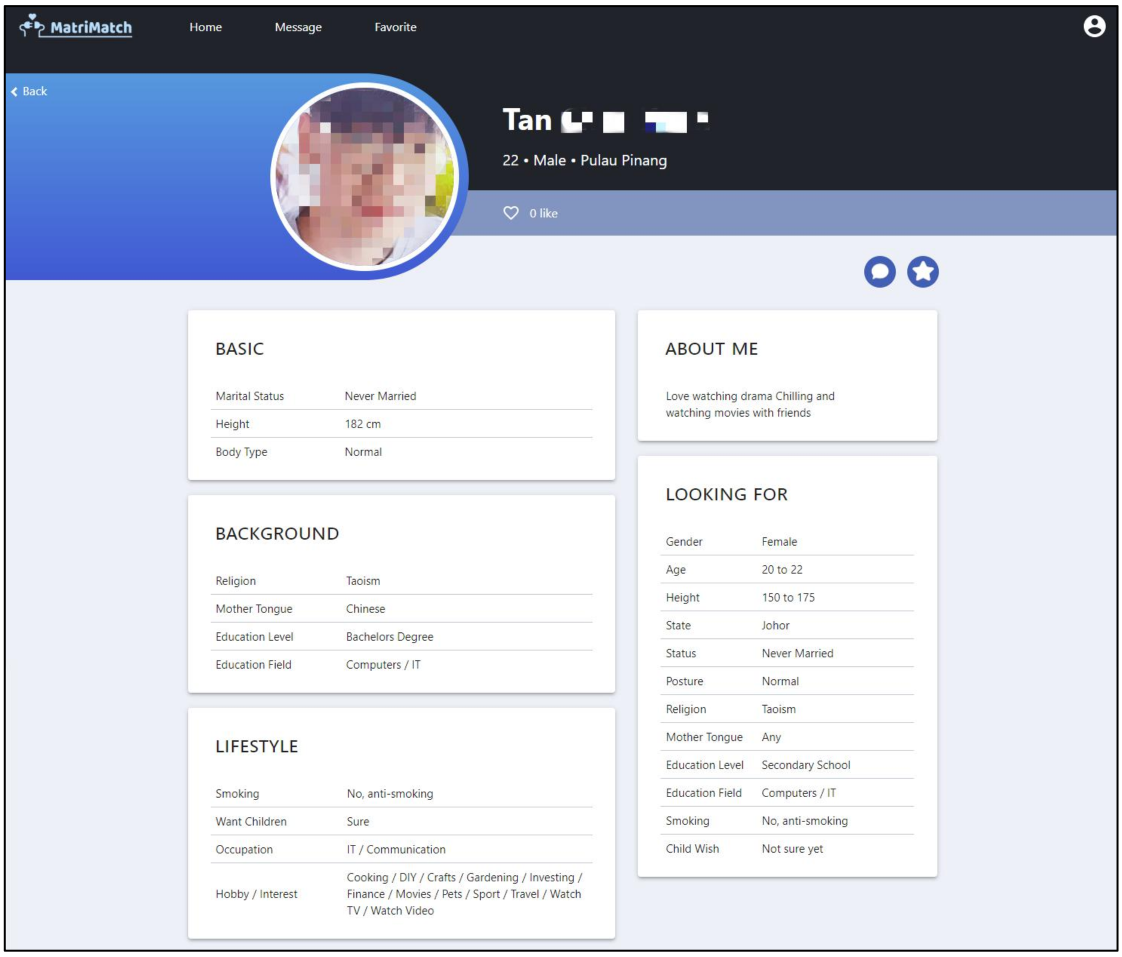
Furthermore, the application will check whether the current profile was added to favourites. The application will send a POST request to API endpoint (/api/checkFavourite) on the server. The server will return the favourite list retrieved from Firestore favourites collection as the response.



```
router.post('/api/checkFavourite', async(request, response) => {

    const uid = request.body.uid;

    admin.firestore().doc(`favourites/${uid}`).get()
    .then((doc)=>{
        if (doc.exists && doc.data() !== undefined)
            return response.status(200).json(doc.data());
        else
            return response.status(404).send("No record");
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.16 Section Code for Check Favourite (Server)

If the current profile exists in the favourite list, the “Favourite” button will be displayed as pink colour instead of white colour. Besides, if the users search the matches by similarity measures, the application will display the similarity percentage in the profile as Figure 6.17. The display of similarity percentage will make the user easier to select the desired partner when viewing the profile information.


MatriMatch Home Message Favorite

Back

Tan [redacted]

22 • Male • Pulau Pinang

0 like

BASIC

Marital Status	Never Married
Height	182 cm
Body Type	Normal

BACKGROUND

Religion	Taoism
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Computers / IT

LIFESTYLE

Smoking	No, anti-smoking
Want Children	Sure
Occupation	IT / Communication
Hobby / Interest	Cooking / DIY / Crafts / Gardening / Investing / Finance / Movies / Pets / Sport / Travel / Watch TV / Watch Video

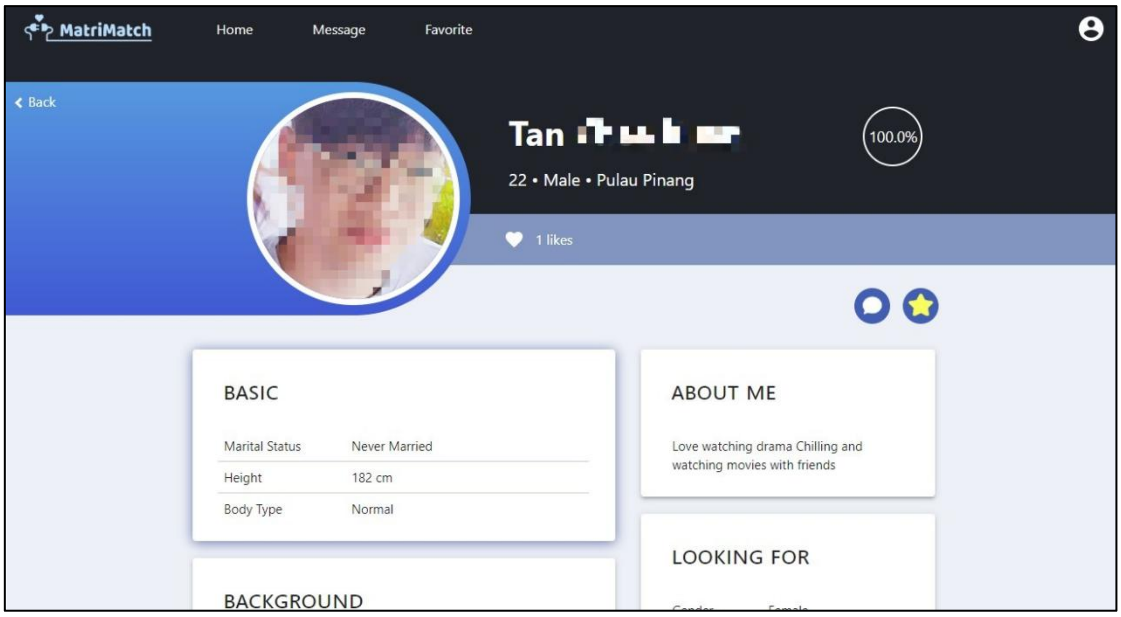
ABOUT ME

Love watching drama Chilling and watching movies with friends

LOOKING FOR

Gender	Female
Age	20 to 22
Height	150 to 175
State	Johor
Status	Never Married
Posture	Normal
Religion	Taoism
Mother Tongue	Any
Education Level	Secondary School
Education Field	Computers / IT
Smoking	No, anti-smoking
Child Wish	Not sure yet

Figure 6.17 Member Profile Page (Without Similarity Measure)


MatriMatch Home Message Favorite

Back

Tan [redacted]

100.0%

22 • Male • Pulau Pinang

1 likes

BASIC

Marital Status	Never Married
Height	182 cm
Body Type	Normal

BACKGROUND

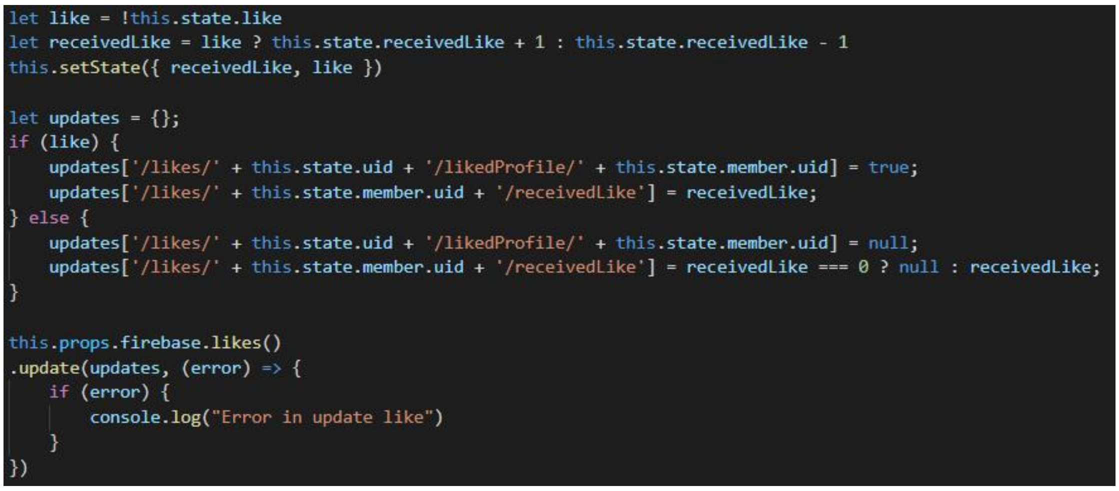
ABOUT ME

Love watching drama Chilling and watching movies with friends

LOOKING FOR

Figure 6.18 Member Profile Page (With Similarity Measure)

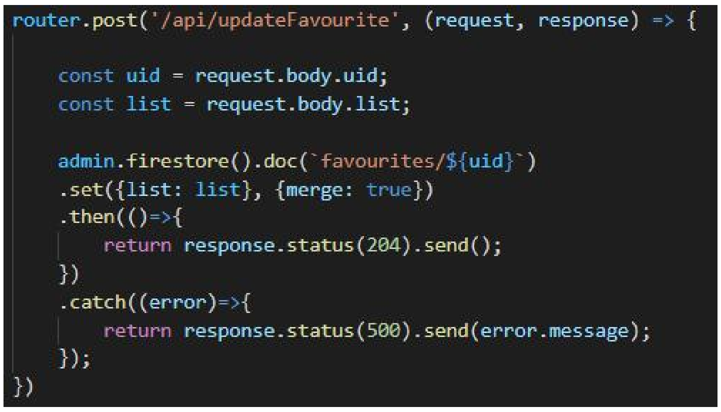
If the users wish to like/unlike the profile, they can click the “Love” button. The application will update the like status in Firebase Real-time Database. If the user likes the profile, the number of likes will be increased. On the other hand, if the user unlike the profile, the number of likes will be decreased.



```
let like = !this.state.like
let receivedLike = like ? this.state.receivedLike + 1 : this.state.receivedLike - 1
this.setState({ receivedLike, like })

let updates = {};
if (like) {
    updates['/likes/' + this.state.uid + '/likedProfile/' + this.state.member.uid] = true;
    updates['/likes/' + this.state.member.uid + '/receivedLike'] = receivedLike;
} else {
    updates['/likes/' + this.state.uid + '/likedProfile/' + this.state.member.uid] = null;
    updates['/likes/' + this.state.member.uid + '/receivedLike'] = receivedLike === 0 ? null : receivedLike;
}

this.props.firebase.likes()
.update(updates, (error) => {
    if (error) {
        console.log("Error in update like")
    }
})
```

Figure 6.19 Section Code for Update Like Status (Front-end)

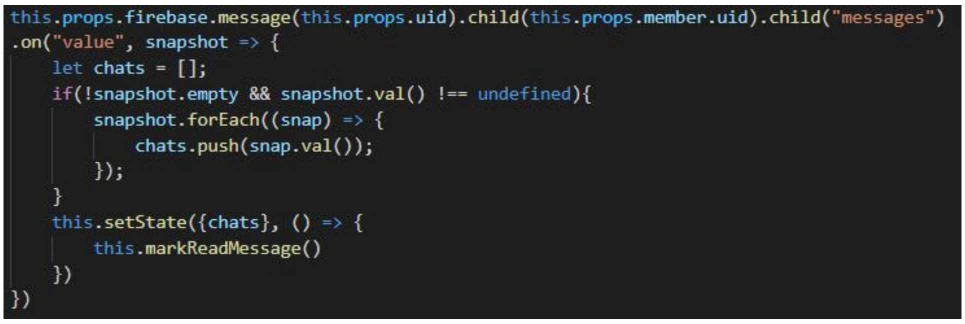
If the users wish to add the profile to favourite or remove the profile from favourite, they can click the “Favourite” button. The application will send a POST request to API endpoint (/api/updateFavourite) on the server. The server will update the favourite list in Firestore favourites collection. After that, the users can check the favourite list on the Favourite List Page.



```
router.post('/api/updateFavourite', (request, response) => {

    const uid = request.body.uid;
    const list = request.body.list;

    admin.firestore().doc(`favourites/${uid}`)
    .set({list: list}, {merge: true})
    .then(()=>{
        return response.status(204).send();
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

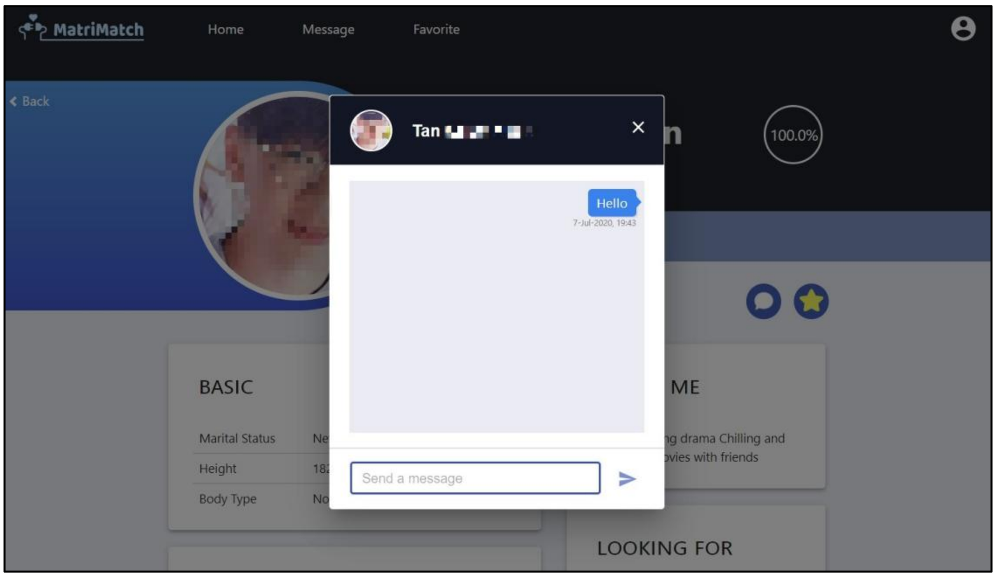
Figure 6.20 Section Code for Update Favourite (Server)

If the users wish to send a message to the profile, they can click the “Chat” button. A pop-up window will be displayed. The application will retrieve any chat history between the user and the target profile from Firebase Real-time Database. The users can view chat history through the window.



```
this.props.firebase.message(this.props.uid).child(this.props.member.uid).child("messages")
.on("value", snapshot => {
    let chats = [];
    if(!snapshot.empty && snapshot.val() !== undefined){
        snapshot.forEach((snap) => {
            chats.push(snap.val());
        });
    }
    this.setState({chats}, () => {
        this.markReadMessage()
    })
})
```

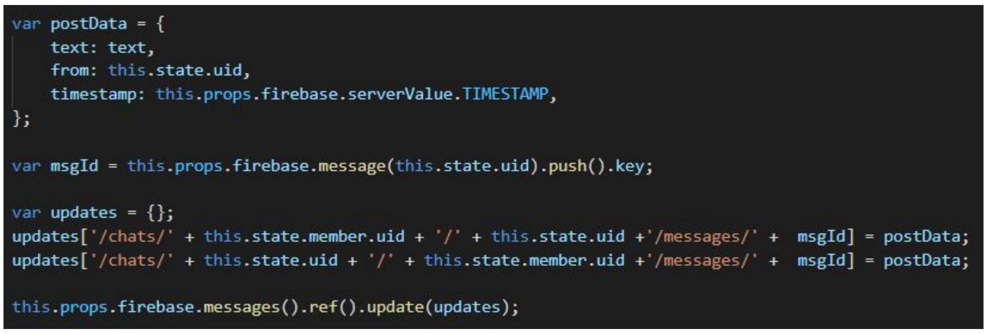
Figure 6.21 Section Code for Load Message (Front-end)



Hello
7-Jul-2020, 19:43

Figure 6.22 Instant Message Pop-up Window

After the users enter the message and click “Send” button, the application will update the messages in Firebase Real-time Database. The message sent will be synced with the receiver. Hence, the receiver can receive the new message within milliseconds.

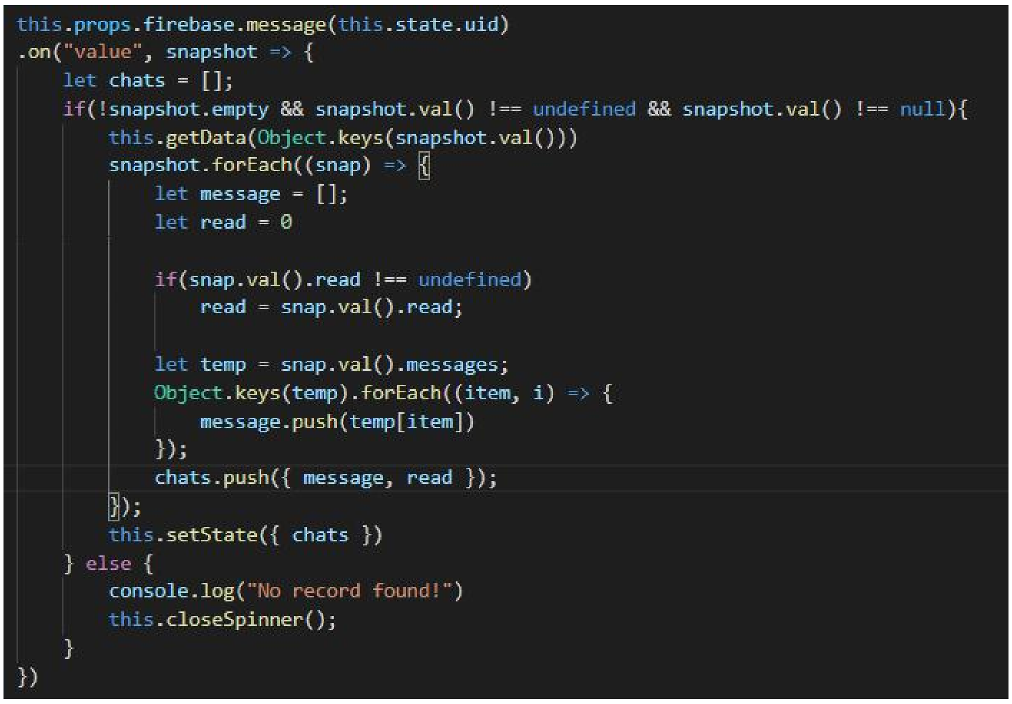


```
var postData = {
    text: text,
    from: this.state.uid,
    timestamp: this.props.firebase.serverValue.TIMESTAMP,
};

var msgId = this.props.firebase.message(this.state.uid).push().key;

var updates = {};
updates['/chats/' + this.state.member.uid + '/' + this.state.uid +'/messages/' +  msgId] = postData;
updates['/chats/' + this.state.uid + '/' + this.state.member.uid +'/messages/' +  msgId] = postData;

this.props.firebase.messages().ref().update(updates);
```

Figure 6.23 Section Code for Send Message (Front-end)

6.2.6 View Chat Message

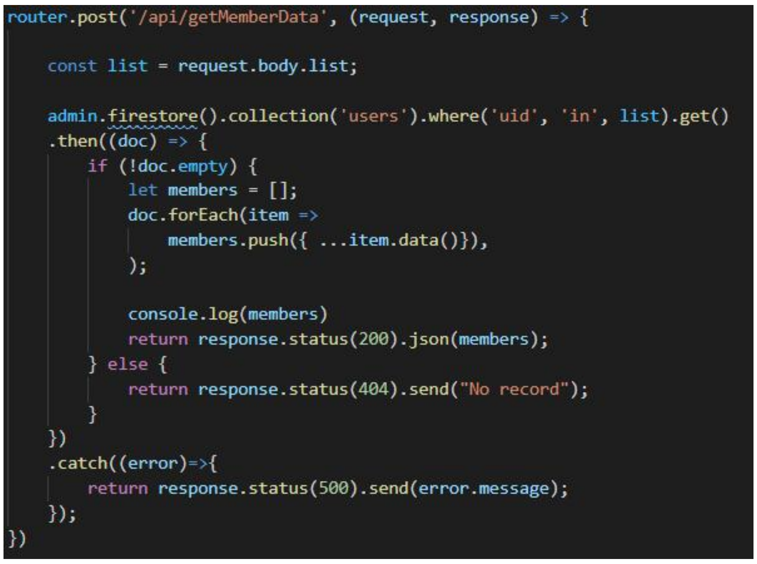
When the users click the “Message” option on the navigation bar, they will be redirected to the Chat List Page. The application will get the messages from Firebase Real-time Database. The read attribute is used to detect the unread messages. If there is any unread message, a red badge will be displayed to inform the users about the unread message.



```
this.props.firebase.message(this.state.uid)
.on("value", snapshot => {
    let chats = [];
    if(!snapshot.empty && snapshot.val() !== undefined && snapshot.val() !== null){
        this.getData(Object.keys(snapshot.val()))
        snapshot.forEach((snap) => {
            let message = [];
            let read = 0

            if(snap.val().read !== undefined)
                read = snap.val().read;

            let temp = snap.val().messages;
            Object.keys(temp).forEach((item, i) => {
                message.push(temp[item])
            });
            chats.push({ message, read });
        });
        this.setState({ chats })
    } else {
        console.log("No record found!")
        this.closeSpinner();
    }
})
```

Figure 6.24 Section Code for Get Chat Messages (Front-end)

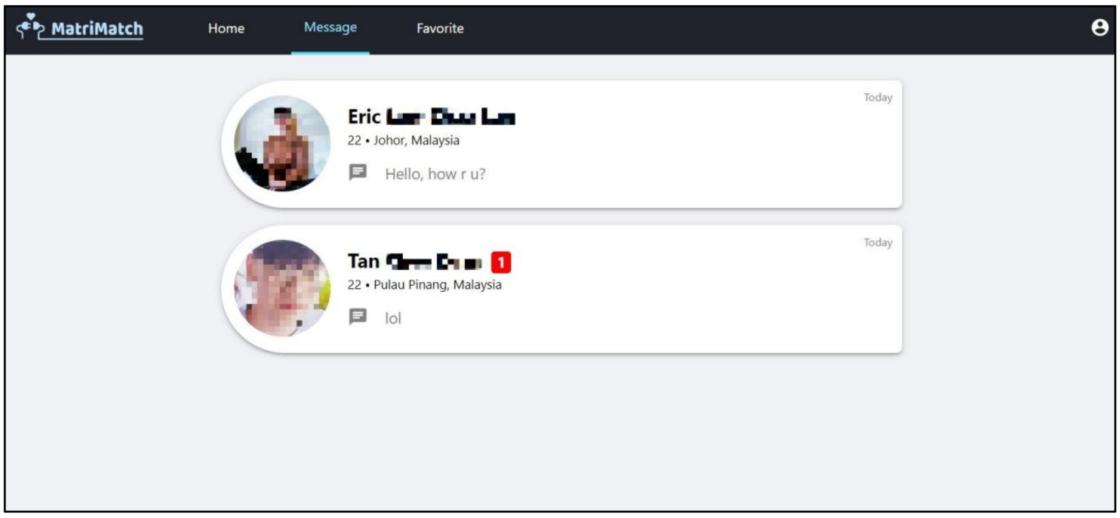
The application will also send a POST request to API endpoint (/api/getMemberData) on the server to get receiver’s data from Firestore. The server will retrieve the requested user data from users collection and return it as a response to the application.



```
router.post('/api/getMemberData', (request, response) => {

    const list = request.body.list;

    admin.firestore().collection('users').where('uid', 'in', list).get()
    .then((doc) => {
        if (!doc.empty) {
            let members = [];
            doc.forEach(item =>
                members.push({ ...item.data()}),
            );

            console.log(members)
            return response.status(200).json(members);
        } else {
            return response.status(404).send("No record");
        }
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.25 Section Code for Get Member Data (Server)

The chat list will display the most recent conversations between users and the receivers. The chat list will also display some information of the receivers such as name, age and living state.



Eric [redacted]
22 • Johor, Malaysia
Hello, how r u?
Today

Tan [redacted] 1
22 • Pulau Pinang, Malaysia
lol
Today

Figure 6.26 Chat List Page

The users can click into one of the chat to view the chat history or send messages. If the users wish to delete the chat history, they can click the “Delete” button to clear the whole chat history. There is also a button next to the “Delete” button which allows the users to view the receiver’s profile.

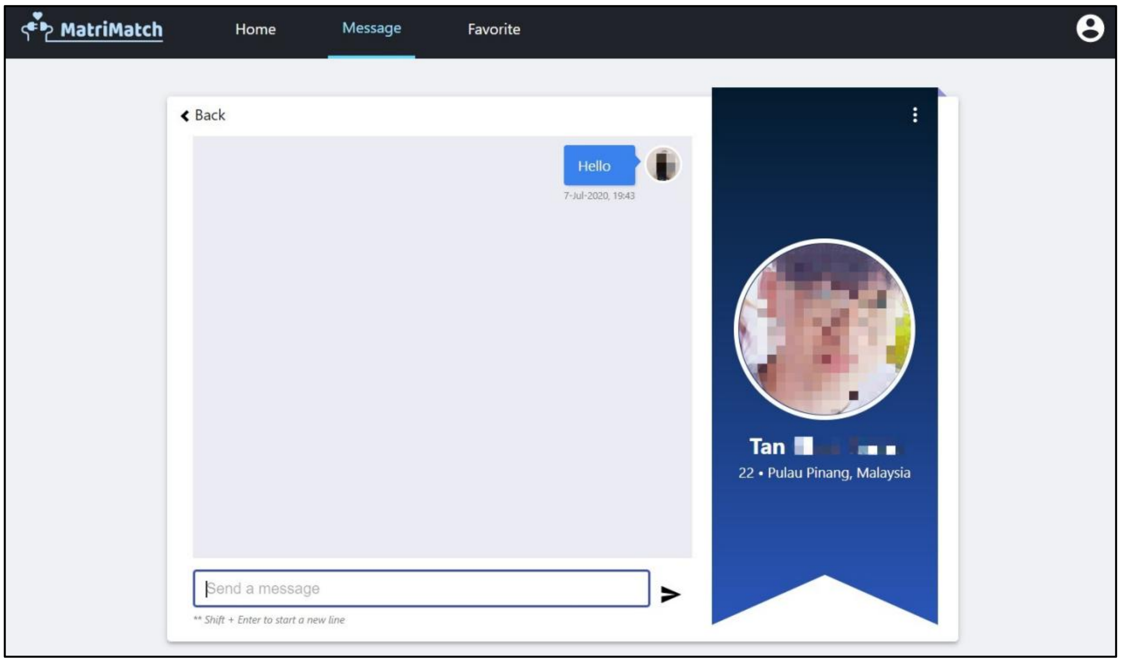
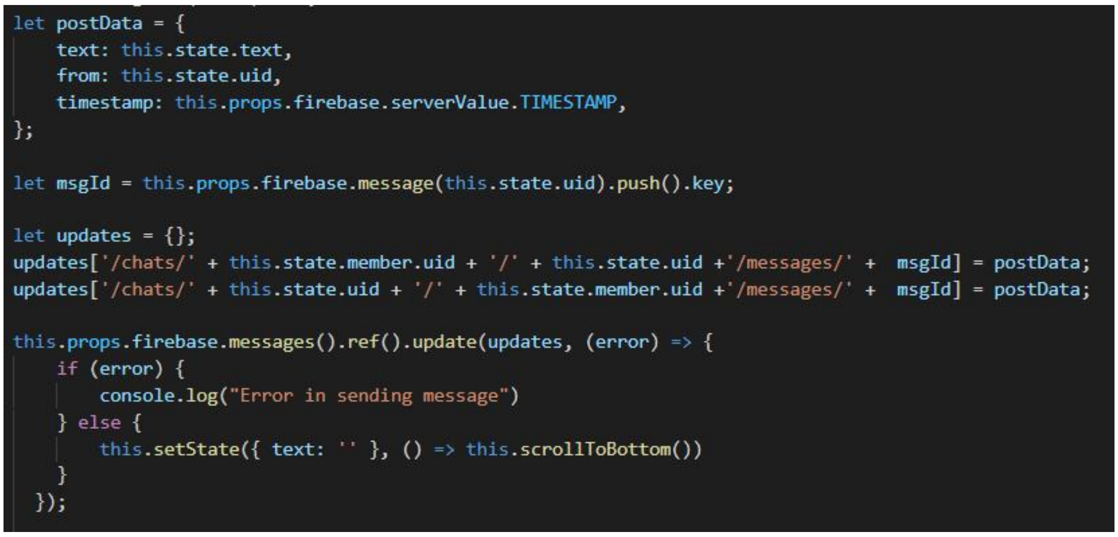


Figure 6.27 Individual Chat History Page

When a user sends a message, the application will update the messages in Firebase Real-time Database. The message sent will be synced with the receiver. Hence, the receiver can receive the new message within milliseconds.



```
let postData = {
    text: this.state.text,
    from: this.state.uid,
    timestamp: this.props.firebase.serverValue.TIMESTAMP,
};

let msgId = this.props.firebase.message(this.state.uid).push().key;

let updates = {};
updates['/chats/' + this.state.member.uid + '/' + this.state.uid +'/messages/' +  msgId] = postData;
updates['/chats/' + this.state.uid + '/' + this.state.member.uid +'/messages/' +  msgId] = postData;

this.props.firebase.messages().ref().update(updates, (error) => {
    if (error) {
        console.log("Error in sending message")
    } else {
        this.setState({ text: '' }, () => this.scrollToBottom())
    }
  });
```

Figure 6.28 Section Code for Send Message (Front-end)

When the users click the "Menu" button with three dots on the top-right, they can choose to view the member's profile or delete the chat history.

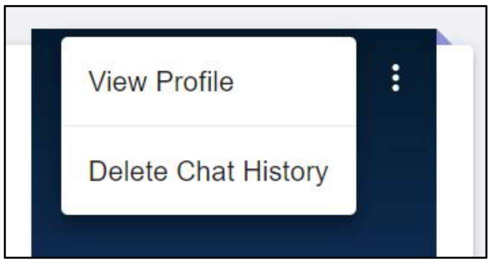
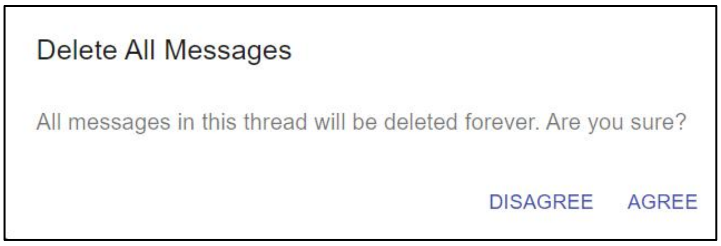


Figure 6.29 Menu Option

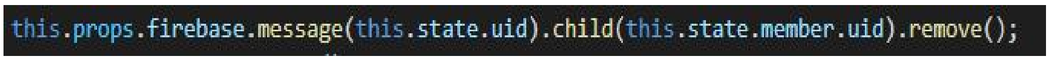
If the users choose to delete the chat history, a delete confirmation message will be pop-up. Once the users choose "Agree", the chat history will be removed permanently from the Firebase Real-time Database. However, the users will only delete their records. The receivers will still be able to view the chat history.



Delete All Messages

All messages in this thread will be deleted forever. Are you sure?

Figure 6.30 Chat History Delete Confirmation Message

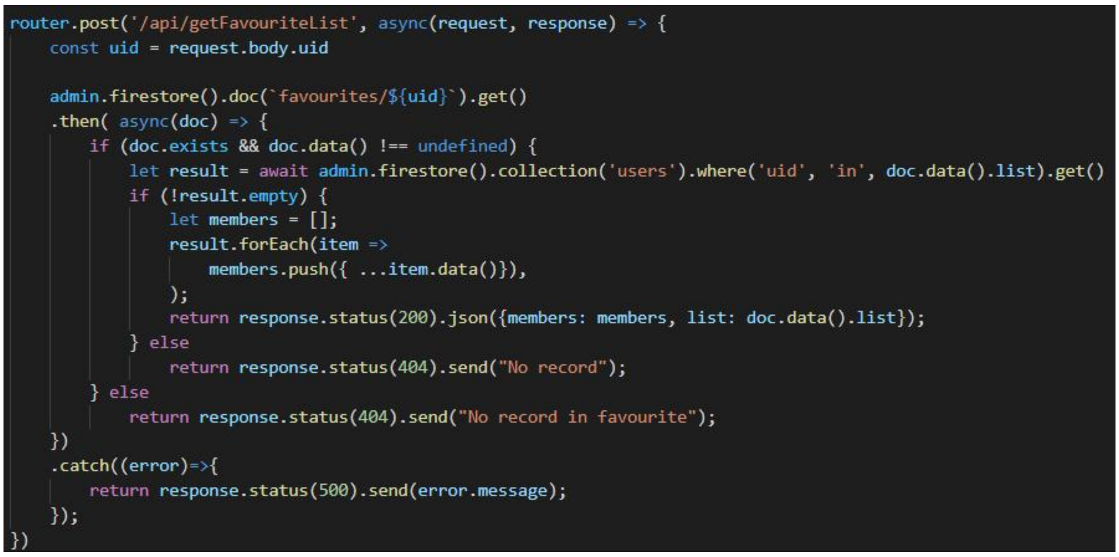


```
this.props.firebase.message(this.state.uid).child(this.state.member.uid).remove();
```

Figure 6.31 Section Code for Delete Chat History (Front-end)

6.2.7 View Favourite List

When the users click the "Favourite" option on the navigation bar, they will be redirected to Favourite List Page. The application will send a POST request to API endpoint (/api/getFavouriteList) on the server. The server will retrieve the favourite list from Firestore favourites collection. Then, the server will retrieve the member data according to the list. The favourite list and the data of the members on the list will be returned to the application.



```
router.post('/api/getFavouriteList', async(request, response) => {
    const uid = request.body.uid

    admin.firestore().doc(`favourites/${uid}`).get()
    .then( async(doc) => {
        if (doc.exists && doc.data() !== undefined) {
            let result = await admin.firestore().collection('users').where('uid', 'in', doc.data().list).get()
            if (!result.empty) {
                let members = [];
                result.forEach(item =>
                    members.push({ ...item.data()}),
                );
                return response.status(200).json({members: members, list: doc.data().list});
            } else
                return response.status(404).send("No record");
        } else
            return response.status(404).send("No record in favourite");
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.32 Section Code for Get Favourite List (Server)

The result of the favourite list will be displayed in the Favourite List Page (Figure 6.27). The users can click on one of the profile to view the detailed profile information. The users can also delete the unwanted profiles from the favourite list.

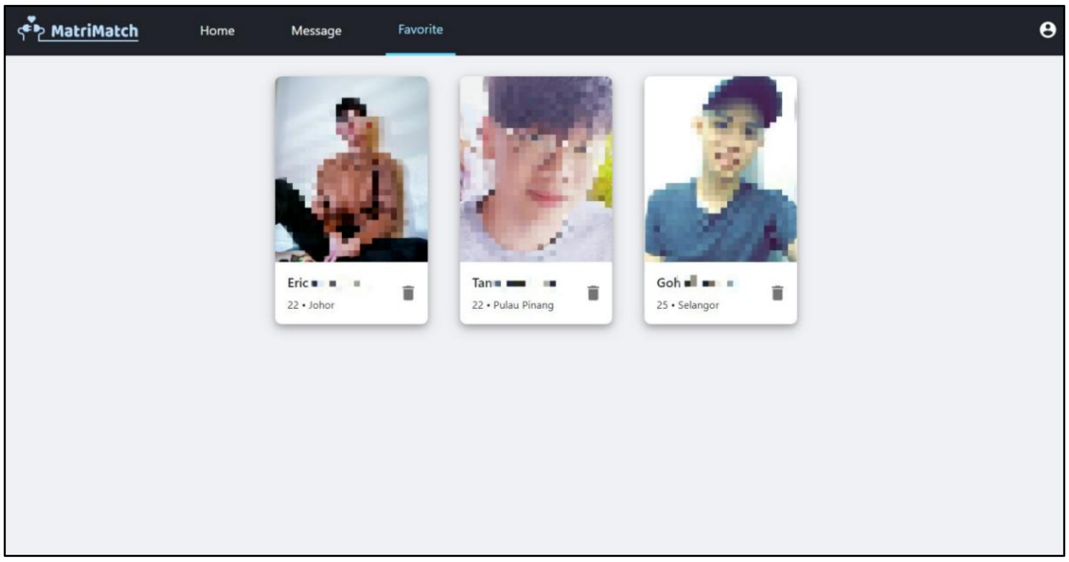
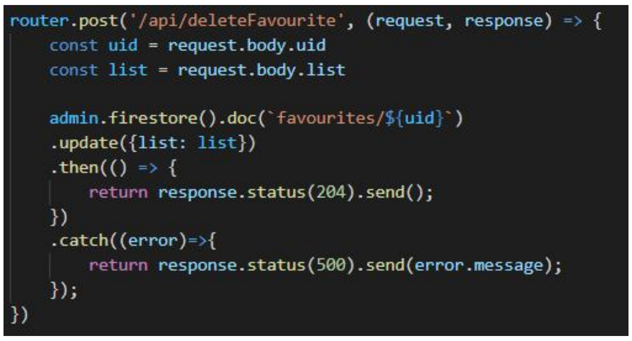


Figure 6.33 Favourite List Page

If the “Delete” button is clicked, the selected profile will be removed from the favourite list. The application will send a POST request to the API endpoint (/api/deleteFavourite) on the server. The server will update the favourite list in Firestore favourite collection to remove the unwanted profiles.

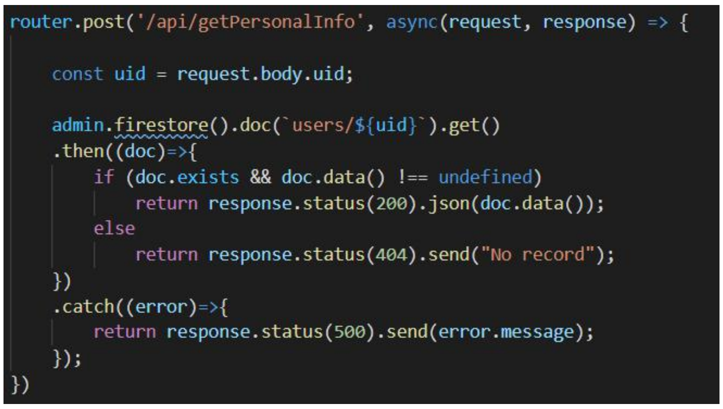


```
router.post('/api/deleteFavourite', (request, response) => {
    const uid = request.body.uid
    const list = request.body.list

    admin.firestore().doc(`favourites/${uid}`)
    .update({list: list})
    .then(() => {
        return response.status(204).send();
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.34 Section Code for Delete Profile from Favourite (Server)

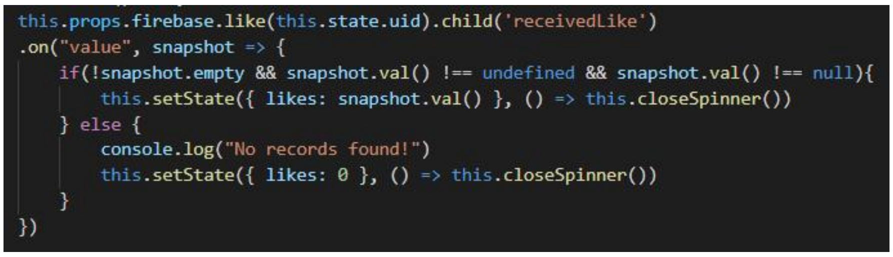
6.2.8 View Personal Profiles

When the users click the button on the top-right of the navigation, they can choose to view their profile. The application will send a POST request to API endpoint (/api/getProfileDetails) on the server to retrieve their profile information. The user data will be retrieved from Firestore users collection and returned as a response to the application. The response will then be displayed in the Personal Profile Page. Besides, the application will display the number of likes retrieved from Firebase Real-time Database. The users are able to know how many people liked their profiles.



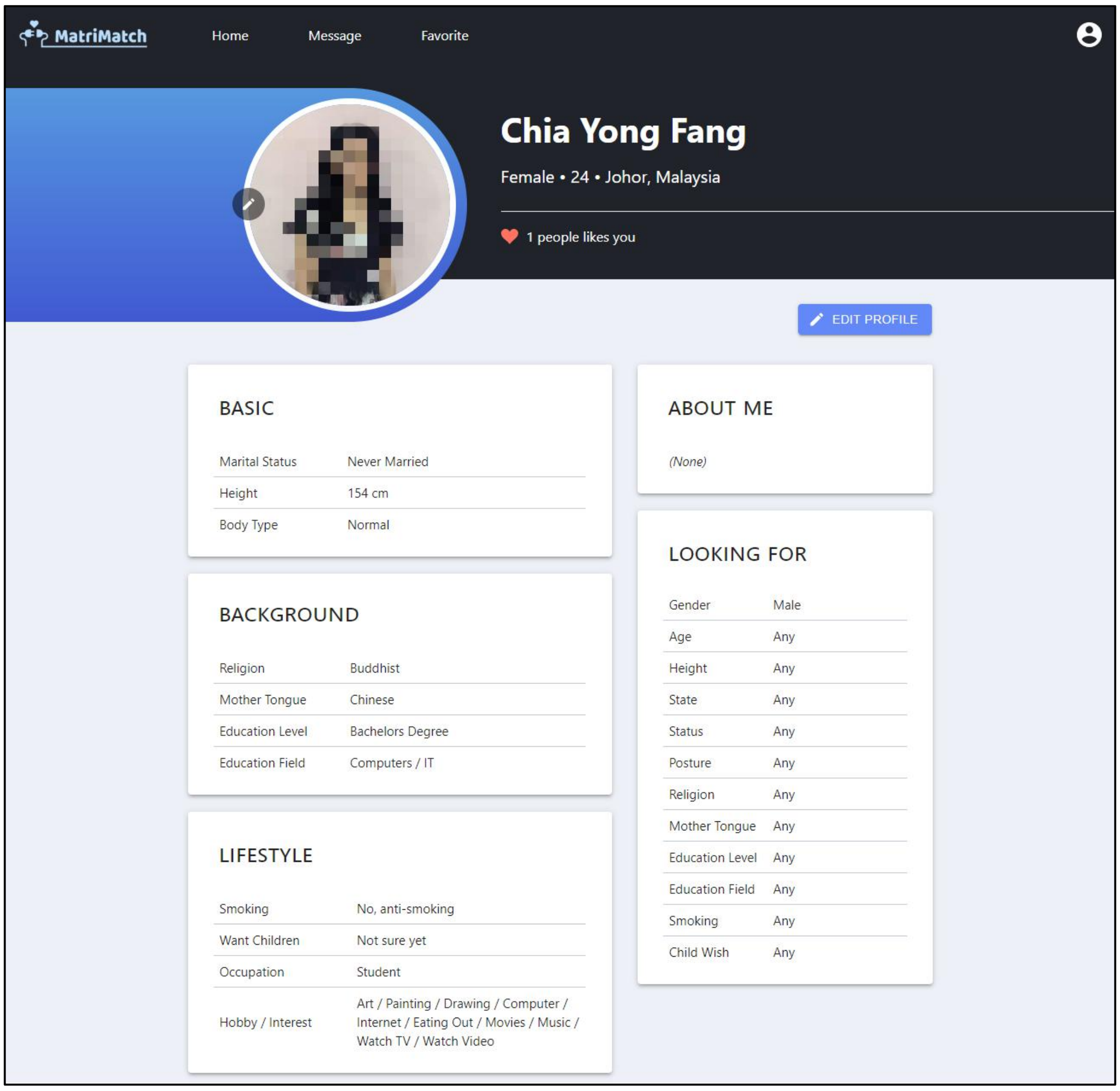
```
router.post('/api/getPersonalInfo', async(request, response) => {

    const uid = request.body.uid;

    admin.firestore().doc(`users/${uid}`).get()
    .then((doc)=>{
        if (doc.exists && doc.data() !== undefined)
            return response.status(200).json(doc.data());
        else
            return response.status(404).send("No record");
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.35 Section Code for Get Profile Details (Server)



```
this.props.firebase.like(this.state.uid).child('receivedLike')
.on("value", snapshot => {
    if(!snapshot.empty && snapshot.val() !== undefined && snapshot.val() !== null){
        this.setState({ likes: snapshot.val() }, () => this.closeSpinner())
    } else {
        console.log("No records found!")
        this.setState({ likes: 0 }, () => this.closeSpinner())
    }
})
```

Figure 6.36 Section Code for Get Number of Likes (Front-end)



Chia Yong Fang

Female • 24 • Johor, Malaysia

1 people likes you

BASIC

Marital Status	Never Married
Height	154 cm
Body Type	Normal

BACKGROUND

Religion	Buddhist
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Computers / IT

LIFESTYLE

Smoking	No, anti-smoking
Want Children	Not sure yet
Occupation	Student
Hobby / Interest	Art / Painting / Drawing / Computer / Internet / Eating Out / Movies / Music / Watch TV / Watch Video

ABOUT ME

(None)

LOOKING FOR

Gender	Male
Age	Any
Height	Any
State	Any
Status	Any
Posture	Any
Religion	Any
Mother Tongue	Any
Education Level	Any
Education Field	Any
Smoking	Any
Child Wish	Any

Figure 6.37 Personal Profile Page

If the users wish to change their profile pictures, they can click the “Edit” button beside the profile picture. A small window with the user profile picture will pop-up. The users can change the profile picture by uploading a new photo through the “Upload Photo” button.

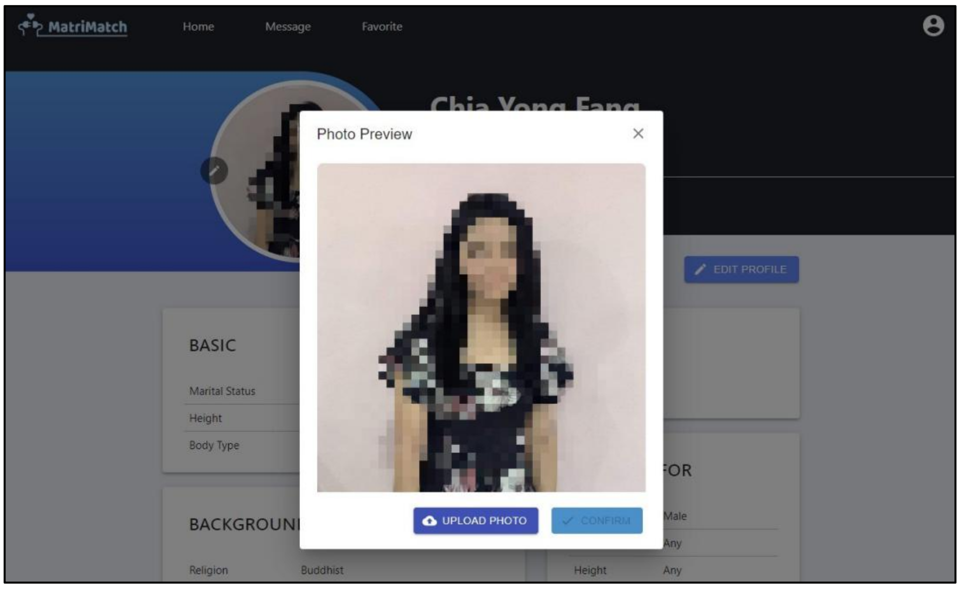
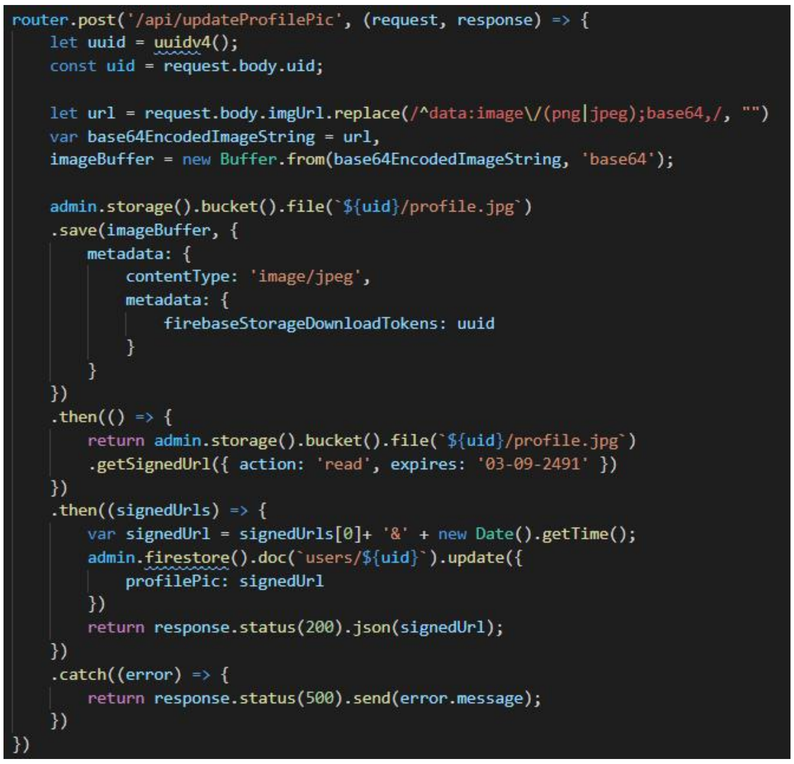


Figure 6.38 Pop-up for Update Profile Picture

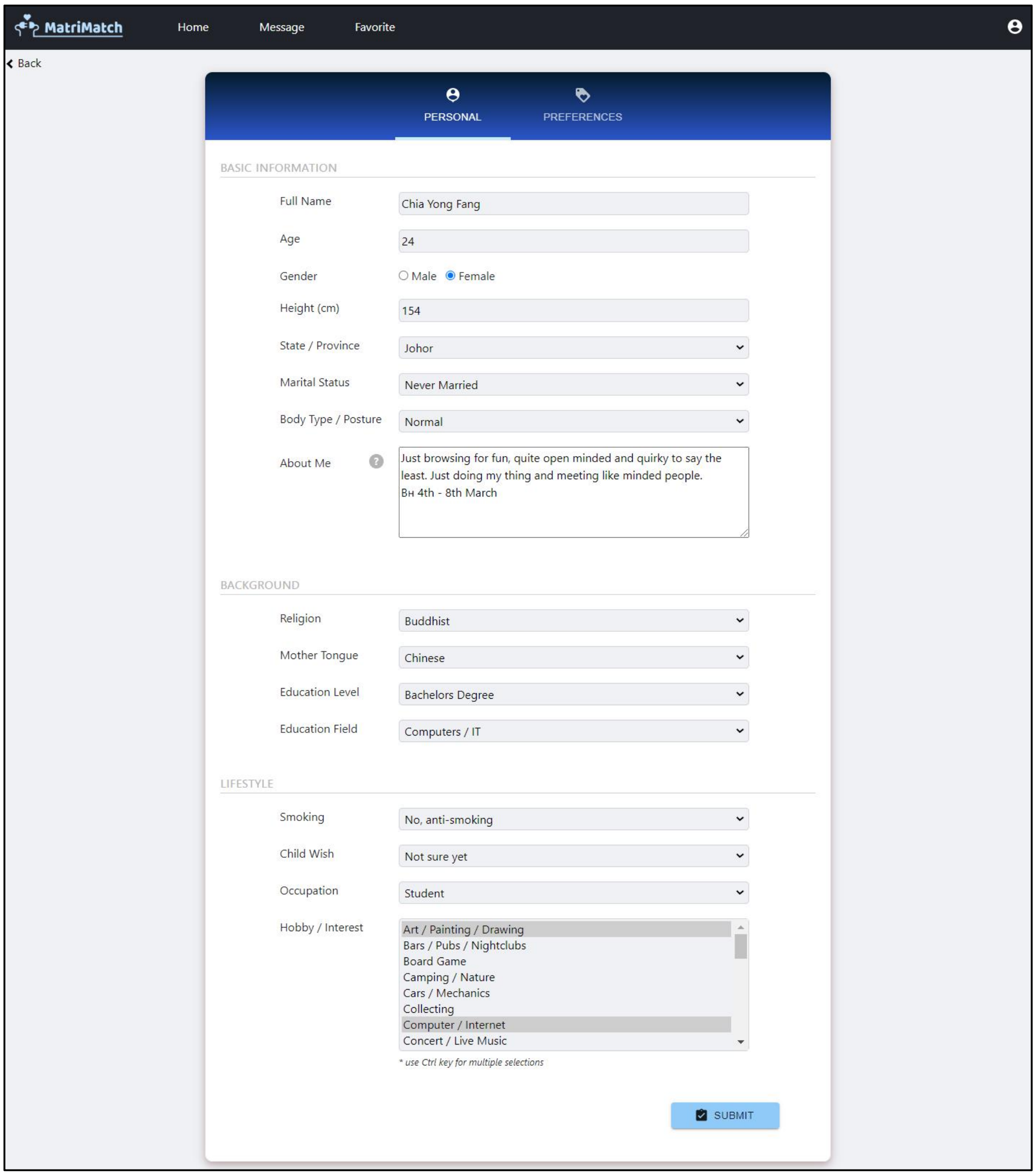
When the users click the "Confirm" button, the application will send a POST request to API endpoint (/api/updateProfilePic) on the server. The server will update the image stored in Firebase Cloud Storage. Then, the server will update profile picture attribute in Firestore users collection with the image URL that retrieved from Firebase Cloud Storage.



```
router.post('/api/updateProfilePic', (request, response) => {
    let uuid = uuidv4();
    const uid = request.body.uid;

    let url = request.body.imgUrl.replace(/^data:image\/(png|jpeg);base64,/, "")
    var base64EncodedImageString = url,
    imageBuffer = new Buffer.from(base64EncodedImageString, 'base64');

    admin.storage().bucket().file(`${uid}/profile.jpg`)
    .save(imageBuffer, {
        metadata: {
            contentType: 'image/jpeg',
            metadata: {
                firebaseStorageDownloadTokens: uuid
            }
        }
    })
    .then(() => {
        return admin.storage().bucket().file(`${uid}/profile.jpg`)
        .getSignedUrl({ action: 'read', expires: '03-09-2491' })
    })
    .then((signedUrls) => {
        var signedUrl = signedUrls[0]+ '&' + new Date().getTime();
        admin.firestore().doc(`users/${uid}`).update({
            profilePic: signedUrl
        })
        return response.status(200).json(signedUrl);
    })
    .catch((error) => {
        return response.status(500).send(error.message);
    })
})
```

Figure 6.39 Section Code for Update Profile Picture (Server)

If the users wish to change their profile information, they can click the “Edit” button beside their name (Figure 6.30). The users will be redirected to the Edit Page. They can choose to edit their personal information or their match preferences.



BASIC INFORMATION

Full Name	Chia Yong Fang
Age	24
Gender	Male [✓ Female]
Height (cm)	154
State / Province	Johor
Marital Status	Never Married
Body Type / Posture	Normal
About Me	Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded people. BH 4th - 8th March

BACKGROUND

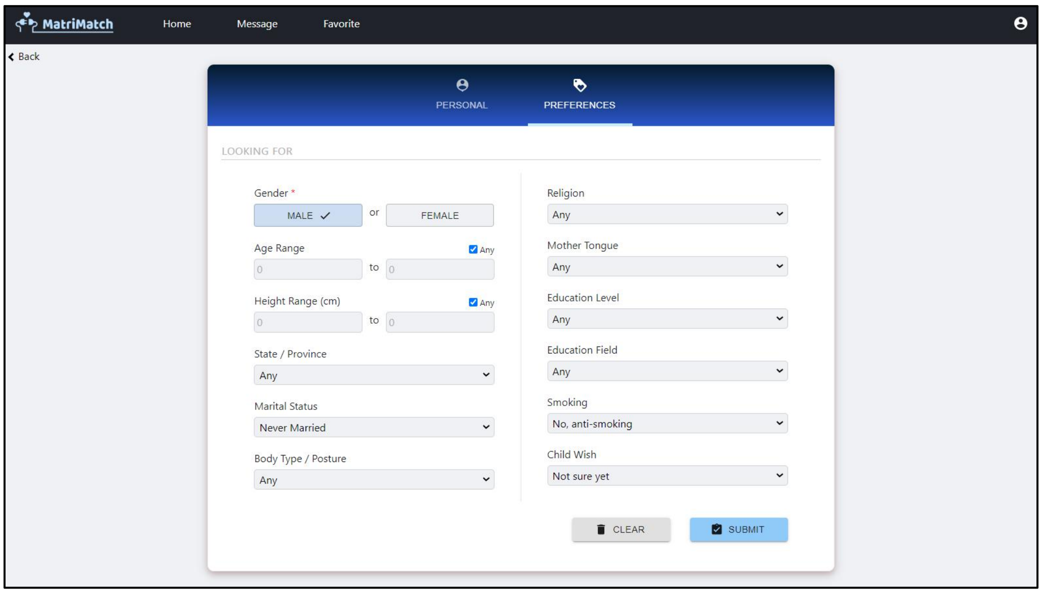
Religion	Buddhist
Mother Tongue	Chinese
Education Level	Bachelors Degree
Education Field	Computers / IT

LIFESTYLE

Smoking	No, anti-smoking
Child Wish	Not sure yet
Occupation	Student
Hobby / Interest	Art / Painting / Drawing; Bars / Pubs / Nightclubs; Board Game; Camping / Nature; Cars / Mechanics; Collecting; Computer / Internet; Concert / Live Music

* use Ctrl key for multiple selections

Figure 6.40 Edit Personal Information Page



LOOKING FOR

Gender *: MALE ✓ or FEMALE

Age Range: 0 to 0 ✓ Any

Height Range (cm): 0 to 0 ✓ Any

State / Province: Any

Marital Status: Never Married

Body Type / Posture: Any

Religion: Any

Mother Tongue: Any

Education Level: Any

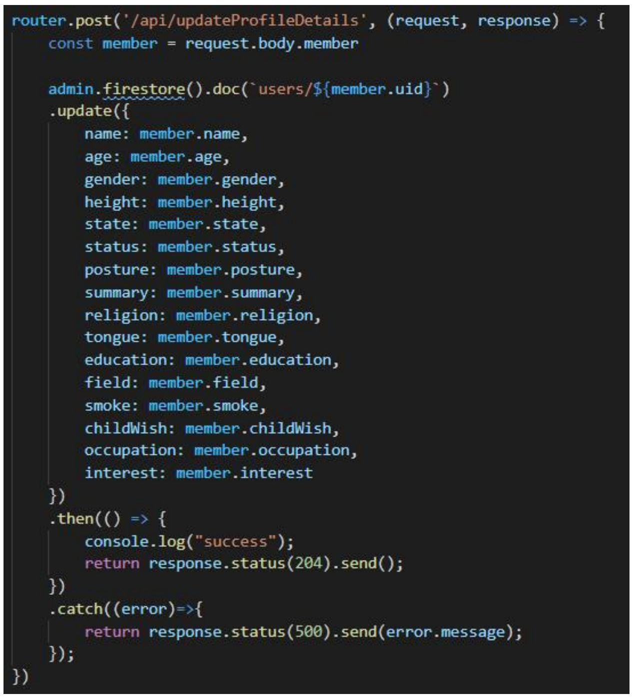
Education Field: Any

Smoking: No, anti-smoking

Child Wish: Not sure yet

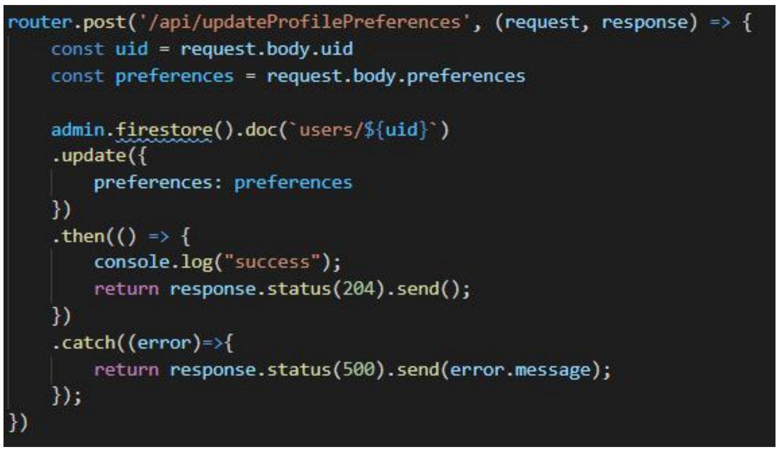
Figure 6.41 Edit Match Preferences Page

If the users edit the personal information in Figure 6.33 and click the “Submit” button, the application will send a POST request to API endpoint (/api/updateProfileDetails). If the users edit the preferences in Figure 6.34, the application will send the POST request to API endpoint (/api/updateProfilePreferences). The server will update the user data in Firestore users collection.



```
router.post('/api/updateProfileDetails', (request, response) => {
    const member = request.body.member

    admin.firestore().doc(`users/${member.uid}`)
    .update({
        name: member.name,
        age: member.age,
        gender: member.gender,
        height: member.height,
        state: member.state,
        status: member.status,
        posture: member.posture,
        summary: member.summary,
        religion: member.religion,
        tongue: member.tongue,
        education: member.education,
        field: member.field,
        smoke: member.smoke,
        childWish: member.childWish,
        occupation: member.occupation,
        interest: member.interest
    })
    .then(() => {
        console.log("success");
        return response.status(204).send();
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.42 Section Code for Update Profile Details (Server)



```
router.post('/api/updateProfilePreferences', (request, response) => {
    const uid = request.body.uid
    const preferences = request.body.preferences

    admin.firestore().doc(`users/${uid}`)
    .update({
        preferences: preferences
    })
    .then(() => {
        console.log("success");
        return response.status(204).send();
    })
    .catch((error)=>{
        return response.status(500).send(error.message);
    });
})
```

Figure 6.43 Section Code for Update Profile Preferences (Server)

CHAPTER 7

TESTING

7.1 Unit Testing

Unit testing was conducted with all components in the application client-side. When performing unit testing, Enzyme was combined with Jest to test the components. Jest is an open-source Javascript testing framework. It offers the “matchers” that enables the assertion easier to read. Similar to Jest, Enzyme is also one of the Javascript testing frameworks. By using the shallow rendering of Enzyme, each component can be tested as a unit. The tables below show the unit test cases of each component in the application.

Table 7.1 Unit Test Case - Home

ID	1			
Module Name	Home			
Description	Test the Home Page Functionality			
Steps	**Details**	**Expected Result**	**Actual Result**	**Status**
1	Render Home component	Should call componentDidMount method when load	componentDidMount was called	Pass

Table 7.2 Unit Test Case - Create Profile

ID	2				
Module Name	Create Profile				
Description	Test the Create Profile Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Render Create Profile component	N/A	Should render create profile page correctly with 1 form, 1 stepper, 15 text labels, 3 next buttons, 3 back buttons and 1 submit button	1 form, 1 stepper, 15 text labels, 3 next buttons, 3 back buttons and 1 submit button were displayed	Pass
2	Enter personal details	Name: Chia Yong Fang State: Pahang Interest: Art / Painting / Drawing', 'Bars / Pubs / Nightclubs'	Should change state value when input for personal details changed	State: { Name: “Chia Yong Fang”, State: Pahang, Interest: ['Art / Painting / Drawing', 'Bars / Pubs / Nightclubs'] }	Pass
3	Submit Form	N/A	Should call the method that handles form submission after click submit button	The handleSubmit method was called after clicked submit button	Pass

Table 7.3 Unit Test Case - Search Profile

ID	3				
Module Name	Search Profile				
Description	Test the Search Profile Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Render Search Profile component	N/A	Should render search profile page correctly with 1 form, 1 header, 12 search criteria, 12 search labels, 4 numeric input fields, 1 clear button and 1 submit button	1 form, 1 header, 12 search criteria, 12 search labels, 4 numeric input fields, 1 clear button and 1 submit button were displayed	Pass
2	Enable age range input and height range input	N/A	Should enable inputs for age range and height range after unchecked "Any" checkbox	Inputs for age range and height range were enabled	Pass
3	Disable age range input and height range input	N/A	Should disable inputs for age range and height range after checked "Any" checkbox	Inputs for age range and height range were disabled	Pass
4	Enter preferences	Gender: Female State: Pahang	Should change state value when input for preferences changed	State: { Gender: Female, State: Pahang }	Pass
5	Submit Form	N/A	Should call method that handle form submission after click submit button	The handleSubmit method was called after clicked submit button	Pass

Table 7.4 Unit Test Case - Search Result

ID	4				
Module Name	Search Result				
Description	Test the Search Result Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Search Result component after API call success	Mock API response to return 2 members' profile details.	Should render search result page correctly with 2 profile cards, 2 chat buttons and 2 favourite buttons	2 profile cards, 2 chat buttons and 2 favourite buttons were displayed	Pass
3	Choose similarity measure as the matching method	N/A	Should call the method that handles similarity measures and render similarity percentage	The handleSimilarity method is called and similarity percentages were displayed for 2 member profiles	Pass
4	Click send message button in the child component	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass
5	Click favourite button	N/A	Should call the method that handles add/remove favourite	The handleFavourite method was called	Pass

Table 7.5 Unit Test Case - Show Profile

ID	5				
Module Name	Show Profile				
Description	Test the Search Result Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Show Profile component after API call success (Exact Match)	N/A	Should render show profile page correctly with 5 tables, 2 cells per table row, 1 profile name field, 1 profile demographic field, 1 chat button and 1 favourite button	5 tables, 2 cells per table row, 1 profile name field, 1 profile demographic field, 1 chat button and 1 favourite button were displayed	Pass
3	Render Show Profile component after API call success (Similarity Measures)	N/A	Should render similarity percentage	The similarity percentages were displayed	Pass
4	Click view picture button	N/A	Should render Upload Profile component	Upload Profile component was rendered	Pass
5	Click chat button	N/A	Should render Instant Message component	Instant Message component was rendered	
6	Click favourite button	N/A	Should call the method that handles add/remove favourite	The updateFavourite method was called	Pass

7	Click send message button in the child component	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass

Table 7.6 Unit Test Case - Instant Message

ID	6				
Module Name	Instant Message				
Description	Test the Instant Message Pop-up Window Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Render Instant Message component	N/A	Should render instant message pop-up window correctly with 1 header, 1 content division, 1 close button, 1 send button and 1 input text field.	1 header, 1 content division, 1 close button, 1 send button and 1 input text field were displayed	Pass
2	Click send message button	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass
3	Click enter key	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass
4	Enter message	Text: “Bye”	Should change state value when giving input for message changed	State: { text: “Bye” }	Pass

Table 7.7 Unit Test Case - Chat List

ID	7				
Module Name	Chat List				
Description	Test the Chat List Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Chat List component after API call success	Mock Firebase Real-time Database response to return 2 chat details.	Should render chat list page correctly with 2 chat record divisions	2 chat record divisions were displayed	Pass
3	Render Chat List component after API call success (without any chat record)	Mock Firebase Real-time Database response to return empty record.	Should display message “Browse your matches and start a conservation today” with 1 search button	Message “Browse your matches and start a conservation today” and 1 search button were displayed	Pass

Table 7.8 Unit Test Case - Individual Chat

ID	8				
Module Name	Individual Chat				
Description	Test the Individual Chat Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Individual Chat component after API call success	Mock Firebase Real-time Database response to return chat details.	Should render individual chat page correctly with 1 banner that consists of user information, 1 option menu, 1 chat display section, 1 send button and 1 input text field	1 banner, 1 option menu, 1 chat display section, 1 send button and 1 input text field were displayed	Pass
3	Enter message	Text: “Bye”	Should change state value when input for message changed	State: { text: “Bye” }	Pass
4	Click send message button	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass
5	Click enter key	N/A	Should call the method that handles sending message	The handleMessage method was called	Pass

Table 7.9 Unit Test Case - Favourite

ID	9				
Module Name	Favourite				
Description	Test the Favourite Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Favourite component after API call success	Mock API response to return 1 member profile.	Should render the favourite page correctly with 1 profile and 1 delete button	1 profile and 1 delete button were displayed	Pass
3	Render Favourite component after API call success (without any favourite record)	Mock API response to return empty record.	Should display message “You haven't added any favourites yet” with 1 search button	Message “You haven't added any favourites yet” and 1 search button were displayed	Pass
4	Click delete button on one profile	N/A	Should call the method that handles delete profile from favourite	The handleDelete method was called	Pass

Table 7.10 Unit Test Case - Personal Profile

ID	10				
Module Name	Personal Profile				
Description	Test the Personal Profile Page Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render Personal Profile component after API call success	Mock API response to return 1 member profile.	Should render show profile page correctly with 5 tables, 2 cells per table row, 1 profile name field, 1 profile demographic field, 1 update picture button and 1 edit profile button	5 tables, 2 cells per table row, 1 profile name field, 1 profile demographic field, 1 update picture button and 1 edit profile button were displayed	Pass
3	Click edit profile button	N/A	Should call the method that handles edit	The handleEdit method was called	Pass
4	Click update picture button	N/A	Should render Upload Photo component	Upload Photo component was rendered	
5	Click confirm button in Upload Photo component (child component)	N/A	Should call the method that handles update profile photo	The handleUpdateProfile method was called	Pass

6	Click close button in Upload Photo component	N/A	Should call the method that handles close Upload Photo component	The handleClose method was called	Pass

Table 7.11 Unit Test Case - Upload Photo

ID	11				
Module Name	Upload Photo				
Description	Test the Upload Photo Pop-up Window Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Click update photo button from the parent component	N/A	Should render upload photo pop-up window correctly with 1 upload button, 1 confirm button and 1 close button	1 upload button, 1 confirm button and 1 close button were displayed	Pass
2	Click view photo button from the parent component	N/A	Should render upload photo pop-up window correctly with without upload button and cancel button	The upload button and cancel button were not displayed	Pass
3	Click confirm button	N/A	Should render a loading sign	A loading spinner was displayed	Pass

Table 7.12 Unit Test Case - Edit Profile

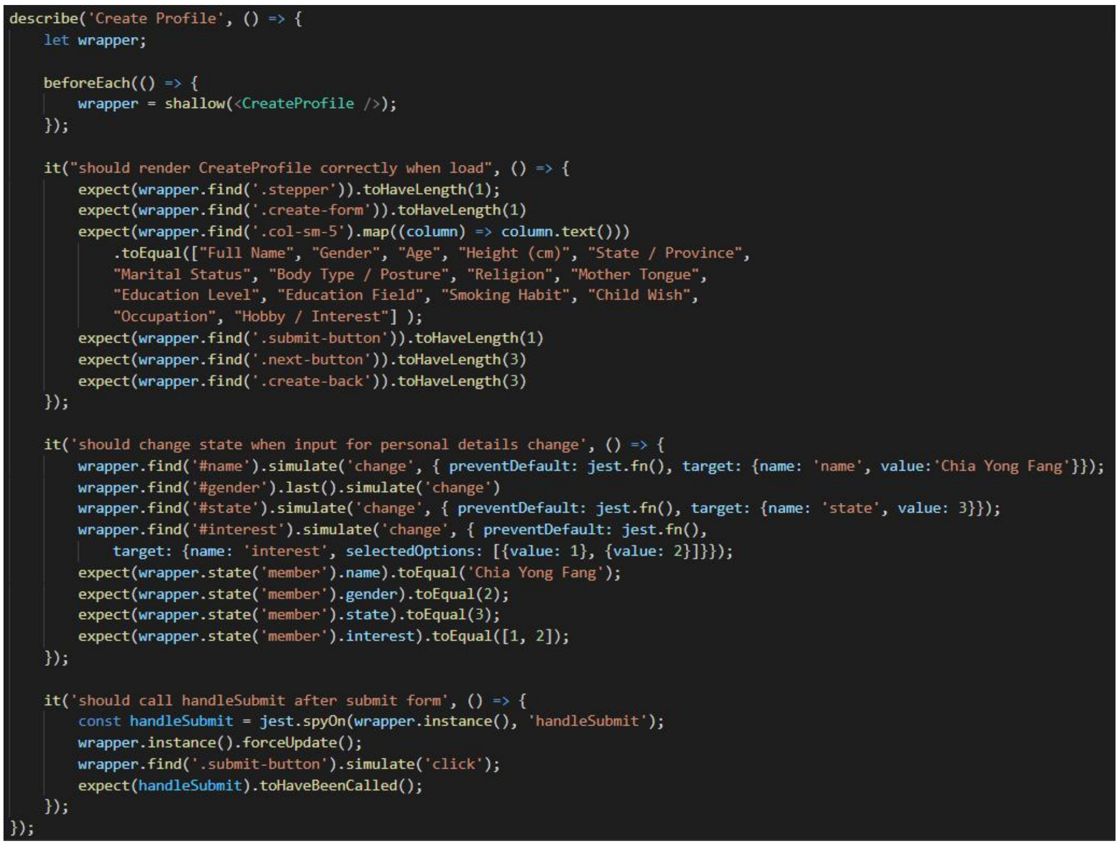
ID	12				
Module Name	Edit Profile				
Description	Test the Edit Page's Navigation Bar Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Before API call success	N/A	Should render a loading sign	A loading spinner was displayed	Pass
2	Render edit navigation bar after API call success	Mock API response to return 1 member profile details.	Should render edit navigation bar correctly with 2 navigation tabs and 1 back button	2 navigation tabs and 1 back button were displayed	Pass
3	Click back button	N/A	Should call method that handle back to previous page action	The handleBack method was called	Pass
4	Click one of the navigation tab	N/A	Should call method that handle the changing of info panel	The handleChange method was called	Pass

Table 7.13 Unit Test Case - Edit Info

ID	13				
Module Name	Edit Info				
Description	Test the Edit Info Panel Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Render Edit Info Panel when loading	Mock member profile details	Should render edit info panel correctly with 1 form, 3 sub-headers, 16 input labels, and 1 submit button. User information should also be displayed correctly.	1 form, 3 sub-headers, 16 input labels, and 1 submit button were displayed. User name, living state and profile summary were displayed correctly.	Pass
2	Click the radio button to select gender	N/A	Should checked radio button for male/female when clicked	The radio button for gender was checked when clicked	Pass
3	Enter personal information	Name: Chia Yong Fang State: Pahang	Should change state value when giving input for personal information changed	State: { Name: “Chia Yong Fang”, State: Pahang }	Pass
4	Click submit button	N/A	Should call the method that handle submit form	The handleSubmit method was called	Pass

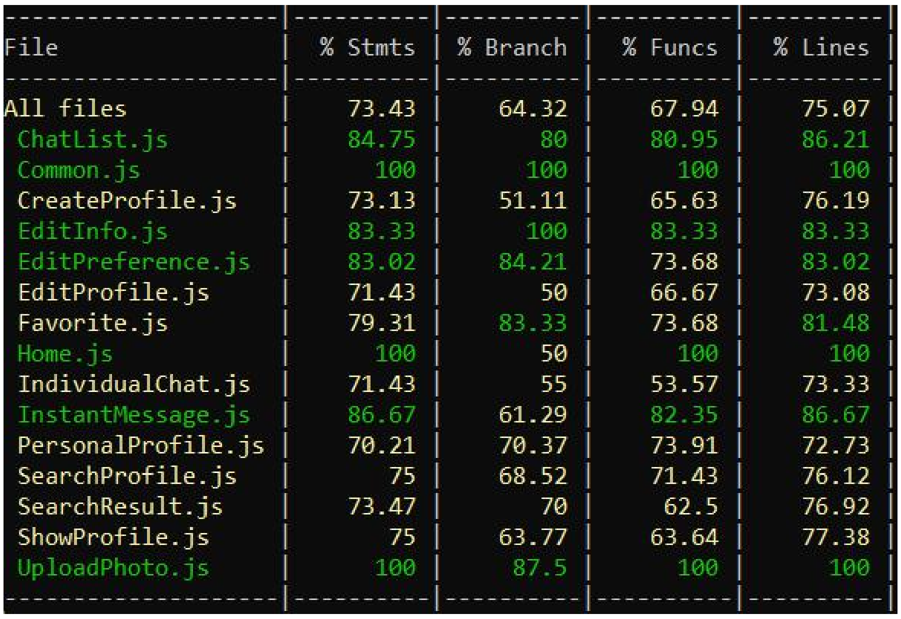
Table 7.14 Unit Test Case - Edit Preferences

ID	14				
Module Name	Edit Preferences				
Description	Test the Edit Preferences Panel Functionality				
Steps	**Details**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Render Edit Preferences Panel when load	Mock member preferences details	Should render edit info panel correctly with 1 form, 1 sub-headers, 12 input labels, 1 clear button and 1 submit button. User preferences should also be displayed correctly.	1 form, 1 sub-headers, 12 input labels, 1 clear button and 1 submit button. User preferences for age range and child wish were displayed correctly.	Pass
2	Enable age range input and height range input	N/A	Should enable inputs for age and height range after unchecked “Any” checkbox	Inputs for age range and height range were enabled	Pass
3	Disable age range input and height range input	N/A	Should disable inputs for age range and height range after checked “Any” checkbox	Inputs for age range and height range were disabled	Pass
4	Enter preferences	Gender: Female State: Pahang	Should change state value when giving input for preferences changed	State: { Gender: Female, State: Pahang }	
5	Submit Form	N/A	Should call the method that handles form submission after click submit button	The handleSubmit method was called after clicked submit button	Pass



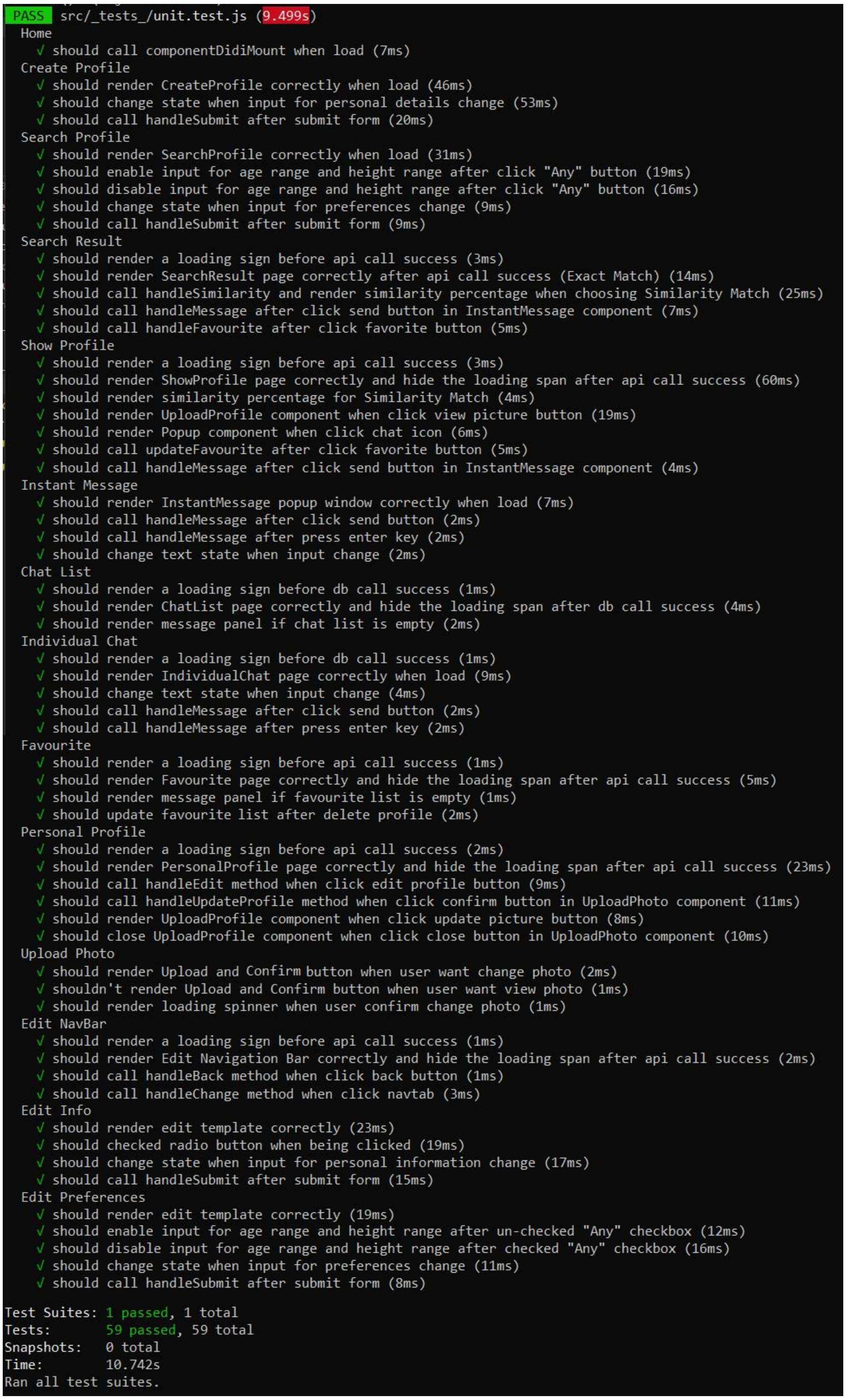
```
describe('Create Profile', () => {
    let wrapper;

    beforeEach(() => {
        wrapper = shallow(<CreateProfile />);
    });

    it("should render CreateProfile correctly when load", () => {
        expect(wrapper.find('.stepper')).toHaveLength(1);
        expect(wrapper.find('.create-form')).toHaveLength(1)
        expect(wrapper.find('.col-sm-5').map((column) => column.text()))
            .toEqual(["Full Name", "Gender", "Age", "Height (cm)", "State / Province",
            "Marital Status", "Body Type / Posture", "Religion", "Mother Tongue",
            "Education Level", "Education Field", "Smoking Habit", "Child Wish",
            "Occupation", "Hobby / Interest"] );
        expect(wrapper.find('.submit-button')).toHaveLength(1)
        expect(wrapper.find('.next-button')).toHaveLength(3)
        expect(wrapper.find('.create-back')).toHaveLength(3)
    });

    it('should change state when input for personal details change', () => {
        wrapper.find('#name').simulate('change', { preventDefault: jest.fn(), target: {name: 'name', value:'Chia Yong Fang'}});
        wrapper.find('#gender').last().simulate('change')
        wrapper.find('#state').simulate('change', { preventDefault: jest.fn(), target: {name: 'state', value: 3}});
        wrapper.find('#interest').simulate('change', { preventDefault: jest.fn(),
            target: {name: 'interest', selectedOptions: [{value: 1}, {value: 2}]}});
        expect(wrapper.state('member').name).toEqual('Chia Yong Fang');
        expect(wrapper.state('member').gender).toEqual(2);
        expect(wrapper.state('member').state).toEqual(3);
        expect(wrapper.state('member').interest).toEqual([1, 2]);
    });

    it('should call handleSubmit after submit form', () => {
        const handleSubmit = jest.spyOn(wrapper.instance(), 'handleSubmit');
        wrapper.instance().forceUpdate();
        wrapper.find('.submit-button').simulate('click');
        expect(handleSubmit).toHaveBeenCalled();
    });
});
```

Figure 7.1 Section Code of Unit Testing



```
--------------------|----------|----------|----------|----------|
File                |  % Stmts | % Branch |  % Funcs |  % Lines |
--------------------|----------|----------|----------|----------|
All files           |    73.43 |    64.32 |    67.94 |    75.07 |
 ChatList.js        |    84.75 |       80 |    80.95 |    86.21 |
 Common.js          |      100 |      100 |      100 |      100 |
 CreateProfile.js   |    73.13 |    51.11 |    65.63 |    76.19 |
 EditInfo.js        |    83.33 |      100 |    83.33 |    83.33 |
 EditPreference.js  |    83.02 |    84.21 |    73.68 |    83.02 |
 EditProfile.js     |    71.43 |       50 |    66.67 |    73.08 |
 Favorite.js        |    79.31 |    83.33 |    73.68 |    81.48 |
 Home.js            |      100 |       50 |      100 |      100 |
 IndividualChat.js  |    71.43 |       55 |    53.57 |    73.33 |
 InstantMessage.js  |    86.67 |    61.29 |    82.35 |    86.67 |
 PersonalProfile.js |    70.21 |    70.37 |    73.91 |    72.73 |
 SearchProfile.js   |       75 |    68.52 |    71.43 |    76.12 |
 SearchResult.js    |    73.47 |       70 |     62.5 |    76.92 |
 ShowProfile.js     |       75 |    63.77 |    63.64 |    77.38 |
 UploadPhoto.js     |      100 |     87.5 |      100 |      100 |
--------------------|----------|----------|----------|----------|
```

Figure 7.2 Test Coverage of Unit Test



```
 PASS  src/_tests_/unit.test.js (9.499s)
  Home
    √ should call componentDidiMount when load (7ms)
  Create Profile
    √ should render CreateProfile correctly when load (46ms)
    √ should change state when input for personal details change (53ms)
    √ should call handleSubmit after submit form (20ms)
  Search Profile
    √ should render SearchProfile correctly when load (31ms)
    √ should enable input for age range and height range after click "Any" button (19ms)
    √ should disable input for age range and height range after click "Any" button (16ms)
    √ should change state when input for preferences change (9ms)
    √ should call handleSubmit after submit form (9ms)
  Search Result
    √ should render a loading sign before api call success (3ms)
    √ should render SearchResult page correctly after api call success (Exact Match) (14ms)
    √ should call handleSimilarity and render similarity percentage when choosing Similarity Match (25ms)
    √ should call handleMessage after click send button in InstantMessage component (7ms)
    √ should call handleFavourite after click favorite button (5ms)
  Show Profile
    √ should render a loading sign before api call success (3ms)
    √ should render ShowProfile page correctly and hide the loading span after api call success (60ms)
    √ should render similarity percentage for Similarity Match (4ms)
    √ should render UploadProfile component when click view picture button (19ms)
    √ should render Popup component when click chat icon (6ms)
    √ should call updateFavourite after click favorite button (5ms)
    √ should call handleMessage after click send button in InstantMessage component (4ms)
  Instant Message
    √ should render InstantMessage popup window correctly when load (7ms)
    √ should call handleMessage after click send button (2ms)
    √ should call handleMessage after press enter key (2ms)
    √ should change text state when input change (2ms)
  Chat List
    √ should render a loading sign before db call success (1ms)
    √ should render ChatList page correctly and hide the loading span after db call success (4ms)
    √ should render message panel if chat list is empty (2ms)
  Individual Chat
    √ should render a loading sign before db call success (1ms)
    √ should render IndividualChat page correctly when load (9ms)
    √ should change text state when input change (4ms)
    √ should call handleMessage after click send button (2ms)
    √ should call handleMessage after press enter key (2ms)
  Favourite
    √ should render a loading sign before api call success (1ms)
    √ should render Favourite page correctly and hide the loading span after api call success (5ms)
    √ should render message panel if favourite list is empty (1ms)
    √ should update favourite list after delete profile (2ms)
  Personal Profile
    √ should render a loading sign before api call success (2ms)
    √ should render PersonalProfile page correctly and hide the loading span after api call success (23ms)
    √ should call handleEdit method when click edit profile button (9ms)
    √ should call handleUpdateProfile method when click confirm button in UploadPhoto component (11ms)
    √ should render UploadProfile component when click update picture button (8ms)
    √ should close UploadProfile component when click close button in UploadPhoto component (10ms)
  Upload Photo
    √ should render Upload and Confirm button when user want change photo (2ms)
    √ shouldn't render Upload and Confirm button when user want view photo (1ms)
    √ should render loading spinner when user confirm change photo (1ms)
  Edit NavBar
    √ should render a loading sign before api call success (1ms)
    √ should render Edit Navigation Bar correctly and hide the loading span after api call success (2ms)
    √ should call handleBack method when click back button (1ms)
    √ should call handleChange method when click navtab (3ms)
  Edit Info
    √ should render edit template correctly (23ms)
    √ should checked radio button when being clicked (19ms)
    √ should change state when input for personal information change (17ms)
    √ should call handleSubmit after submit form (15ms)
  Edit Preferences
    √ should render edit template correctly (19ms)
    √ should enable input for age range and height range after un-checked "Any" checkbox (12ms)
    √ should disable input for age range and height range after checked "Any" checkbox (16ms)
    √ should change state when input for preferences change (11ms)
    √ should call handleSubmit after submit form (8ms)

Test Suites: 1 passed, 1 total
Tests:       59 passed, 59 total
Snapshots:   0 total
Time:        10.742s
Ran all test suites.
```

Figure 7.3 Result of Unit Test

7.2 Application Programming Interface (API) Testing

API endpoint testing was conducted to test APIs functionality and performance. The testing section was set up in the application's server-side (Express.js). Jest and Supertest were used as the testing tools to conduct API testing. Supertest is an HTTP assertion library that enables the sending of HTTP requests to Node.js HTTP servers. In addition, Jest is used to validate the test results.

Table 7.15 API Test Case - Create Profile

ID	1
Name	Create Profile

Input	Send a POST request to '/api/createProfile' by providing:
	1. user id: 'yw7GCczWDGcZbPX6aA3MezhABZ32'
	2. profile picture: 'profile.jpg' in data URI format
	3. user's personal information: {
	age: 24, childWish: 3, education: 5, field: 13, gender: 2, height: 154, interest: ["1", "7", "13", "19", "21", "29", "30"], name: "Chia Yong Fang", occupation: 26, posture: 2, religion: 1, smoke: 1, state: 1, status: 1, tongue: 1, summary: "Hello", preferences: { childWish: 3, education: 0, field: 0, gender: 1, maxAge: 0, maxHeight: 0, minAge: 0, minHeight: 0, posture: 0, religion: 0, smoke: 1, state: 0, status: 1, tongue: 1 }
	}

Expected Result	Response status: 204
Actual Result	Response status: 204
Test Status	Pass

Table 7.16 API Test Case - Get Personal Info

ID	2
Name	Get Personal Info
Input	Send a POST request to '/api/getPersonalInfo' by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
Expected Result	Return an object as response body which contains: ● uid: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
Actual Result	Return an object as response body which contains ● uid: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
Test Status	Pass

Table 7.17 API Test Case - Update Profile Picture

ID	3
Name	Update Profile Picture
Input	Send a POST request to '/api/updateProfilePic' by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1' 2. profile picture: 'profile.jpg' in data URI format
Expected Result	Response status: 200
Actual Result	Response status: 200
Test Status	Pass

Table 7.18 API Test Case - Update Profile Details

ID	4
Name	Update Profile Details

Input	Send a POST request to '/api/updateProfileDetails' by providing:
	1. user's personal information: {
	age: 24, childWish: 3, education: 5, field: 13, gender: 2, height: 154, interest: ["1", "7", "13", "19", "21", "29", "30"], name: "Chia Yong Fang", occupation: 26, posture: 2, religion: 1, smoke: 1, state: 1, status: 1, tongue: 1, uid: 'Q1aATWz9u3UZWetOlo6vkaNju2j1', summary: "Just browsing for fun, quite open minded and quirky to say the least. Just doing my thing and meeting like minded people." }
	}

Expected Result	Response status: 204
Actual Result	Response status: 204
Test Status	Pass

Table 7.19 API Test Case - Update Profile Preferences

ID	5
Name	Update Profile Preferences

Input	Send a POST request to '/api/updateProfilePreferences' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. preferences: {
	childWish: 3, education: 0, field: 0, gender: 1, maxAge: 0, maxHeight: 0, minAge: 0, minHeight: 0, posture: 0, religion: 0, smoke: 1, state: 0, status: 1, tongue: 1
	}

Expected Result	Response status: 204
Actual Result	Response status: 204
Test Status	Pass

Table 7.20 API Test Case - Get Exact Match Result

ID	6
Name	Get Exact Match Result

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1'
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1'
Test Status	Pass

Table 7.21 API Test Case - Get Similarity Match Result (Jaccard)

ID	7
Name	Get Similarity Match Result (Jaccard)

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. type: 'jaccard'
	3. p-index: 0
	4. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● objects with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● objects with uid 'WknKtEmYcVc9cYYkoRSyLet2JCy1' and similarity 83.33%.
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'WknKtEmYcVc9cYYkoRSyLet2JCy1' and similarity 83.33%.
Test Status	Pass

Table 7.22 API Test Case - Get Similarity Match Result (Cosine)

ID	8
Name	Get Similarity Match Result (Cosine)

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. type: 'cosine'
	3. p-index: 0
	4. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● objects with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● objects with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 99.59%.
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 99.59%.
Test Status	Pass

Table 7.23 API Test Case - Get Similarity Match Result (Euclidean)

ID	9
Name	Get Similarity Match Result (Euclidean)

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. type: 'euclidean'
	3. p-index: 0
	4. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● objects with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● objects with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 76.72%.
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 76.72%.
Test Status	Pass

Table 7.24 API Test Case - Get Similarity Match Result (Manhattan)

ID	10
Name	Get Similarity Match Result (Manhattan)

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. type: 'manhattan'
	3. p-index: 0
	4. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 63.61%.
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 63.61%.
Test Status	Pass

Table 7.25 API Test Case - Get Similarity Match Result (Minkowski)

ID	11
Name	Get Similarity Match Result (Minkowski)

Input	Send a POST request to '/api/getExactMatchResult' by providing:
	1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
	2. type: 'minkowski'
	3. p-index: 3
	4. preferences: {
	childWish: '3', education: '5', field: '13', gender: '1', maxAge: '0', maxHeight: '0', minAge: '0', minHeight: '0', tongue: '1', posture: '2', religion: '2', smoke: '1', state: '12', status: '1'
	}

Expected Result	Response status: 200 Return an array of object as response body which contains: ● objects with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● objects with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 79.94%.
Actual Result	Response status: 200 Return an array of object as response body which contains: ● object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● object with uid 'KX14LPGtsxYA1tnPzNrtgVOYJpr1' and similarity 79.94%.
Test Status	Pass

Table 7.26 API Test Case - Check Favourite

ID	12
Name	Check Favourite
Input	Send a POST request to '/api/checkFavourite'by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
Expected Result	Response status: 200 Return an list as response body which contains: ● uid: 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Actual Result	Response status: 200 Return an list as response body which contains: ● uid: 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Test Status	Pass

Table 7.27 API Test Case - Update Favourite

ID	13
Name	Update Favourite
Input	Send a POST request to '/api/updateFavourite' by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1' 2. new favourite list: ['WZeVJykKnPUdG6w8YC2sAv7R3772', 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1']
Expected Result	Response status: 204
Actual Result	Response status: 204
Test Status	Pass

Table 7.28 API Test Case - Get Favourite List

ID	14
Name	Get Favourite List
Input	Send a POST request to '/api/getFavouriteList' by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'
Expected Result	Response status: 200 Return response body: ● array of objects that contains object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● favourite list that contains data 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Actual Result	Response status: 200 Return response body: ● Array of objects that contains object with uid 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1' and similarity 100.0%. ● Favourite list that contains data 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Test Status	Pass

Table 7.29 API Test Case - Delete Favourite

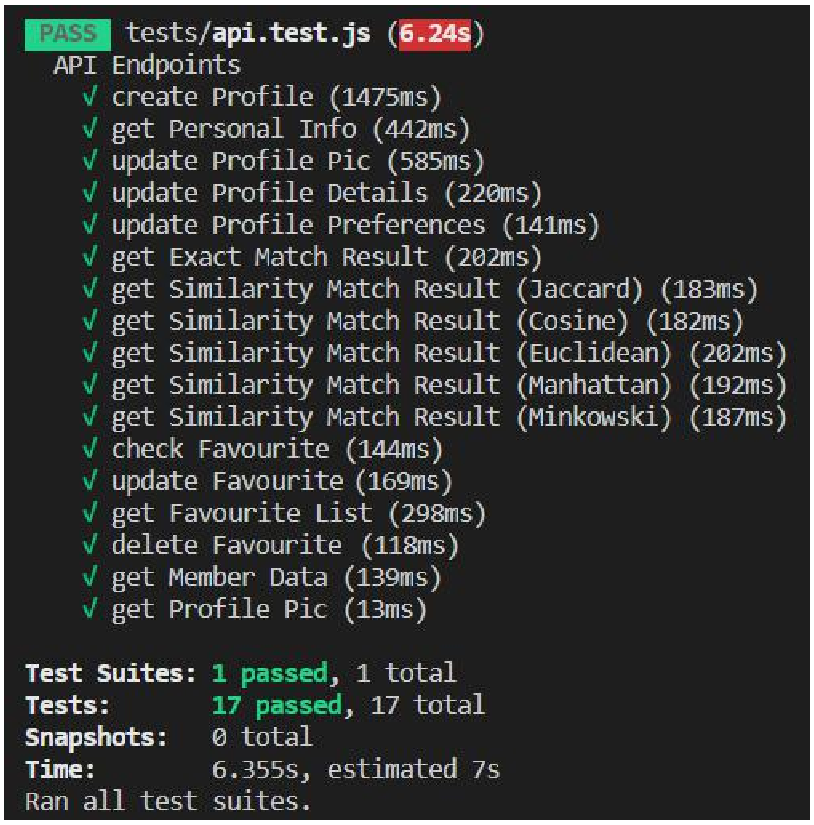
ID	15
Name	Delete Favourite
Input	Send a POST request to '/api/getFavouriteList' by providing: 1. user id: 'Q1aATWz9u3UZWetOlo6vkaNju2j1' 2. new favourite list: ['WZeVJykKnPUdG6w8YC2sAv7R3772']
Expected Result	Response status: 204
Actual Result	Response status: 204
Test Status	Pass

Table 7.30 API Test Case - Get Member Data

ID	16
Name	Get Member Data
Input	Send a POST request to ‘/api/getMemberData’ by providing: 1. list: ['WZeVJykKnPUdG6w8YC2sAv7R3772', 'rSmD4vUJP6NMHsk8FPvPCgPwfAm1']
Expected Result	Response status: 200 Return an array of objects as response body which contains: ● object with uid 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Actual Result	Response status: 200 Return an array of objects as response body which contains: ● object with uid 'WZeVJykKnPUdG6w8YC2sAv7R3772'
Test Status	Pass

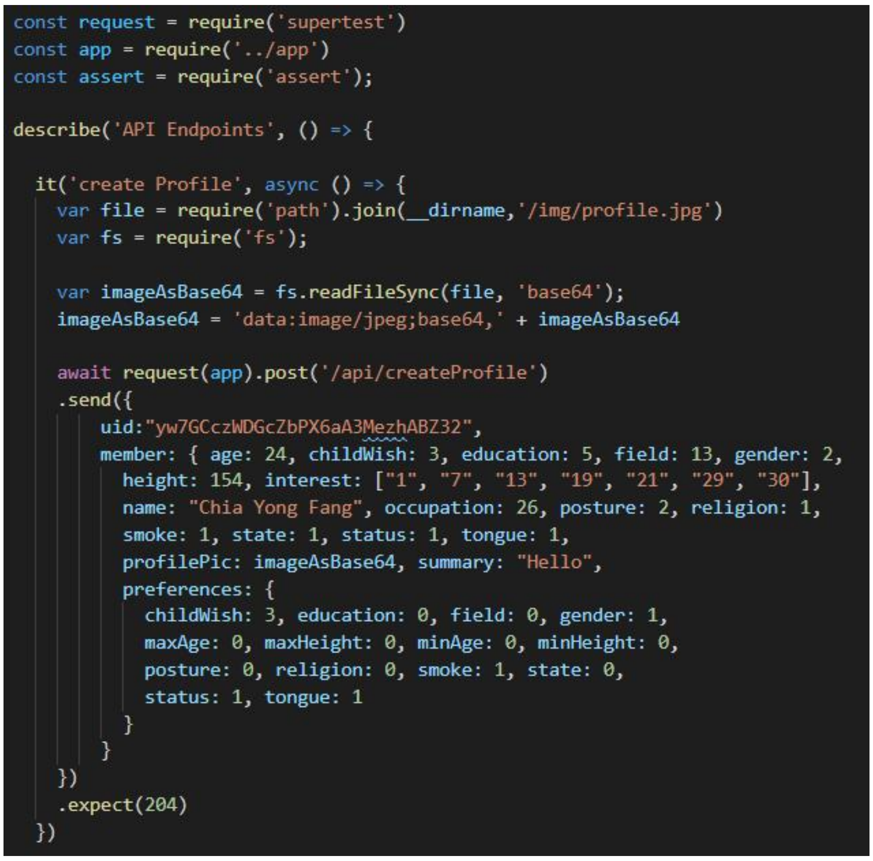
Table 7.31 API Test Case - Get Profile Picture

ID	17
Name	Get Profile Picture
Input	Send a POST request to ‘/api/getProfilePic’ by providing: 1. user id: ‘Q1aATWz9u3UZWetOlo6vkaNju2j1’
Expected Result	Response status: 200
Actual Result	Response status: 200
Test Status	Pass



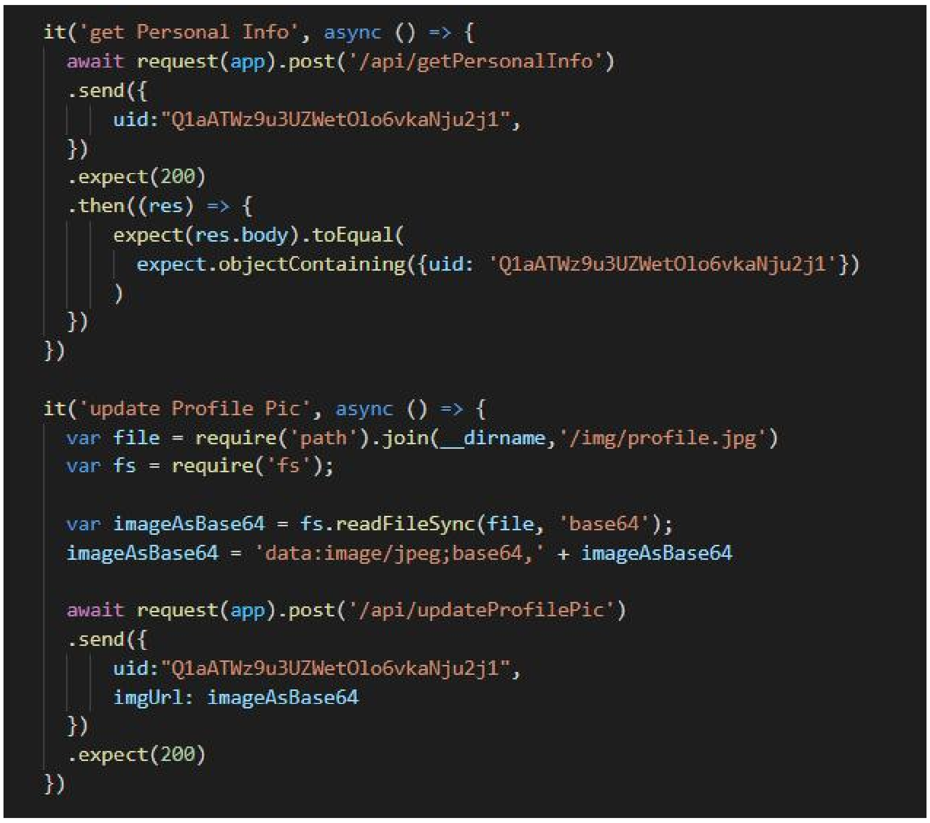
```
 PASS  tests/api.test.js (6.24s)
  API Endpoints
    √ create Profile (1475ms)
    √ get Personal Info (442ms)
    √ update Profile Pic (585ms)
    √ update Profile Details (220ms)
    √ update Profile Preferences (141ms)
    √ get Exact Match Result (202ms)
    √ get Similarity Match Result (Jaccard) (183ms)
    √ get Similarity Match Result (Cosine) (182ms)
    √ get Similarity Match Result (Euclidean) (202ms)
    √ get Similarity Match Result (Manhattan) (192ms)
    √ get Similarity Match Result (Minkowski) (187ms)
    √ check Favourite (144ms)
    √ update Favourite (169ms)
    √ get Favourite List (298ms)
    √ delete Favourite (118ms)
    √ get Member Data (139ms)
    √ get Profile Pic (13ms)

Test Suites: 1 passed, 1 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        6.355s, estimated 7s
Ran all test suites.
```

Figure 7.4 Result of API Test



```
const request = require('supertest')
const app = require('../app')
const assert = require('assert');

describe('API Endpoints', () => {

  it('create Profile', async () => {
    var file = require('path').join(__dirname,'/img/profile.jpg')
    var fs = require('fs');

    var imageAsBase64 = fs.readFileSync(file, 'base64');
    imageAsBase64 = 'data:image/jpeg;base64,' + imageAsBase64

    await request(app).post('/api/createProfile')
    .send({
        uid:"yw7GCczWDGcZbPX6aA3MezhABZ32",
        member: { age: 24, childWish: 3, education: 5, field: 13, gender: 2,
          height: 154, interest: ["1", "7", "13", "19", "21", "29", "30"],
          name: "Chia Yong Fang", occupation: 26, posture: 2, religion: 1,
          smoke: 1, state: 1, status: 1, tongue: 1,
          profilePic: imageAsBase64, summary: "Hello",
          preferences: {
            childWish: 3, education: 0, field: 0, gender: 1,
            maxAge: 0, maxHeight: 0, minAge: 0, minHeight: 0,
            posture: 0, religion: 0, smoke: 1, state: 0,
            status: 1, tongue: 1
          }
        }
    })
    .expect(204)
  })
```

Figure 7.5 Sample Section Code for API Test - Part 1



```
it('get Personal Info', async () => {
  await request(app).post('/api/getPersonalInfo')
  .send({
      uid:"Q1aATWz9u3UZWetOlo6vkaNju2j1",
  })
  .expect(200)
  .then((res) => {
      expect(res.body).toEqual(
        expect.objectContaining({uid: 'Q1aATWz9u3UZWetOlo6vkaNju2j1'})
      )
  })
})

it('update Profile Pic', async () => {
  var file = require('path').join(__dirname,'/img/profile.jpg')
  var fs = require('fs');

  var imageAsBase64 = fs.readFileSync(file, 'base64');
  imageAsBase64 = 'data:image/jpeg;base64,' + imageAsBase64

  await request(app).post('/api/updateProfilePic')
  .send({
      uid:"Q1aATWz9u3UZWetOlo6vkaNju2j1",
      imgUrl: imageAsBase64
  })
  .expect(200)
})
```

Figure 7.6 Sample Section Code for API Test - Part 2

7.3 End-to-end Testing

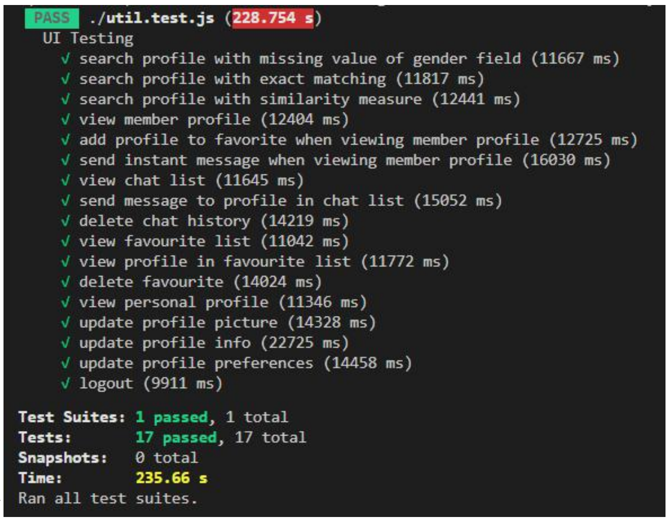
In this project, end-to-end testing was used to test the application's actual flow from start to finish. It can ensure that the application flow will work as expected. Besides, Puppeteer and Jest were used as the testing tools to perform end-to-end testing. Puppeteer is an automation testing tool that enables headless browsing of Chrome. It can interact with the application like a real user.

Table 7.32 End-to-end Test Cases

ID	Test Case	Steps	Expected Result	Status
1	Search profile with missing value of gender field	Login → Select match preferences except gender field → Click "Submit" button	The application is able to display an alert message with text 'Gender field cannot be empty'.	Pass
2	Search profile with exact matching	Login → Select match preferences → Click "Submit" button	The application is able to display matched profile.	Pass
3	Search profile with similarity measure	Login → Select match preferences → Click "Submit" button → Select "Jaccard Coefficient" radio button	The application is able to display profile list with similarity percentage.	Pass
4	View member profile	Login → Select match preferences → Click "Submit" button → Click first profile	The application is able to display member profile page with correct name and demographic info.	Pass
5	Add profile to favourite when viewing member profile	Login → Select match preferences → Click "Submit" button → Click first profile → Click "Favourite" button	The application is able to display a message with text 'Added to Favourite'.	Pass

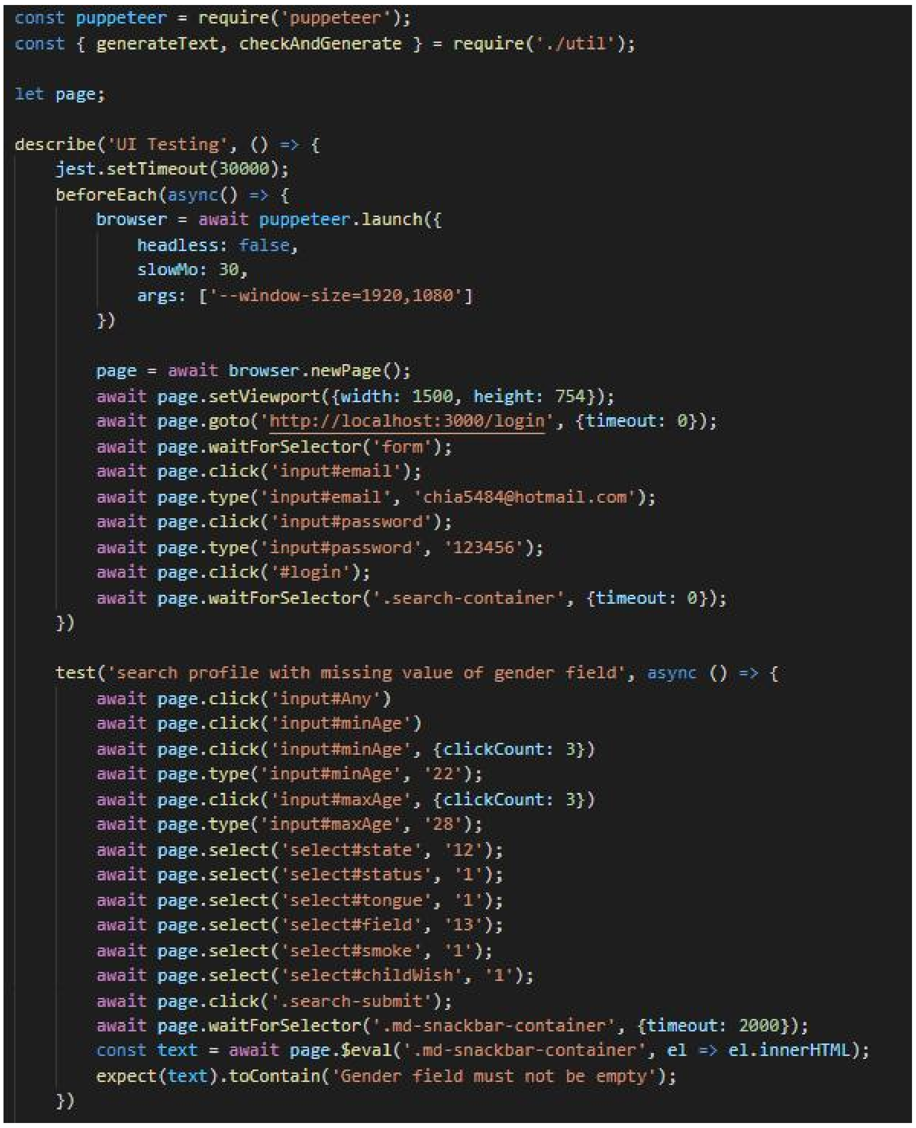
6	Send instant message when viewing member profile	Login → Select match preferences → Click “Submit” button → Click first profile → Click “Chat” button → Enter message in the pop-up window → Click “Send” button	After user sends the message, the application is able to display the message as chat history in the pop-up window.	Pass
7	View chat list	Login → Click “Messages” on navigation	The application is able to display a list of chat records. For each chat in the list, the newest message will be shown.	Pass
8	Send message to profile in chat list	Login → Click “Messages” on navigation → Click on the first record → Enter message → Click “Send” button	After user sends the message, the application is able to display the message as chat history.	Pass
9	Delete chat history	Login → Click “Messages” on navigation → Click on the first record → Click “Delete” button	The application is able to remove the chat record and display the updated chat list.	Pass
10	View favourite list	Login → Click “Favourite” on navigation	The application is able to display a list of profiles added to favourite.	Pass
11	View profile in favourite list	Login → Click “Favourite” on navigation → Click on the first profile	The application is able to display member profile page with correct name.	Pass
12	Delete favourite	Login → Click “Favourite” on navigation → Click “Delete” button	The application is able to remove the profile from favourite and display the updated	Pass

			favourite list.	
13	View personal profile	Login → Click “Avatar” icon on navigation → Click “Profile” option	The application is able to display user’s personal information	Pass
14	Update profile picture	Login → Click “Avatar” icon on navigation → Click “Profile” option → Click “Edit” button beside the profile picture	The application is able to get the new profile picture from the specified file directory.	Pass
15	Update profile info	Login → Click “Avatar” icon on navigation → Click “Profile” option → Click “Edit Profile” button → Edit “About Me” field → Click “Submit” button	The application is able to display a message with text ‘Update Successfully’.	Pass
16	Update profile preferences	Login → Click “Avatar” icon on navigation → Click “Profile” option → Click “Edit Profile” button → Click “Preferences” tab → Edit some fields → Click “Submit” button	The application is able to display a message with text ‘Update Successfully’.	Pass
17	Logout	Login → Click “Avatar” icon on navigation → Click “Logout” option	The application is able to redirect the user to home page after logout	Pass



```
PASS  ./util.test.js (228.754 s)
  UI Testing
    √ search profile with missing value of gender field (11667 ms)
    √ search profile with exact matching (11817 ms)
    √ search profile with similarity measure (12441 ms)
    √ view member profile (12404 ms)
    √ add profile to favorite when viewing member profile (12725 ms)
    √ send instant message when viewing member profile (16030 ms)
    √ view chat list (11645 ms)
    √ send message to profile in chat list (15052 ms)
    √ delete chat history (14219 ms)
    √ view favourite list (11042 ms)
    √ view profile in favourite list (11772 ms)
    √ delete favourite (14024 ms)
    √ view personal profile (11346 ms)
    √ update profile picture (14328 ms)
    √ update profile info (22725 ms)
    √ update profile preferences (14458 ms)
    √ logout (9911 ms)

Test Suites: 1 passed, 1 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        235.66 s
Ran all test suites.
```

Figure 7.7 Result of End-to-end Test



```
const puppeteer = require('puppeteer');
const { generateText, checkAndGenerate } = require('./util');

let page;

describe('UI Testing', () => {
    jest.setTimeout(30000);
    beforeEach(async() => {
        browser = await puppeteer.launch({
            headless: false,
            slowMo: 30,
            args: ['--window-size=1920,1080']
        })

        page = await browser.newPage();
        await page.setViewport({width: 1500, height: 754});
        await page.goto('http://localhost:3000/login', {timeout: 0});
        await page.waitForSelector('form');
        await page.click('input#email');
        await page.type('input#email', 'chia5484@hotmail.com');
        await page.click('input#password');
        await page.type('input#password', '123456');
        await page.click('#login');
        await page.waitForSelector('.search-container', {timeout: 0});
    })

    test('search profile with missing value of gender field', async () => {
        await page.click('input#Any')
        await page.click('input#minAge')
        await page.click('input#minAge', {clickCount: 3})
        await page.type('input#minAge', '22');
        await page.click('input#maxAge', {clickCount: 3})
        await page.type('input#maxAge', '28');
        await page.select('select#state', '12');
        await page.select('select#status', '1');
        await page.select('select#tongue', '1');
        await page.select('select#field', '13');
        await page.select('select#smoke', '1');
        await page.select('select#childWish', '1');
        await page.click('.search-submit');
        await page.waitForSelector('.md-snackbar-container', {timeout: 2000});
        const text = await page.$eval('.md-snackbar-container', el => el.innerHTML);
        expect(text).toContain('Gender field must not be empty');
    })
```

Figure 7.8 Sample Section Code for End-to-end Test

7.4 Usability Testing

A usability testing on the matrimonial application was carried remotely with 8 participants. The usability testing can help to collect information on how real users interact with the application. Besides, usability testing was used to find out which similarity measure is better.

7.4.1 Test Scenario

All the participants were requested to complete all the test scenario provided. The interaction of participants with the application was observed.

Table 7.33 Usability Testing Test Scenarios

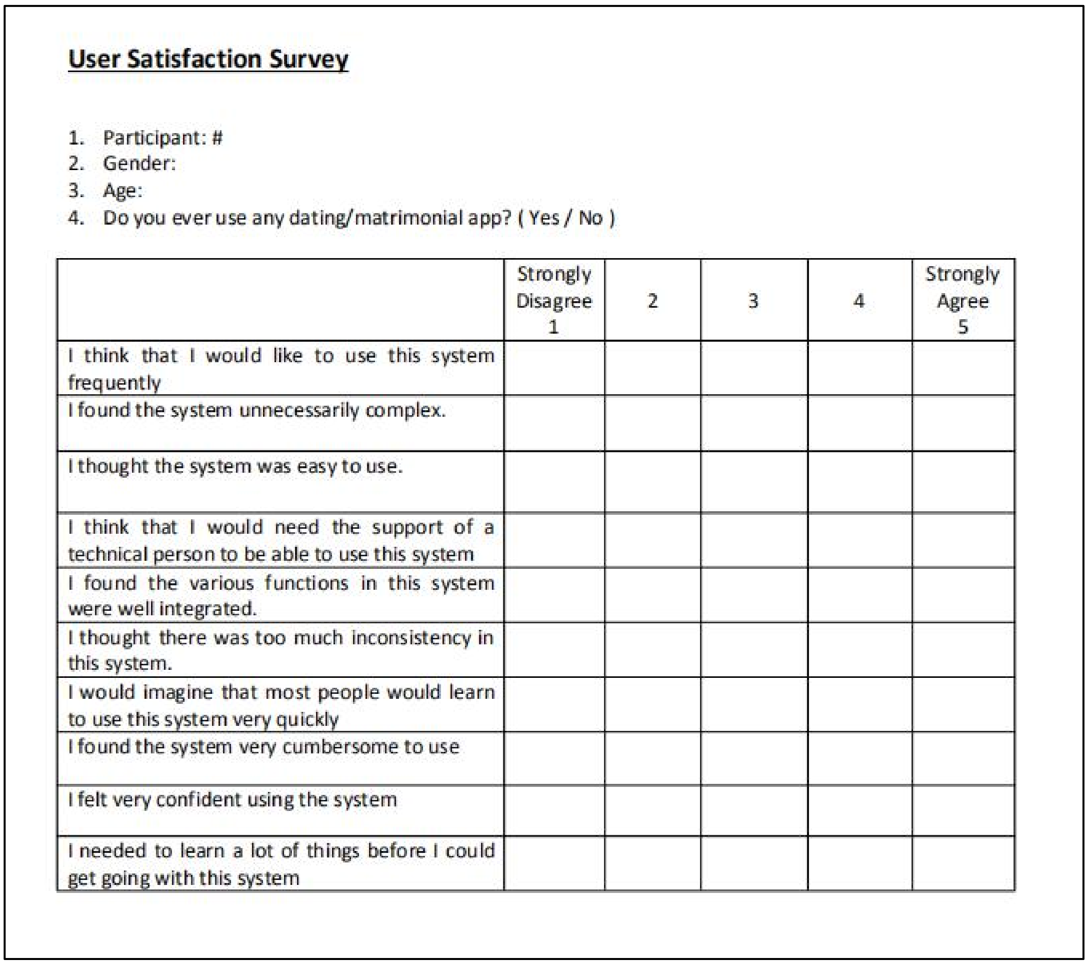
ID	Scenario Name	Scenario Description
1	Create a profile	Imagine that you are a user who wishes to use the MatriMatch application for finding a partner. Task: i. You want to create a profile in the application so that you can search for the desired partner. By creating a profile, you might also have the chance to exist in other members' search result. How do you create a profile?
2	Search for potential matches	Imagine that you are a user who wishes to find for a partner that suit your preferences. Task: i. You wish to set the search criteria and view the search result. ii. After viewing the search result, you wish to edit the search criteria. How would you set the search criteria and view for

		a search result? How would you edit the search criteria?
3	View member profiles	Imagine that you are a user who uses the application to view the potential partner. Task: i. You wish to view more information about a particular member. ii. You want to favourite two profiles that you interested in. iii. You would like to start a conversation with one of the profiles. How would you view the member's profile information? How would you favourite an item? How would you send a message to the profile you interest?
4	View chat history	Imagine that you are a user who wishes to view your chat history with one of the members in the application. Task: i. You want to view the chat between you and one of the members in the application. ii. You want to delete the current chat history. How would you view the chat history? How would you delete the chat history?
5	View favourite list	Imagine that you are a user who wishes to view your favourite list. Task:

		i. You want to view the favourite list ii. You want to delete an unwanted profile from the list How would you view the favourite list? How would you delete a profile from the favourite list?
6	Edit personal profile	Imagine that you are a user who wishes to change your profile information and profile picture. Task: i. You want to view the profile information. ii. You want to change the profile picture. iii. You want to update your match preferences in "Looking For" section. iv. You want to edit your profile summary as follow: <u>Profile summary:</u> Love watching drama Chilling and watching movies with friends How would you change the profile summary? How would you change your match preferences? How would you change the profile picture?

7.4.2 User Satisfaction Survey Form

After the testing section ended, the participants were requested to fill in the user satisfaction survey form. The survey form included the System Usability Scale (SUS) and several questions regarding the accuracy of the similarity measures in finding the matches. The survey form gathered the participants' responses and opinions upon the performance of the system and the favourite match method.

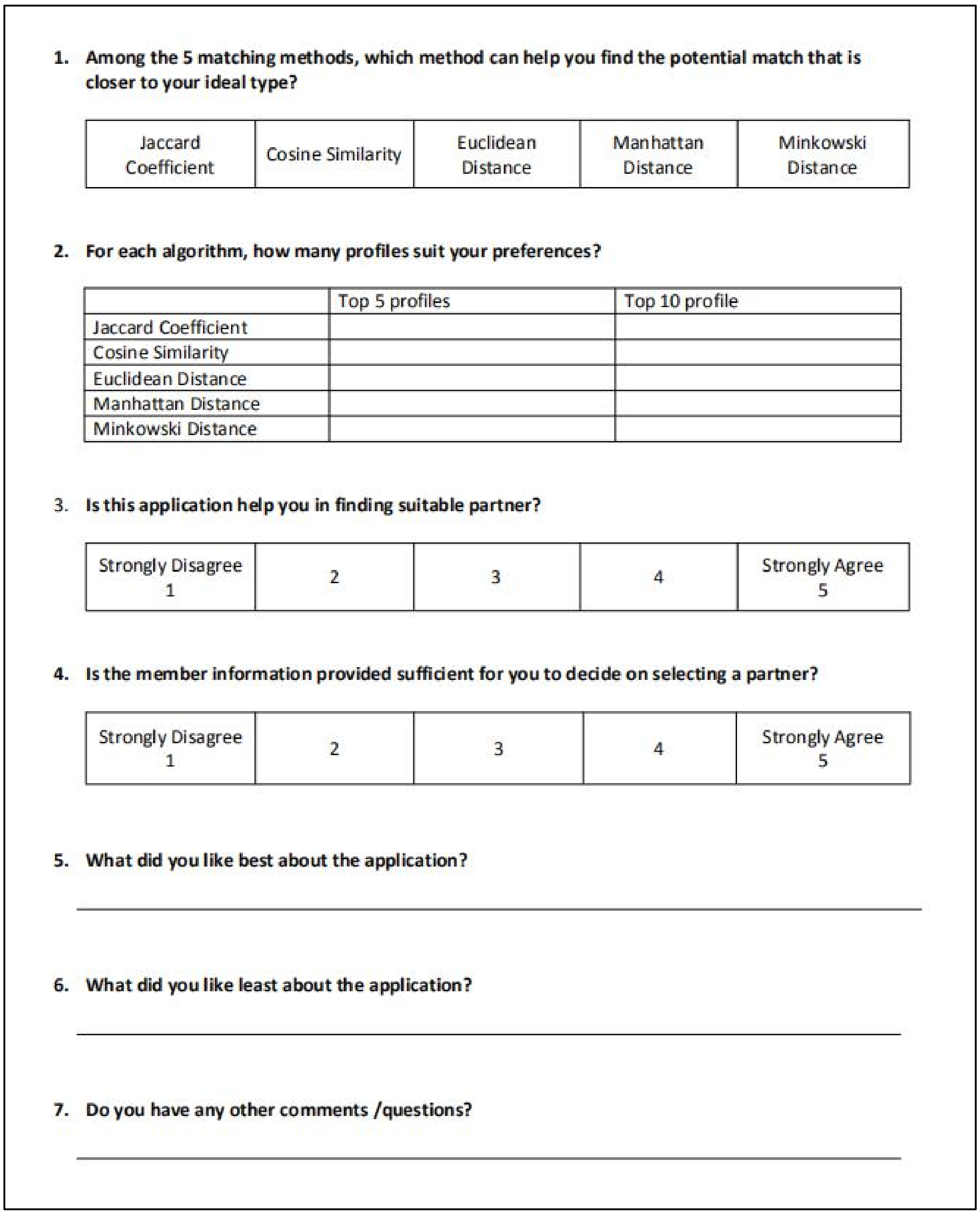


User Satisfaction Survey

1. Participant: #
2. Gender:
3. Age:
4. Do you ever use any dating/matrimonial app? (Yes / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently					
I found the system unnecessarily complex.					
I thought the system was easy to use.					
I think that I would need the support of a technical person to be able to use this system					
I found the various functions in this system were well integrated.					
I thought there was too much inconsistency in this system.					
I would imagine that most people would learn to use this system very quickly					
I found the system very cumbersome to use					
I felt very confident using the system					
I needed to learn a lot of things before I could get going with this system					

Figure 7.9 User Satisfaction Survey Form - Part 1



1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	Euclidean Distance	Manhattan Distance	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient		
Cosine Similarity		
Euclidean Distance		
Manhattan Distance		
Minkowski Distance		

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	4	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	4	Strongly Agree 5

5. **What did you like best about the application?**

____

6. **What did you like least about the application?**

____

7. **Do you have any other comments /questions?**

____

Figure 7.10 User Satisfaction Survey Form - Part 2

7.4.3 Usability Test Result

Based on the results obtained from the SUS survey, The SUS scores collected from each participant ranged from 70 – 92.5. The findings demonstrated that the mean of the SUS score was 80.5, which was far better than the minimum score required, 68. According to the grading scale defined by Sauro and Lewis (2012), the SUS score between 78.9 and 80.7 can be interpreted as a grade of A-. Grade A- reflects that the users have good experience with the application.

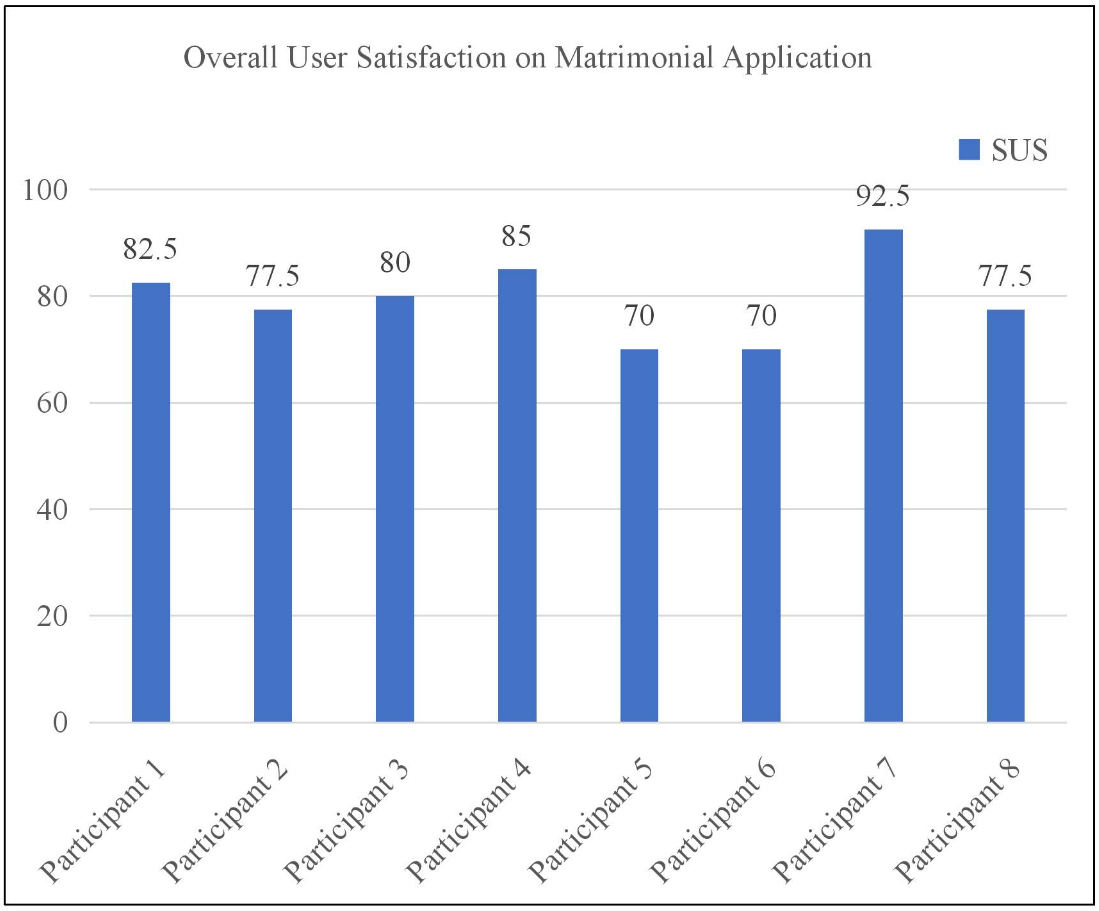


Figure 7.11 Overall User Satisfaction

The table below showed the feedback collected from 8 participants regarding the performance and functionality of the system. Some participants had provided some valuable comments for system improvement. Their suggestions will be considered in future enhancement.

Table 7.34 User Feedback on System's Performance and Functionality

Participant No.	**Like Best?**	**Like Least?**	**Comments**
1	Sufficient information to find the matching person	Need to repeat filling person criteria	Can use more simple words in the app or frequently used word
2	Display percentage	Always need to submit the preferences one more time when clicking the home navigation button.	Prefer multiple selection on the preferences data
3	More filtering options are provided	Need to move the mouse a lot	Non-responsive UI, can try to improve the user experience when using it
4	Able to display the degree of similarity regarding on my preferences with other members in this application	The delete function in the chat section	Some of the icons can make it bigger such as the profile icon (on the top-right side), better indication after user interaction with the system, such as the favourite button. Delete function in the chat section need some amendments, suggestions on displaying beside the chat box.
5	Different matching methods are very fascinating.	The matching method showed might be a little bit	The application is good if it's view as a dating app. If it's for marriage

	Explanation or justification provided is the best.	weird and difficult to comprehend for normal users.	app, more information provided would be way more superior.
6	Chat function	Have to refill preference every time login.	Change join us button to a brighter colour
7	Filter preferences and message people	Looking preferences does not update with search preferences	Maybe have more photo to explore
8	Display percentage	Need to resubmit the preferences for viewing the results after press the home button	Maybe can add a photo gallery that allows user to put more personal photos

In the survey form, the participants were also requested to select the match method that can help to find the potential matches close to their preferences. From the top 5 and top 10 search results returned by each similarity measure, they needed to elect the profiles that suit their preference. Table 7.35 showed the results obtained from the participants during usability testing. The numbers in bold indicated that the best similarity measures selected by the participants. For the top 5 search results, half of the participants selected Manhattan Distance as the best similarity measure. They satisfied with all of the top 5 search results. On the other hand, there were 5 participants satisfied with the top 10 search results of Manhattan Distance. The number of search results they agreed with was higher compared to other similarity measures.

Table 7.35 User Feedback on Similarity Measures

Participant	**Similarity Measures**									
	Top 5 Search Result					**Top 10 Search Result**				
	Jac	**Cos**	**Euc**	**Man**	**Min**	**Jac**	**Cos**	**Euc**	**Man**	**Min**
1	2	3	2	**5**	1	5	6	3	**7**	3
2	**3**	2	2	2	2	**5**	2	2	3	2
3	4	4	**5**	4	4	8	8	8	**9**	**9**
4	3	3	3	**5**	3	**6**	**6**	5	**6**	5
5	**4**	2	2	2	2	4	4	2	**7**	4
6	1	**3**	1	2	2	5	**7**	4	4	3
7	3	**5**	**5**	**5**	**5**	8	6	7	7	**9**
8	4	4	4	**5**	4	7	6	6	**9**	6
Notes: Jac = Jaccard Coefficient; Cos = Cosine Similarity; Euc = Euclidean Distance; Man = Manhattan Distance; Min = Minkowski Distance										

In conclusion, this survey showed that Manhattan Distance is more capable of helping the participants to find the potential matches close to their ideal type. For a application, it is essential to understand what suits the users well, what was their least favourite part and why. Since majority of the participants satisfied with the search results of Manhattan Distance, only Manhattan Distance will be implemented in the final system as the match method. User feedbacks were considered in order to increase user satisfactions.

7.5 User Acceptance Testing

User acceptance testing was conducted to ensure that the system's functionality fulfils the users' expectations. There was a total of 5 end-users took part in this testing. The following tables show the user acceptance test cases executed by the participants.

Table 7.36 User Acceptance Test Cases - Create Profile

ID	1	
Start Time		
End Time		
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account		
Able to insert personal information		
Able to upload profile picture		

Table 7.37 User Acceptance Test Cases - Search Potential Matches

ID	2	
Start Time		
End Time		
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences		
Able to view search results if there are profiles matched the preferences (Exact Matching)		
Able to edit match preferences		
Able to select different match methods		
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance		

Table 7.38 User Acceptance Test Cases - View Potential Matches Profiles

ID	3	
Start Time		
End Time		
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information		
Able to zoom member profile picture		
Able to add member to favourite		
Able to remove member from favourite		
Able to send instant message to member		
Able to like the member		

Table 7.39 User Acceptance Test Cases - Manage Chat History

ID	4	
Start Time		
End Time		
Module	Manage Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list		
Able to view the chat records with one of the members		
Able to delete the chat history		
Able to show confirmation message before delete the chat history		
Able to view the latest chat list		

Table 7.40 User Acceptance Test Cases - Manage Favourite List

ID	5	
Start Time		
End Time		
Module	Manage Favourite List	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view favourite list		
Able to view member profile information that added to favourite		
Able to remove member from favourite		
Able to show confirmation message before remove the profile from favourite		
Able to view the latest favourite list		

Table 7.41 User Acceptance Test Cases - Manage Personal Profile

ID	6	
Start Time		
End Time		
Module	Manage Personal Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view personal profile information		
Able to update profile picture		
Able to view number of likes received		
Able to update profile information		
Able to update match preferences		

CHAPTER 8

CONCLUSION AND RECOMMENDATIONS

8.1 Conclusion

This project had been completed within six months by following the Software Development Life Cycle. According to the research performed in the planning phase, it had been found that online dating has become a new trend for people to meet their potential life partner. Besides, the marriage rate in Malaysia declined over the last few years. Some people feel that it is not easy to find the right partner as the pool of suitable candidate might become smaller due to age increases. It was also discovered that there are some limitations to the rule-based approach and SQL query.

Therefore, with the matrimonial application, individuals can find their potential matches based on their preferences. After understanding the problem domains, the project objectives were declared as follow:

- To develop a web-based application by providing a solution that enables an individual to find their potential matches for marriage as per their priorities.
- To perform matching through similarity measures based on the requirements and priorities set by users.

Various research had been done on several existing similar applications, matching algorithms, software development methodologies and usability testing. All the findings were analysed and translated into the requirements of the application. Moreover, a survey was conducted to analyse the needs of end-users on the match preferences.

After all the necessary specifications had been gathered, the application was designed to satisfy the requirements. Use case diagram, system architecture diagram, data model diagram and preliminary user interfaces design were created to provide a deeper insight into how the application operates.

The development process of the matrimonial application was divided into several phases. The application was developed based on the system design specifications defined in the previous phase. Five similarity measures were implemented in the application, namely Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance.

After the implementation phase completed, unit testing, API testing and end-to-end testing were conducted to ensure that the application will operate as expected. Usability testing was used to evaluate which similarity measure is best suited for the application. Through usability testing, Manhattan Distance was found to be more appropriate for the matrimonial application. Lastly, user acceptance testing was conducted to ensure that the system's functionality fulfils the users' expectations.

In conclusion, a similarity measure based matrimonial application had been delivered at the end of the development life cycle. All the objectives had been accomplished, which were:

- Develop the web-based matrimonial application that enables an individual to find their potential matches for marriage as per their priorities.
- Implement the similarity measures in the application that enables the users to search for a potential partner based on their requirements.

8.2 Limitation and Suggestion

Although the system developed has met all the requirements, there are some limitations to be noted. First, through usability testing, it was found that most of the participants preferred using Manhattan Distance as the match method. During the test, the participants tended to focus more on appearance when selecting for the matched profiles. Hence, the participants had been advised to make their selection based on the member criteria rather than appearance.

It was unable to deny that physical attraction is a significant factor in selecting a partner. According to the study from Ha, Overbeek and Engels (2009), people tend to pursue relationships with those who attractive to them. Some participants also mentioned that the first impression is very important while selecting the partner that match their preferences.

A "Like" function had been added into the application to deal with this condition. The users can express their preferences for each profile by clicking the "Like" button. This is a quantitative measurement to reflect the physical appearance. The number of likes will indicate the popularity of a profile. However, the "Like" function was just a temporary solution. In future research and development, a better approach may be provided to include physical appearance into the measurement.

Second, the dataset collected for the usability testing is small because it only consists of 40 male data and 45 female data. A male user, for example, can only search through 45 female members for potential matches. Besides, the results for the five similarity measures will be similar due to the small dataset. Hence, future research may need to be conducted once more data is collected.

8.3 Future Enhancements

Although the application had been completed, there was still room for improvement. Besides, the participants of usability testing had given some valuable recommendations for potential improvement of the application. Those recommendations will also be considered in future enhancement to improve system functionality.

- Enable multiple selections on match preferences. For example, users can choose "Chinese" and "English" as their preferred mother tongue.
- Develop a profile photo album function that enables the users to upload more profile photo.
- Collect more user detailed information when the users register their accounts. For example, the application will prompt users for their annual income, weight and family background.
- Enhance member profile interface by highlighting the profile details that match user's preferences. This can help the users to identify the matched member details more easily.
- Develop a function that will recommend some potential matches for user everyday, which is similar to Double Take feature in OkCupid.com. The users can choose to approach or "pass" the recommended profile.

REFERENCES

Almeida, F., 2016. *Practical SQL Guide For Relational Databases*. [ebook] ISSUU Publishing, pp.8-22. Available at: <https://www.researchgate.net/publication/319852714_Practical_SQL_Guide_for_Relational_Databases> [Accessed 15 March 2020].

Alshamrani, A. and Bahattab, A., 2015. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, [online] 12(1), pp.106-111. Available at: <https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf> [Accessed 12 Feb. 2020].

Arnaboldi, V., Passarella, A., Conti, M., Dunbar, R. I. M., 2015. *Online social networks : human cognitive constraints in Facebook and Twitter personal graphs*. Amsterdam: Elsevier.

Babich, N., 2017. *A Guide To The Art Of Guerrilla UX Testing*. [online] Medium. Available at: <https://medium.springboard.com/a-guide-to-the-art-of-guerrilla-ux-testing-69a1411d34fb> [Accessed 12 March 2020].

Beynon-Davies, P., Tudhope, D. and Mackay, H.,1999. Information Systems Prototyping in Practice. *Journal of Information Technology*, [online] 14(1), pp.107-120. Available at: <https://www.researchgate.net/publication/233602440_Information_systems_prototyping_in_practice> [Accessed 21 Feb. 2020].

Bourgeois, D. T., 2014. *Information Systems For Business And Beyond*. 1st ed. [ebook] Saylor Academy Open Textbook Challenge. Available at: <https://bus206.pressbooks.com/> [Accessed 13 March 2020].

Campbell, L., Chin, K. and Stanton, S., 2016. Initial Evidence that Individuals Form New Relationships with Partners that More Closely Match their Ideal Preferences. *Collabra*, 2(1), p.2.

Chen, S. C., Gulatit, S., Hamid, S., Huang, X., Luo, L., Morisseau-Leroy, N., Powell, M. D., Zhan, C. and Zhang, C., 2003. A three-tier system architecture design and development for hurricane occurrence simulation. In: *International Conference on Information Technology: Research and Education*. [online] Newark, New Jersey, USA: IEEE, pp.113-117. Available at: <https://pdfs.semanticscholar.org/5a99/d4cdc10988ba2c7a920d1ac2d06b9cdc3eef.pdf> [Accessed 24 June 2020].

Collado, J., Mora, P. and Parham, E., 2013. A guerrilla usability lab with free software. *interactions*, [online] 20(3), p.62. Available at:

<https://interactions.acm.org/archive/view/may-june-2013/a-guerrilla-usability-lab-with-free-software> [Accessed 7 Mar. 2020].

Department of Statistics Malaysia, 2019a. *Marriage and Divorce Statistics, Malaysia, 2019*. [online] Available at: <https://newss.statistics.gov.my/newss-portalx/ep/epFreeDownloadContentSearch.seam?cid=198734> [Accessed 15 Feb. 2020].

Department of Statistics Malaysia, 2019b. *Vital Statistics, Malaysia, 2019*. [online] Available at: <https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=165&bul_id=bE5PcWNLaFBMUy9FK3ZhOEpTdG0xdz09&menu_id=L0pheU43NWJwRWVSZklWdzQ4TlhUUT09> [Accessed 15 Feb. 2020].

Dicks, R. S., 2002. Mis-Usability: On the Uses and Misuses of Usability Testing. In: *SIGDOC 02: Proceedings of the 20st annual international conference on Computer documentation*. [online] New York, United States: ACM, pp.26-30. Available at: <https://tecfa.unige.ch/tecfa/maltt/ergo/1415/UtopiaPeriode4/articles/Dicks_2002.pdf > [Accessed 7 Mar. 2020].

Finkel, E. J., Eastwick, P. W., Karney, B. R., Reis, H. T., & Sprecher, S., 2012. Online Dating: A Critical Analysis From the Perspective of Psychological Science, *Psychological Science in the Public Interest,* 13(1), p3-66.

Firebase, 2020. *Privacy and Security in Firebase*. [online] Available at: <https://firebase.google.com/support/privacy> [Accessed 8 Feb. 2020].

Geambasu, C., Jianu, I., Jianu, I. and Gavrila, A., 2011. INFLUENCE FACTORS FOR THE CHOICE OF A SOFTWARE DEVELOPMENT METHODOLOGY. *Accounting and Management Information Systems*, [online] 10(4), pp.479-494. Available at: <https://core.ac.uk/download/pdf/6261795.pdf >[Accessed 23 Feb. 2020].

Ha, T., Overbeek, G. and Engels, R.C., 2010. Effects of attractiveness and social status on dating desire in heterosexual adolescents: An experimental study. *Archives of Sexual Behavior*, *39*(5), pp.1063-1071.

Internetworldstats.com., 2020. *World Internet Users Statistics and 2019 World Population Stats*. [online] Available at: <https://www.internetworldstats.com/stats.htm> [Accessed: 4 Feb. 2020].

Kannan, V., Jhajharia, S. and Verma, S., 2020. Agile vs waterfall: A Comparative Analysis. *International Journal of Science, Engineering and Technology Research (IJSETR)*, [online] 3(10), pp.2680-2686. Available at: <http://ijsetr.org/wp-content/uploads/2014/10/IJSETR-VOL-3-ISSUE-10-2680-2686.pdf> [Accessed 19 Feb. 2020].

Kwasny S.C., Faisal K.A., 1990. Overcoming Limitations of Rule-Based Systems: An Example of a Hybrid Deterministic Parser. In: G. Dorffner, eds. *Konnektionismus in Artificial Intelligence und Kognitionsforschung. Informatik-Fachberichte.* Berlin: Springer. pp.48-57.

Lewis, J. and Sauro, J., 2009. The Factor Structure of the System Usability Scale. In: *Human Centered Design: First International Conference, HCD 2009*. [online] Berlin, Heidelberg: Springer, pp.94-103. Available at: <https://measuringu.com/wp-content/uploads/2017/07/Lewis_Sauro_HCII2009.pdf> [Accessed 19 March 2020].

Liu B., Ma Y., Wong CK., 2001. Classification Using Association Rules: Weaknesses and Enhancements. In: R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, R. Namburu, ed. (2013) *Data Mining for Scientific and Engineering Applications. Massive Computing*. New York: Springer. pp.591-605.

Liu, H., Gegov, A. and Cocea, M., 2016. Rule-based systems: a granular computing perspective. *Granular Computing*, [online] 1(4), pp.259-274. Available at: <https://link.springer.com/article/10.1007/s41066-016-0021-6> [Accessed 13 March 2020].

Malaysia Population Research Hub, 2019. *Malaysian Getting Married Later in Life*. [online] Available at: <http://mprh.lppkn.gov.my/2019/04/09/malaysian-getting-married-later-in-life/> [Accessed 15 Feb. 2020].

Moran, K. and Pernice, K., 2020. *Remote Moderated Usability Tests: How and Why to Do Them*. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/moderated-remote-usability-test/> [Accessed 7 Mar. 2020].

NICHD (National Institute of Child Health and Human Development), 2020. *What age-related factors may be involved with infertility in females and males?*. [online] Available at: <https://www.nichd.nih.gov/health/topics/infertility/conditioninfo/causes/age> [Accessed 18 Feb. 2020].

Nielsen, J., 1993. *Usability engineering*. 1st ed. United States: Academic Press, Inc., pp.26-37.

Polamuri, S., 2015. *Five most popular similarity measures implementation in python*. [online] Available at: <https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/> [Accessed 1 Mar. 2020].

Reactjs.org., 2020. *React – A JavaScript library for building user interfaces*. [online] Available at: <https://reactjs.org/> [Accessed 8 Feb. 2020].

Rodríguez-Martínez, L. C., Mora, M. and Alvarez, J., 2009. *A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles (PM-SDLCs)*. In: 2009 Mexican International Conference on Computer Science. [online] IEEE, pp.298-303. Available at: <https://ieeexplore.ieee.org/document/5452522> [Accessed 20 Feb. 2020].

Rosenfeld, M., Thomas, R. and Hausen, S., 2019. Disintermediating your friends: How online dating in the United States displaces other ways of meeting. *Proceedings of the National Academy of Sciences*, 116(36), pp.17753-17758.

Ross, T., 2010. Fuzzy logic with engineering applications. 3rd ed. USA: John Wiley & Sons Ltd.

Sauro, J. and Lewis, J. R., 2012. *Quantifying The User Experience: Practical Statistics For User Research*. United State: Elsevier Inc., pp.203-204.

Sharma, S., Sarkar, D. and Gupta, D., 2012. Agile Processes and Methodologies : A Conceptual Study. *International Journal on computer science and Engineering*, [online] 4(5), pp.892–899. Available at: <https://www.researchgate.net/publication/267706023_Agile_Processes_and_Methodologies_A_Conceptual_Study> [Accessed 21 February 2020].

Shirkhorshidi, A., Aghabozorgi, S. and Wah, T., 2015. A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. *PLOS ONE*, 10(12).

Stevenson, B. and Wolfers, J., 2007. Marriage and Divorce: Changes and Their Driving Forces. *Journal of Economic Perspectives*, 21(2), pp.27-52.

Tegarden, D., Dennis, A. and Wixom, B., 2009. *Systems analysis design UML version 2.0*. 3rd ed. United States: John Wiley & Sons.

Usability.gov., 2020a. *Usability Testing | Usability.gov*. [online] Available at: <https://www.usability.gov/how-to-and-tools/methods/usability-testing.html> [Accessed 1 Mar. 2020].

Usability.gov., 2020b. *System Usability Scale (SUS) | Usability.Gov*. [online] Usability.gov. Available at: <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html> [Accessed 19 March 2020].

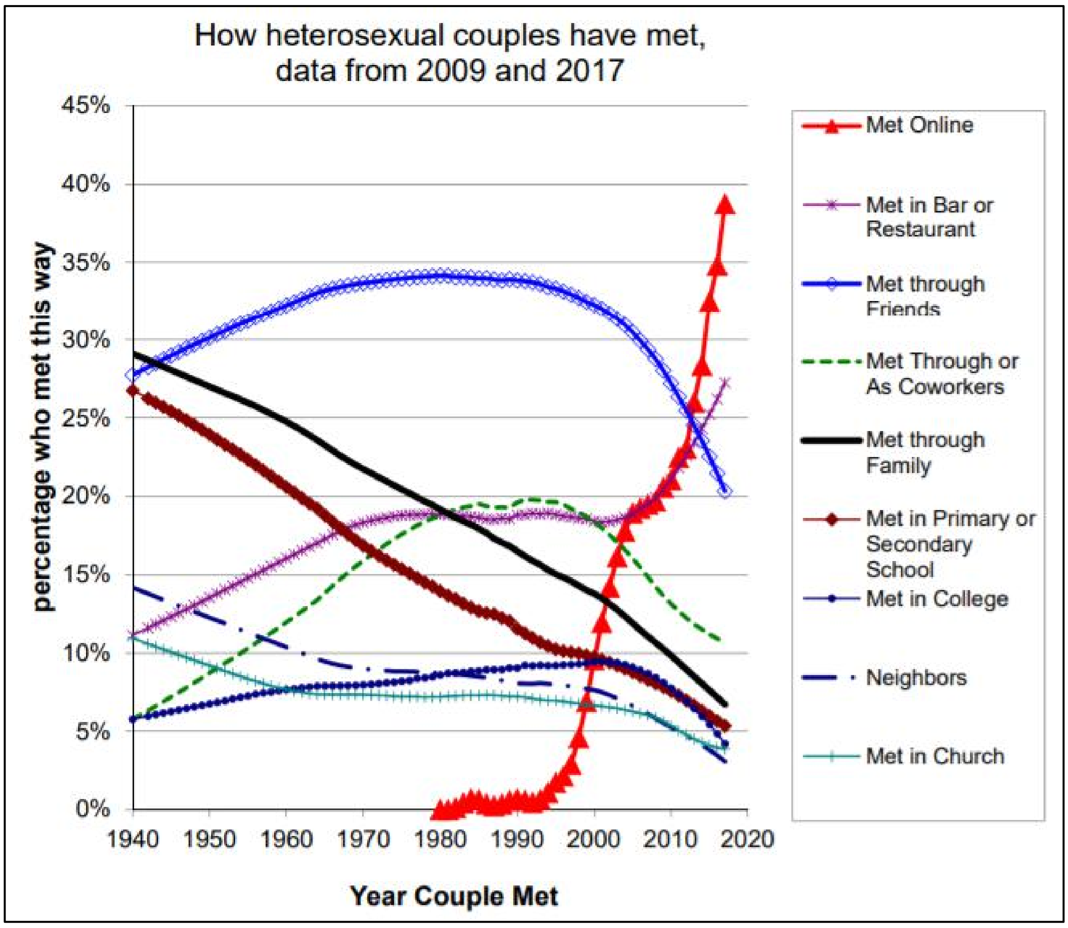
Usability.gov., 2020c. *Remote Testing | Usability.gov*. [online] Available at: <https://www.usability.gov/how-to-and-tools/methods/remote-testing.html> [Accessed 7 Mar. 2020].

Yadav, K., Yasvi, M. and Shubhika, 2019. Review On Extreme Programming-XP. In: *International Conference on Robotics, Smart Technology and Electronics Engineering*. [online] Available at: https://www.researchgate.net/publication/332465869_Review_On_Extreme_Programming-XP [Accessed 23 Feb. 2020].

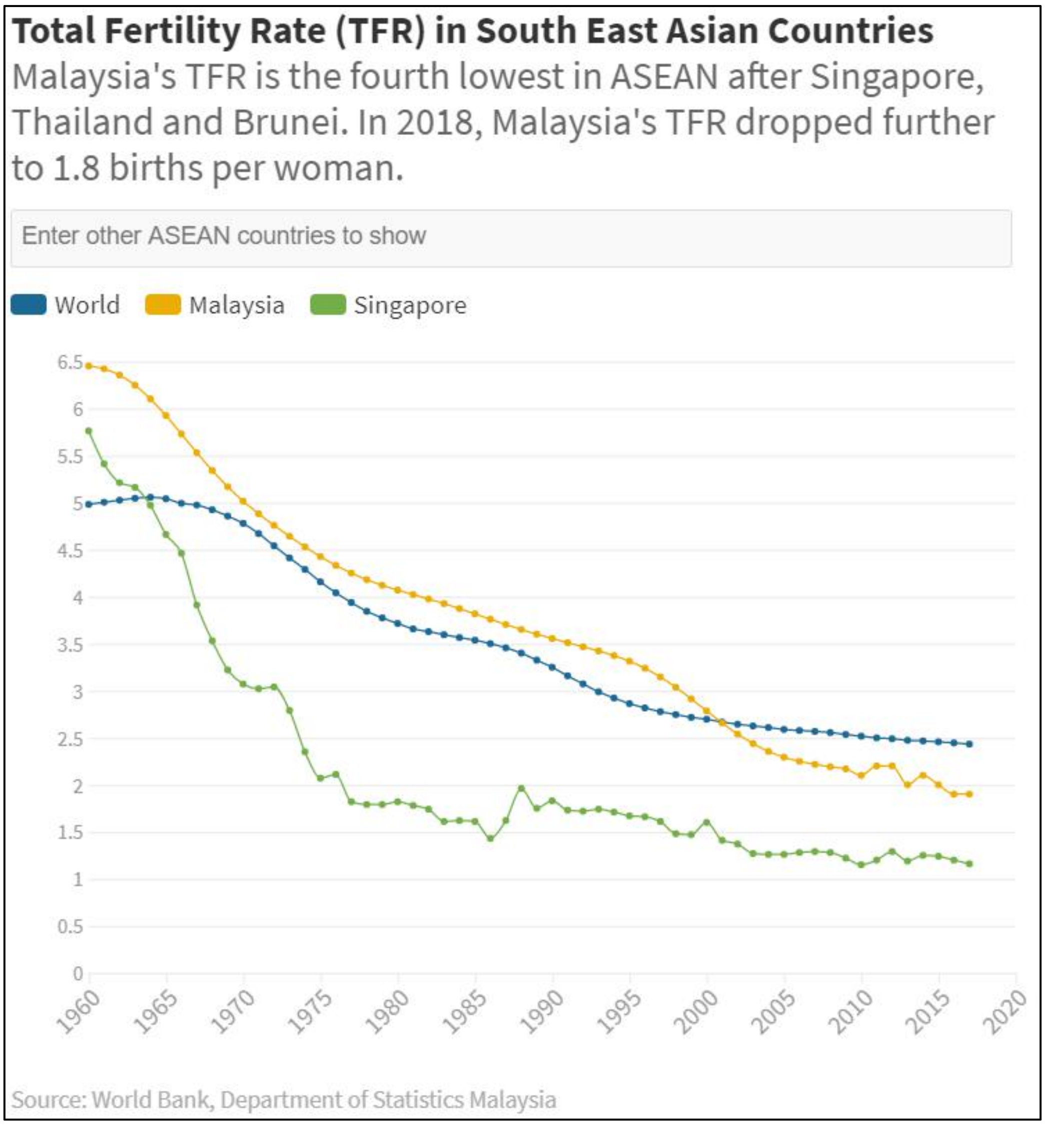
Yuen, M. K., 2019. *Feature: Malaysia's shrinking families*. [online] The Star Online. Available at: <https://www.thestar.com.my/news/nation/2019/11/10/malaysia039s-shrinking-families> [Accessed 15 Feb. 2020].

APPENDICES

APPENDIX A: The continued rise of meeting online for heterosexual couples



APPENDIX B: Total Fertility Rate in Malaysia

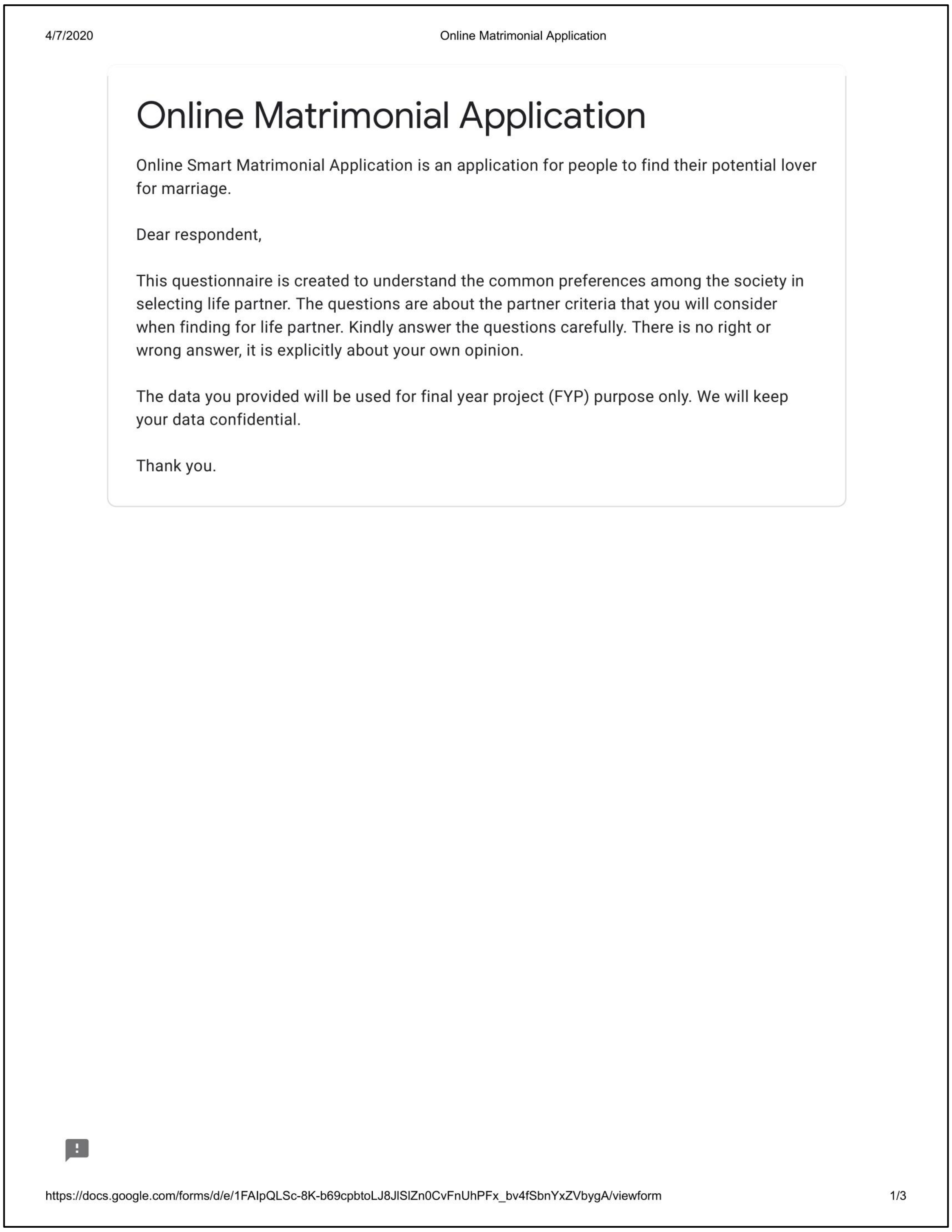


Total Fertility Rate (TFR) in South East Asian Countries

Malaysia's TFR is the fourth lowest in ASEAN after Singapore, Thailand and Brunei. In 2018, Malaysia's TFR dropped further to 1.8 births per woman.

Source: World Bank, Department of Statistics Malaysia

APPENDIX C: Questionnaire for Collecting Match Preferences



4/7/2020 Online Matrimonial Application

Online Matrimonial Application

Online Smart Matrimonial Application is an application for people to find their potential lover for marriage.

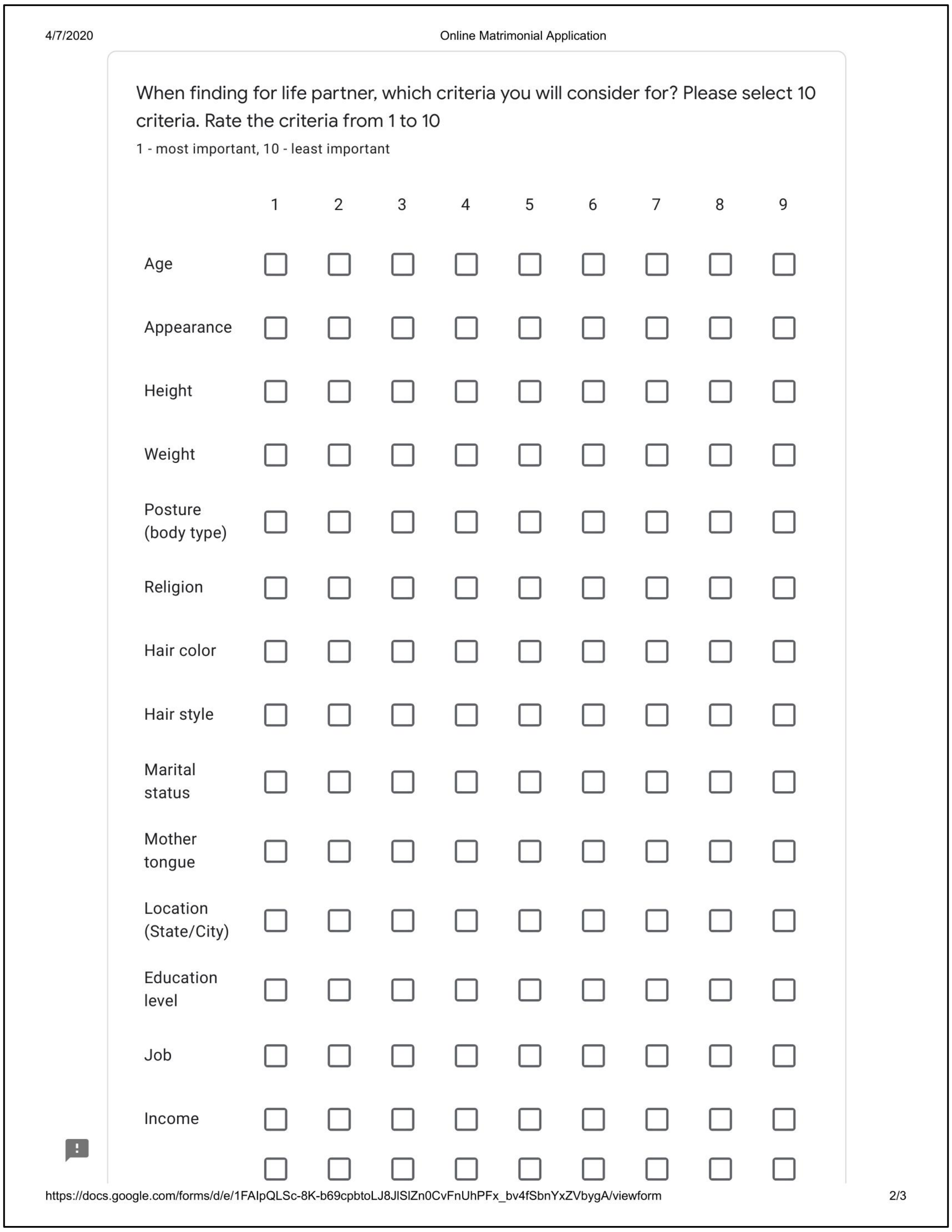
Dear respondent,

This questionnaire is created to understand the common preferences among the society in selecting life partner. The questions are about the partner criteria that you will consider when finding for life partner. Kindly answer the questions carefully. There is no right or wrong answer, it is explicitly about your own opinion.

The data you provided will be used for final year project (FYP) purpose only. We will keep your data confidential.

Thank you.

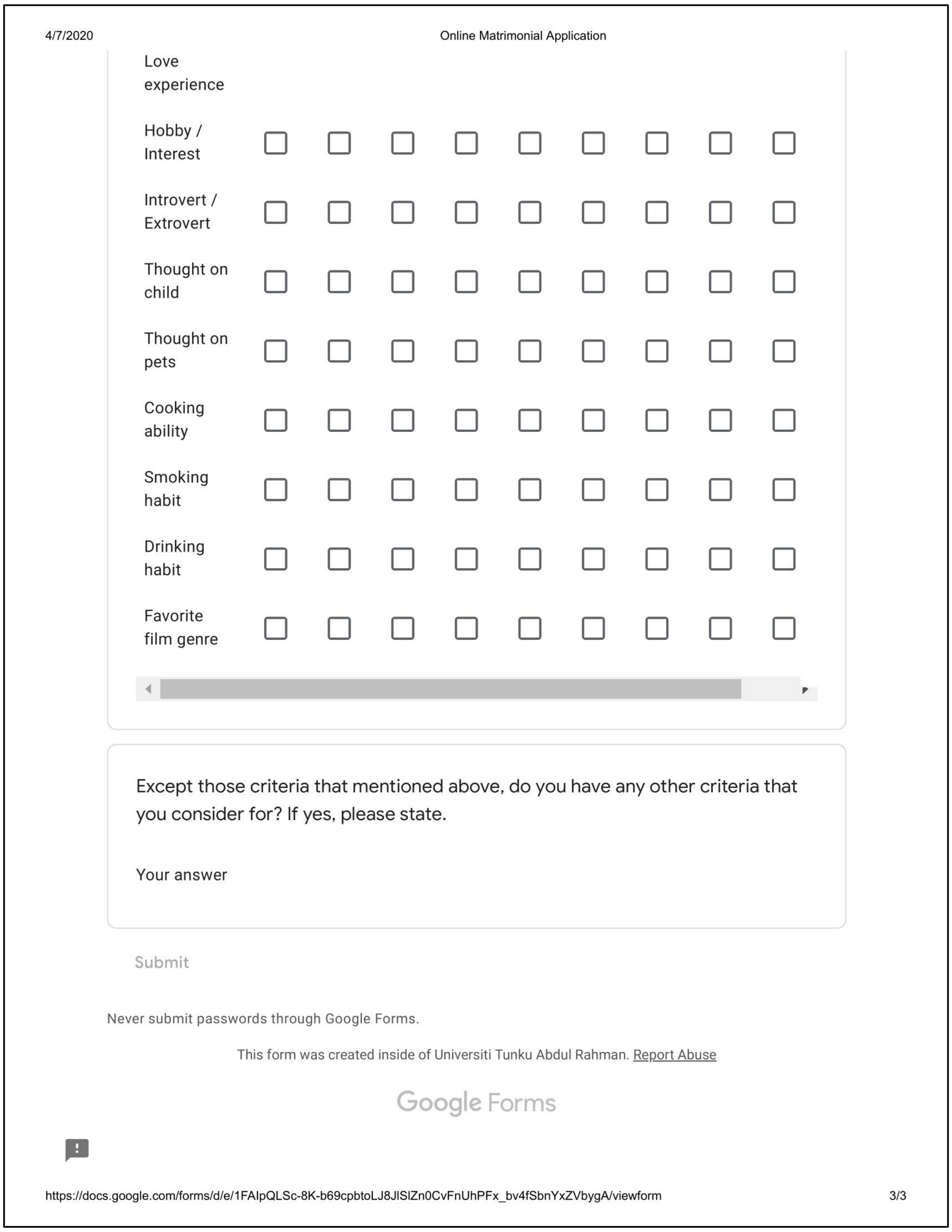
https://docs.google.com/forms/d/e/1FAIpQLSc-8K-b69cpbtoLJ8JlSlZn0CvFnUhPFx_bv4fSbnYxZVbygA/viewform 1/3



When finding for life partner, which criteria you will consider for? Please select 10 criteria. Rate the criteria from 1 to 10

1 - most important, 10 - least important

	1	2	3	4	5	6	7	8	9
Age	☐	☐	☐	☐	☐	☐	☐	☐	☐
Appearance	☐	☐	☐	☐	☐	☐	☐	☐	☐
Height	☐	☐	☐	☐	☐	☐	☐	☐	☐
Weight	☐	☐	☐	☐	☐	☐	☐	☐	☐
Posture (body type)	☐	☐	☐	☐	☐	☐	☐	☐	☐
Religion	☐	☐	☐	☐	☐	☐	☐	☐	☐
Hair color	☐	☐	☐	☐	☐	☐	☐	☐	☐
Hair style	☐	☐	☐	☐	☐	☐	☐	☐	☐
Marital status	☐	☐	☐	☐	☐	☐	☐	☐	☐
Mother tongue	☐	☐	☐	☐	☐	☐	☐	☐	☐
Location (State/City)	☐	☐	☐	☐	☐	☐	☐	☐	☐
Education level	☐	☐	☐	☐	☐	☐	☐	☐	☐
Job	☐	☐	☐	☐	☐	☐	☐	☐	☐
Income	☐	☐	☐	☐	☐	☐	☐	☐	☐
	☐	☐	☐	☐	☐	☐	☐	☐	☐



Love experience									
Hobby / Interest	☐	☐	☐	☐	☐	☐	☐	☐	☐
Introvert / Extrovert	☐	☐	☐	☐	☐	☐	☐	☐	☐
Thought on child	☐	☐	☐	☐	☐	☐	☐	☐	☐
Thought on pets	☐	☐	☐	☐	☐	☐	☐	☐	☐
Cooking ability	☐	☐	☐	☐	☐	☐	☐	☐	☐
Smoking habit	☐	☐	☐	☐	☐	☐	☐	☐	☐
Drinking habit	☐	☐	☐	☐	☐	☐	☐	☐	☐
Favorite film genre	☐	☐	☐	☐	☐	☐	☐	☐	☐

Except those criteria that mentioned above, do you have any other criteria that you consider for? If yes, please state.

Your answer

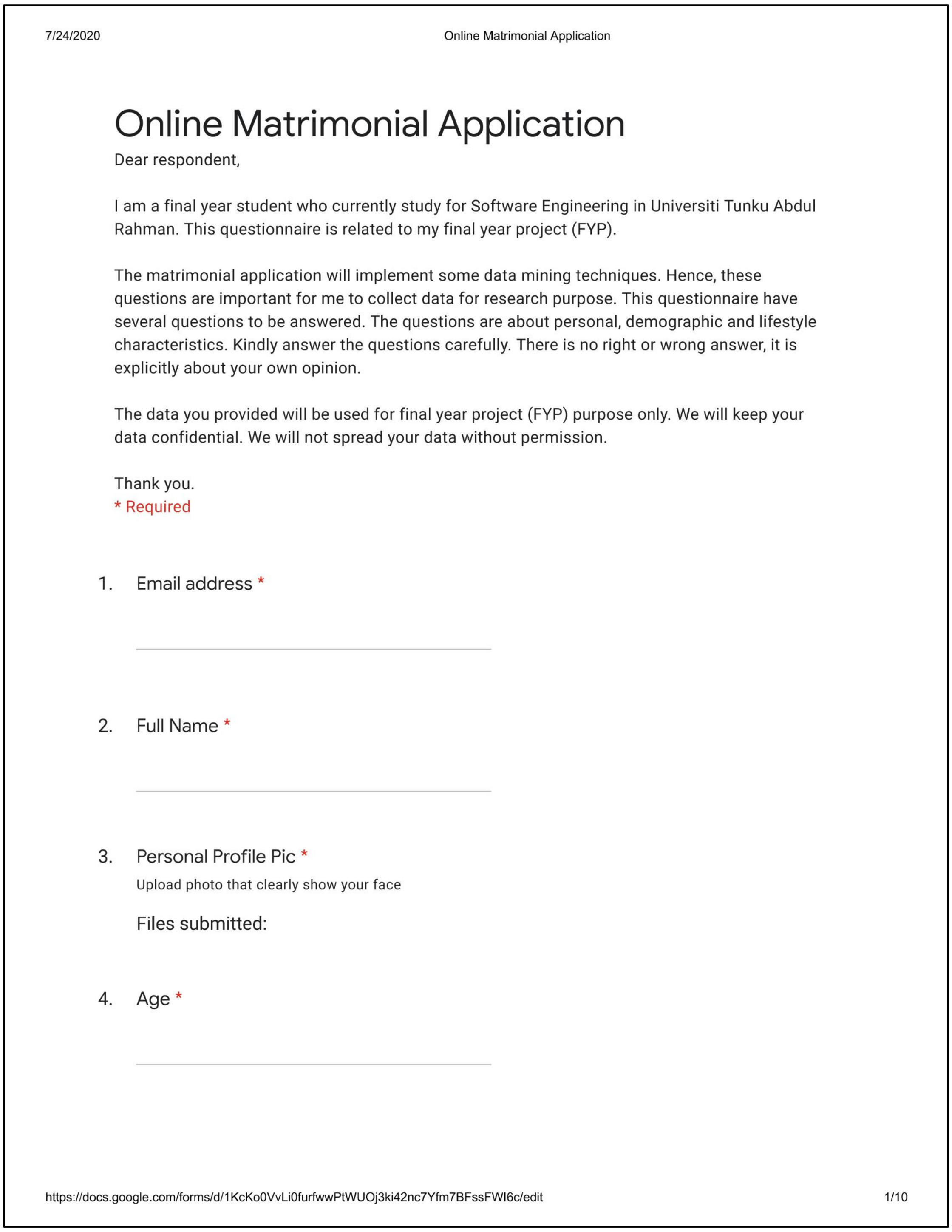
Submit

Never submit passwords through Google Forms.

This form was created inside of Universiti Tunku Abdul Rahman. Report Abuse

Google Forms

APPENDIX D: Questionnaire for Collecting User Data



Online Matrimonial Application

Dear respondent,

I am a final year student who currently study for Software Engineering in Universiti Tunku Abdul Rahman. This questionnaire is related to my final year project (FYP).

The matrimonial application will implement some data mining techniques. Hence, these questions are important for me to collect data for research purpose. This questionnaire have several questions to be answered. The questions are about personal, demographic and lifestyle characteristics. Kindly answer the questions carefully. There is no right or wrong answer, it is explicitly about your own opinion.

The data you provided will be used for final year project (FYP) purpose only. We will keep your data confidential. We will not spread your data without permission.

Thank you.
* Required

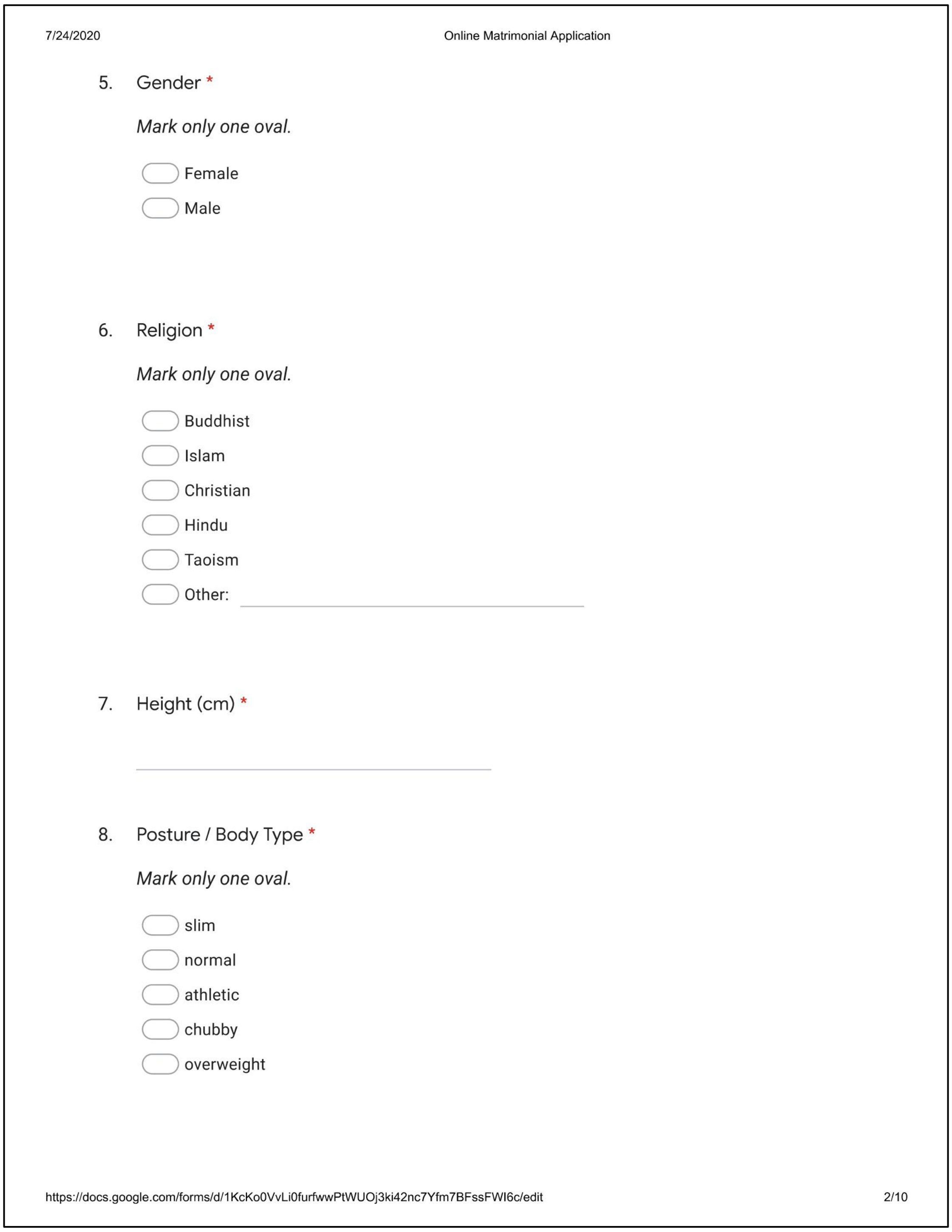
1. Email address *

2. Full Name *

3. Personal Profile Pic *
   Upload photo that clearly show your face

   Files submitted:

4. Age *



5. Gender *

Mark only one oval.

◯ Female
◯ Male

6. Religion *

Mark only one oval.

◯ Buddhist
◯ Islam
◯ Christian
◯ Hindu
◯ Taoism
◯ Other: ____________________

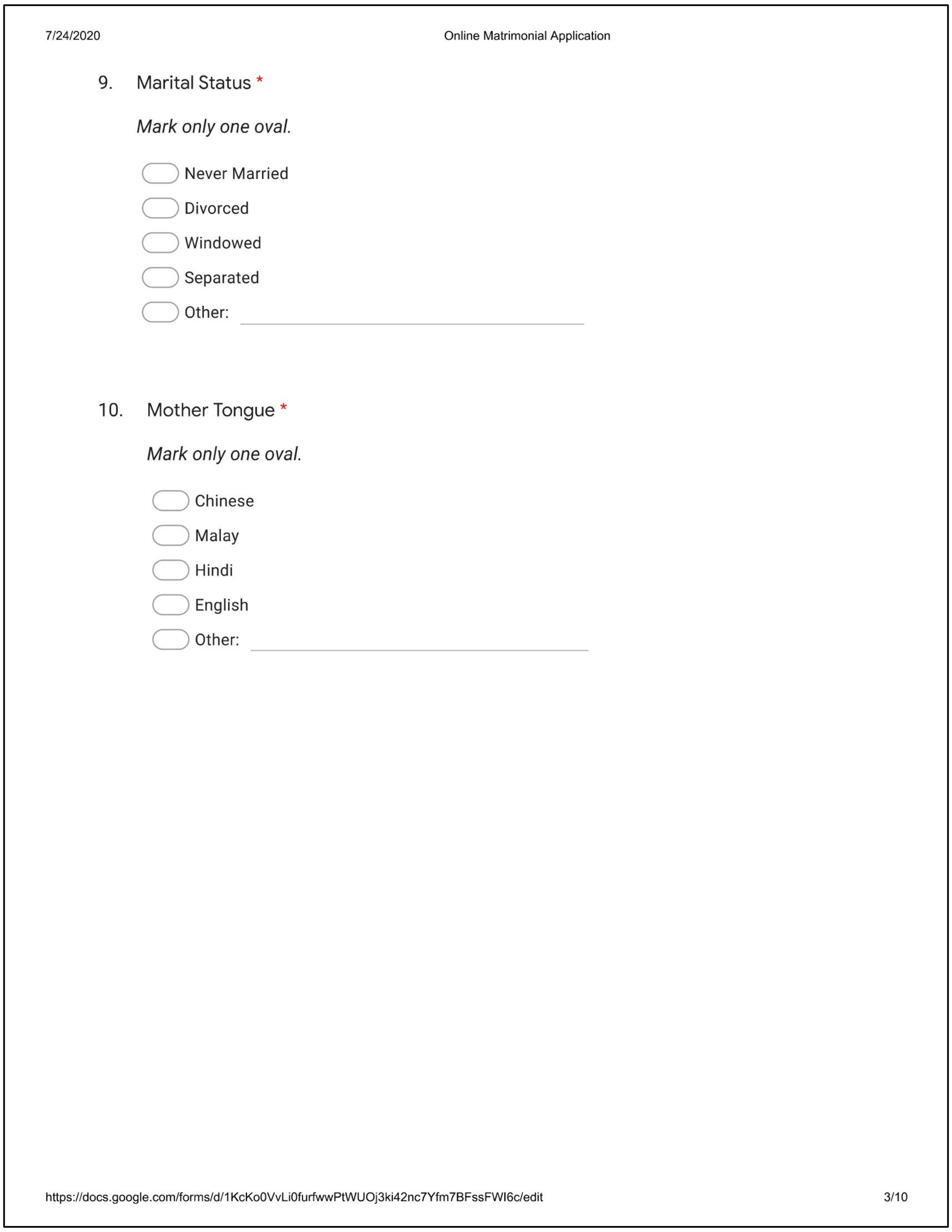
7. Height (cm) *

____

8. Posture / Body Type *

Mark only one oval.

◯ slim
◯ normal
◯ athletic
◯ chubby
◯ overweight



9. Marital Status *

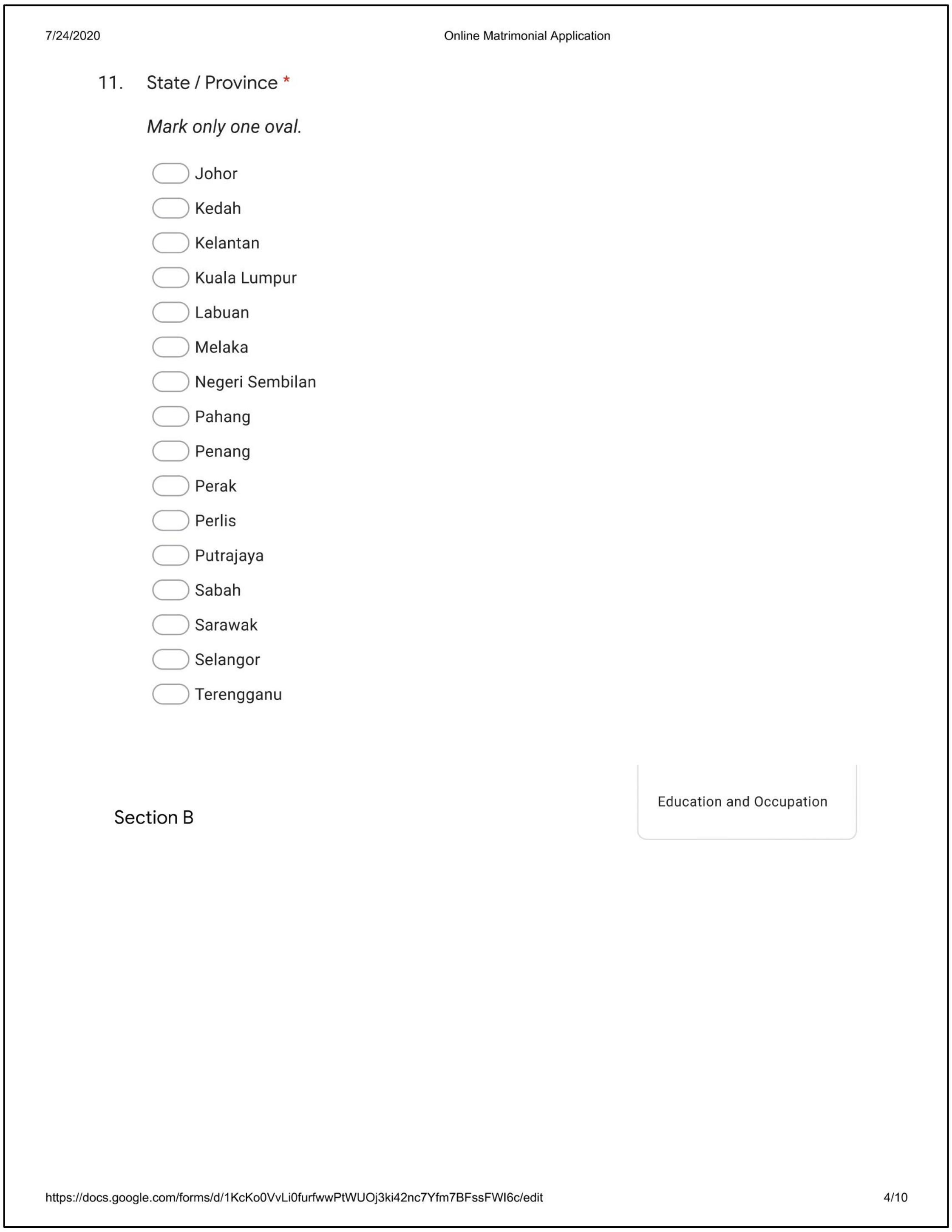
Mark only one oval.

- Never Married
- Divorced
- Windowed
- Separated
- Other: ________________________

10. Mother Tongue *

Mark only one oval.

- Chinese
- Malay
- Hindi
- English
- Other: ________________________



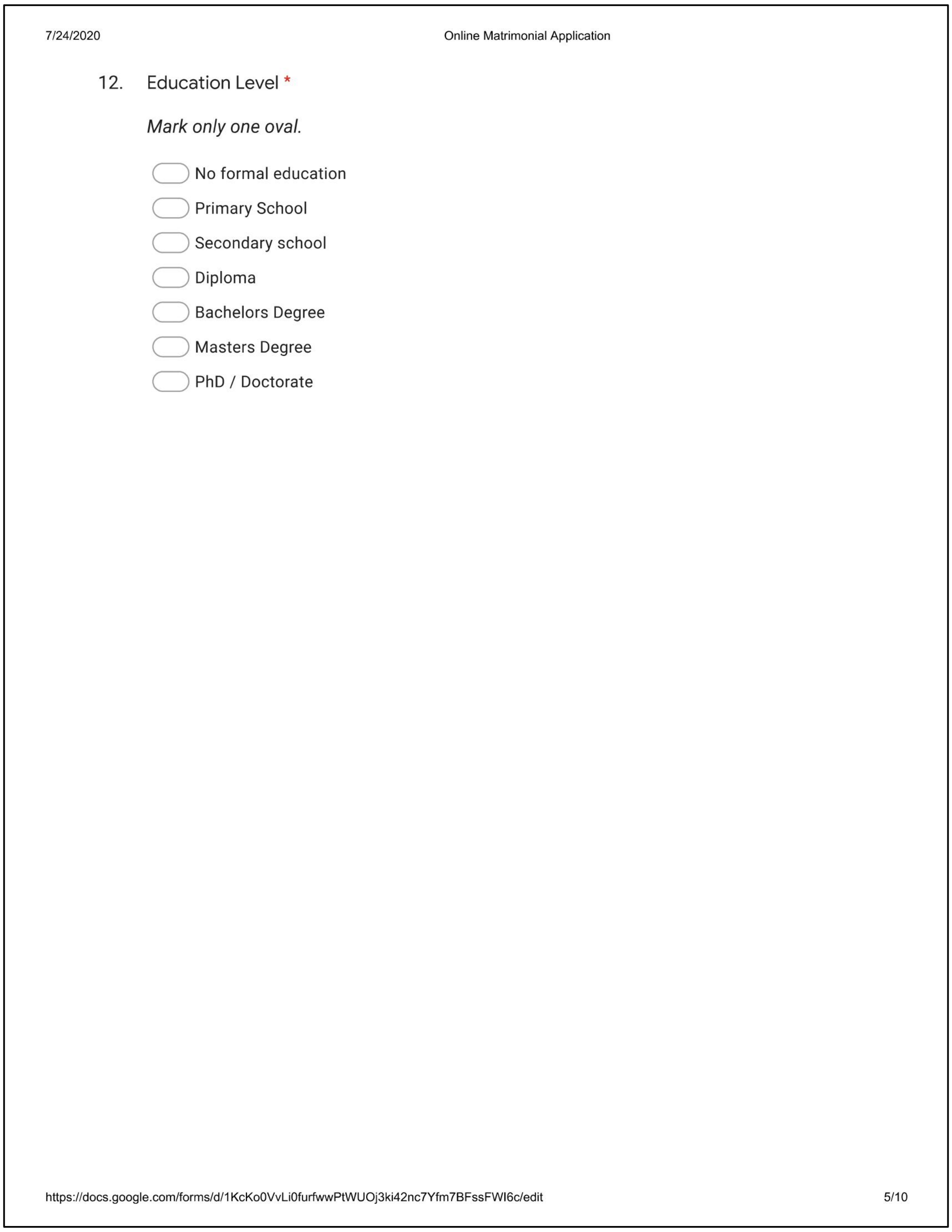
11. State / Province *

Mark only one oval.

- Johor
- Kedah
- Kelantan
- Kuala Lumpur
- Labuan
- Melaka
- Negeri Sembilan
- Pahang
- Penang
- Perak
- Perlis
- Putrajaya
- Sabah
- Sarawak
- Selangor
- Terengganu

Section B

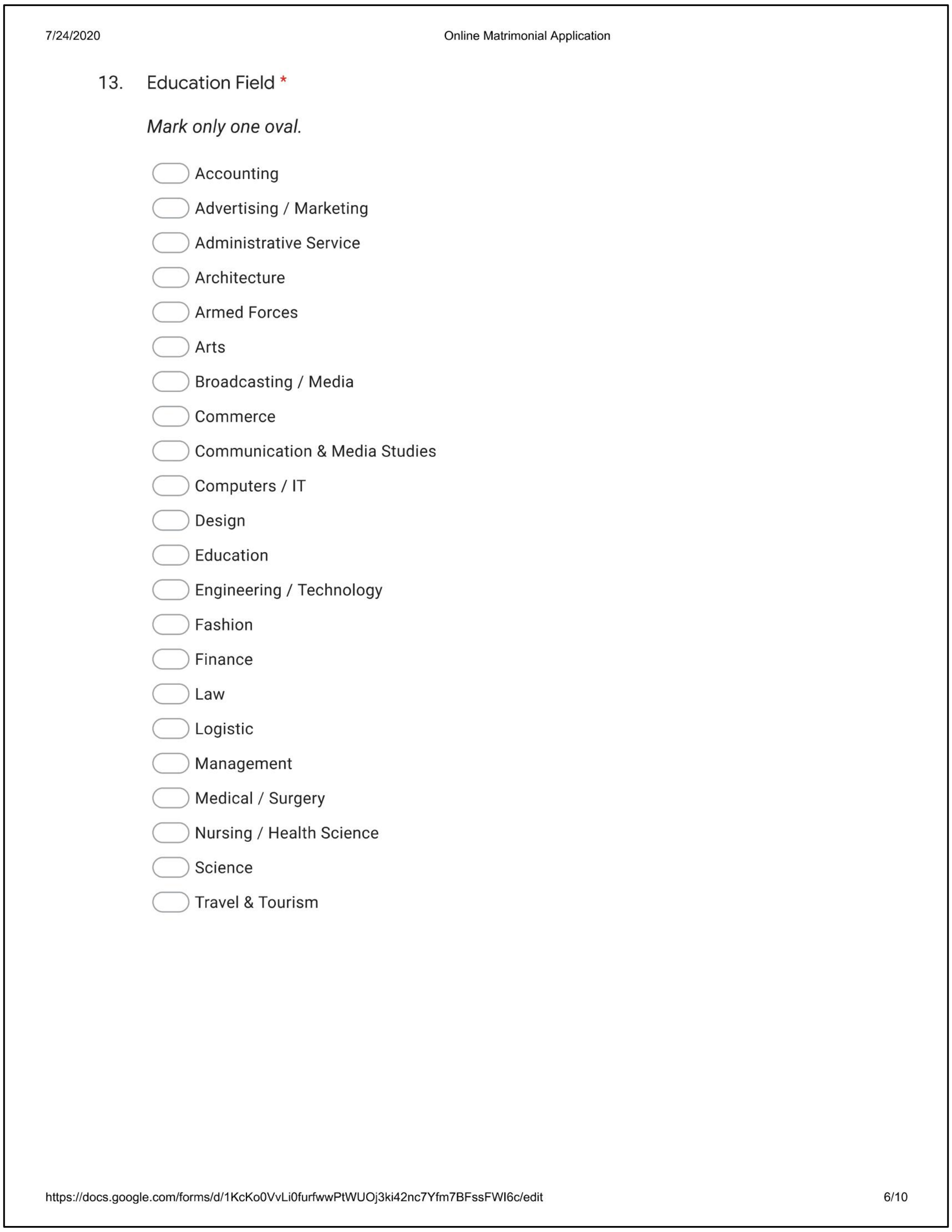
Education and Occupation



12. Education Level *

Mark only one oval.

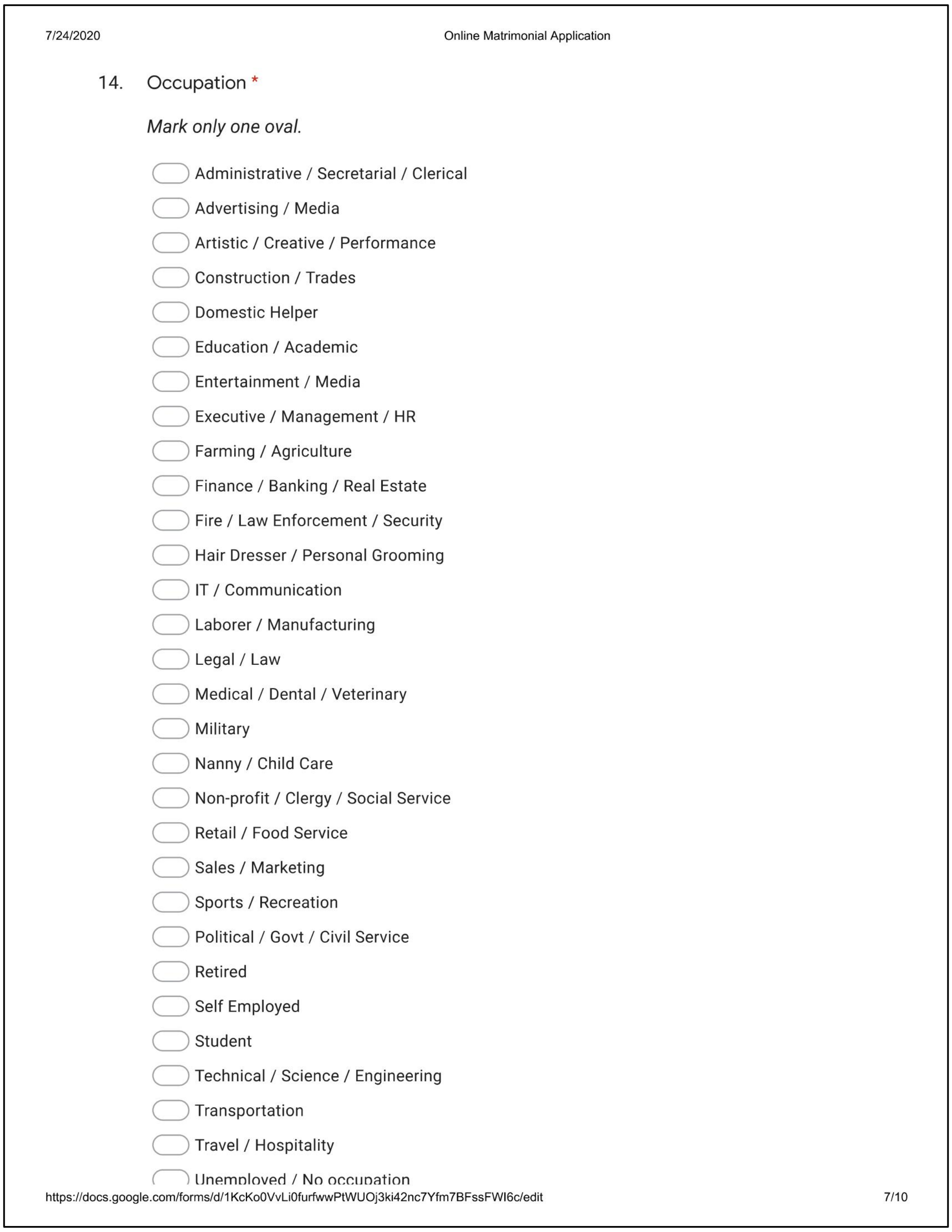
- No formal education
- Primary School
- Secondary school
- Diploma
- Bachelors Degree
- Masters Degree
- PhD / Doctorate



13. Education Field *

Mark only one oval.

- Accounting
- Advertising / Marketing
- Administrative Service
- Architecture
- Armed Forces
- Arts
- Broadcasting / Media
- Commerce
- Communication & Media Studies
- Computers / IT
- Design
- Education
- Engineering / Technology
- Fashion
- Finance
- Law
- Logistic
- Management
- Medical / Surgery
- Nursing / Health Science
- Science
- Travel & Tourism



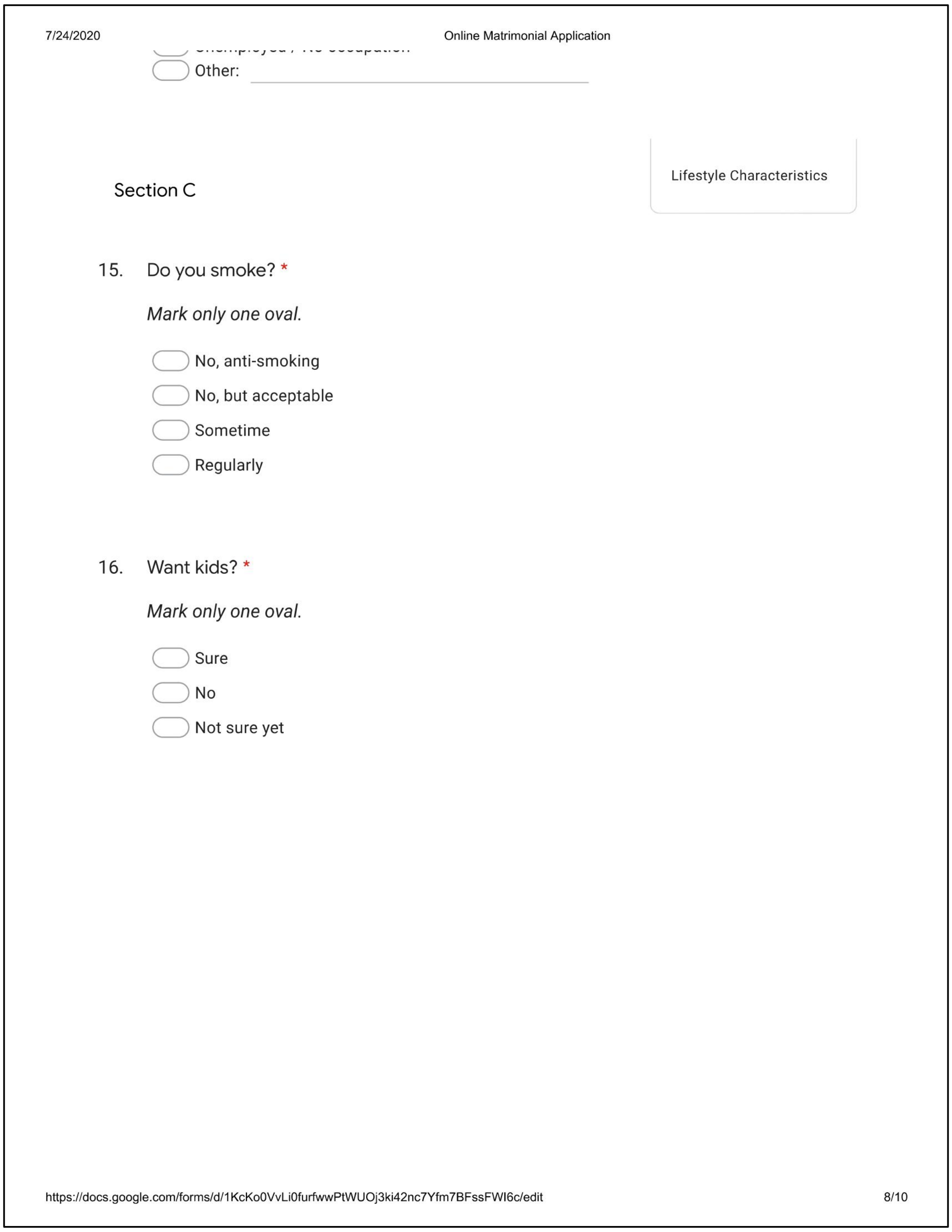
7/24/2020 Online Matrimonial Application

14. Occupation *

Mark only one oval.

- ◯ Administrative / Secretarial / Clerical
- ◯ Advertising / Media
- ◯ Artistic / Creative / Performance
- ◯ Construction / Trades
- ◯ Domestic Helper
- ◯ Education / Academic
- ◯ Entertainment / Media
- ◯ Executive / Management / HR
- ◯ Farming / Agriculture
- ◯ Finance / Banking / Real Estate
- ◯ Fire / Law Enforcement / Security
- ◯ Hair Dresser / Personal Grooming
- ◯ IT / Communication
- ◯ Laborer / Manufacturing
- ◯ Legal / Law
- ◯ Medical / Dental / Veterinary
- ◯ Military
- ◯ Nanny / Child Care
- ◯ Non-profit / Clergy / Social Service
- ◯ Retail / Food Service
- ◯ Sales / Marketing
- ◯ Sports / Recreation
- ◯ Political / Govt / Civil Service
- ◯ Retired
- ◯ Self Employed
- ◯ Student
- ◯ Technical / Science / Engineering
- ◯ Transportation
- ◯ Travel / Hospitality
- ◯ Unemployed / No occupation

https://docs.google.com/forms/d/1KcKo0VvLi0furfwwPtWUOj3ki42nc7Yfm7BFssFWI6c/edit 7/10



( ) Other: ______________________

Section C

Lifestyle Characteristics

15. Do you smoke? *

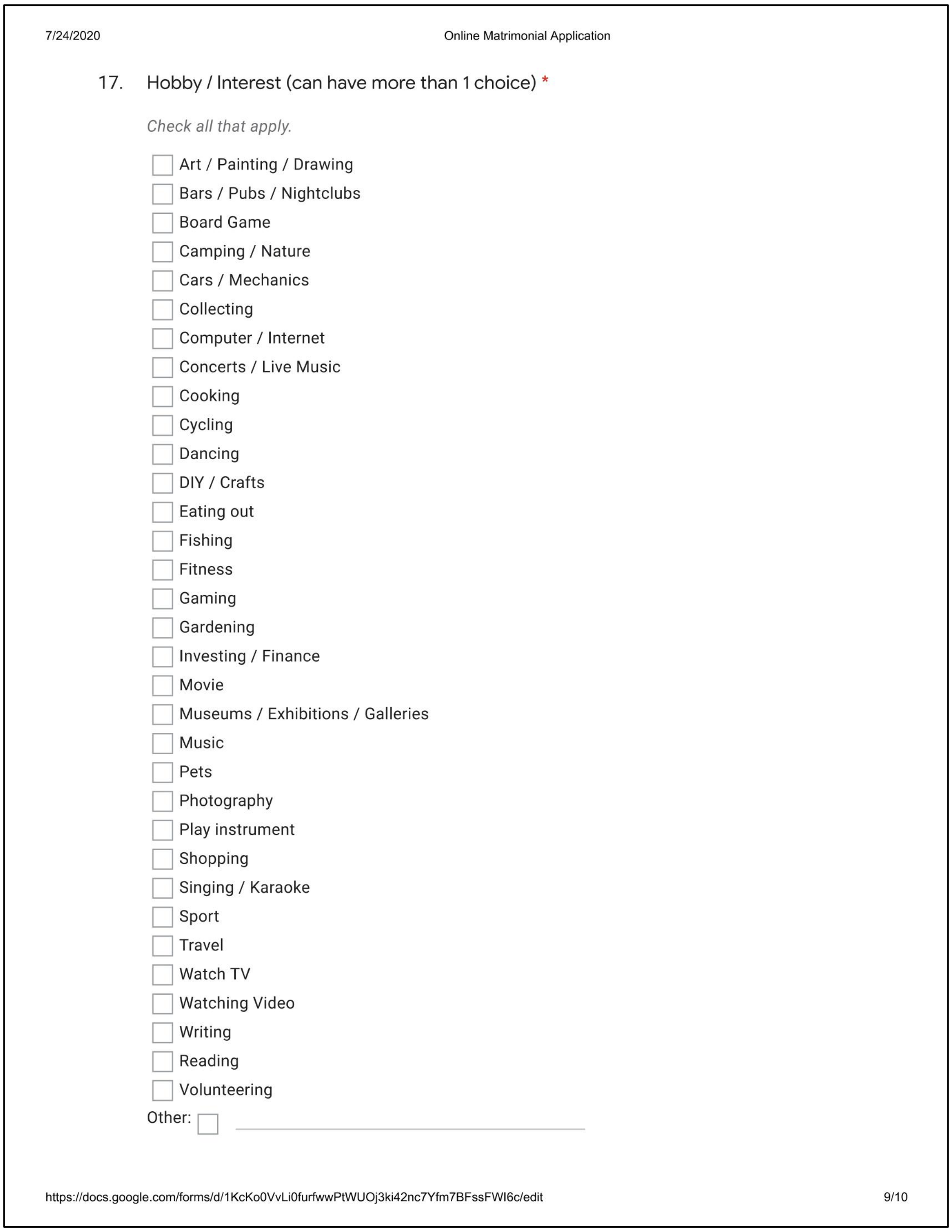
Mark only one oval.

( ) No, anti-smoking
( ) No, but acceptable
( ) Sometime
( ) Regularly

16. Want kids? *

Mark only one oval.

( ) Sure
( ) No
( ) Not sure yet



7/24/2020 Online Matrimonial Application

17. Hobby / Interest (can have more than 1 choice) *

Check all that apply.

- ☐ Art / Painting / Drawing
- ☐ Bars / Pubs / Nightclubs
- ☐ Board Game
- ☐ Camping / Nature
- ☐ Cars / Mechanics
- ☐ Collecting
- ☐ Computer / Internet
- ☐ Concerts / Live Music
- ☐ Cooking
- ☐ Cycling
- ☐ Dancing
- ☐ DIY / Crafts
- ☐ Eating out
- ☐ Fishing
- ☐ Fitness
- ☐ Gaming
- ☐ Gardening
- ☐ Investing / Finance
- ☐ Movie
- ☐ Museums / Exhibitions / Galleries
- ☐ Music
- ☐ Pets
- ☐ Photography
- ☐ Play instrument
- ☐ Shopping
- ☐ Singing / Karaoke
- ☐ Sport
- ☐ Travel
- ☐ Watch TV
- ☐ Watching Video
- ☐ Writing
- ☐ Reading
- ☐ Volunteering

Other: ☐ ______________________

https://docs.google.com/forms/d/1KcKo0VvLi0furfwwPtWUOj3ki42nc7Yfm7BFssFWI6c/edit 9/10

APPENDIX E: Usability Test Results

User Satisfaction Survey

1. Participant: #1
2. Gender: female
3. Age: 22
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently					*
I found the system unnecessarily complex.		*			
I thought the system was easy to use.					*
I think that I would need the support of a technical person to be able to use this system				*	
I found the various functions in this system were well integrated.				*	
I thought there was too much inconsistency in this system.		*			
I would imagine that most people would learn to use this system very quickly				*	
I found the system very cumbersome to use	*				
I felt very confident using the system					*
I needed to learn a lot of things before I could get going with this system	*				

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	Euclidean Distance	[✓ Manhattan Distance]	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	2	5
Cosine Similarity	3	6
Euclidean Distance	2	3
Manhattan Distance	5	7
Minkowski Distance	1	3

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	4	[✓ Strongly Agree 5]

5. **What did you like best about the application?**

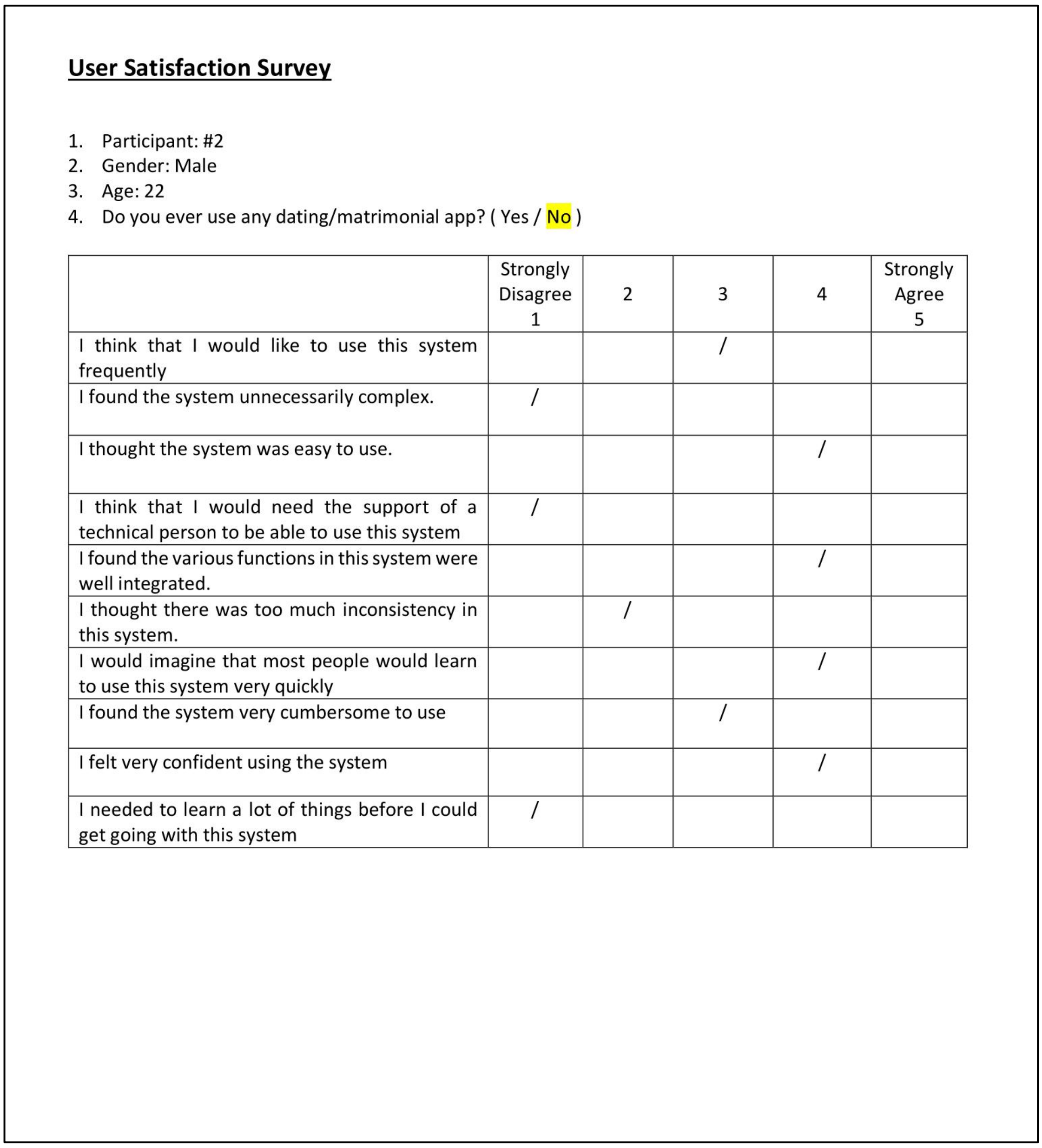
Sufficient information to find the matching person

6. **What did you like least about the application?**

Need to repeat filling person criteria

7. **Do you have any other comments /questions?**

Can use more simple words in the app or frequently used word



User Satisfaction Survey

1. Participant: #2
2. Gender: Male
3. Age: 22
4. Do you ever use any dating/matrimonial app? (Yes / [✓ No])

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently			/		
I found the system unnecessarily complex.	/				
I thought the system was easy to use.				/	
I think that I would need the support of a technical person to be able to use this system	/				
I found the various functions in this system were well integrated.				/	
I thought there was too much inconsistency in this system.		/			
I would imagine that most people would learn to use this system very quickly				/	
I found the system very cumbersome to use			/		
I felt very confident using the system				/	
I needed to learn a lot of things before I could get going with this system	/				

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

[✓ Jaccard Coefficient]	Cosine Similarity	Euclidean Distance	Manhattan Distance	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	3	5
Cosine Similarity	2	2
Euclidean Distance	2	2
Manhattan Distance	2	3
Minkowski Distance	2	2

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	[✓ 3]	4	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	[✓ 3]	4	Strongly Agree 5

5. **What did you like best about the application?**

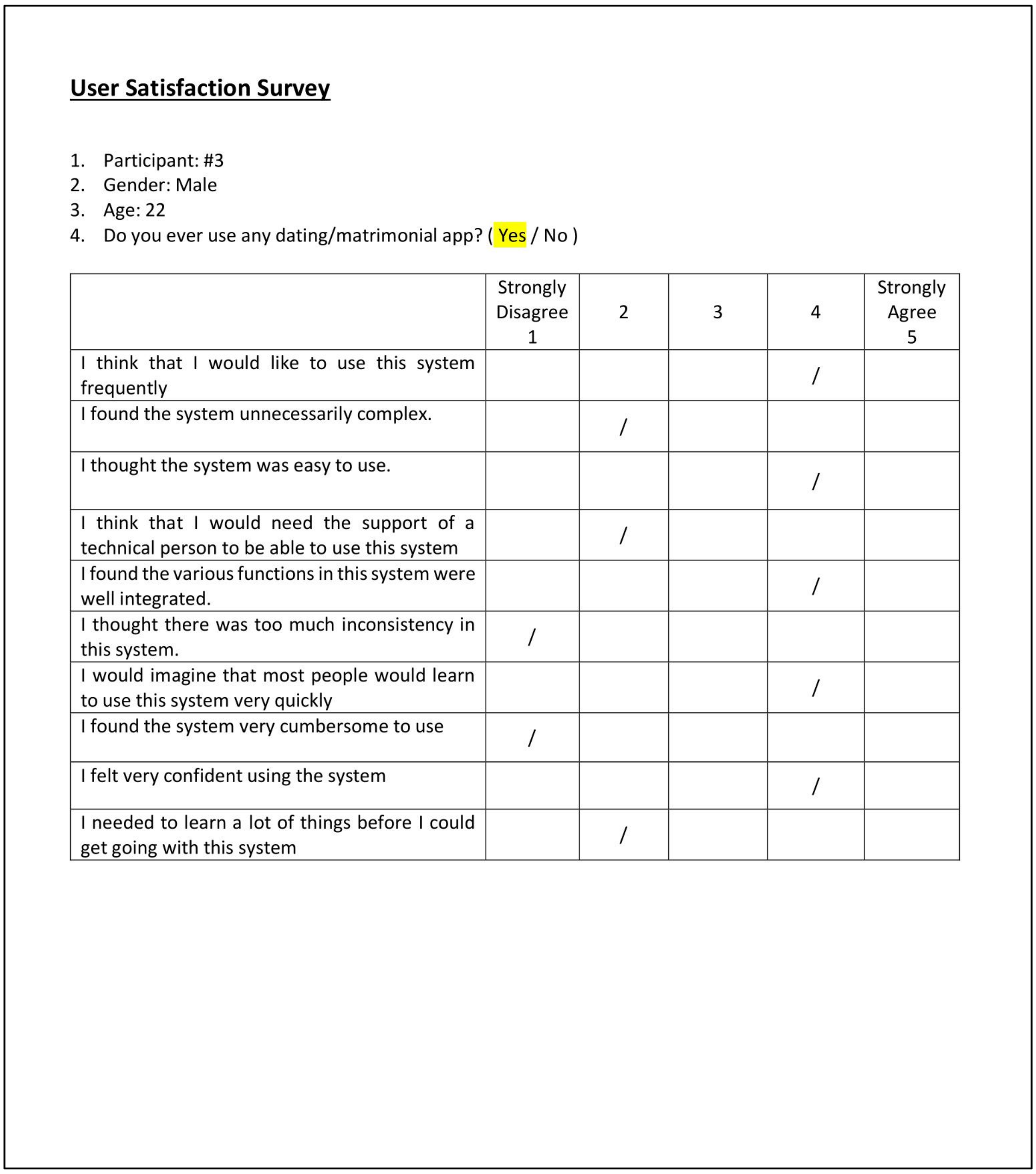
Display percentage

6. **What did you like least about the application?**

Always need to submit the preferences one more time when clicking the home navigation button.

7. **Do you have any other comments /questions?**

Prefer multiple selection on the preferences data.



User Satisfaction Survey

1. Participant: #3
2. Gender: Male
3. Age: 22
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently				/	
I found the system unnecessarily complex.		/			
I thought the system was easy to use.				/	
I think that I would need the support of a technical person to be able to use this system		/			
I found the various functions in this system were well integrated.				/	
I thought there was too much inconsistency in this system.	/				
I would imagine that most people would learn to use this system very quickly				/	
I found the system very cumbersome to use	/				
I felt very confident using the system				/	
I needed to learn a lot of things before I could get going with this system		/			

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	[✓ Euclidean Distance]	Manhattan Distance	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	4	8
Cosine Similarity	4	8
Euclidean Distance	5	8
Manhattan Distance	4	9
Minkowski Distance	4	9

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

5. **What did you like best about the application?**

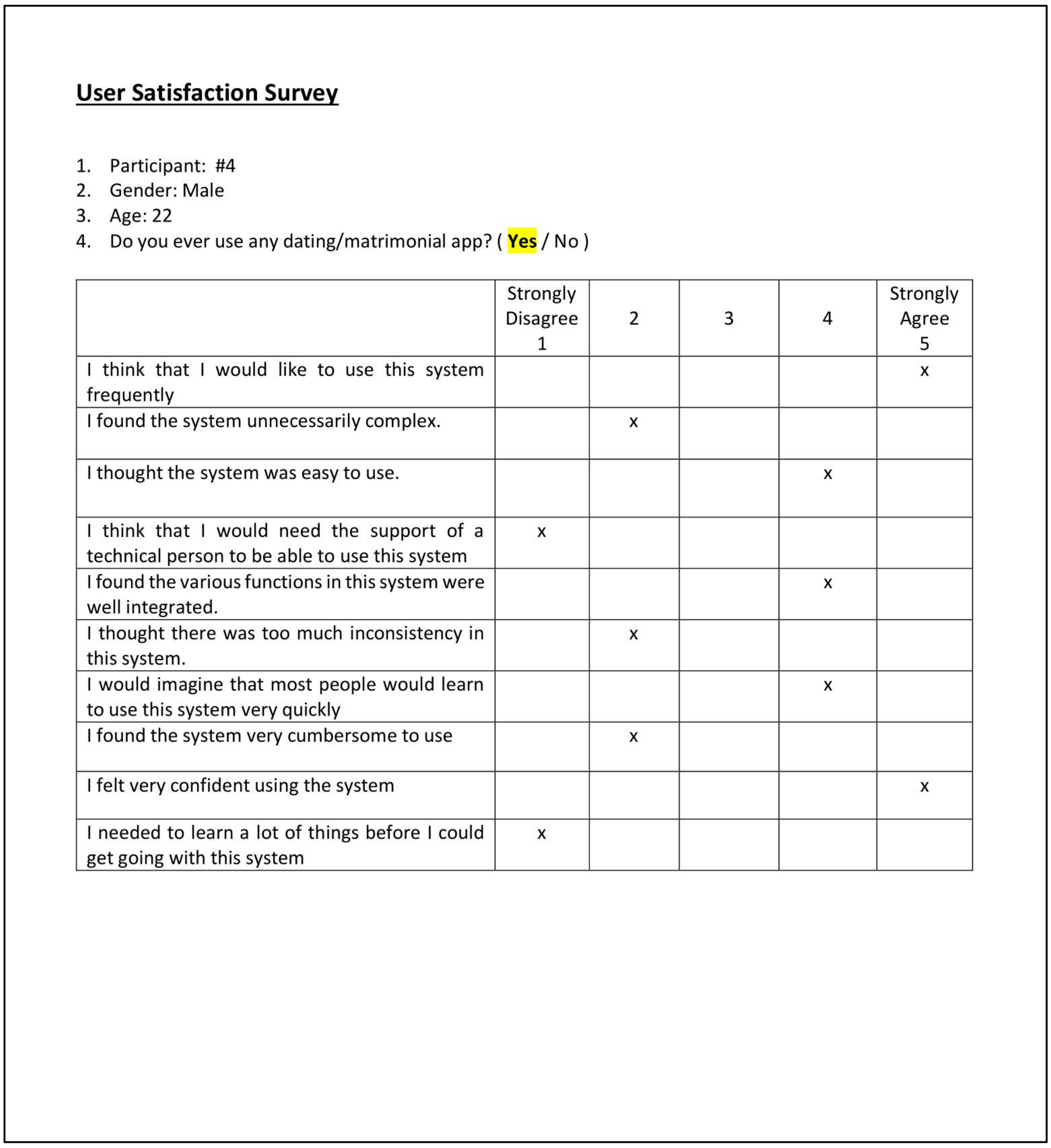
More filtering options are provided

6. **What did you like least about the application?**

Need to move the mouse a lot

7. **Do you have any other comments /questions?**

Non responsive UI, can try to improve the user experience when using it



User Satisfaction Survey

1. Participant: #4
2. Gender: Male
3. Age: 22
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently					x
I found the system unnecessarily complex.		x			
I thought the system was easy to use.				x	
I think that I would need the support of a technical person to be able to use this system	x				
I found the various functions in this system were well integrated.				x	
I thought there was too much inconsistency in this system.		x			
I would imagine that most people would learn to use this system very quickly				x	
I found the system very cumbersome to use		x			
I felt very confident using the system					x
I needed to learn a lot of things before I could get going with this system	x				

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	Euclidean Distance	[✓ Manhattan Distance]	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	3	6
Cosine Similarity	3	6
Euclidean Distance	3	5
Manhattan Distance	5	6
Minkowski Distance	3	5

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

5. **What did you like best about the application?**

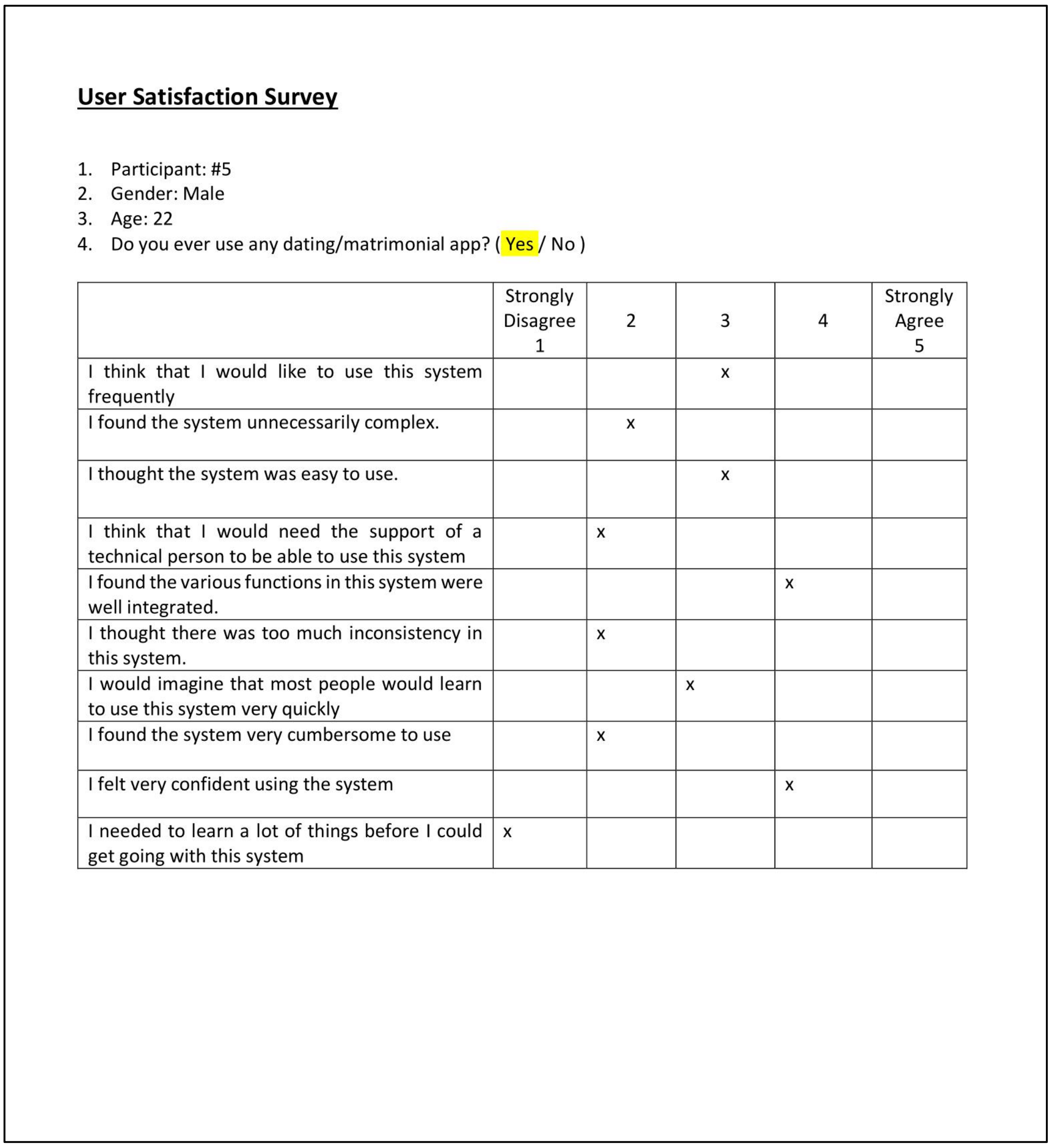
   Able to display the degree of similarity regarding on my preferences with other members in this application

6. **What did you like least about the application?**

   The delete function in the chat section

7. **Do you have any other comments /questions?**

   Some of the icons can make it bigger such as the profile icon (on the top-right side), better indication after user interaction with the system, such as the favourite button. Delete function in the chat section need some amendments, suggestions on displaying beside the chat box.



User Satisfaction Survey

1. Participant: #5
2. Gender: Male
3. Age: 22
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently			x		
I found the system unnecessarily complex.		x			
I thought the system was easy to use.			x		
I think that I would need the support of a technical person to be able to use this system		x			
I found the various functions in this system were well integrated.				x	
I thought there was too much inconsistency in this system.		x			
I would imagine that most people would learn to use this system very quickly			x		
I found the system very cumbersome to use		x			
I felt very confident using the system				x	
I needed to learn a lot of things before I could get going with this system	x				

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

[✓ Jaccard Coefficient]	Cosine Similarity	Euclidean Distance	Manhattan Distance	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	4	4
Cosine Similarity	2	4
Euclidean Distance	2	2
Manhattan Distance	2	7
Minkowski Distance	2	4

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	[✓ 3]	4	Strongly Agree 5

5. **What did you like best about the application?**

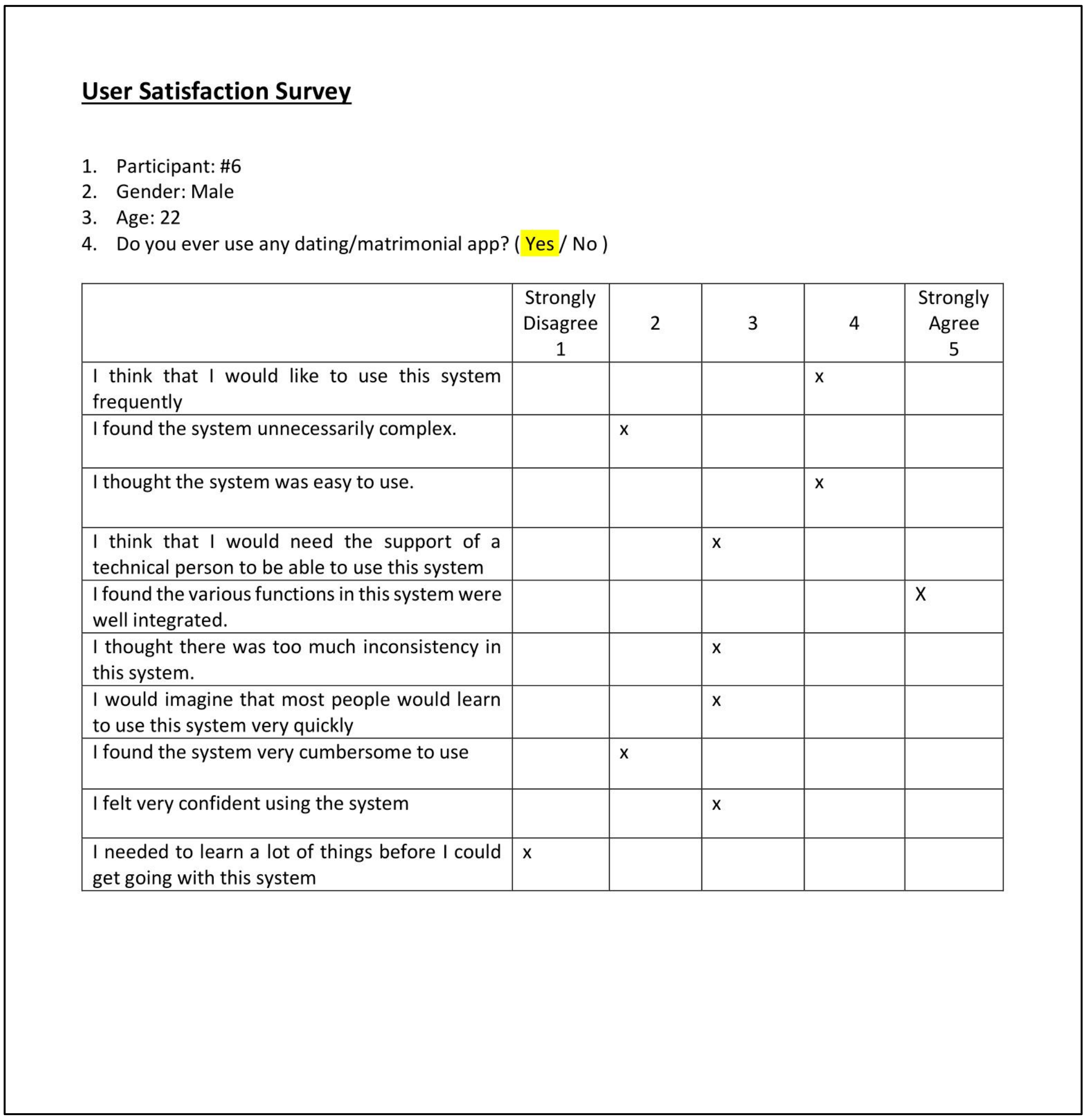
   Different matching methods are very fascinating. Explanation or justification provided are the best.

6. **What did you like least about the application?**

   The matching method showed might be a little bit weird and difficult to comprehend for normal users.

7. **Do you have any other comments /questions?**

   The application is good if it's view as a dating app. If it's for marriage app, more information provided would be way more superior.



User Satisfaction Survey

1. Participant: #6
2. Gender: Male
3. Age: 22
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently				x	
I found the system unnecessarily complex.		x			
I thought the system was easy to use.				x	
I think that I would need the support of a technical person to be able to use this system			x		
I found the various functions in this system were well integrated.					X
I thought there was too much inconsistency in this system.			x		
I would imagine that most people would learn to use this system very quickly			x		
I found the system very cumbersome to use		x			
I felt very confident using the system			x		
I needed to learn a lot of things before I could get going with this system	x				

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	[✓ Cosine Similarity]	Euclidean Distance	Manhattan Distance	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	1	5
Cosine Similarity	3	7
Euclidean Distance	1	4
Manhattan Distance	2	4
Minkowski Distance	2	3

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

5. **What did you like best about the application?**

   Chat function

6. **What did you like least about the application?**

   Have to refill preference every time login.

7. **Do you have any other comments /questions?**

   Change join us button to brighter color

User Satisfaction Survey

1. Participant: #7
2. Gender: MALE
3. Age: 25
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently				✓	
I found the system unnecessarily complex.	✓				
I thought the system was easy to use.					✓
I think that I would need the support of a technical person to be able to use this system	✓				
I found the various functions in this system were well integrated.					✓
I thought there was too much inconsistency in this system.	✓				
I would imagine that most people would learn to use this system very quickly					✓
I found the system very cumbersome to use	✓				
I felt very confident using the system					✓
I needed to learn a lot of things before I could get going with this system			✓		

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	Euclidean Distance	Manhattan Distance	[✓ Minkowski Distance]

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	3	8
Cosine Similarity	5	6
Euclidean Distance	5	7
Manhattan Distance	5	7
Minkowski Distance	5	9

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	[✓ 3]	4	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	4	Strongly Agree [✓ 5]

5. **What did you like best about the application?**

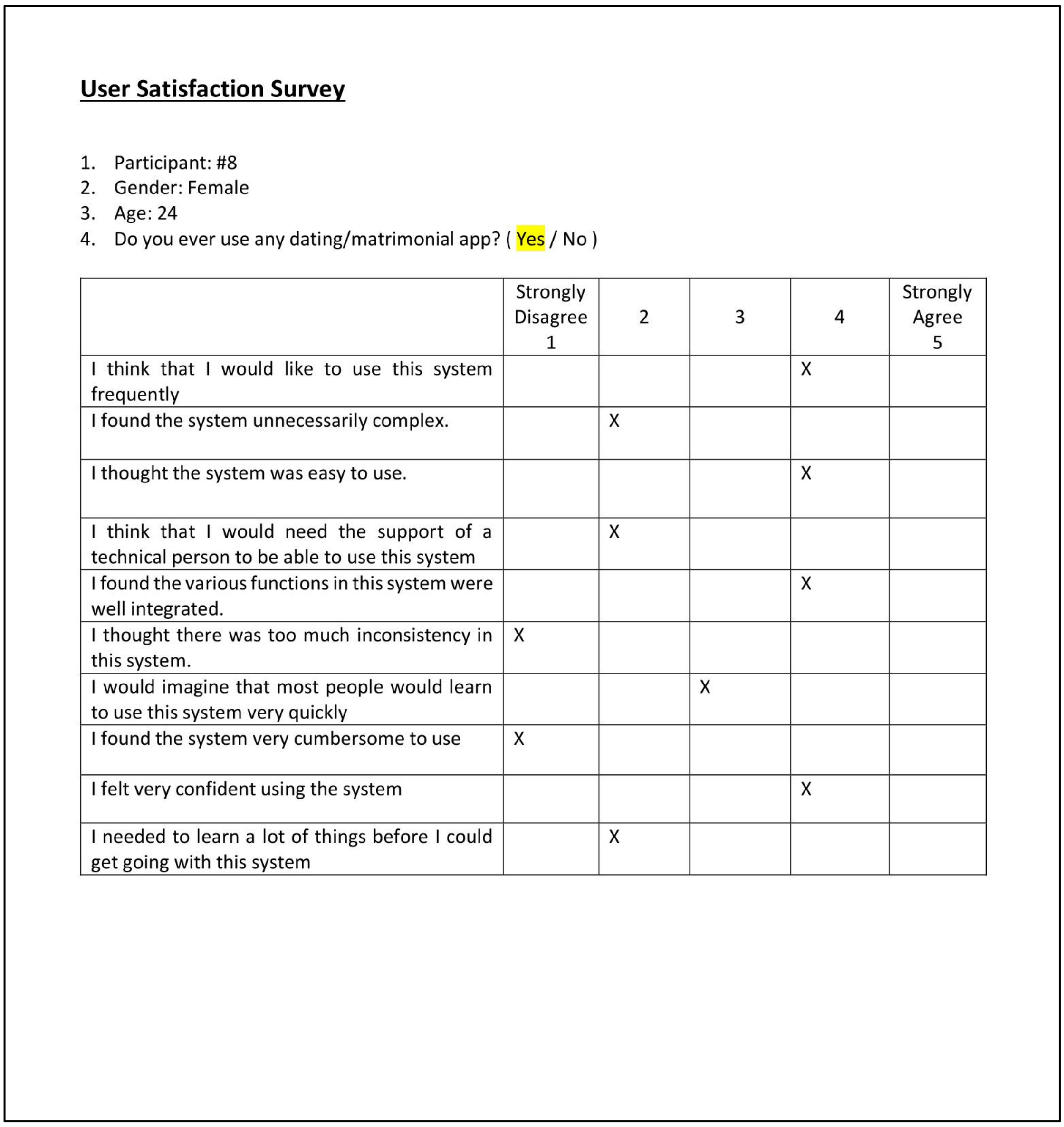
Filter Preferences and Message People

6. **What did you like least about the application?**

Looking preferences does not update with search preferences

7. **Do you have any other comments /questions?**

Maybe have more photo to explore



User Satisfaction Survey

1. Participant: #8
2. Gender: Female
3. Age: 24
4. Do you ever use any dating/matrimonial app? ([✓ Yes] / No)

	Strongly Disagree 1	2	3	4	Strongly Agree 5
I think that I would like to use this system frequently				X	
I found the system unnecessarily complex.		X			
I thought the system was easy to use.				X	
I think that I would need the support of a technical person to be able to use this system		X			
I found the various functions in this system were well integrated.				X	
I thought there was too much inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly			X		
I found the system very cumbersome to use	X				
I felt very confident using the system				X	
I needed to learn a lot of things before I could get going with this system		X			

1. **Among the 5 matching methods, which method can help you find the potential match that is closer to your ideal type?**

Jaccard Coefficient	Cosine Similarity	Euclidean Distance	[✓ Manhattan Distance]	Minkowski Distance

2. **For each algorithm, how many profiles suit your preferences?**

	Top 5 profiles	Top 10 profile
Jaccard Coefficient	4	7
Cosine Similarity	4	6
Euclidean Distance	4	6
Manhattan Distance	5	9
Minkowski Distance	4	6

3. **Is this application help you in finding suitable partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

4. **Is the member information provided sufficient for you to decide on selecting a partner?**

Strongly Disagree 1	2	3	[✓ 4]	Strongly Agree 5

5. **What did you like best about the application?**

   Display percentage

6. **What did you like least about the application?**

   Need to resubmit the preferences for viewing the results after press the home button

7. **Do you have any other comments /questions?**

   Maybe can add a photo gallery that allow user to put more personal photos

APPENDIX F: User Acceptance Test Results

User Acceptance Test

Tester's Name: Tan Chee Kuan

Date of Testing: 24 July 2020

Test Case 1

ID	1	
Start Time	10.28pm	
End Time	10.30pm	
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account	Pass	-
Able to insert personal information	Pass	-
Able to upload profile picture	Pass	-

Test Case 2

ID	2	
Start Time	10.30pm	
End Time	10.35pm	
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences	Pass	-
Able to view search results if there are profiles matched the preferences (Exact Matching)	Pass	-
Able to edit match preferences	Pass	-
Able to select different match methods	Pass	-
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance	Pas	Result are not that accurate. Maybe the data is insufficient

Test Case 3

ID	3	
Start Time	12:05 AM	
End Time	12:08 AM	
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information	Pass	-
Able to zoom member profile picture	Pass	-
Able to add member to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to send instant message to member	Pass	-
Able to like the member	Pass	-

Test Case 4

ID	4	
Start Time	12:08 AM	
End Time	12:09 AM	
Module	Manage Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list	Pass	-
Able to view the chat records with one of the members	Pass	-
Able to delete the chat history	Pass	-
Able to show confirmation message before delete the chat history	Pass	-
Able to view the latest chat list	Pass	-

Test Case 5

ID	5	
Start Time	12:09 AM	
End Time	12:11 AM	
Module	Manage Favourite List	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view favourite list	Pass	-
Able to view member profile information that added to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to show confirmation message before remove the profile from favourite	Pass	-
Able to view the latest favourite list	Pass	-

Test Case 6

ID	6	
Start Time	12:11 AM	
End Time	12:13 AM	
Module	Manage Personal Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view personal profile information	Pass	-
Able to update profile picture	Pass	-
Able to view number of likes received	Pass	-
Able to update profile information	Pass	-
Able to update match preferences	Pass	-

User Acceptance Test

Tester's Name: Tan Chee Kuan

Date of Testing: 24 July 2020

Test Case 1

ID	1	
Start Time	10.28pm	
End Time	10.30pm	
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account	Pass	-
Able to insert personal information	Pass	-
Able to upload profile picture	Pass	-

Test Case 2

ID	2	
Start Time	10.30pm	
End Time	10.35pm	
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences	Pass	-
Able to view search results if there are profiles matched the preferences (Exact Matching)	Pass	-
Able to edit match preferences	Pass	-
Able to select different match methods	Pass	-
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance	Pas	Result are not that accurate. Maybe the data is insufficient

Test Case 3

ID	3	
Start Time	10.37pm	
End Time	10.49pm	
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information	Pass	-
Able to zoom member profile picture	Pass	-
Able to add member to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to send instant message to member	Pass	-
Able to like the member	Pass	-

Test Case 4

ID	4	
Start Time	10.40pm	
End Time	10.44pm	
Module	Manage Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list	Pass	-
Able to view the chat records with one of the members	Pass	-
Able to delete the chat history	Pass	-
Able to show confirmation message before delete the chat history	Pass	-
Able to view the latest chat list	Pass	-

Test Case 5

ID	5	
Start Time	10.44pm	
End Time	10.46pm	
Module	Manage Favourite List	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view favourite list	Pass	-
Able to view member profile information that added to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to show confirmation message before remove the profile from favourite	Pass	-
Able to view the latest favourite list	Pass	-

Test Case 6

ID	6	
Start Time	10.47pm	
End Time	10.50pm	
Module	Manage Personal Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view personal profile information	Pass	-
Able to update profile picture	Pass	-
Able to view number of likes received	Pass	-
Able to update profile information	Pass	-
Able to update match preferences	Pass	-

User Acceptance Test

Tester's Name: Tan Wei Seng

Date of Testing: 26/7/2020

Test Case 1

ID	1	
Start Time	11:40pm	
End Time	11:47pm	
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account	Pass	-
Able to insert personal information	Pass	-
Able to upload profile picture	Pass	-

Test Case 2

ID	2	
Start Time	11:48pm	
End Time	11:52pm	
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences	Pass	-
Able to view search results if there are profiles matched the preferences (Exact Matching)	Pass	-
Able to edit match preferences	Pass	-
Able to select different match methods	Pass	-
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance	Pass	-

Test Case 3

ID	3	
Start Time	11:53pm	
End Time	12:00pm	
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information	Pass	-
Able to zoom member profile picture	Pass	-
Able to add member to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to send instant message to member	Pass	-
Able to like the member	Pass	-

Test Case 4

ID	4	
Start Time	12:00pm	
End Time	12:00pm	
Module	View Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list	Pass	-
Able to view the chat records with one of the members	Pass	-
Able to delete the chat history	Pass	-
Able to show confirmation message before delete the chat history	Pass	-
Able to view the latest chat list	Pass	-

Test Case 5

ID	5	
Start Time	12:00pm	
End Time	12:02pm	
Module	View Favourite List	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view favourite list	Pass	-
Able to view member profile information that added to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to show confirmation message before remove the profile from favourite	Pass	-
Able to view the latest favourite list	Pass	-

Test Case 6

ID	6	
Start Time	12:02pm	
End Time	12:04pm	
Module	View Personal Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view personal profile information	Pass	-
Able to update profile picture	Pass	-
Able to view number of likes received	Pass	-
Able to update profile information	Pass	-
Able to update match preferences	Pass	-

User Acceptance Test

Tester's Name: Ling Kah Sin

Date of Testing: 27/7/2020

Test Case 1

ID	1	
Start Time	3.05pm	
End Time	3.07pm	
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account	Pass	-
Able to insert personal information	Pass	-
Able to upload profile picture	Pass	-

Test Case 2

ID	2	
Start Time	3.08pm	
End Time	3.11pm	
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences	Pass	-
Able to view search results if there are profiles matched the preferences (Exact Matching)	Pass	-
Able to edit match preferences	Pass	-
Able to select different match methods	Pass	-
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance	Pass	-

Test Case 3

ID	3	
Start Time	3.13pm	
End Time	3.14pm	
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information	Pass	-
Able to zoom member profile picture	Pass	-
Able to add member to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to send instant message to member	Pass	-
Able to like the member	Pass	-

Test Case 4

ID	4	
Start Time	3.15pm	
End Time	3.16pm	
Module	Manage Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list	Pass	-
Able to view the chat records with one of the members	Pass	-
Able to delete the chat history	Pass	-
Able to show confirmation message before delete the chat history	Pass	-
Able to view the latest chat list	Pass	-

Test Case 5

ID	5	
Start Time	3.16pm	
End Time	3.17pm	
Module	Manage Favourite List	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view favourite list	Pass	-
Able to view member profile information that added to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to show confirmation message before remove the profile from favourite	Pass	-
Able to view the latest favourite list	Pass	-

Test Case 6

ID	6	
Start Time	3.17pm	
End Time	3.19pm	
Module	Manage Personal Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view personal profile information	Pass	-
Able to update profile picture	Pass	-
Able to view number of likes received	Pass	-
Able to update profile information	Pass	-
Able to update match preferences	Pass	-

User Acceptance Test

Tester's Name: Ong Shu Xian
Date of Testing: 28/7/2020

Test Case 1

ID	1	
Start Time	1.30pm	
End Time	1.34pm	
Module	Create Profile	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to register an account	Pass	-
Able to insert personal information	Pass	Do not have scroll bar, need to use keyboard to scroll down.
Able to upload profile picture	Pass	-

Test Case 2

ID	2	
Start Time	1.35 pm	
End Time	1.40 pm	
Module	Search Potential Matches	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to set match preferences	Pass	-
Able to view search results if there are profiles matched the preferences (Exact Matching)	Pass	-
Able to edit match preferences	Pass	-
Able to select different match methods	Pass	-
Able to view search results for Jaccard Coefficient, Cosine Similarity, Euclidean Distance, Manhattan Distance and Minkowski Distance	Pass	-

Test Case 3

ID	3	
Start Time	1.41 pm	
End Time	1.44 pm	
Module	View Potential Matches Profiles	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view member profile information	Pass	-
Able to zoom member profile picture	Pass	-
Able to add member to favourite	Pass	-
Able to remove member from favourite	Pass	-
Able to send instant message to member	Pass	-
Able to like the member	Pass	-

Test Case 4

ID	4	
Start Time	1.45 pm	
End Time	1.48 pm	
Module	Manage Chat History	
Test Descriptions	**Status (Pass / Fail)**	**Comments**
Able to view chat list	Pass	-
Able to view the chat records with one of the members	Pass	-
Able to delete the chat history	Pass	-
Able to show confirmation message before delete the chat history	Pass	-
Able to view the latest chat list	Pass	-

Test Case 5

ID	5		
Start Time	1.50 pm		
End Time	1.52 pm		
Module	Manage Favourite List		
Test Descriptions		**Status (Pass / Fail)**	**Comments**
Able to view favourite list		Pass	-
Able to view member profile information that added to favourite		Pass	-
Able to remove member from favourite		Pass	-
Able to show confirmation message before remove the profile from favourite		Pass	-
Able to view the latest favourite list		Pass	-

Test Case 6

ID	6		
Start Time	1.52 pm		
End Time	1.55 pm		
Module	Manage Personal Profile		
Test Descriptions		**Status (Pass / Fail)**	**Comments**
Able to view personal profile information		Pass	-
Able to update profile picture		Pass	-
Able to view number of likes received		Pass	-
Able to update profile information		Pass	-
Able to update match preferences		Pass	-