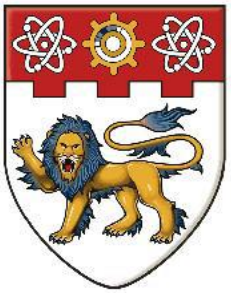


NANYANG TECHNOLOGICAL UNIVERSITY

Lab Group: SSP4

CZ2006 SOFTWARE ENGINEERING

Final Submission

Team Name: UnderConstruction/Group 4

GymBuddies Application
Software Requirement Specifications Document

Francis Lim Yong Wen (U1821950H)

Soh Yan Quan, Kenneth (U1821541B)

Hsiao Chia Yu (U1821839L)

Sam Jian Shen (U1821296L)

Kong Alson (U1822899B)

Tang Wei Cong (U1820899G)

Due Date of Submission: 10 November 2019 (Sunday)

School of Computer Science and Engineering

Nanyang Technological University

Software Requirements Specification

for

GymBuddies Android Application

Version 2.2 approved

Prepared by Francis Lim

NTU Team UnderConstruction

10 November 2019

Contents

Revision History

Name	Date	Reason For Changes	Version
Francis Lim	14/10/19	Initial draft version	0.1
Francis Lim	20/10/19	Addition of UML diagrams and design patterns	0.2
All	22/10/19	First version	1.0
Kenneth Soh	27/10/19	Fixed some erroneous errors	1.1
All	5/11/19	Added white and black box testing	2.0
Francis Lim	8/11/19	Added video link of app demo and finalisation	2.1
Francis Lim	10/11/19	Clarified on operating environment	2.2

Introduction

Project Mission Statement

The "Under Construction" team has developed an android application called "Gym Buddies". The application allows user to search for a gym buddy to work out in any public gym across Singapore. This project will be considered complete when the application has been tested and approved for release by ZEA. This project supports the Smart Nation movement and is contesting in the fourth Data-Driven Smart Nation Competition.

This application is targeted at Android smartphone users looking for another user "Gym Buddy", to go to the gym with.

Purpose

The "Gym Buddies" Android Application aims to introduce a buddy system in which application users can pair up with other users, who have similar interests, through the app, to search for a gym based on the device's GPS system and visit for recreational usage. The process of searching a gym will also retrieve carpark information relative to the distance of the specified gym.

In a nutshell, the main purposes of the GymBuddies Android Application is to:

1) Facilitate users to find and add other users as "GymBuddies" based on filtered preferences;
2) To search for gyms based on either their GPS location, or through search filters;
3) To retrieve carpark information relative to the specified gym's location, and;

Scope

The GymBuddies Android Application allows users to add other users as "GymBuddies", so as to visit a nearby gym with their "GymBuddies" for recreational usage. On top of that, the application also allows viewing of nearby carparks relative to the gym, as well as allowing users to post discussion topics related to the application.

Users and Stakeholders

The stakeholders of this project consists of gym establishment owners, Android users of this mobile application, and GymBuddies software development team. Information of gym establishments is retrieved via the Data.gov.sg dataset on gym establishments, information on carpark locations is retrieved via the Data.gov.sg Singapore Carpark Locations API; the application uses the Google Maps API for location searching and location retrieval via coordinates.

Android mobile users will use the app to search for other users as "GymBuddies", search for gyms as well as their nearby carpark information. The developers of GymBuddies aim to interconnect between frequent gym users such that they can find friends that have common interests at neighbourhood gyms, as well as heading to the gym to do workout with their friends, providing a networking platform between users.

The intended audience for this application is mainly Android mobile phone users who have a habit of exercising at gyms.

Assumptions and Constraints

The developer of this application assumes that users should have a GPS-enabled Android phone such that it is capable of running this application, as well as an Android version of at least 5.0.

With GPS services, location services on their Android phone must be enabled to enjoy the location features of this application, as well as internet access to link with other users.

Dependencies
GymBuddies requires to use of multiple APIs: Google Maps API to display to location of gyms and carparks, government APIs and datasets such as Carpark API and Gym Dataset to prepopulate Gym and Carpark objects for our application.

References
Gyms@SG KML Dataset – Data.gov.sg: https://data.gov.sg/dataset/gymssg

Carpark Availability API – LTA Datamall:
http://datamall2.mytransport.sg/ltaodataservice/CarParkAvailabilityv2

Summary of Operating Environment
Android Application
- Minimum Android Version - Android 5 Lollipop (API 21)
- Targeted Android Version - Android 10 (API 29)
- App Support from Android 5 Lollipop and above

Android Dependencies
- Android Room Persistence Library - 2.2.0
- Google Firebase Firestore - 21.2.0
- Google Firebase Remote Config - 19.0.3
- Google Firebase Storage - 19.1.0
- Google Firebase Analytics - 17.2.0
- Google Firebase Crashlytics - 2.10.1
- Google Firebase Authentication - 19.1.0
- Google Firebase UI - 4.3.1
- AndroidX Support Libraries - 1.1.0
- AndroidX Jetpack Compose Navigation Library - 2.1.0
- Kotlin - 1.3.50
- Google Play Services - 17.0.0
- Google Play Services - Maps - 17.0.0
- Google Play Services - Location - 17.0.0
- Attribouter - 0.1.5
- Google Gson - 2.8.5
- Google Material Design - 1.1.0
- Material Rating Bar - 1.3.2
- Apache Commons Text - 1.8

Server
- Firebase BaaS (Backend-as-a-Service)
    - Cloud Functions for server-side code
        - Microsoft Typescript - 3.2.2

Database Software
- Firebase Firestore NoSQL Database
- Android Local SQLite Database (accessed through the Android Room Library)

Development Environment

- Android Studio 3.5
- Android Build Tools 29.0.1

Functional Requirements

The functional requirements of the application are as follows:

1. The **application user** must be able to **register** for their own account such as Google Account, in order to use the application.
   1.1. The **user** must **enter** their e-mail address during registration.
   1.2. The **system** shall then **send** a verification e-mail to the user's registered e-mail address after initially registering for the application.
      1.2.1. The user must receive the e-mail from the system within 1 minute.
   1.3. The **user** must able to **request** the system to resend the confirmation e-mail.
2. The **user** must be able to **login** to application in order to use its features.
   2.1. The user must be able to log in with their registered account via their email.
   2.2. The user must be able to log in with their Google account.
   2.3. The user must be able to log out after being logged in.
      2.3.1. The log out process shall be a maximum of 5 seconds.
      2.3.2. If any, the system must save user data changed during the login session.
      2.3.3. If any, the system must save user preferences changed during the login session.
   2.4. The **system** shall **lock** the user out of the application after 5 consecutive failed attempts.
      2.4.1. After this, the user will be timed-out. After the time-out ends, the user can re-enter the password again
3. The **user** must be able to **reset** his password at the login page.
   3.1. The **user** shall **enter** his e-mail address in order to verify his account.
   3.2. The **system** shall **send** an e-mail notification to the user within 1 minute, in order to reset the password.
   3.3. The user shall click a confirmation link in their person e-mail inbox.
   3.4. The **user** shall be able to **enter** a new password after clicking the confirmation link.
4. The **user** must **fill in** their own personal profile to fill up information such as their username or nickname, gender, day and time preferences, and gym location preferences.
   4.1. The user must perform this after logging in for the first time.
   4.2. The **user** must be able to **enter** a full name.
   4.3. The **user** must be able to **select** their gender.
      4.3.1. The gender shall be in the form of a radio box.
      4.3.2. The gender options shall only be limited to Male or Female.
   4.4. The **user** must **enter** their region of stay during registration.
      4.4.1. The region of stay is restricted to 5 regions, namely North, South, East, West and Central.
   4.5. The **user** must be able to **select** their day preferences.
      4.5.1. The day preferences shall be in the form of a multiple selection checkbox.
      4.5.2. The day preferences shall contain Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.
   4.6. The **user** must be able to **select** their time preferences.
      4.6.1. The time preferences shall be in the form of a radio button.

4.6.2. The time preferences shall contain either AM or PM, and not both.

4.7. The **user** must be able to **select** their gym location preferences.

4.7.1. The gym location preferences shall be selected through the user's favorite gym list.

5. The **user** must be able to **search** for other users ("Gym Buddy") after logging in.

5.1. The user must be able to search for other users through filtering.

5.1.1. The filtering options shall include filtering the days.

5.1.1.1. The user shall be able to filter through multiple days.

5.1.2. The filtering options shall include filtering the timings.

5.1.2.1. The user shall be able to filter through either timings.

5.1.3. The filtering options shall include filtering the gender preferences.

5.1.3.1. The user shall be able to filter through either genders.

6. The **user** must be able to **favourite** another user as their gym buddy.

6.1. The **user** must **click** the favourite icon to favourite the buddy.

6.1.1. The favourited buddy will be added to the favourited buddy list.

7. The **user** must be able to **unfavourite** gym buddies.

7.1.1. The user shall be able to do this after favouriting a gym buddy.

8. The **user** must be able to **view** a list of their favourited gym buddies after logging in.

8.1. The **system** should **display** a list of gym buddies.

8.1.1. The gym buddy's brief information shall contain the name.

8.2. The **user** must be able to **view** detailed information of a gym buddy.

8.2.1. The gym buddy's detail information shall contain the name.

8.2.2. The gym buddy's detail information shall contain the region.

8.2.3. The gym buddy's detail information shall contain their gender.

8.2.4. The gym buddy's detail information shall contain their day preferences.

8.2.5. The gym buddy's detail information shall contain their time preferences.

9. The **user** must be able to **chat** with other users after logging in.

9.1. The **user** must be able to **chat** with users in their gym buddy list.

9.1.1. The **user** must be able to **send** messages to the other user.

10. The **user** must be able to **search** for gyms located in Singapore after logging in.

10.1. The user must be able to search by filtering out specific gyms.

10.1.1. The user shall be able to filter by gym rating.

10.1.2. The user shall be able to filter by distance from user's current position.

10.1.3. The user shall be able to filter by gym popularity.

10.2. The **system** must **return** the search results in less than 15 seconds.

10.2.1. The system shall show the results in the form of a map.

10.2.2. The system shall show the results in list view.

10.3. The **user** must be able to **view** up to 10 nearby carparks around the gym.

10.3.1. The nearby carparks shall be within a radius of 1 kilometre from the gym.

10.3.1.1. The system shall show the results in the form of a map.

10.3.1.2. The system shall show the results in list view.

10.3.2. The system must be able to show the carpark's information.

10.3.2.1. The system shall show carpark rates of the carpark.

10.3.2.2. The system shall show the address of the carpark.

11. The **user** must be able to **rate** gyms.

11.1. The user shall leave a rating of a minimum of 1 stars.

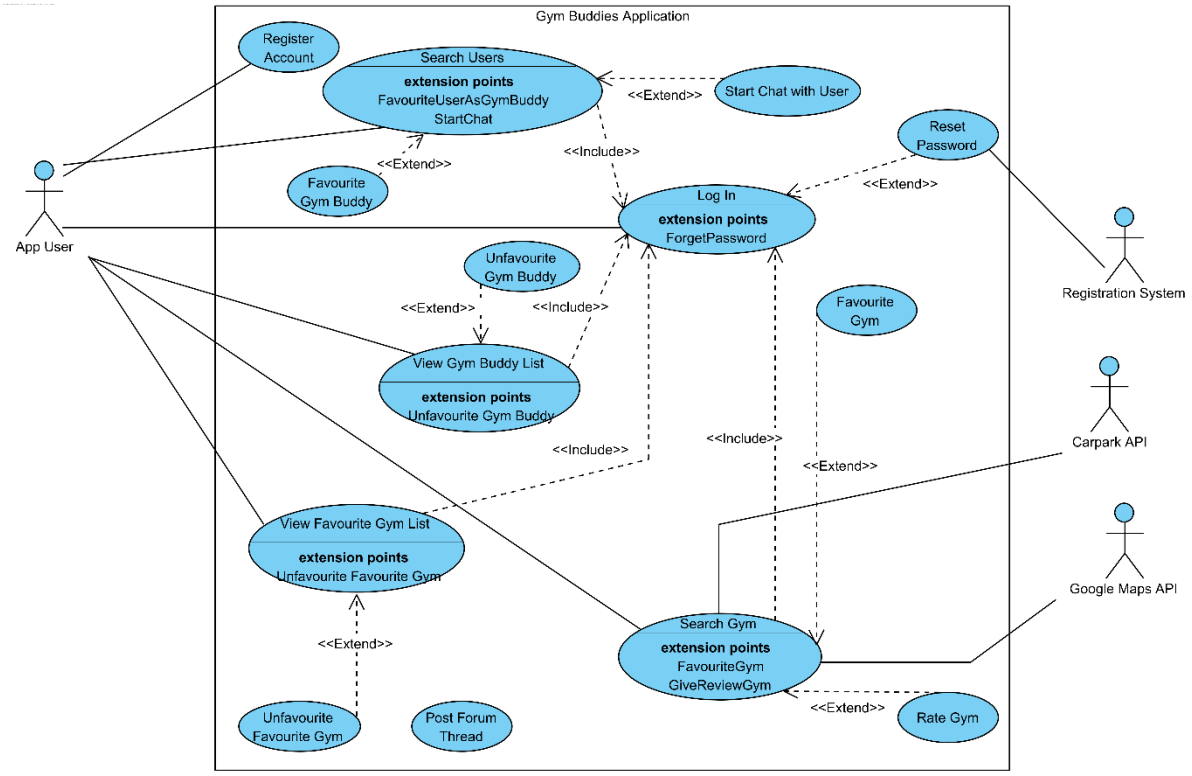
11.2. The user shall leave a rating of a maximum of 5 stars.

11.3. The user must also be able to leave a review.

11.3.1. The review shall be no more than 512 characters.

12. The **user** must be able to **add** a list of favourite gyms.
13. The **user** must be able to **delete** favourite gyms from their list.

Initial Use Case Model



Use Case Description

UC-01: Register User

Use Case ID:	UC-01		
Use Case Name:	Register User		
Created By:	Francis Lim	Last Updated By:	Kong Alson
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows an app user to register for an account which will be stored in the system.
Preconditions:	None
Postconditions:	1. The app user's account is successfully created. 2. The system successfully stores the data of the account created.
Priority:	High
Frequency of Use:	0-1 times per day
Flow of Events:	1. Use case starts when app user selects Create Account. 2. System displays the Account Creation interface. 3. User enters his e-mail address, password and password confirmation. 4. User selects Register Account. 5. System validates information in real-time. 6. System sends a confirmation e-mail to the app user's registered e-mail for validation.

	7. System stores app user's registered account into the database.
Alternative Flows:	**Alternate Flow 1: Password and Confirm Password fields do not match** 5.1. System, upon validating information in real time, displays error message indicating the passwords do not match. 5.2. Use case returns to Step 4. **Alternate Flow 2: The email address has been registered by another user** 2.1. System, upon validating that the email has been registered by other user, displays error message informing the user to register with another email address. 2.2. Use case returns to Step 4.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-02: User Log In

Use Case ID:	UC-02		
Use Case Name:	User Log In		
Created By:	Francis Lim	Last Updated By:	Kong Alson
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the app user to login to use the features of the app.
Preconditions:	1. User must have a registered account. 2. User account must not be locked. 3. Invocation as an included use case by Search Gym 4. Invocation as an included use case by Search Gym Buddy. 5. Invocation as an included use case by View Gym Buddy List. 6. Invocation as an included use case by View Favourite Gym List.
Postconditions:	1. User is successfully logged in.
Priority:	High
Frequency of Use:	0-3 times per day
Flow of Events:	1. Use case starts at the log in page of the application. 2. The user types in the e-mail address and password. 3. System verifies account. 4. User is successfully logged in. 5. User is then allowed to execute use cases Search Gym, Search Gym Buddy, View Gym Buddy List and View Favourite Gym List.
Alternative Flows:	**Alternate Flow 1: User Logs In with Gmail Account**

	1.1. User can also choose to select the login with Gmail Account button. 1.2. System verifies account through Firebase 1.3. Use case returns to Step 3. **Alternate Flow 2: User enters invalid credentials** 2.1. System returns error message indicating invalid username and/or password 2.1. Use case returns to Step 1. **Alternate Flow 3: User's account is locked** 3.1. System will display error message informing the user that the user has keyed in invalid Password for 5 consecutive times and prompt the user to reset Password. 3.2. System invokes UC-03 Reset Password alternate flow 2.
Exceptions:	**Exception Flow 1: Gmail Account not Registered** 2.1. System verifies through Firebase that the Gmail account is not registered with the application. 2.2. Register User use case is invoked.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-03: Reset Password

Use Case ID:	UC-03		
Use Case Name:	Reset Password		
Created By:	Francis Lim	Last Updated By:	Sam Jian Shen
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the app user to reset the account password.
Preconditions:	1. User has an existing account in the app database
Postconditions:	1. System replaces password for the user account in the database.
Priority:	Low
Frequency of Use:	0-1 times per day
Flow of Events:	1. Use case UC-02 User Log In is invoked until Step 1. 2. User selects Forget Password button. 3. System displays Forget Password interface 4. User enters e-mail in the text field. 5. System verifies e-mail address and sends password reset e-mail to user's email address. 6. User clicks on link in user's e-mail inbox 7. System displays interface to change password. 8. User enters new password and confirms password input, and selects Change Password button. 9. System verifies and successfully stores new password into database.
Alternative Flows:	**Alternate Flow 1: E-mail does not exist in system** 4.1. System, upon validating information in real time, displays that no such e-mail exists.

	4.2. Use case returns to Step 3. **Alternate Flow 2: System invokes Reset Password use case.** 2.1. Use case UC-02 User Log In alternate flow 3 is invoked. 2.2. Use case continues from Step 3. **Alternate Flow 3: Verification e-mail not received.** 6.1. User does not receive e-mail in their inbox. 6.2. User can request a resend of verification email. 6.3. Use case restarts from Step 5. **Alternate Flow 2: Password and Confirm Password fields do not match** 8.1. System, upon validating information in real time, displays error message indicating the passwords do not match. 8.2. Use case returns to Step 7.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	1. When the user retrieves the e-mail sent by the system, assume that the user uses a third-party e-mail application on a smartphone. 2. When the user clicks on the link of the reset password e-mail, the link will re-launch a website to allow the user to change their password
Notes and Issues:	-

UC-04: Search Gym

Use Case ID:	UC-04		
Use Case Name:	Search Gym		
Created By:	Francis Lim	Last Updated By:	Kong Alson
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, Carpark API, Google Maps API, System
Description:	Allows the user to search for a gym.
Preconditions:	1. User must be logged in. 2. The device running the application must allow permissions from location services.
Postconditions:	1. The system must display the search result accordingly.
Priority:	High
Frequency of Use:	0-5 times per day
Flow of Events:	1. Use case UC-02 User Log In is invoked. 2. Use case begins when user selects Search. 3. System transitions into interface for Search. 4. User type in the search box and selects the search icon. 5. System returns a Google Maps view of gyms that case-matches the user's input in the search box. 6. User selects list view to view the list of gyms in list view. 7. User selects a specific gym from the list. 8. System returns the location of the specific gym in Google Maps view with a pin indicating the exact location of the gym and its information,

	9. User selects a specific carpark icon. 10. System returns the location of the specific carpark, as well as its information and its carpark rates.
Alternative Flows:	**Alternate Flow 1: User selects a gym to favourite.** 8.1.1. User selects the favourite icon in the gym's information page. 8.1.2. Use case UC-05 Favourite Gym is invoked. **Alternate Flow 2: User selects a gym to rate/review** 8.2.1. User selects the rate icon in the gym's information page. 8.2.2. Use case UC-08 Rate Gym is invoked. **Alternate Flow 3: User selects Current Location button when Searching** 3.1. User may select the current location icon 3.2. System returns a Google Map view of gyms that is within 1km radius from the user. 3.3. Use case returns to Step 5. **Alternate Flow 4: No gym found** 4.1. System returns a message in the results box showing no gyms found. 4.2. Use case ends.
Exceptions:	**Exception Flow 1: App requires permission to access location services** 3.1.1. User selects current location icon. 3.1.2. App requests for permission to allow location services.
Includes:	- UC-02 User Log In
Special Requirements:	
Assumptions:	
Notes and Issues:	

UC-05: Favourite Gym

Use Case ID:	UC-05		
Use Case Name:	Favourite Gym		
Created By:	Francis Lim	Last Updated By:	Kenneth Soh
Date Created:	30/8/2019	Date Last Updated:	27/10/2019

Actor:	App User, System
Description:	Allows the user to add a gym to the user's favourite gym list.
Preconditions:	1. User must be searching for a gym.
Postconditions:	1. The specific gym is stored in the user's favourite gyms list.
Priority:	Medium
Frequency of Use:	0-5 times per day
Flow of Events:	1. Use case UC-04 Search Gym is invoked until Step 7 2. User selects the Favourite icon located in the information display of the gym. 3. System successfully stores the specific gym into the user's favourite list.
Alternative Flows:	-
Exceptions:	-
Includes:	-

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-06: View Favourite Gym List

Use Case ID:	UC-06		
Use Case Name:	View Favourite Gym List		
Created By:	Francis Lim	Last Updated By:	Kong Alson
Date Created:	2/9/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to view their favourite gym list
Preconditions:	1. User must be logged in.
Postconditions:	-
Priority:	High
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-02 User Log In is invoked. 2. Use case begins when user selects Favourite Gym function 3. System displays the user's list of favourite gyms in list view. 4. User filters the gym list according to name 5. System displays and reorders the list to the user's filter preference in alphabetical order. 6. User filters the gym list according to nearest location from user 7. System displays and reorders the list to the user's filter preference, starting from the gym with the shortest distance to the user's current location.
Alternative Flows:	**Alternate Flow 1: User unfavourites a gym.** 3.1. User selects the favourited icon on a gym in the favourite list. 3.2. Use case UC-07 Unfavourite Gym is invoked.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-07: Unfavourite Gym

Use Case ID:	UC-07		
Use Case Name:	Unfavourite Gym		
Created By:	Francis Lim	Last Updated By:	Tang Wei Cong
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to unfavourite gym list.
Preconditions:	1. User must be viewing their favourite gym list.
Postconditions:	1. The user must be able to unfavorite the gym.
Priority:	Low

Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-06 View Favourite Gym List is invoked until Step 3. 2. User press the favorite icon to unfavorite the gym. 3. System updates the content of the list.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-08: Rate Gym

Use Case ID:	UC-08		
Use Case Name:	Rate Gym		
Created By:	Francis Lim	Last Updated By:	Tang Wei Cong
Date Created:	1/9/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to rate and review a gym between 1 to 5 stars.
Preconditions:	1. User must be logged in 2. User must be searching for a gym
Postconditions:	1. The specific gym's rating and review is updated in the system, and its average rating is recalculated.
Priority:	Medium
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-04 Search Gym is invoked until Step 7 2. User selects Rate Gym button on the gym information interface. 3. System displays the gym rate and review interface. 4. User can click between 1 to 5 stars to leave a star-rating for the gym, and can leave a review in the rich textbox up to 512 characters. 5. User selects Submit Rate and Review 6. System processes user's rating and updates the gym's rate and review page.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-09: Search Buddy

Use Case ID:	UC-09		
Use Case Name:	Search Users		
Created By:	Francis Lim	Last Updated By:	Tang Wei Cong
Date Created:	1/9/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to search for other buddies
Preconditions:	1. User must be logged in.
Postconditions:	-
Priority:	High
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-02 User Log In is invoked. 2. Use case begins when user selects the Buddy Search option 3. System displays a gym buddy search interface. 4. User selects their preferences of days, timings and region they desire. 5. User selects Confirm Filter 6. System processes and displays list of other buddies that fits the filters the user applied. 7. User can select a specific user and view that user's profile information.
Alternative Flows:	**Alternate Flow 1: No available user with the applied filters** 7.1.1. System returns an error message indicating that no users matches the filters. 7.1.2. Use case ends. **Alternate Flow 2: User favourites another user as gym buddy** 7.2.1. Use case Favourite Gym Buddy is invoked. **Alternate Flow 3: User starts a chat with another user.** 7.3.1. Use case Start Chat with User is invoked.
Exceptions:	-
Includes:	- UC-02 User Log In
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-10: Favourite Gym Buddy

Use Case ID:	UC-10		
Use Case Name:	Add Gym Buddy		
Created By:	Francis Lim	Last Updated By:	Kenneth Soh
Date Created:	1/9/2019	Date Last Updated:	3/9/2019

Actor:	App User, Reciprocating App User, System
Description:	Allows the user to add a gym buddy to the user's gym buddy list.
Preconditions:	1. User must be logged in 2. User must be searching for a user.
Postconditions:	1. The specific user is added into the user's gym buddy list.
Priority:	Medium
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-09 Search Users is invoked with alternate flow 2. 2. User clicks on the favourite icon on the user's profile information page.

	3. System processes user request and adds target user to current user's gym buddy list.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-11: View Gym Buddy List

Use Case ID:	UC-11		
Use Case Name:	View Gym Buddy List		
Created By:	Francis Lim	Last Updated By:	Kenneth Soh
Date Created:	2/9/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to view their gym buddy list.
Preconditions:	1. User must be logged in.
Postconditions:	-
Priority:	High
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-02 User Log In is invoked. 2. Use case begins when user selects Gym Buddy List function 3. System displays the user's list of gym buddies in list 4. System displays and reorders the list to the user's filter preference in alphabetical order.
Alternative Flows:	**Alternate Flow 1: User unfavourites a gym buddy.** 3.1. User selects a specific gym buddy. 3.2. System displays a page containing the user's profile. 3.3. User selects the favourite icon on the user's profile page to unfavourite. 3.4. Use case UC-12 Unfavourite Gym Buddy is invoked.
Exceptions:	-
Includes:	- UC-02 User Log In
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-12: Unfavourite Gym Buddy

Use Case ID:	UC-12		
Use Case Name:	Unfavourite Gym Buddy		
Created By:	Francis Lim	Last Updated By:	Hsiao Chiayu
Date Created:	30/8/2019	Date Last Updated:	3/9/2019

Actor:	App User, System
Description:	Allows the user to search and delete from the user's gym buddy list.
Preconditions:	1. User must be logged in.

	2. User must have at least 1 other user in the gym buddy list
Postconditions:	1. The user's gym buddy list is modified accordingly.
Priority:	Low
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-11 View Gym Buddy List alternate flow 1 is invoked. 2. User press the favorite icon to unfavorite the buddy. 3. System updates the content of the list.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

UC-13: Start Chat with User

Use Case ID:	UC-13		
Use Case Name:	Start Chat with User		
Created By:	Francis Lim	Last Updated By:	Hsiao Chiayu
Date Created:	1/9/2019	Date Last Updated:	3/9/2019

Actor:	App User(s), System
Description:	Allows the user to message and chat with another user.
Preconditions:	1. User must be logged in. 2. User must be searching for another user.
Postconditions:	1. Chat history is stored in the system.
Priority:	Medium
Frequency of Use:	0~5 times per day
Flow of Events:	1. Use case UC-09 is invoked. 2. User can click the chat icon of the other user to enter a conversation. 3. System displays a messaging interface. 4. System sends the message to the reciprocating app user. 5. Reciprocating app user receives the user's message. 6. System stores message history between the 2 users.
Alternative Flows:	**Alternate Flow 1: Reciprocating user sends back a message** 6.1. Reciprocating app user enters a message up to 1024 characters and selects the send message icon. 6.2. System sends the message to the original user. 6.3. The original user receives the reciprocating user's message. 6.4. System stores message history between 2 users. 6.5. Use case ends.
Exceptions:	**Exception Flow 1: Message failed to deliver** 5.1. System prompts user that the message did not deliver successfully. 5.2. User presses the unsent message to attempt to send again 5.3. If failed again, return to Exception Flow 1 Step 5.1. 5.4. Use case continues.
Includes:	-

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Class Diagram

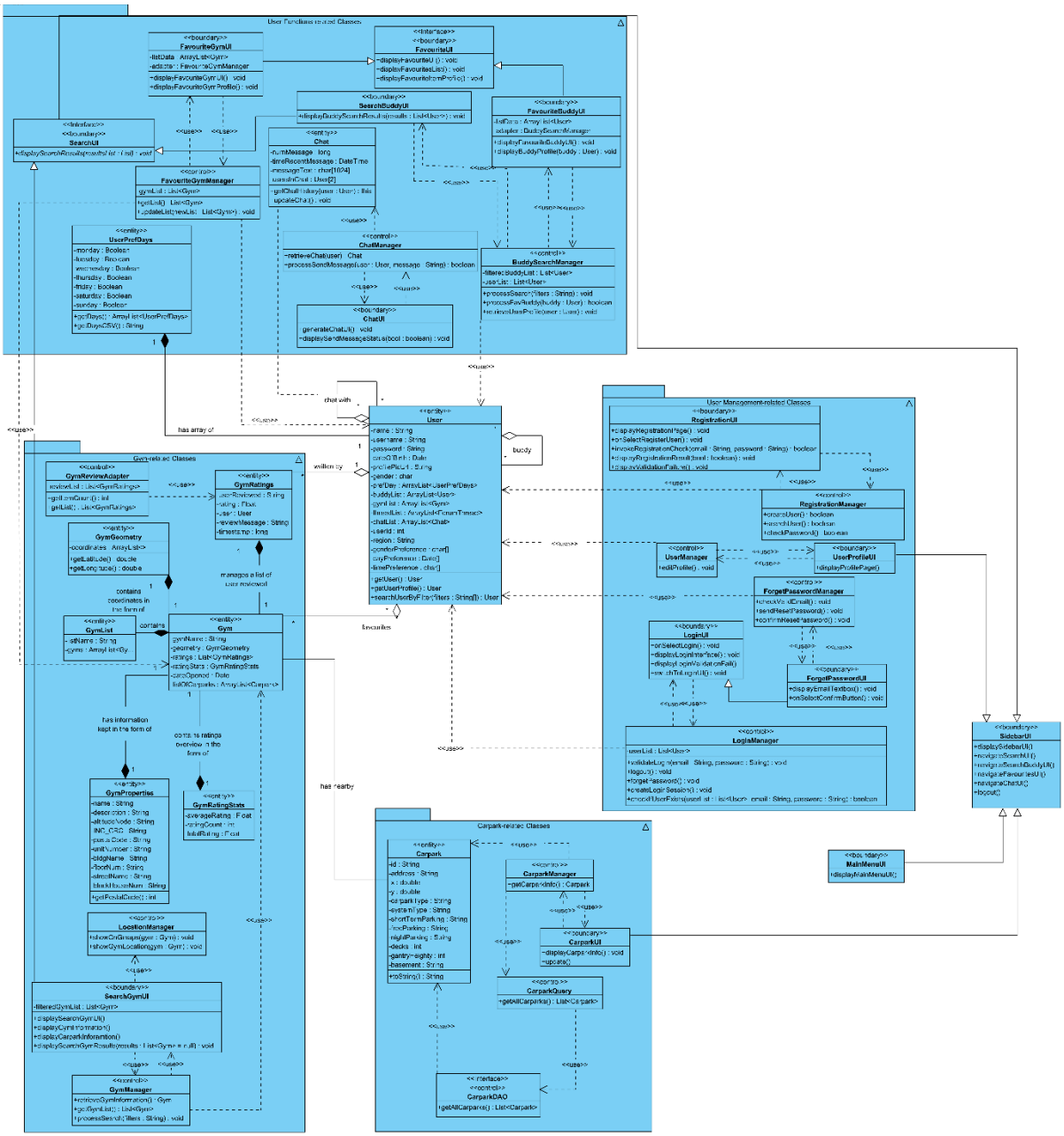


Figure 1: Class Diagram of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above diagram is the Design Level Class Diagram for the GymBuddies Application.

The team has applied certain assumptions to design the above diagram, as follows:

1. Class diagram does not strictly depict the actual implementation of the application, but however a close depiction of the application
2. The team is developing an Android Application, which contains activity windows and fragments that would be complex to be illustrated here.
3. Attributes and operations in the classes are related to the main sequence diagrams depicted in the same project, and may not reflect the actual operation implementation of the application.

State Machine/Dialog Diagram

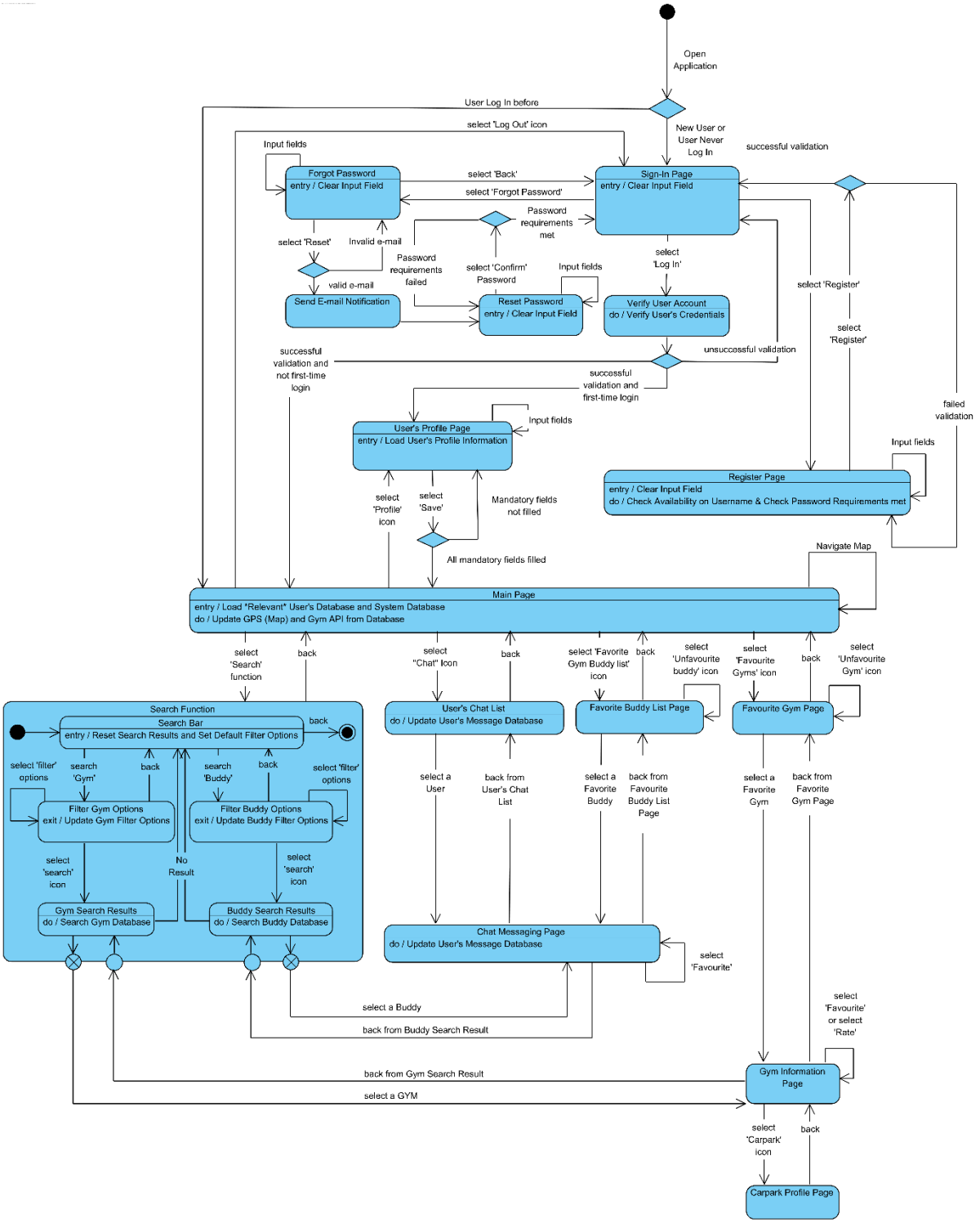


Figure 2: State Machine Diagram of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above diagram is the Initial Dialog Map, or State Machine Diagram for the GymBuddies Application.

The team has applied certain assumptions to design the above diagram, as follows:

1. Users can arbitrarily terminate the application. In this case, the system will store the session of the user, thereafter when the user logs in again, he will be returned to the initial state.
2. If user loses focus of the application, upon gaining focus, the application shall return to the last page the user was at before losing focus.
3. Unsuccessful validation includes the following conditions:

   i. The user account has been banned
   ii. The user account does not exist or has entered invalid e-mail address and/or password.
   iii. The user has attempted to log in more than 5 times previously, and is now blocked from signing in.
   iv. Validation has timed out.

   If the negation of these conditions are all fulfilled, the validation is successful and user is redirected to the application main page.

4. If user do not select filter option during search, it will be search in default settings of the filtering option.
5. The team is developing an Android Application, which contains activity windows and fragments that would be complex to be illustrated here.
6. Attributes and operations in the classes are related to the main sequence diagrams depicted in the same project and may not reflect the actual operation ir ıplementation of the application.

Sequence Diagrams

User Log In

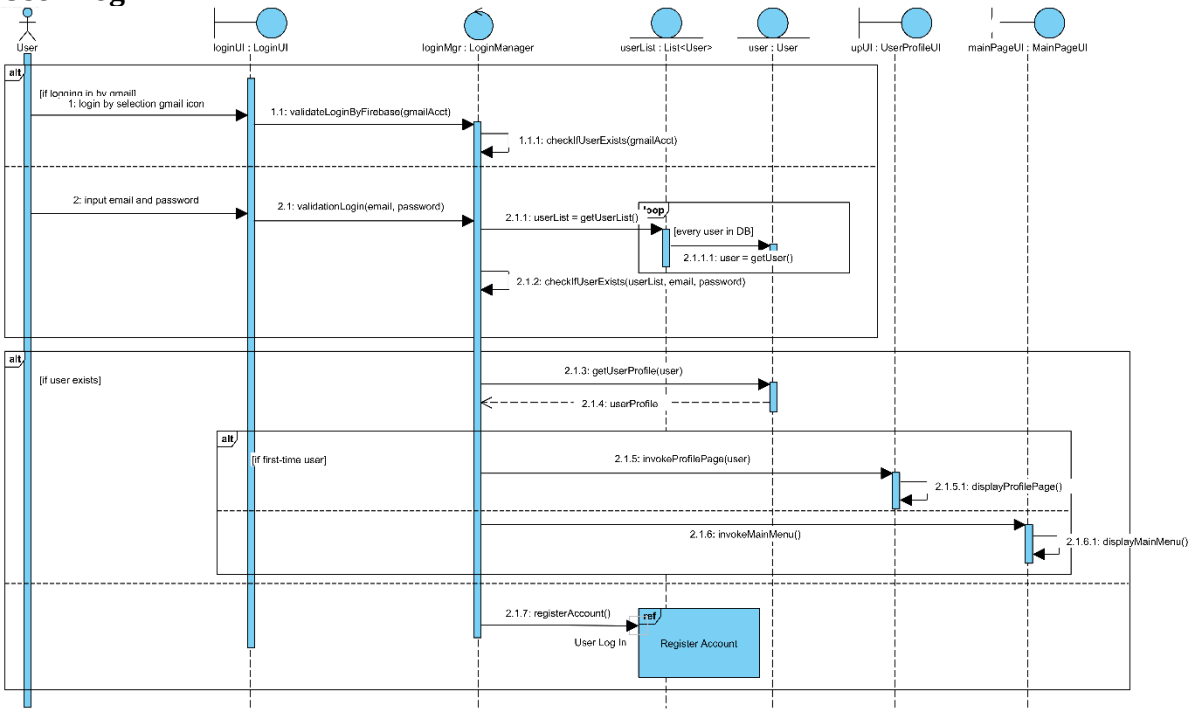


Figure 3: Sequence Diagram of User Log In flow of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above sequence diagram shows the flow of the User Log In Use Case. Assumptions made includes:

- In the first if-else fragment, the else fragment assumes the user logs in using the primitive e-mail and password method.

Register Account

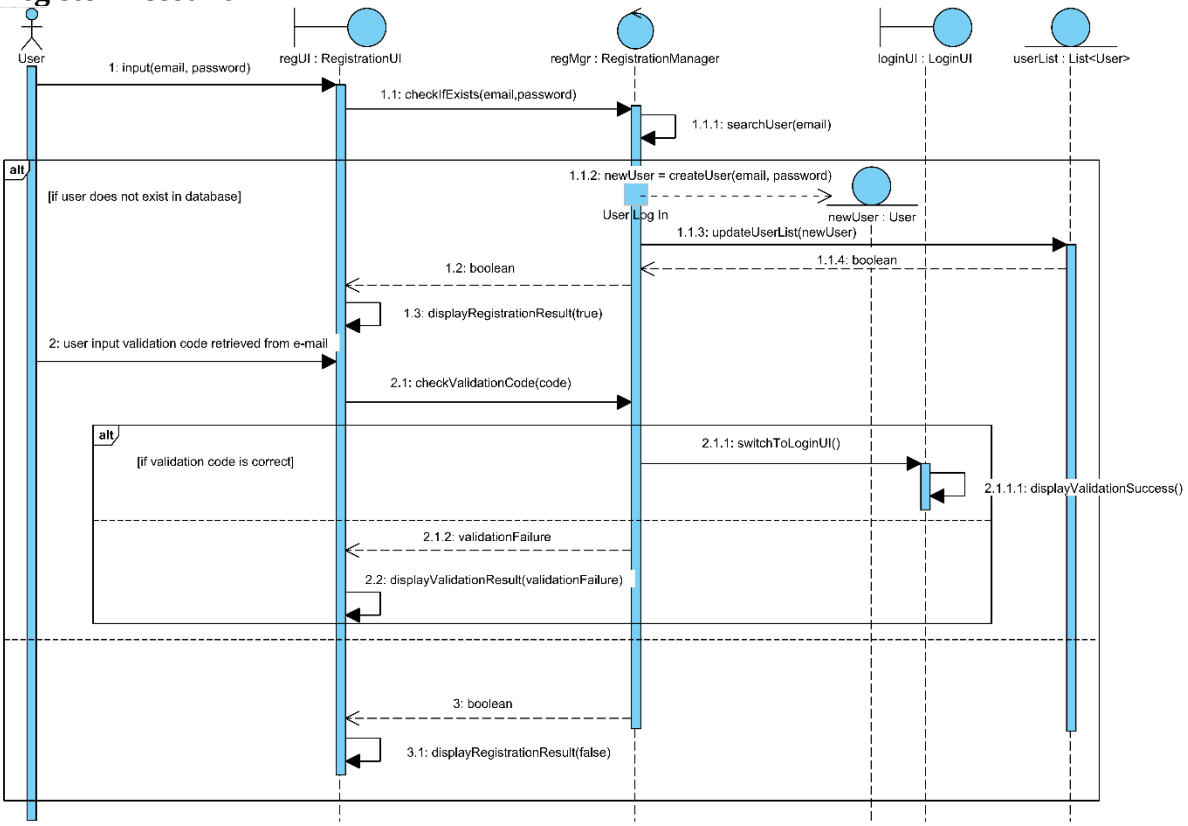


Figure 4: Sequence Diagram of Register Account flow of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above sequence diagram shows the flow of the Register Account Use Case. No specific assumptions were made in the diagram.

Search Gym

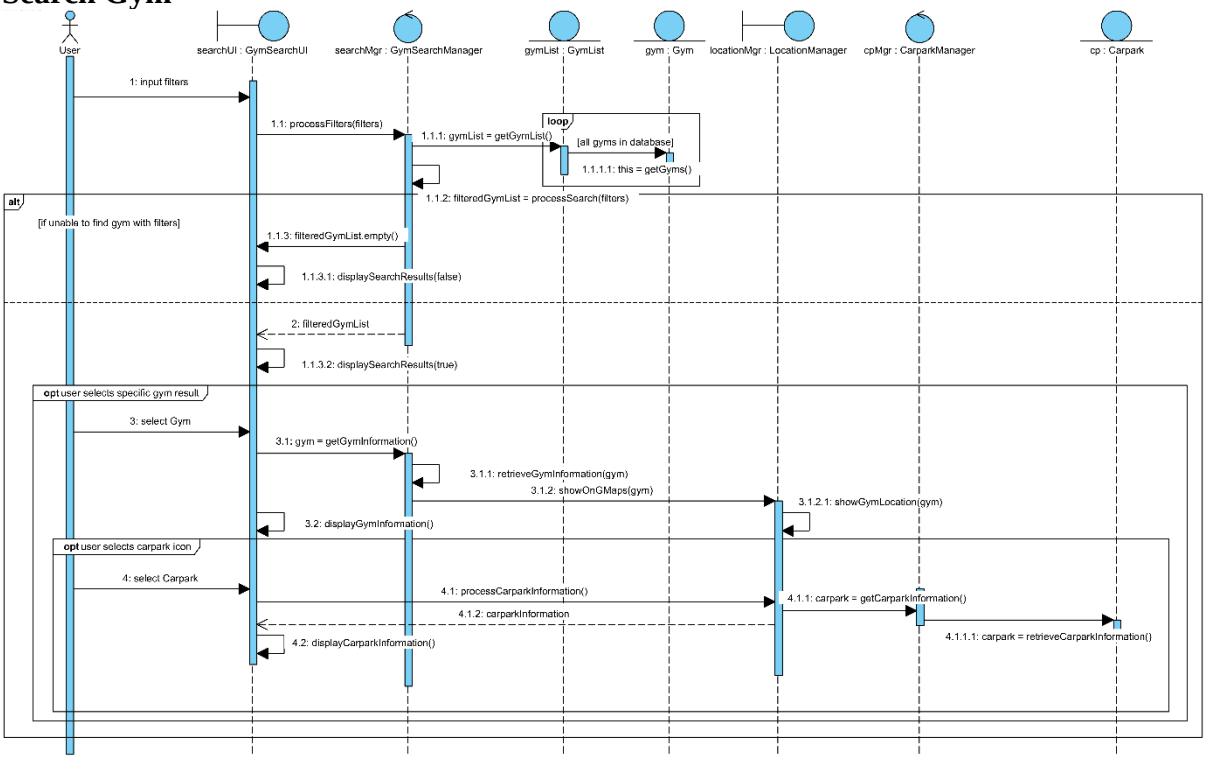


Figure 5: Sequence Diagram of Search Gym flow of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above sequence diagram shows the flow of the Search Gym Use Case. Assumptions made includes:

- Upon successful search, i.e. search results return 1 or more gym, the user can choose to rate or favourite the gym
- The Search Result will show in Google Map View as well as list view for user to select

Search Gym Buddy

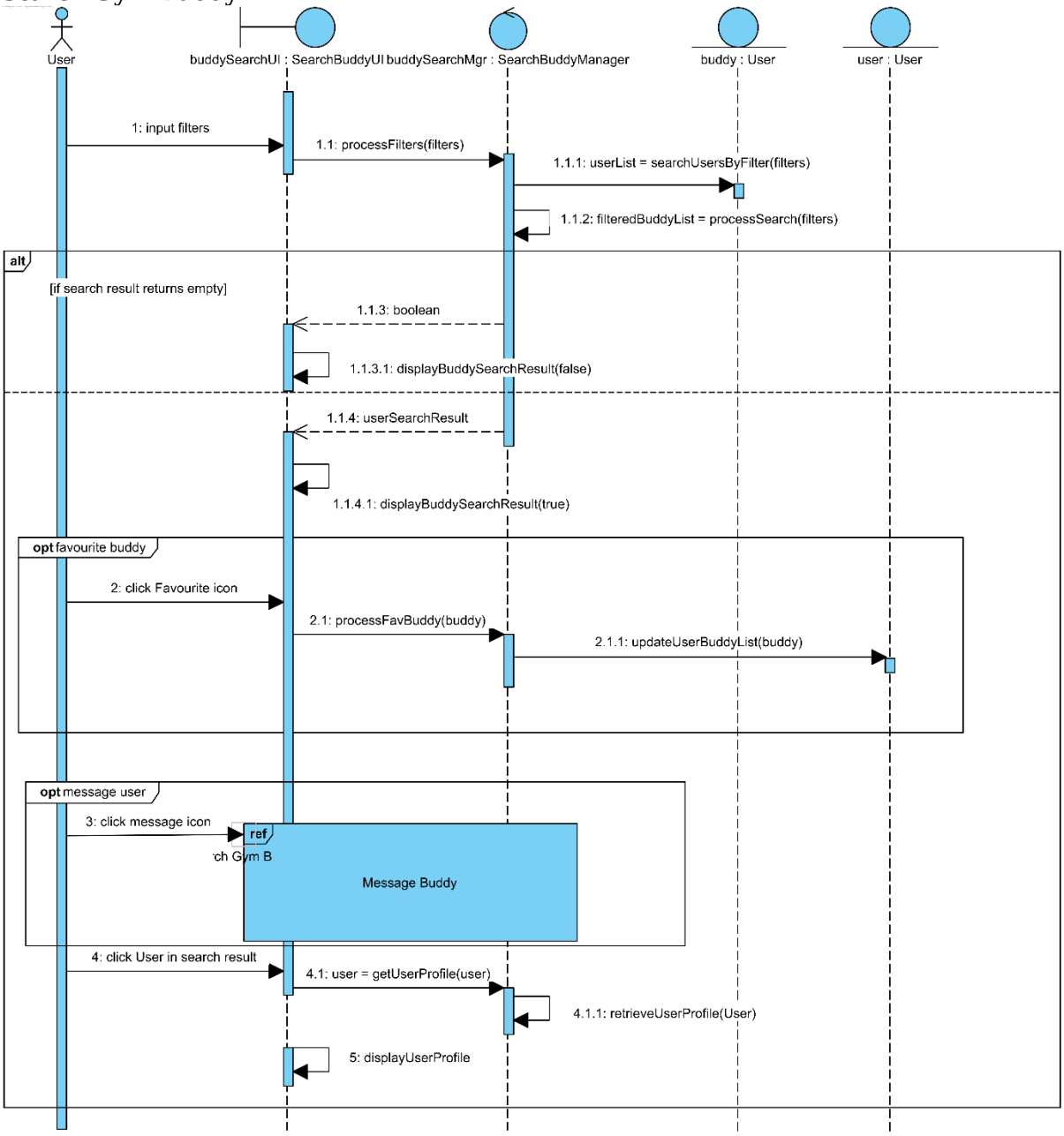


Figure 6: Sequence Diagram of Searvh Gym Buddies of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above sequence diagram shows the flow of the Search Gym Buddy Use Case. Assumptions made includes:

- Upon successful search, i.e. search results return 1 or more users, the user can choose to favourite the buddy or chat with the buddy

Message Buddy

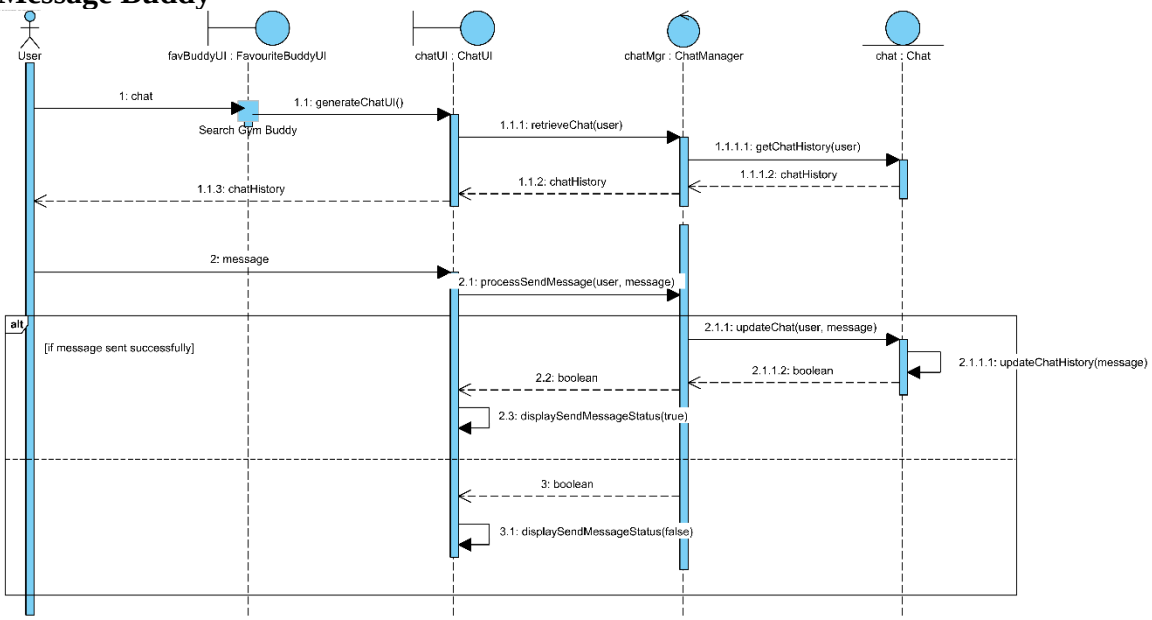


Figure 7: Sequence Diagram of Message Buddy flow of GymBuddies application. The full size diagram can be found in the VPP file in the submission folder, as well is in PNG format.

The above sequence diagram shows the flow of the Message Buddy Use Case. Assumptions made includes:

- This sequence diagram assumes messaging starts from the Favourite Buddy interface; a gate is available to access this sequence from Search Gym Buddy SD, where users can directly choose to chat with users from the search result list
- For the if-else fragment: if the message failed to send, the system will prompt the user to resend again

Non-Functional Requirements

The non-functional requirements of the project are as follows:

Usability

- User Interfaces' fonts size should be scalable to certain degree from "Roboto" size 8 to 24. To ensure the text clearly and comfortably towards the user.
- 1st time login users must be given a guide option to facilitates the functionality offered by the application.
- User Interfaces' layout must be consistency to ensure smooth and pleasant experiences for the users.
- Main Menu icon should be visible in all pages except itself to ensure user quick access to all possible functions.
- All pages should have a function that allows user to go back to previous pages.

Supportability

- The GymBuddies application will support users with Android smartphones that has Android Version 5.0 (Lollipop, API Level 21) or above installed.

Reliability

- Informative error messages will be displayed to the user when the application returns exceptional cases.
- New buddies must be shown when added into the buddy list.

Security

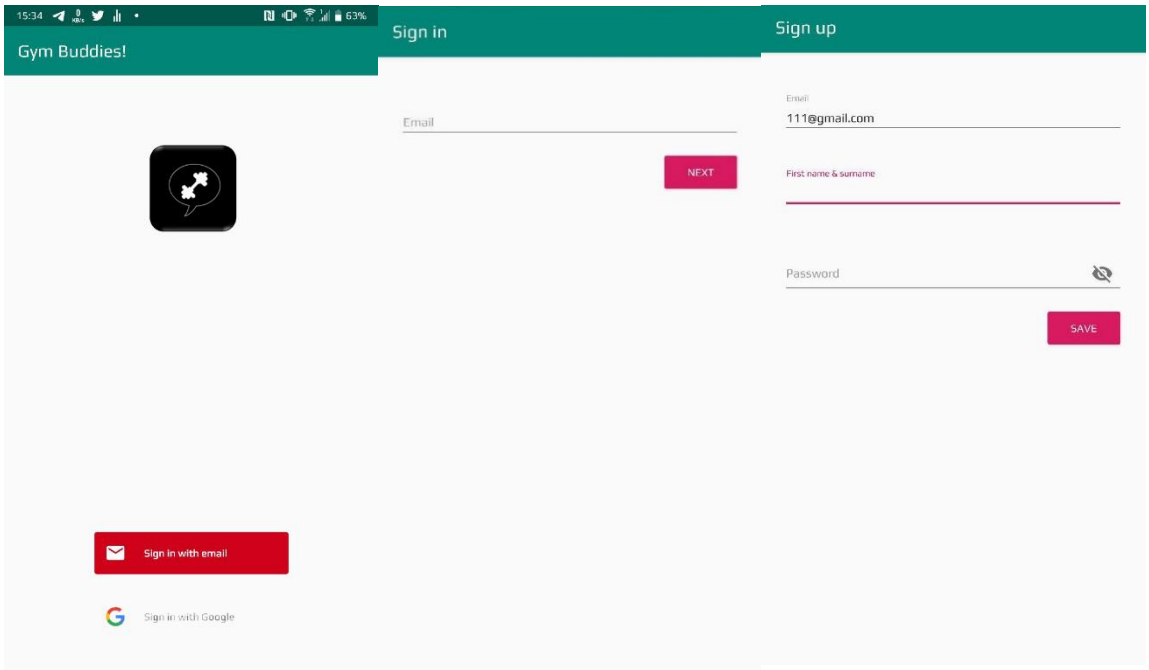
- User's profiling data is shared across other users.
- User's should not be able to see other users' real identity vice versa. Exceptional case is when user shared his/her information in their mutual chat.

Performances

- The application system must be able to support from a minimum of 2000 users online concurrently.
- If there were to be future modifications to the application, the system's performance should be affected to a maximum reduction of 10%

User Interface

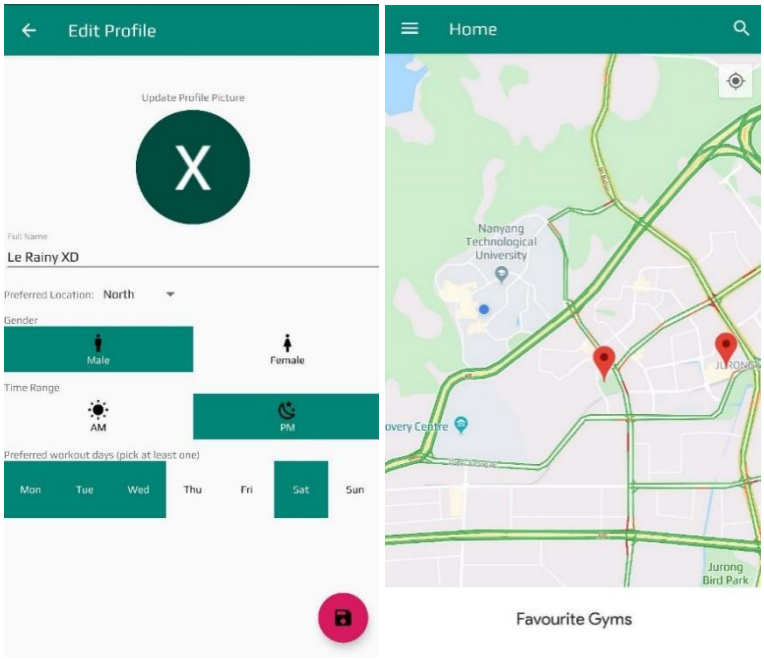
The screenshots in this section are initial mockups of the application created by the team. The mockups do not reflect the actual interfaces of the live application.



Screenshot 1: Login screen mock-up of GymBuddies

Screenshot 1 shows the login screen for the application. Basing the login screen off other applications which require login and registration, the design is kept minimalistic and intuitive. Users will be able to sign in using their e-mail, or via their Google account. The application utilizes Google Firebase to keep track of database records such as registered users. When the user clicks on sign in by e-mail icon, the user will be prompted to enter their e-mail and password. If they do not have an account, the user can proceed to click on Register to register for an account. The system will send an e-mail to the user's registered

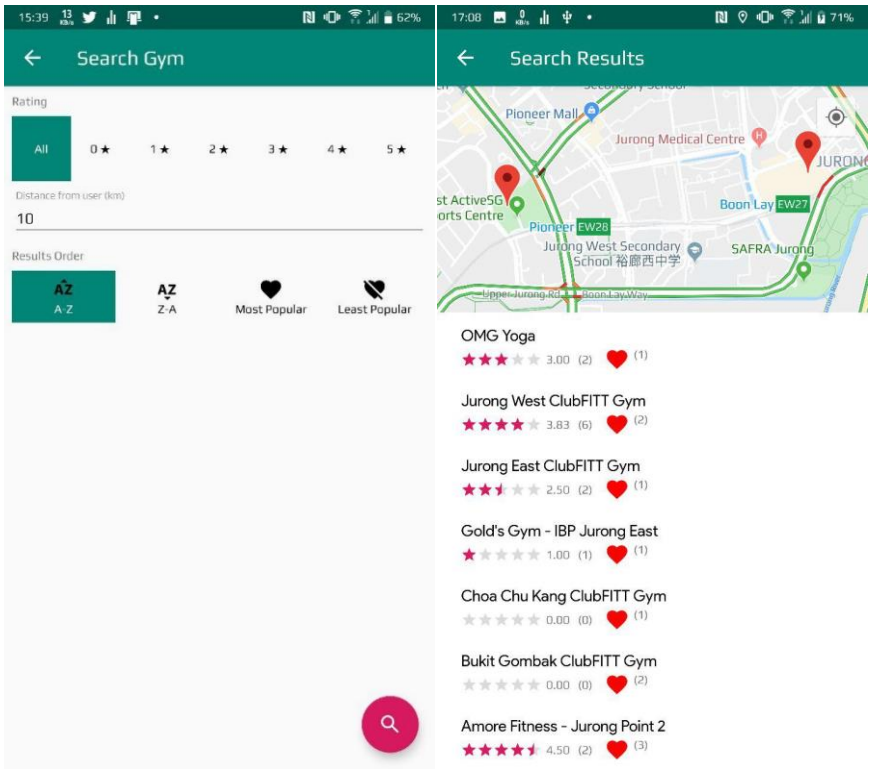
email address, which the user needs to access their inbox and hit accept. The application will validate whether the e-mail exists, and if it does not, the user will be prompted to register.



Screenshot 2: User profiling and main page mock-up of GymBuddies

Screenshot 2 shows the user profile page and the main landing page of the application. When users log in for the first time – after having successfully registered and validated their account – they will need to fill up their profile with their particulars such as their name, a nickname for the application, the region which the user stays at, their gender, their day preference and time preferences before saving the information. After saving, the app will transit to the main landing page on the right – this will be the default landing page for users re-logging in (i.e. it is not their first time logging in).

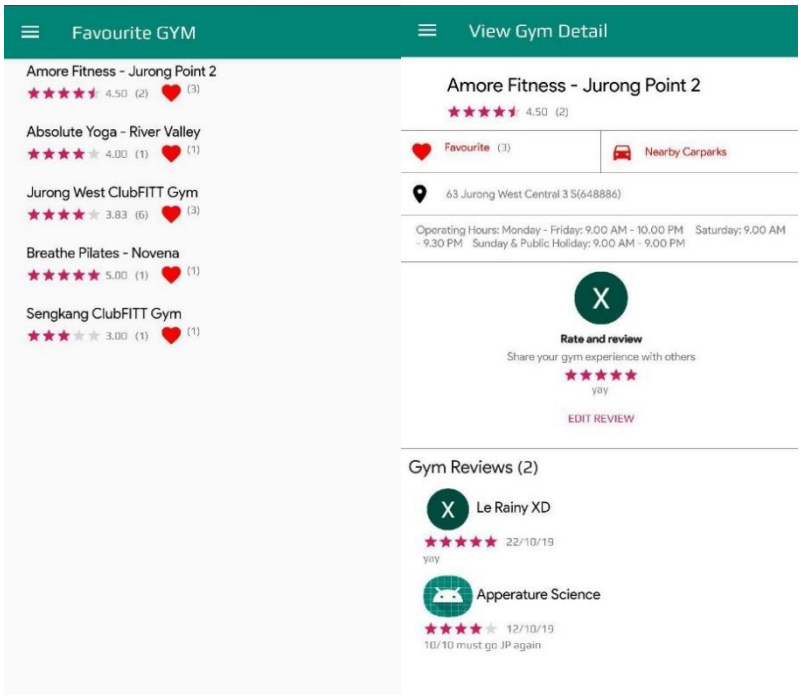
The landing page, or home page, will show a Google Maps screen of their current location, as well as nearby gyms to a radius of 1 kilometer from their current location. There is also a sidebar to access functions like their profile page, list of gym buddies, searching for gym, chat list, as well as an option to log out.



Screenshot 3: Search Gym and Search Results Mock-up of GymBuddies Application

The left screenshot above shows the search gym interface of the GymBuddies Application. The search interface allows the users to search the gyms based on the gym's average ratings, based on the distance from user – the radius in kilometers, as well as sorting by alphabetical order, and by popularity order.

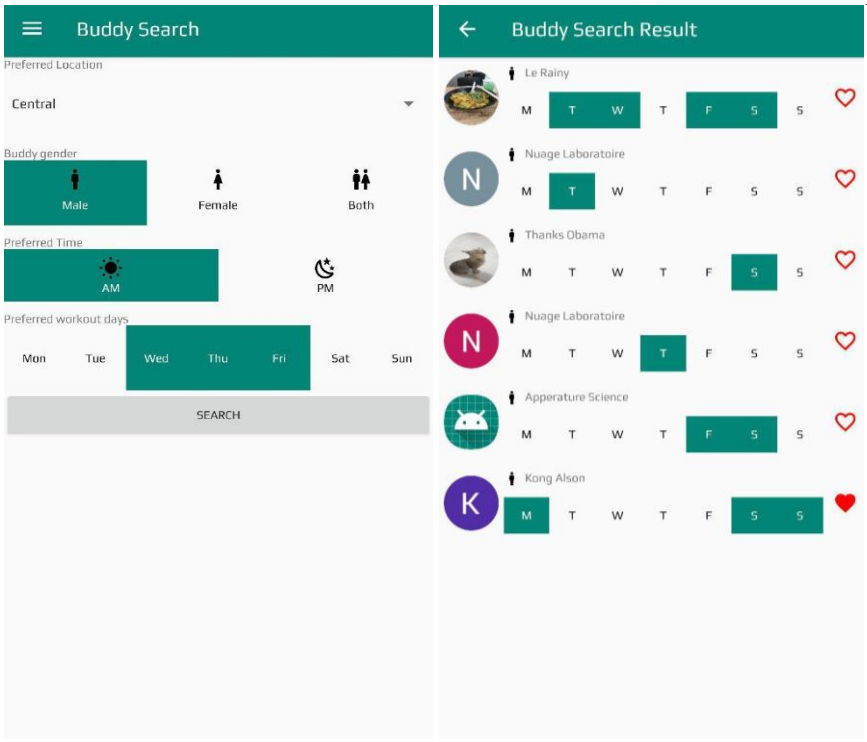
If there are results, search results will be shown as per the screenshot on the right above. In the screenshot, the search results are ordered in descending order, with a distance of 10km from the user. The user can then press into a specific search result to see informaton of the gym, or press on the favourite icon to add the gyms into their favourite gym list.



Screenshot 4: Gym Information Page and Favourite Gym List Mock-up of GymBuddies Application

The left interface on Screenshot 4 shows the gym information page, where users, after tapping on a particular pin on the map pointing to a gym, can swipe up from the bottom to view information of the gym, such as the name, location, description, operating hours, rating and reviews, as well as the option to rate and review the gym as well.

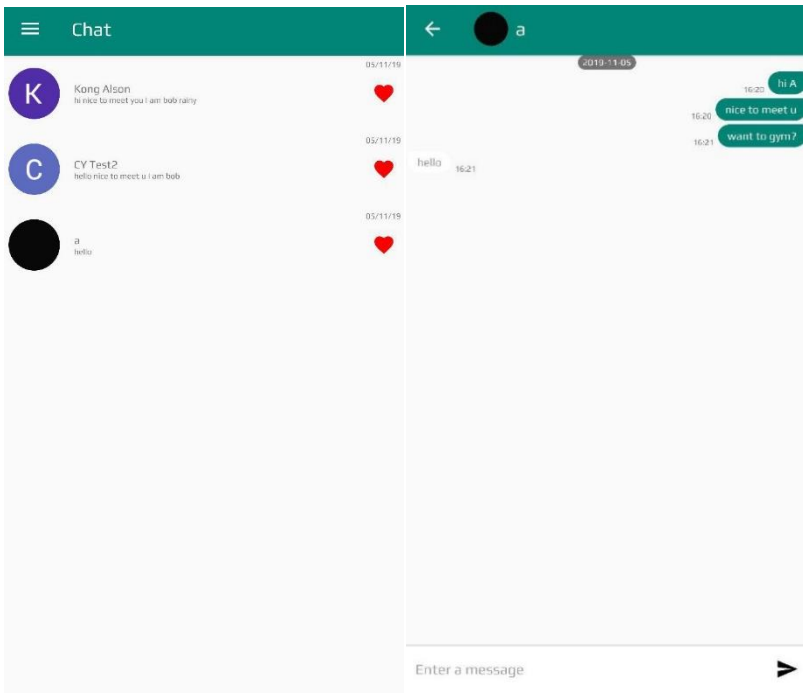
The right interface on Screenshot 3 shows the favourite gym list screen of the application. The screen shows a list of gyms the users has favourited. Each of the gym will show details such as the name, number of people reviewed, averaging rating out of 5 stars, and an option to un-favourite the gym. When a gym is tapped on, the gym information page is shown.



Screenshot 5: Search Buddy and Buddy Search Results Mockup of GymBuddies Application

Screenshot 5 shows the buddy search screen and the buddy search results screen of the application. The current user can choose the search buddy function, which the application will show a screen with filter options for searching of other users by preference. It displays options such as region of stay, gender, time preference of AM or PM, and day of preference from Monday to Sunday. When searched, the system will search for other users based on their profile preferences. If the preferences match the user's search filters, that user will be added to the search results list.

The search results list will display a list of matching users according to the preference chosen from the filter search, with each row display the name, gender, region of stay, as well as an option to initiate a chat with them.



Screenshot 6: Chat List and Chat History Mock-up of GymBuddies Applicaiion

Screenshot 6 shows the chat list and chat history of the application, where users can head into existing chats and message the other user. The chat list will be a scrollable list view with each of the individual chat history with the user. Each simplified chat history will contain the other user's name, the last message sent, the time the last message was sent, and a heart icon to indicate whether the other user is on the user's gym buddy list.

The chat history list will list out all the messages conversed between the users. On the top it will show the other user's display picture as well as their name, and an option to block the user. The main interface shows the messages, and at the bottom is a text box enabling the user to type in the message, with a send button on the bottom-right corner of the screen.

External Interface Requirements

Hardware Interfaces

GymBuddies will be built on Android devices with location services enabled to locate user location.

Software Interfaces

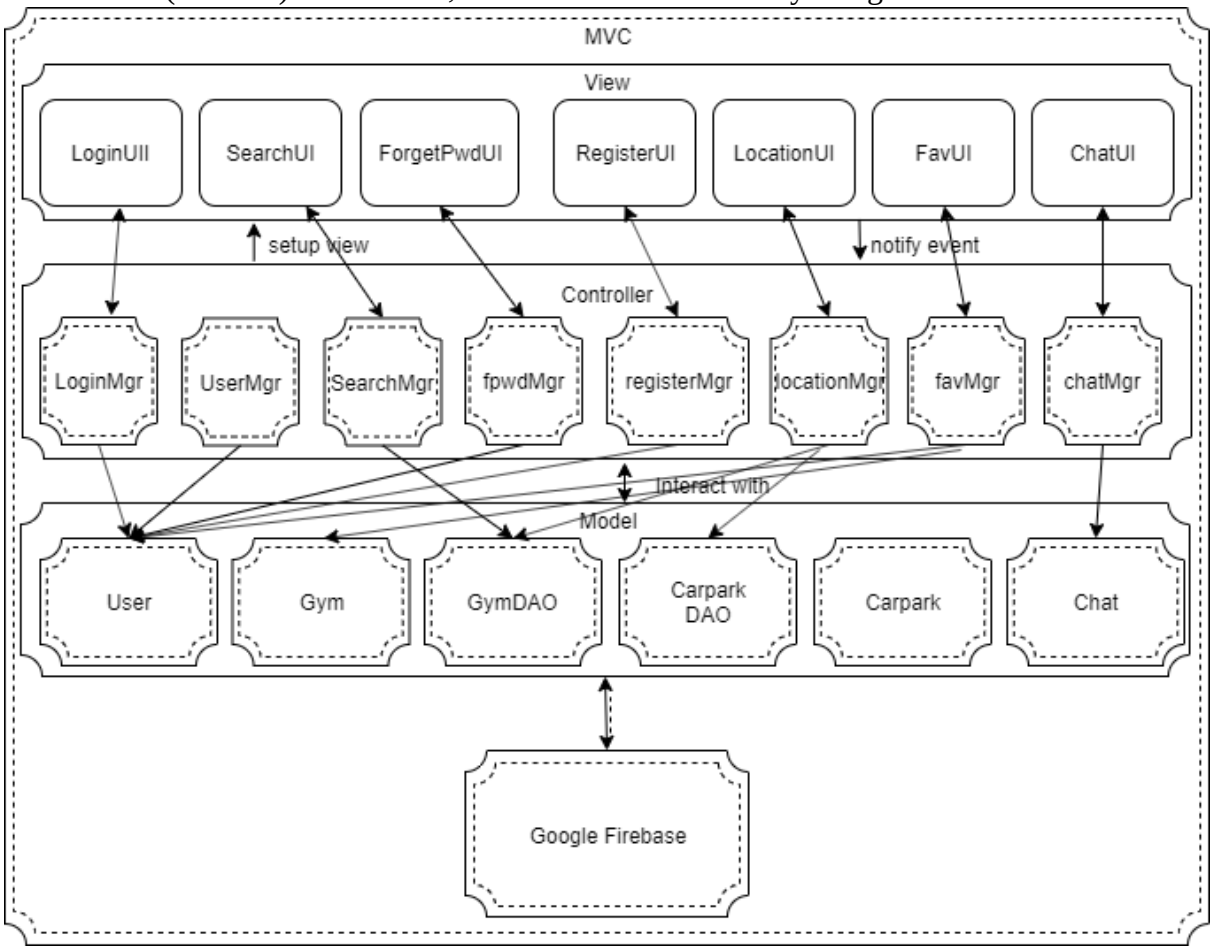
GymBuddies application is being designed to work on Android devices supported with Android 5.X(Lollipop, API Level 21) or above installed. GymBuddies application is utilizing Google's Firebase, which will provide an authentication mechanism on a database. Carpark API is used to display carpark information and Gym JSON file which contains all the gyms in Singapore.

Communications Interfaces

For Communication Interfaces, HTTPS will be used for communication with Firebase for authentication, API for retrieving carpark availability information, DB to retrieve carpark information, and JSON file to retrieve gym details.

Architecture Design

The team is developing an Android application for the GymBuddies Application, therefore the default system architecture design for Android applications is the Model-View-Controller (MVC) architecture. The team has also incorporated MVC along with Model-View-ViewModel (MVVM) architecture, but however is still mainly using the **MVC architecture**.



Screenshot 5: MVC architecture of GymBuddies

Design Patterns

Singleton Pattern

Problem: In our application, certain lists containing data retrieved from APIs will need to be called multiple times: when retrieving the lists, the controller needs to retrieve the list from slow storage, which is very inefficient as it increases the effective access time of data. The team predicts that such lists in the application will be used frequently by the user, and we aim to improve the efficiency of loading.

Solution: We introduce a singleton instance of the list, for the example, the list of gyms retrieved from the dataset.

```
/**
 * An internal property that stores parsed gym lists so that we do not need to parse it again
 */
@JvmStatic private var gymList: GymList? = null

/**
 * Gets the list of gym objects from the JSON file in the application
 * @param context Context Application Context
 * @return GymList? Gyms saved in the JSON file and parsed
 */
@JvmStatic
fun getGymList(context: Context): GymList? {
    if (gymList != null) return gymList
    val json = JsonHelper.readFromRaw(context, R.raw.gymlist)
    val gson = Gson()
    gymList = gson.fromJson(json, GymList::class.java)
    return gymList
}
```

The Kotlin code above in the applications shows the implementation of the Singleton list variable.

When this function is first invoked, it checks if the said list already exists. It will proceed to parse the data from the file stored in slow storage and returns the list. When the function is invoked thereafter, the list will already exist, and therefore there is no need parse data into the list again for subsequently accesses.

The Singleton instance will exist until the application session is killed, and thereafter the need to load from slow storage.

Data Access Object Pattern

Problem: The application needs to retrieves a list of Carparks within a certain distance from the Gym. To do that, we need to retrieve the Carpark information from the application database (which has pulled data from the Carpark API). However, this requires direct access from the application logic to the persistent mechanism, which is the database. This requires the application logic to know the information about the database.

Solution: We implement a Data Access Object (DAO) for Carpark such that the application logic will interact with the CarparkDAO in order to access the database.

```
@Dao
public interface CarParkDao {

    /**
     * getting list of all car parks in room database
     */
    @Query("SELECT * FROM hdbcarparks")
    LiveData<List<CarPark>> getAllCarParks();

    /**
     * Get a list of all carparks from the database
     */
    @Query("SELECT * FROM hdbcarparks")
    List<CarPark> getAllCarParksNow();


}
```

The CarparkDAO abstracts the underlying data access implementation for the objects to enable transparent access to the data source. The objects also delegates data load and store operations to the DAO.

User Acceptance Testing

Black Box Testing

Gmail Login

Test Id	Scenario	Expected Result	Actual Result
1	Login with valid account username and password	The system displays the main page	The system displays the main page
2	Login without valid email	The system prompts the user to enter the email again	The system prompts the user to enter the email again
3	Login with incorrect password	The system will display "wrong password" and prompt the user to enter the correct password	The system will display "wrong password" and prompt the user to enter the correct password
4	Login without filling up the required fields	The system prompts the user to fill up the required field	The system prompts the user to fill up the required field

Specific Case (Gmail Login)

Test Id	Email Address	Expected Result	Actual Result
1	test@gmail.com	Approved	Approved
2	test	Reject	Reject
3	test@gmail	Reject	Reject

Normal Login

Test Id	Scenario	Expected Result	Actual Result
1	Login with valid account username and password	The system displays the main page	The system displays the main page
2	Login without valid email (non-existing email)	The system will display register page	The system will display register page
3	Login without valid email (incomplete email)	The system will prompt the enter the email again	The system will prompt the enter the email again
4	Login with incorrect password	The system will display "wrong password" and prompt the user to enter the correct password	The system will display "wrong password" and prompt the user to enter the correct password
5	Login without filling up the required fields	The system prompts the user to fill up the required field	The system prompts the user to fill up the required field

Specific Case (Normal Login Email Address)

Test Id	Email Address	Expected Result	Actual Result
1	test@yahoo.com(Existing email)	Approved	Approved
2	Non-Existing user	Prompt to Register page	Prompt to Register page
3	test	Reject	Reject
4	test@yahoo	Reject	Reject

Specific Case (Password)(After signing in with correct email)

Test Id	Password	Expected Result	Actual Result
1	testpass	Approved	Approved
2	wrongpass	Reject	Reject
3	Empty	Reject	Reject

Register

Test Id	Scenario	Expected Result	Actual Result
1	Register with valid account username and password	The system will send the verification code and go to login page	The system will send the verification code and go to login page
2	Register without filling up the required fields	The system prompts the user to fill up the required field	The system prompts the user to fill up the required field
3	Register without strong password	The system will prompt the user to enter stronger password	The system will prompt the user to enter stronger password

Specific Case (After entering email and go to register page)

Test Id	email	First name & Surname	Password	Expected Result	Actual Result
1	test@yahoo.com	Abc	testpass	Approved	Approved
2	Empty	Abc	testpass	Reject	Reject
3	test@yahoo.com	Empty	testpass	Reject	Reject
4	test@yahoo.com	Abc	Empty	Reject	Reject
5	test	Abc	testpass	Reject	Reject
6	test@yahoo	Abc	testpass	Reject	Reject

Search Buddy

Test Id	Scenario	Expected Result	Actual Result
1	There exists at least a buddy who satisfy the filter condition	The system displays the available buddies in a list	The system displays the available buddies in a list
2	There do not exists any buddy who satisfy the filter condition	The system displays an empty list	The system displays an empty list
3	Click the favourite icon and if the buddy exists in the buddy list	The system will remove the buddy from buddy list and the icon will become non-filled heart	The system will remove the buddy list and the icon become non-filled heart
4	Click the favourite icon and if the buddy does not exist in the buddy list	The system will add the buddy into the buddy list and the icon will become filled heart.	The system will add the buddy into the buddy list and the icon will become filled heart.

Search Gym

Test Id	Scenario	Expected Result	Actual Result
1	There exists at least a gym who satisfy the filter condition	The system displays the available gyms on the map and in the slide up bar	The system displays the available gyms on the map and in the slide up bar
2	There do not exist any gym who satisfy the filter condition	The system displays the map with no available gym	The system displays the map with no available gym
3	Search without filling up required distance	The system will prompt the user to reenter the distance	The system will prompt the user to reenter the distance
4	The distance is less than or equal to 0	The system will display “distance cannot be less than or equal to 0” and prompt the user to reenter.	The system will display “distance cannot be less than or equal to 0” and prompt the user to reenter.
5	Select the gym	The system will go to the gym information page	The system will go to the gym information page
6	Click the favourite icon and if the gym exists in the gym list	The system will remove the gym from gym list and the icon will become non-filled heart	The system will remove the gym from gym list and the icon will become non-filled heart
7	Click the favourite icon and if the gym does not exist in the gym list	The system will add the gym into the gym list and the icon will become filled heart.	The system will add the gym into the gym list and the icon will become filled heart.

Specific Case (Distance)

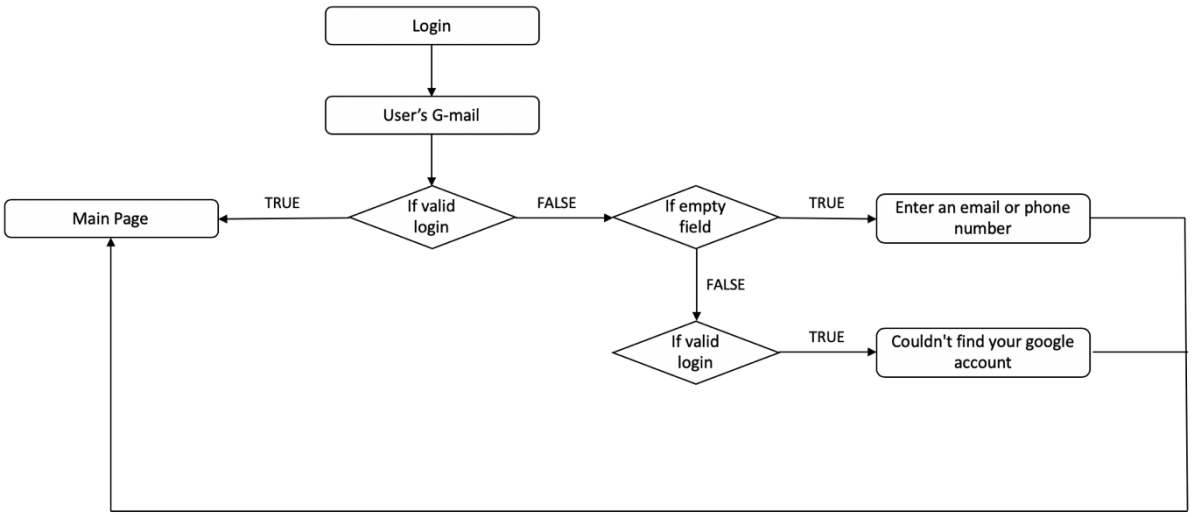
Test Id	Distance	Expected Result	Actual Result
1	1	Approved	Approved
2	1000	Approved	Approved
3	0.1	Approved	Approved
4	0	Reject	Reject
5	-1	Reject	Reject
6	-1000	Reject	Reject
7	-0.1	Reject	Reject

Message Buddy

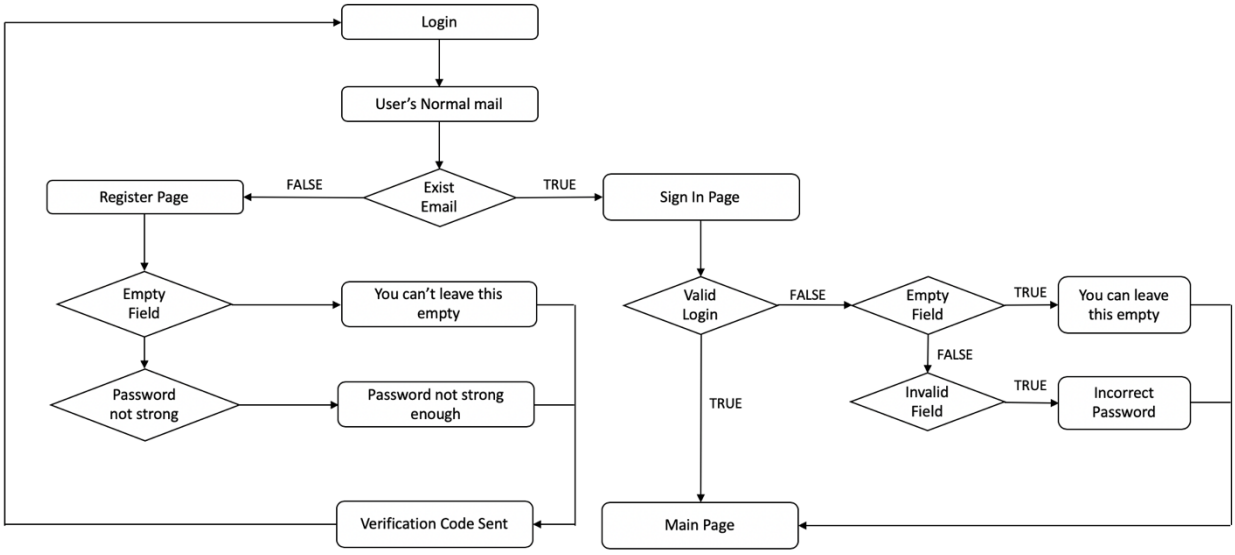
Test Id	Scenario	Expected Result	Actual Result
1	User clicks the chat tab on the side bar	The system displays the chat list	The system displays the chat list
2	If user has been chatting with some other buddies	The system displays the chat list with the buddies	The system displays the chat list with the buddies
3	If user has not been chatting with any other buddy	The system displays empty chat list	The system displays empty chat list
4	User select buddy from the favourite buddy list	The system will display the individual chat page and the user can chat with the buddy	The system will display the individual chat page and the user can chat with the buddy
5	User select buddy from the search function	The system will display the individual chat page and the user can chat with the buddy	The system will display the individual chat page and the user can chat with the buddy

White Box Testing

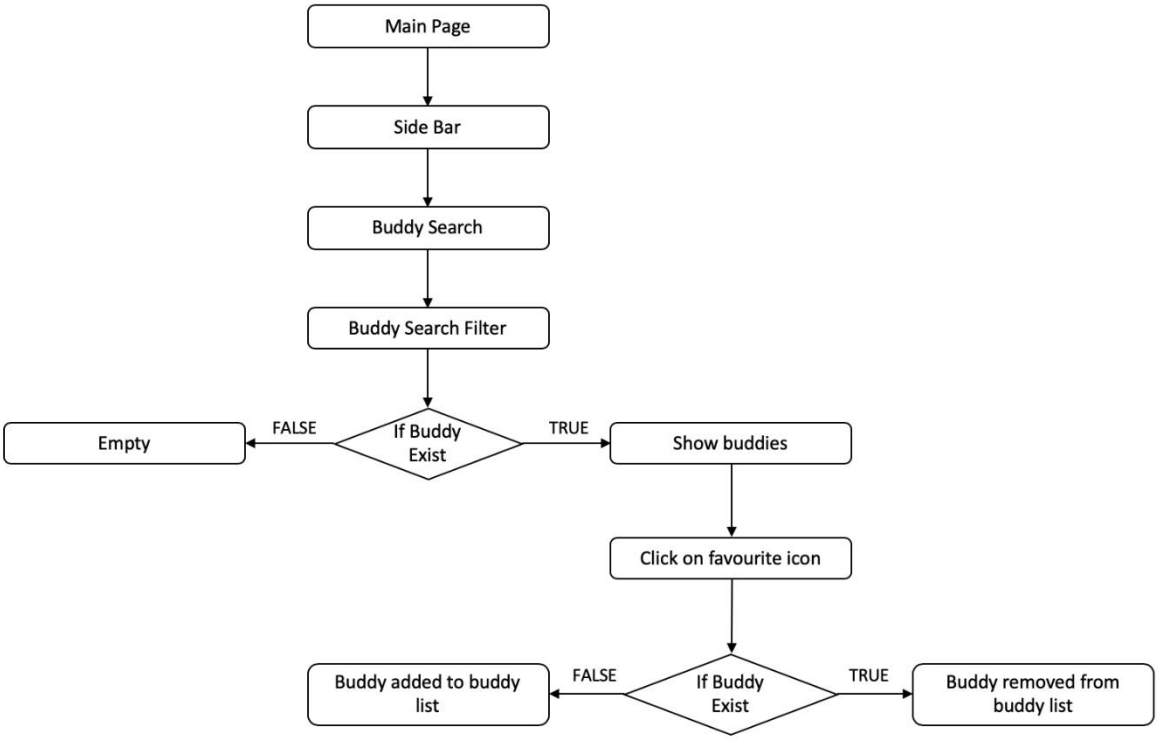
Gmail Login



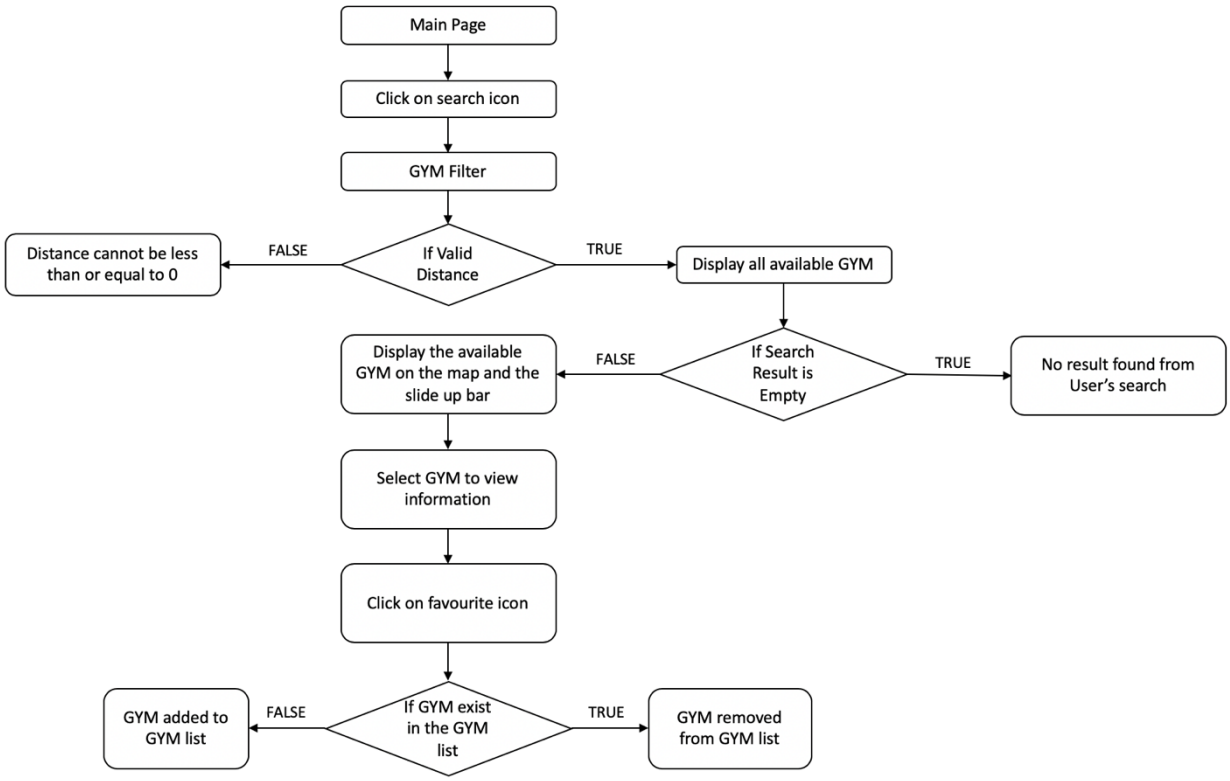
Normal Login & Register



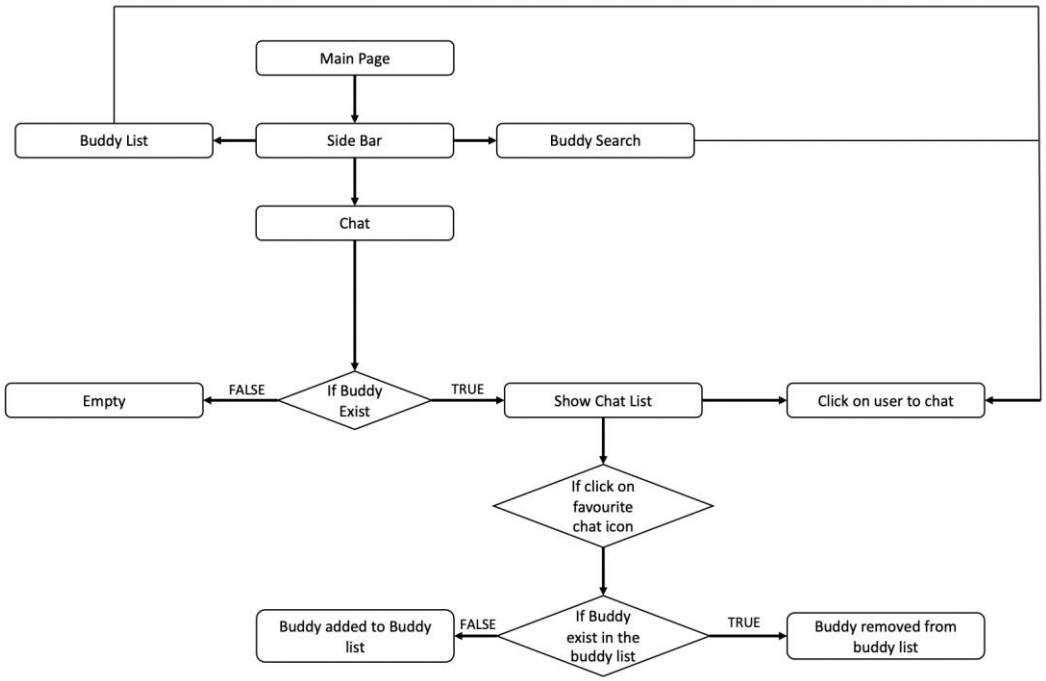
Search Buddy



Search Gym


Main Page
Click on search icon
GYM Filter
If Valid Distance
FALSE
Distance cannot be less than or equal to 0
TRUE
Display all available GYM
If Search Result is Empty
TRUE
No result found from User's search
FALSE
Display the available GYM on the map and the slide up bar
Select GYM to view information
Click on favourite icon
If GYM exist in the GYM list
FALSE
GYM added to GYM list
TRUE
GYM removed from GYM list

Chat



Further Viewing

The video of the application demonstration for GymBuddies is available as an unlisted YouTube video in the following link: https://youtu.be/nBBD-W37vwY

Appendix A: Glossary

Term	Definition
User	A user is a person who is using the application to find gym buddy in Singapore and carpark nearby its location
System	A system refers to the 'GymBuddy' Android mobile application
Rating	A classification of the gym, given by the user, usually from 0 stars to 5 stars, with 0 stars being poor and 5 stars being excellent. The rating is given based on users' opinion towards the gym.
Review	A paragraph containing the user's personal view (i.e. facilities available, cleanliness, etc) of the gym.
Filter Search	A feature that allows the users to search for gym buddy based on his/her own preferences. Search also allows the user to search for gym location.
Profile	Refers to the user's individual information and preference that he/she required to fill up after account registration
Chat	A messaging system in the application that allows users to communicate via peer-to-peer communication.
Gym	A facility that the user searches for the application, and can be favourited and reviewed. The system allows the user to search for gyms.
Gym Buddy	A user that has been favourited by another user, and thus adding them to the second user's favourite list.
Favourites List	A set of lists that stores the user's favourite gyms, and the user's gym buddies.
Report	A user-submitted complaint against another user for violation of terms of use. The report is stored in the system after the user submits it.
Carpark	A location where users pay to allocate their cars to a parking lot. The system checks if there are carparks near gyms.
Location	A well-defined piece of information containing the street name, postal code, and building name as well as unit number (if any).
API	API stands for Application Programming Interface. It is a set of clearly defined methods of communication among various component for building software
User Interface (UI)	The user interface, or UI, is the medium where the user interacts with the system. Each major function in the application (e.g. Search, Favourites) has a separate UI.
MVC	Model-View-Controller: a software architecture for implementing interfaces in the application. Android applications use MVC architecture.
MVVM	Model-View-ViewModel: a software architecture for implementing view-models in the application.

Appendix B: UI Mockup Flow