SW Engineering CSC648/848 Summer 2021

FitHub

Project Application And Name: Find a fitness partner - “FitHub”

Team Info: Team 07 | Error 404

Team Members:

1. Vidhi Vora (Team Lead and GitHub Master) - vvora@sfsu.edu
2. Roberto Simental (Front-End Lead) - rsimental@mail.sfsu.edu
3. Johnson Nguyen (Back-End Lead) - Jnguyen63@mail.sfsu.edu
4. Zhinan Zhao
5. Eduardo Hernandez
6. Ziming Wang
7. Michael Satumba

Milestone: Milestone 2

Date: 08 July 2021

History:

Version	Date
M1V1	22 June 2021
M1V2	07 July 2021
M2V1	08 July 2021

Table Of Contents

1. Data Definitions 3
2. Prioritized Functional Requirements 5
3. UI Mockups and Storyboards (high level only) 10
4. High level database architecture and organization 21
5. High Level APIs and Main Algorithms 30
6. High Level UML Diagrams 31
7. High Level Application Network and Deployment Diagrams 32
8. Identify actual key risks for your project at this time 34
9. Project management 36
10. Detailed list of contributions (this section must be done by the team lead) 37

1. Data Definitions

1. Guest user: A user who hasn't registered or provided their information to FitHub
   1.1 Registration: Ability to register for an account
   1.2 User name: Necessary to have username
   1.3 Email: Necessary to have email
   1.4 Password: Necessary to have password
   1.5 Accept terms of use: Necessary to accept terms of use

2. Registered user: A user who has successfully created their account with FitHub and can access all the app features
   2.1 Login: Necessary to have username and Password
   2.2 User interests: This information will be used by FitHub to show recommendations of people with similar interest
   2.3 User Profile: All the user-related information that can be used to create its profile
   2.4 User logs: Previous workout partners or events that the user has been to can be looked at in the user logs
   2.5 Activities: All the indoor and outdoor activities that Fithub allows a user to search or select as interest.

3. Gym membership: A user's gym membership information. This information will be used in order to sent buddy pass request by other users
   3.1 Buddy Pass Owner: A user having gym membership and willing to share his gym pass with a friend so that they can workout together

4. Events:
   4.1 A user shall post events
   4.2 Events shall be viewed by other users
   4.3 Users can join the event

5. Friends:
    5.1 Users that match based on interest
    5.2 Users that match based on location
    5.3 Users can send friend request
    5.4 Users can accept friend request

6. Friend List:
    6.1 User shall have a friends list

7. Private chat:
    7.1 Users shall be friends
    7.2 Users can send private messages from friends list

8. Group chat:
    8.1 Users shall be friends
    8.2 Users can have group conversation
    8.3 Users can invite other friends

9. Clubs:
    9.1 Users can create clubs
    9.2 Users can join/send request
    9.3 clubs can be public/private

10. Club owner:
    10.1 Clubs can only have one owner

11. Club List:
    11.1 List of clubs users can view
    11.2 List of clubs users can join

2. Prioritized Functional Requirements

P1 (Mandatory)

Guest User:

1. Guest users shall be able to view events posted by other users.
2. Guest users shall be able to select their interests.
3. Guest users shall see other users' user names of people looking to work out nearby.
4. Guest users shall be able to register.
5. Guest users shall be able to Log in as registered users.
6. Guest users shall be able to access the homepage.
7. Guest users shall be able to access the About us page.
8. Guest users shall be able to access the Contact us page.
9. Guest users shall be able to access the Support page
10. Guest users shall be able to delete their account from FitHub

Registered User:

11. Registered users shall be able to access homepage
12. Registered users shall be able to access About us page
13. Registered users shall be able to access Contact us page
14. Registered users shall be able to access Support Page
15. Registered users shall be able to send workout invites to other users.
16. Registered users shall be able to select their interests
17. Registered users shall be able to edit their information
18. Registered users shall be able to update their profile picture
19. Registered users shall be able to delete their profile picture
20. Registered users shall be able to deactivate their account
22. Registered users shall be able to change their account passwords
26. Registered users shall be able to delete their account from Fithub

Searching People

27. Registered users shall be able to have a certain number of swipes per day.
28. Registered users shall be able search for buddies with similar interest
29. Registered users shall be able search for buddies nearby
30. Registered users shall be able to filter search options based on their interests.
31. Registered users shall be able to filter search options based on how far they are willing to travel.

Friends

33. Registered users shall be able to find friends to exercise with.
34. Registered users shall be able to add other registered users as friends.
35. Registered users shall be able to access their friend list
36. Registered users shall DM (direct message) other registered users only if they are friends.
37. Registered users shall have multiple friends if they choose to.
38. Registered users shall have 0 friends if they choose to.
39. Registered users shall have the ability to unfriend a former friend.
40. Register users shall have the ability to block a former friend.
41. Register users shall have the ability to report a former friend.
43. Registered users shall view friend's event postings.
44. Registered users shall be able to decline friend requests.
45. Registered users shall be able to accept friend requests.

Events

46. Registered users shall be able to create events
47. Registered users shall be able to delete events they created
48. Registered users shall be able to edit the events they created
49. Registered users shall be able to join an event
50. Registered users shall be able to exit from an event
51. Registered users shall be able to rejoin an event
52. Registered users shall invite people to events they created

Chats

70. Registered users shall be able to create a private chat
78. Registered users shall be able to reject to join in a private chat
80. Registered users shall be able to accept to join in a private chat

Web Application

89. Web Application shall have About us Page
90. Web Application shall have Contact us page
91. Web Application shall ask user to log in
92. Web Application shall display user's profile
93. Web Application shall show notifications to the user
94. Web Application shall show friend requests
95. Web Application shall allow user to check their messages
96. Web Application shall allow user to check event dates they are planning to go
98. Web Application shall allow user to change password of their account
99. Web Application shall allow user to deactivate their account
101. Web Application shall show the logs of user activities
102. Web application shall show events occurring nearby
103. Web Application shall show people recommendation with similar interest
104. Web Application shall allow user to log out
105. Web Application shall allow user to delete the account
106. Web Application shall allow user to see list of friends
108. Web Application shall allow user to search for people using various filter options

P2 (Desired)

Registered User:

24. Registered users shall be able to send an introductory message to a different user in their search result.
25. Registered users shall be able to share their real time location with FitHub

Buddy Pass

82. Registered users shall be able to view people with gym membership
83. Registered users shall be able to put request to access buddy pass
84. Registered users shall be able to accept the request to access buddy pass
85. Registered users shall be able to reject the request to buddy pass

Web Application shall show buddy pass requests

History logs

86. Registered users shall access their event visits in logs
87. Registered users shall access their workout pairing information in logs
88. Registered users shall be able to access their workout partner's information from logs

P3 (Opportunistic)

Registered User:

21. Registered users shall be able to save their frequent searches

Searching People

32. Registered users shall be able to remove people from their search results

Friends

42. Registered users shall view friend's club postings.

Clubs

53. Registered users shall be able to create the club
54. Registered users shall be able to update the picture of the club they created
55. Registered users shall be able to edit the club name they created
56. Registered users shall be able to delete the club they created
57. Registered users shall be able to delete the club picture they created

58. Registered users shall be able to send request to join the club
59. Registered users shall be able to leave a club
60. Registered users shall be able to send request rejoin a club
61. Registered users shall be able to add members to the club they created
62. Registered users shall be able to remove members from the club they created
63. Registered users shall be able to post in club
64. Registered users shall be member of multiple clubs
65. Registered users shall be admin of multiple clubs
66. Registered users shall be able to invite people to club
67. Registered users shall be able to accept the joining request for club they created
68. Registered users shall be able to reject the joining request for club they created

Chats

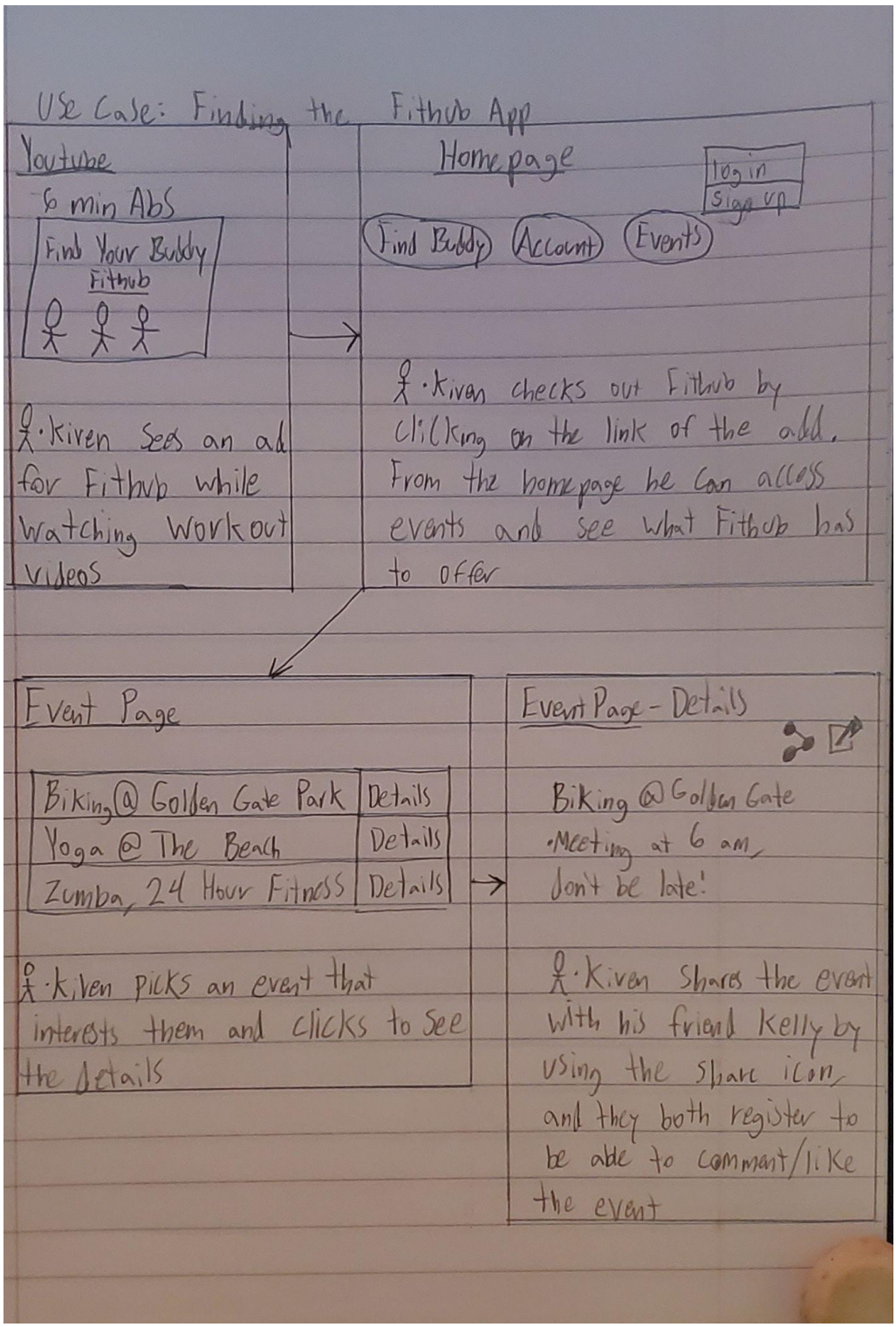
69. Registered users shall be able to create a group chat
71. Registered users shall be able to invite friends to the group chat
72. Registered users shall be able to remove people from group chat
73. Registered users shall be able to add unknown people from the chat
74. Registered users shall be able to post in the group chat
75. Registered users shall be able to leave a group chat
76. Registered users shall be able to dissolution the group chat they cheated
77. Registered users shall be able to recall their posts
79. Registered users shall be able to reject to join in a group chat
81. Registered users shall be able to accept to join in a group chat

Web Application

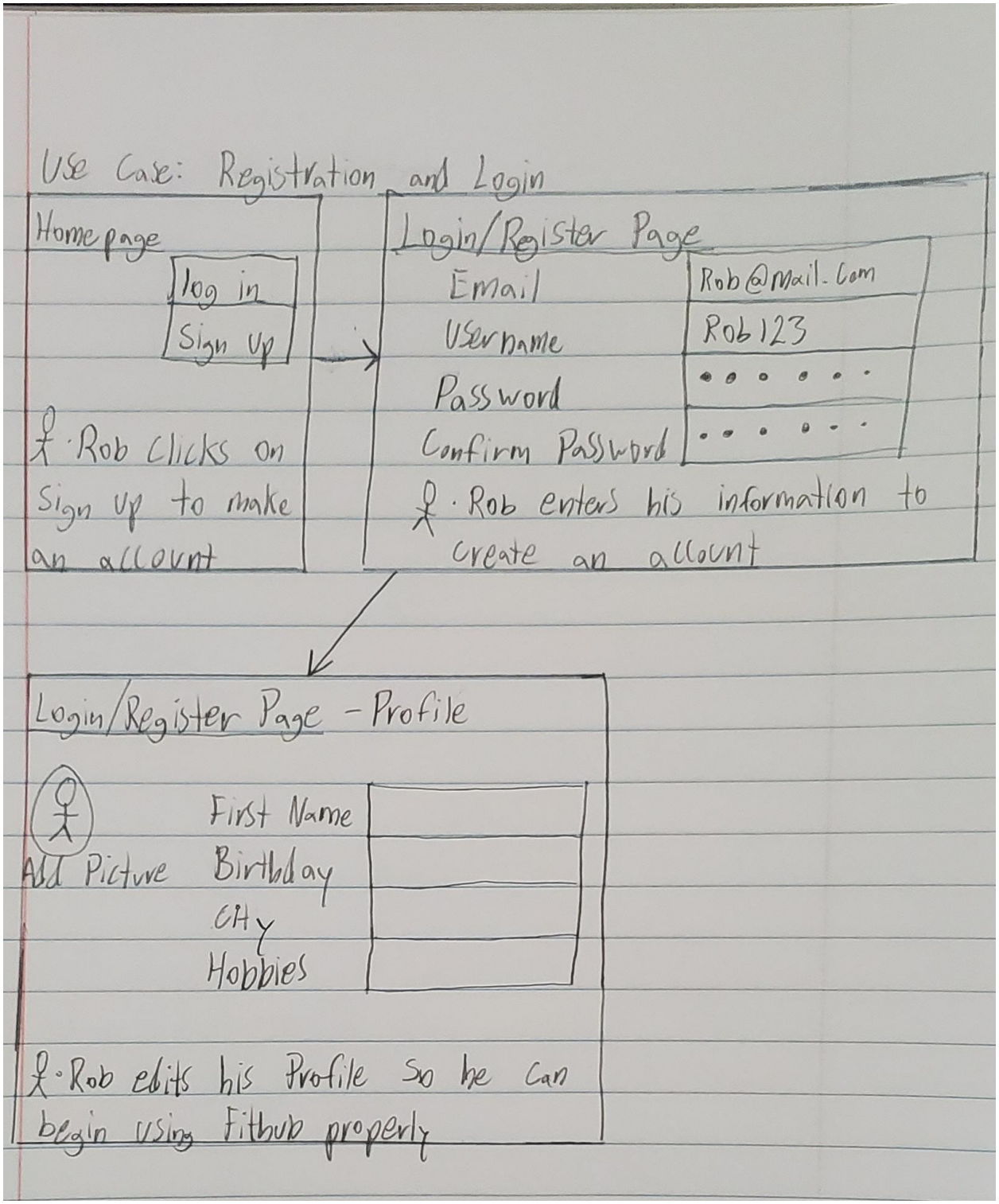
97. Web Application shall allow user to check updates of the club
103. Web Application shall show people recommendation with similar interest
107. Web Application shall allow user to see list of clubs

3. UI Mockups and Storyboards (high level only)

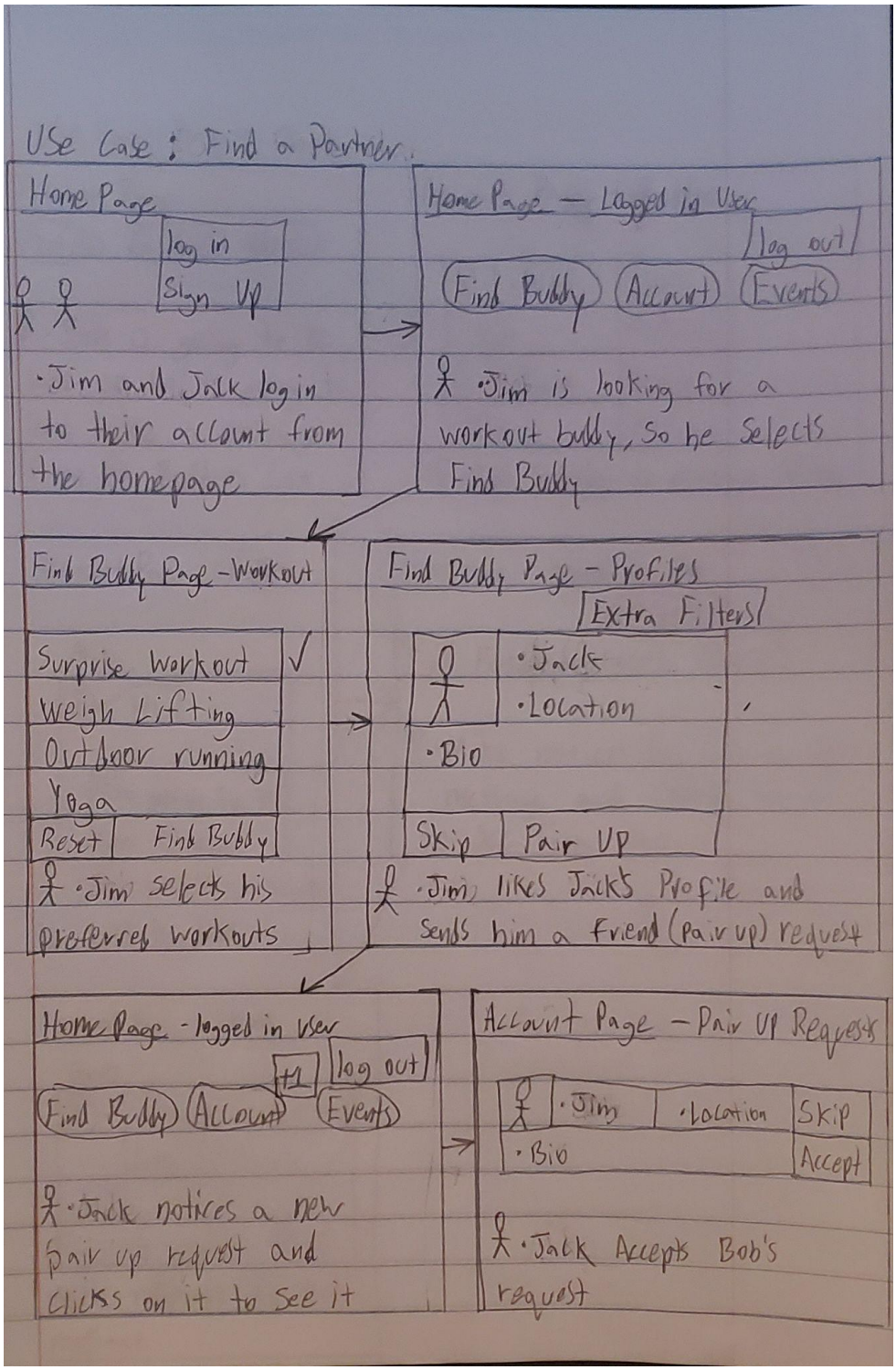
Use Case 1: Finding the Fithub Website



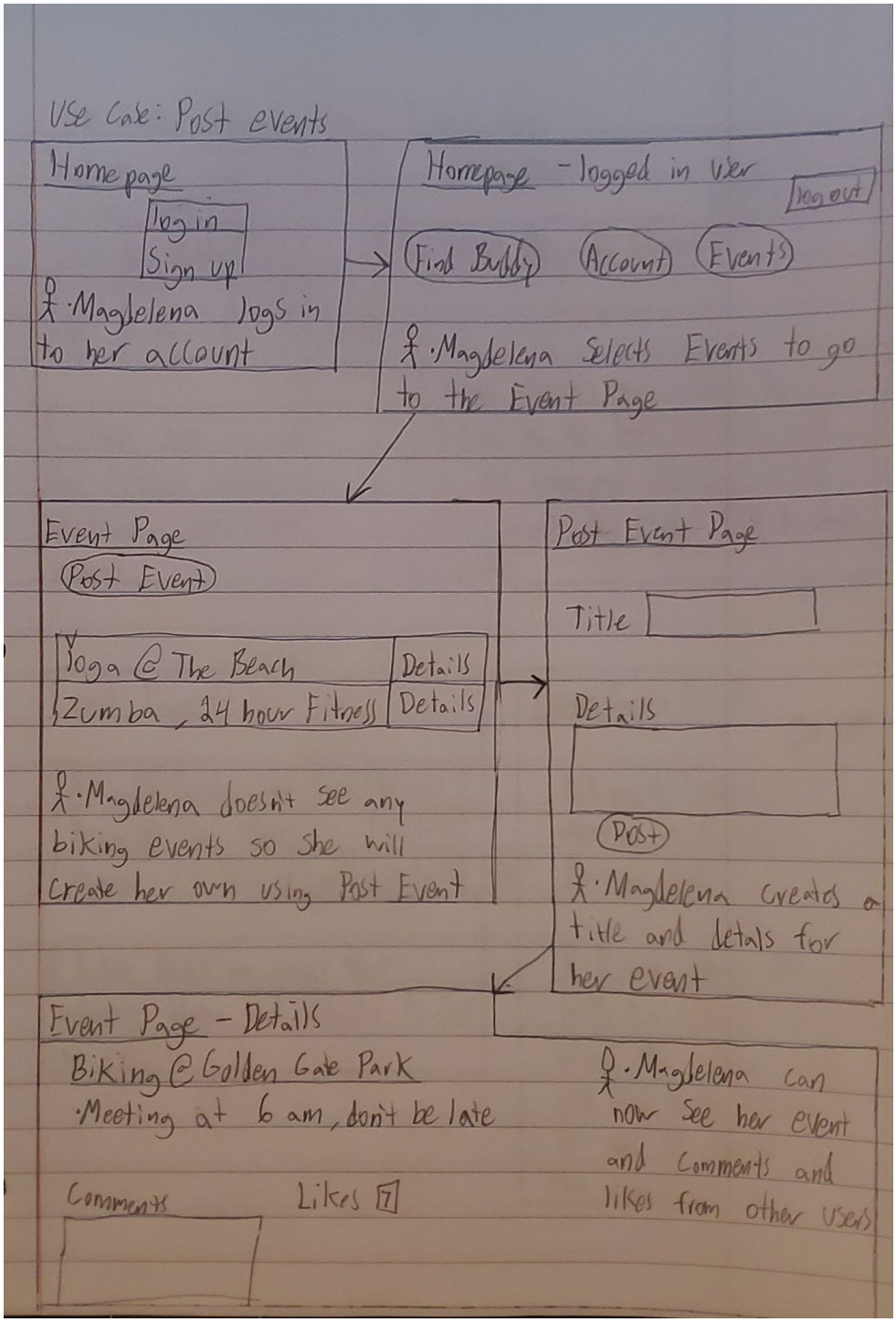
Use Case 2: Registration and Login



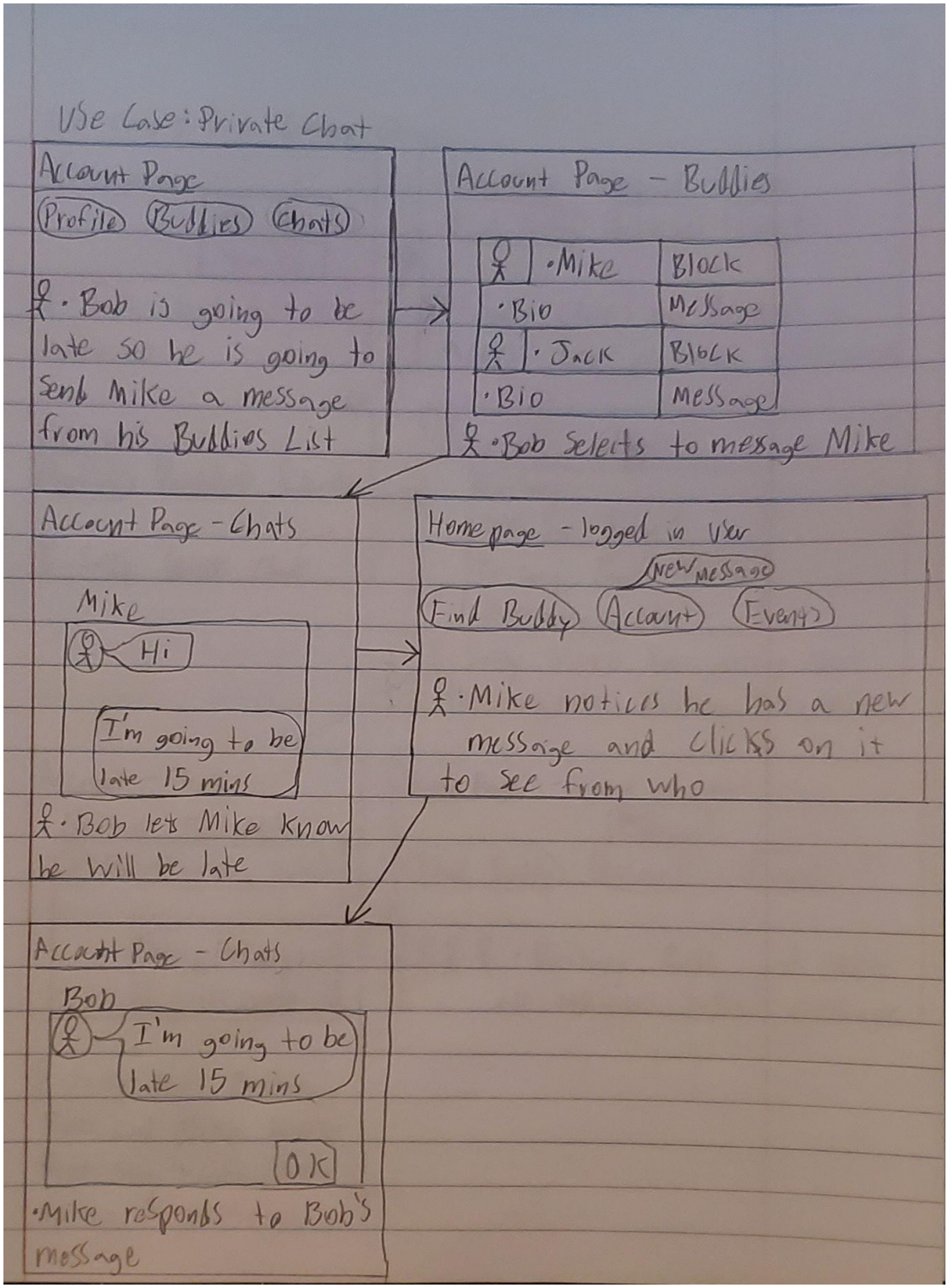
Use Case 3: Finding a Partner



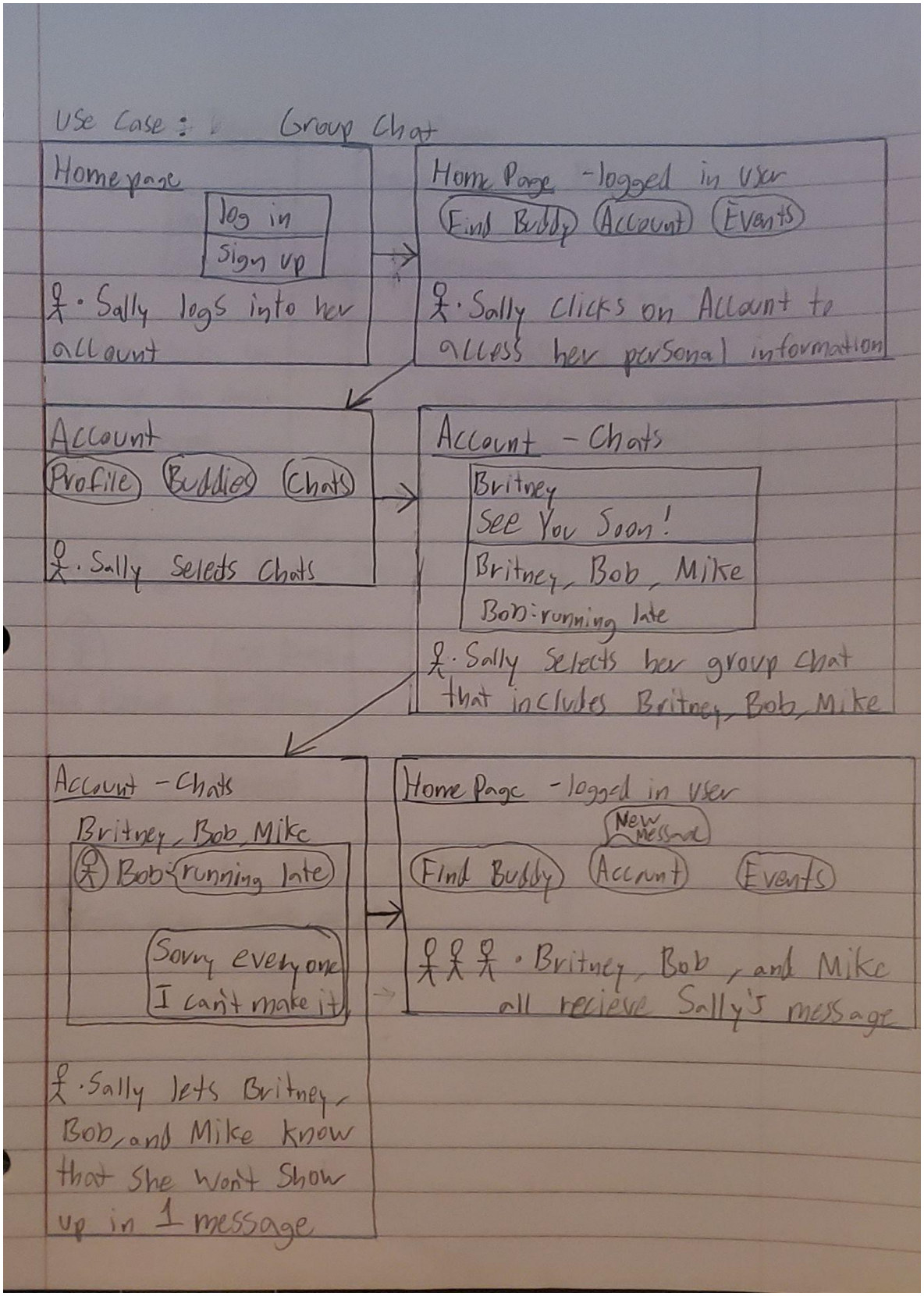
Use Case 4: Posting Events



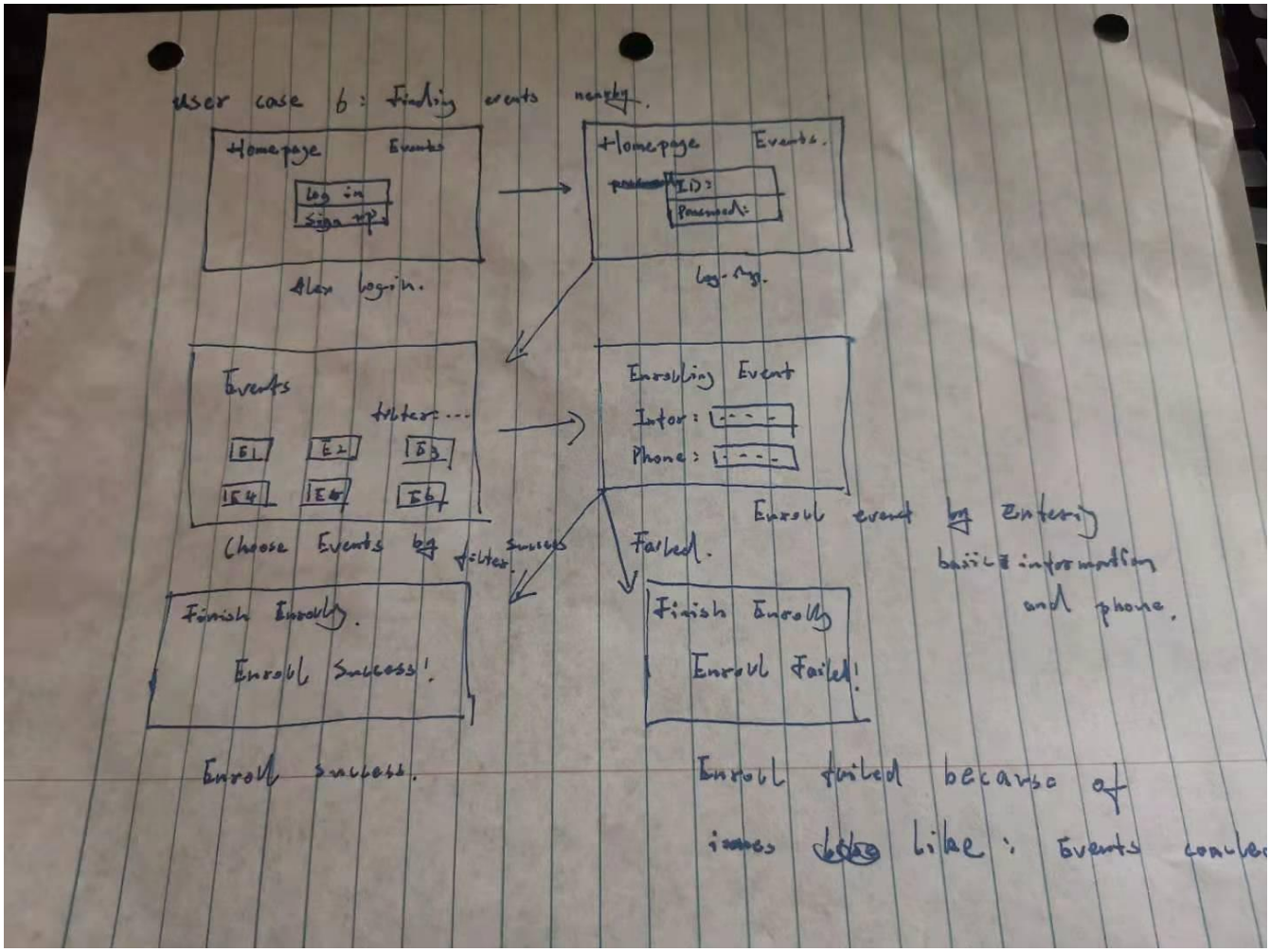
Use Case 5: Private Chat



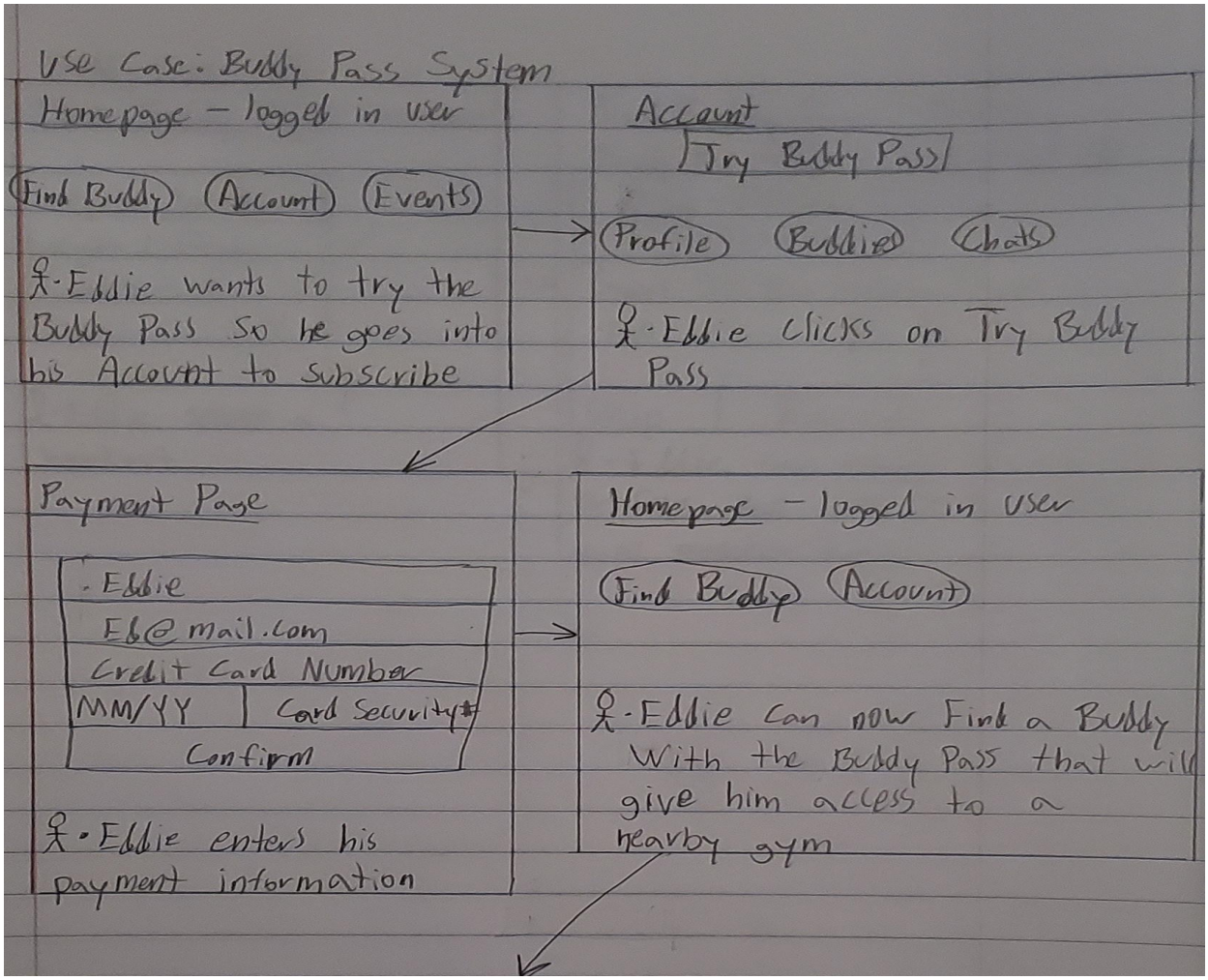
Use Case 6: Group Chat

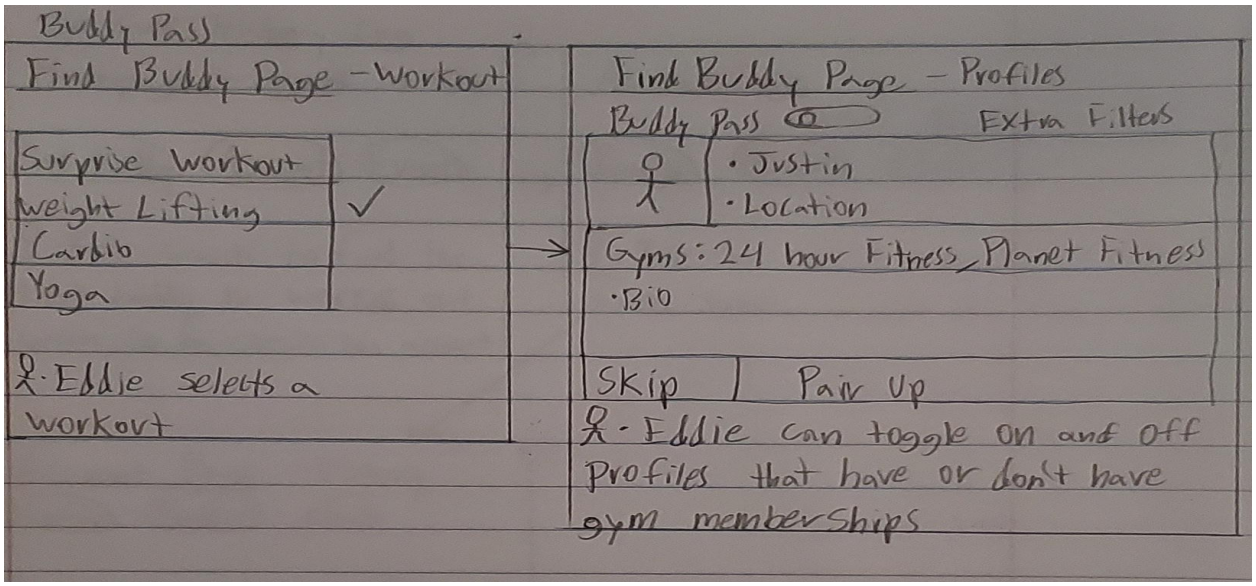


Use Case 7: Finding events

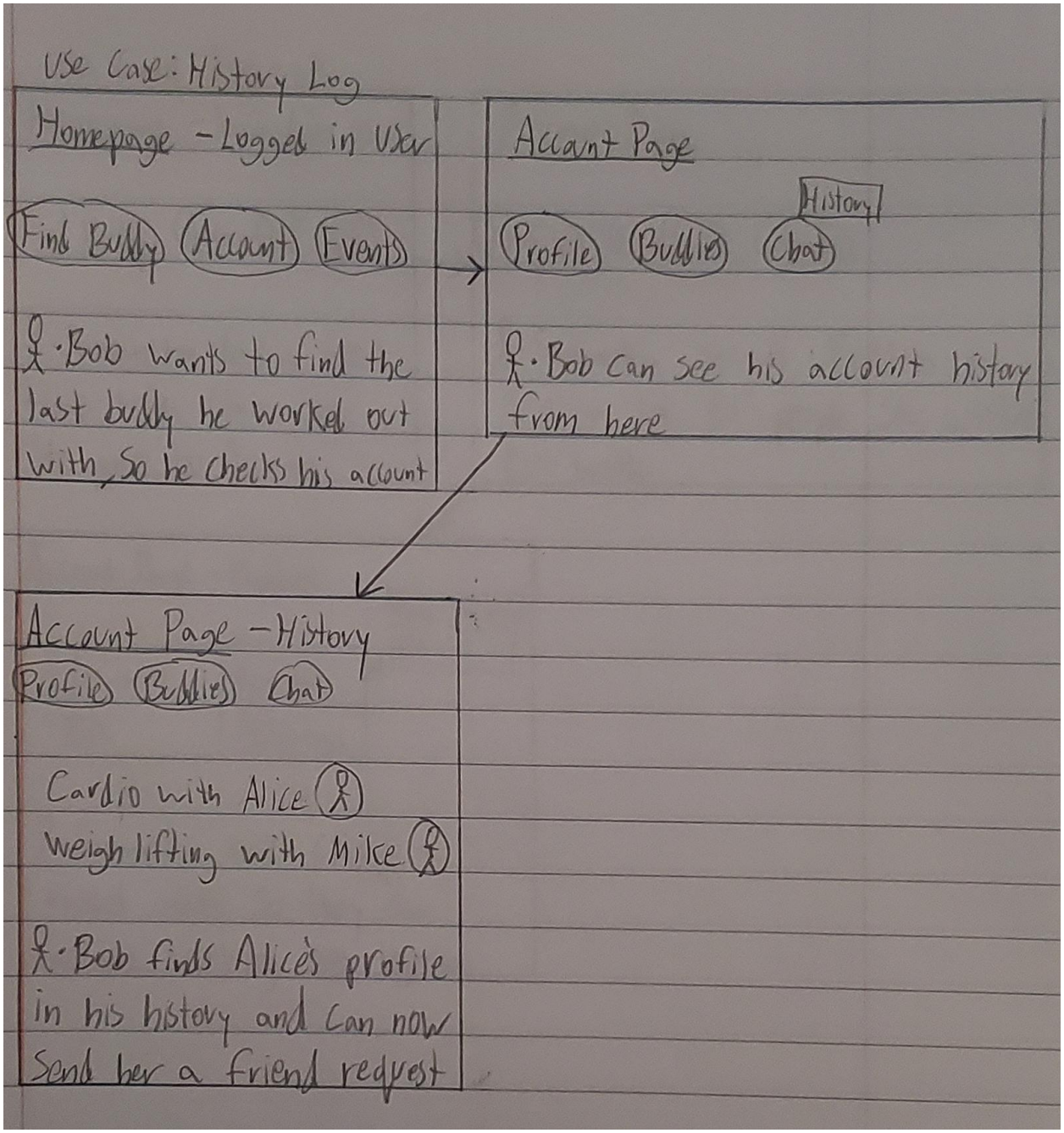


Use Case 8: Buddy Pass System

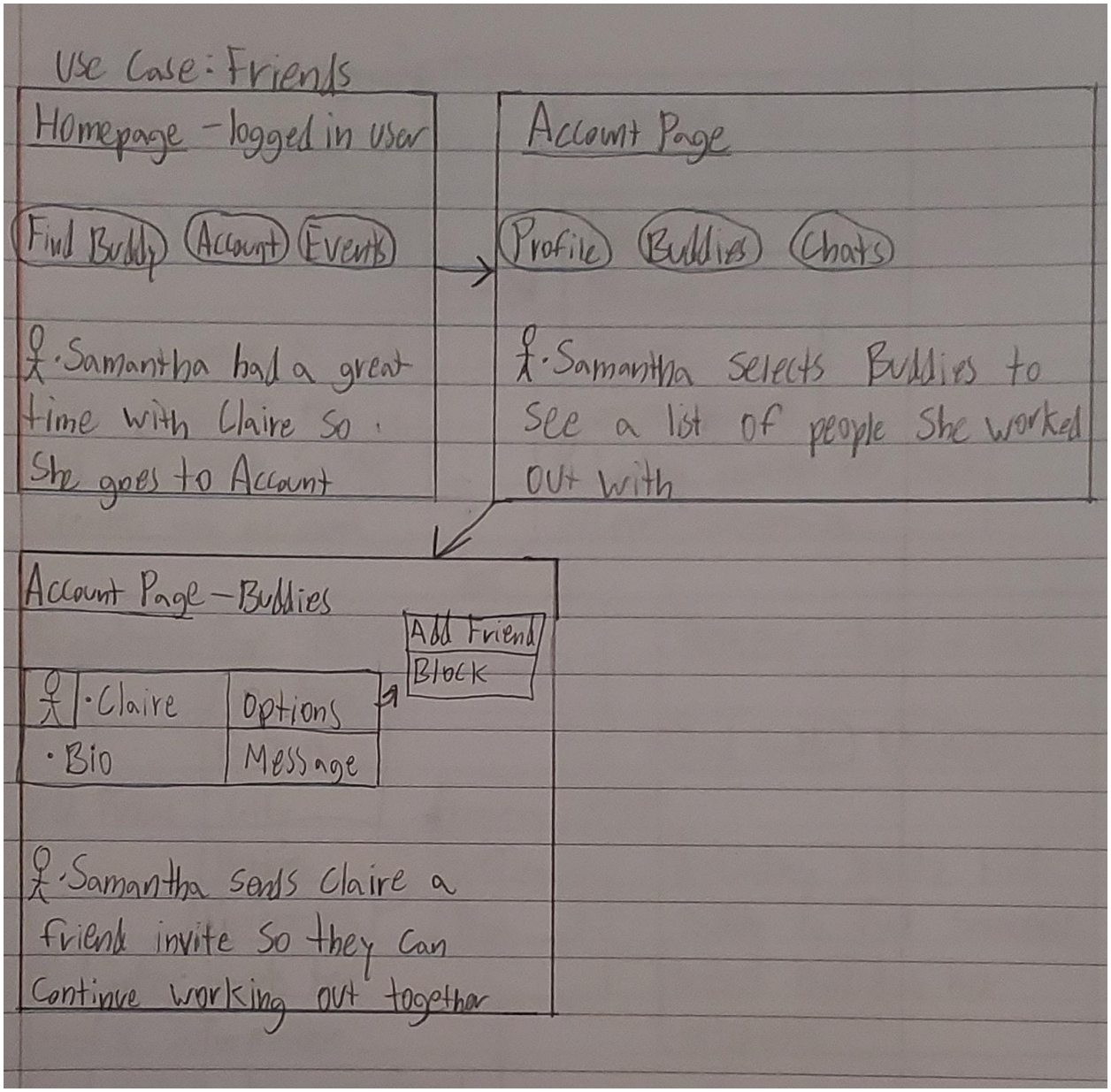




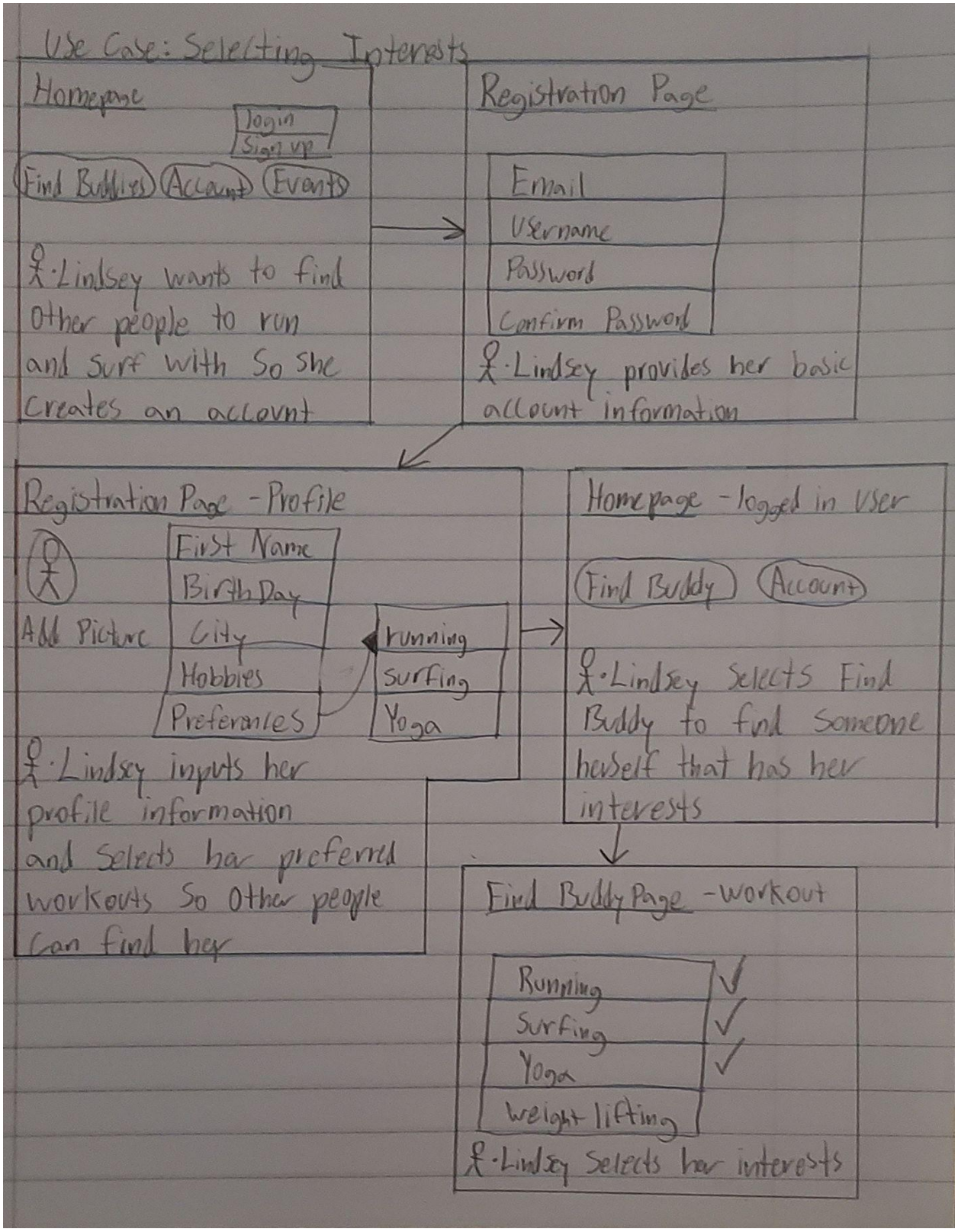
Use Case 9: History Log



Use Case 10: Friends



Use Case 11: Selecting Interests



4. High level database architecture and organization

- ***DB organization:***
  1. ***Business rules:***

a. Guest User:
   - A guest shall have one account
   - A guest shall be at most one registered user
   - A guest shall be able to search many Users
   - A guest shall be able to view many events

b. Registered User:
   - A registered user shall have one account
   - A registered user shall be able to search many Users
   - A registered user shall be able to create and delete many events
   - A registered user shall be able to send, cancel, accept and reject many workout requests
   - A registered user shall have many gym memberships
   - A registered user shall send, cancel, accept and reject many friend requests

c. Account:
   - An account shall be associated with a guest user or a registered user
   - An account shall have all the information of a user
   - An account shall be accessed by a single credentials combination only (username-password)

d. Events:
   - An event shall be created and deleted by a Registered User
   - An event shall be viewed by many Users

e. Clubs:
   - An club shall be created and deleted by a Registered User
   - An club shall be joined by many Registered Users

f. Gym Membership:
   - A gym membership shall belong to a registered user
   - A gym membership shall have option for atmost one buddy pass

g. History Logs:
   - A history log shall belong to a registered user
   - A history log shall have information related to user activities

h. Blocked Users:
   - A block list shall have many users blocked by other users
   - A block list shall be used by many registered users

i. Buddy pass:
   - A buddy pass belongs to a gym membership
   - A buddy pass shall be used by any registered user

j. Friend request
   - A friend request shall be sent by a registered user
   - A friend request shall be canceled by a registered user
   - A friend request shall be accepted by a registered user
   - A friend request shall be rejected by a registered user

k. Workout request
   - A workout request shall be sent by a registered user
   - A workout request shall be canceled by a registered user
   - A workout request shall be accepted by a registered user
   - A workout request shall be rejected by a registered user

2. Entities:

a. Guest User (Strong)

- user_id: unique user id to identify the guest user
- email_id: unique email id associated with a user
- is_registered: if the user is a registered user

b. Registered User (Weak)

- reg_id: unique user id to identify the registered user
- user_id: id associated with a guest user
- phone: contact no of the registered user
- address: address of the registered user
- location: city name of the registered user
- zipcode: zipcode of the registered user
- activity_type: like if the user is interested in indoor/outdoor activities
- workout_type: kind of workout user prefers to carry out

c. Account (Weak)

- acc_id: unique id for the account entity
- reg_id: id associated with a registered user
- username: used to log in account
- password: password to log in account

d. Events (Weak)

- event_id: unique id for the events entity
- reg_id: id associated with registered user
- description: more info about the event
- start_time: start time for the event
- end_time: end time for the event
- from_date: start date for the event
- to_date: end date for the event

e. Clubs (Weak)
   - club_id: unique id for the clubs entity
   - reg_id: id associated with a registered user
   - description: more information on the club
   - open_to_all: if other registered users can join without invite

f. Gym Membership (Weak)
   - gym_id: unique id for the gym membership info
   - reg_id: id associated with a registered user
   - gym_name: gym name the user has membership for
   - gym_loc: location of the gym
   - gym_zipcode: zipcode in which the gym is located
   - buddy_pass: does it have the option for buddy pass
   - share_pass: if the registered user is willing to share his buddy pass
   - membership_start_date: start date for gym membership
   - membership_expiry_date: end date for gym membership

g. History Logs (Weak)
   - log_id: unique id to identify the log of a registered user
   - reg_id: id associated with a registered user
   - activty_info: information on the user activity
   - is_workout_request: if log is for workout request
   - is_friend_request: if log is for friend request
   - is_buddy_pass_request: if log is for buddy pass request
   - is_block_user: if log is for blocking a user
   - is_club_request: if log is for a club request

h. Blocked Users (Weak)
   - block_id: unique id to for blocking entity
   - block_from_reg_id: id associated with a registered user who is blocking

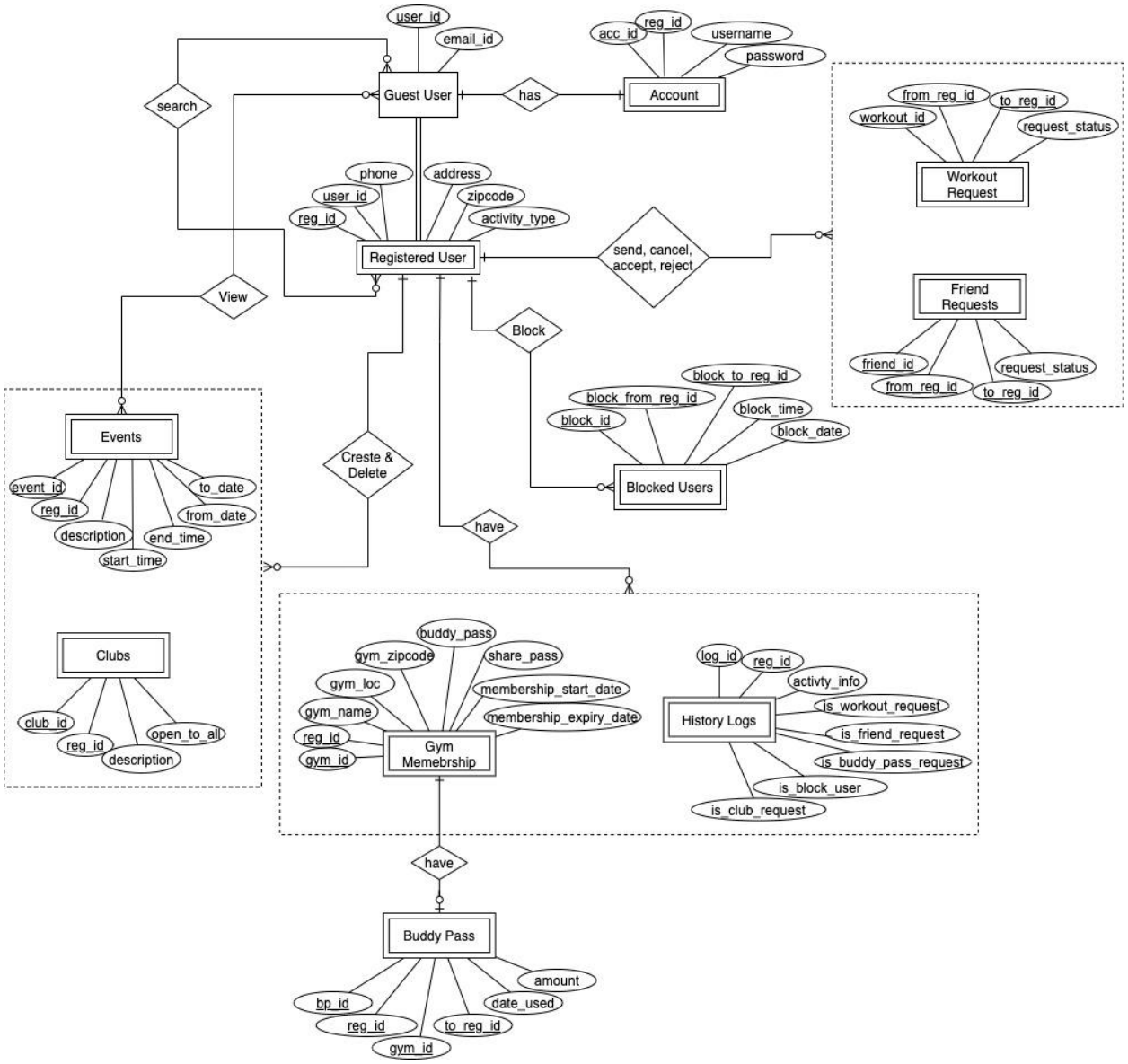
- block_to_reg_id: id of the registered user who is being blocked
- block_date: date the user was blocked
- block_time: time user was blocked

i. Buddy pass (Weak)
- bp_id: unique id for the buddy pass entity
- reg_id: id associated with a registered user
- gym_id: buddy pass belong to which gym
- to_reg_id: registered user id with whom the buddy pass is being shared
- date_used: the date on which the user is sharing buddy pass
- amount: amount user collects for the buddy pass mechanism

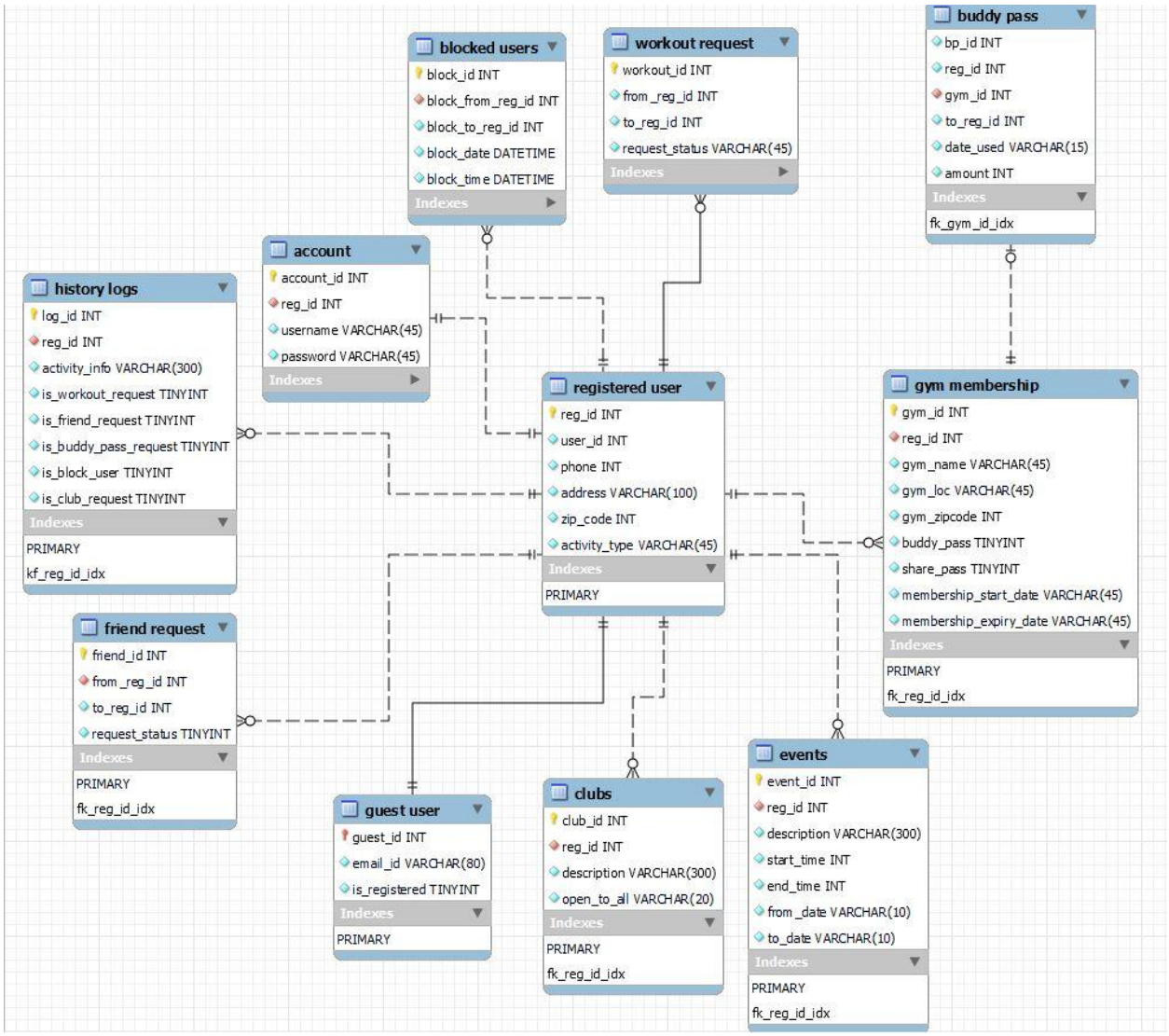
j. Friend request (Weak)
- friend_id: unique id for the friend entity
- from_reg_id: id of a registered user who is sending friend request
- to_reg_id: id of a registered user to whom the request is sent
- request_status: status of the request as sent, cancel, accept, reject

k. Workout request (Weak)
- workout_id: unique id for the workout entity
- from_reg_id: id of a registered user who is sending workout request
- to_reg_id: id of a registered user to whom the request is sent
- request_status: status of the request as sent, cancel, accept, reject

3. ERD:


user_id
email_id
reg_id
acc_id
username
password
Guest User
has
Account
search
from_reg_id
to_reg_id
workout_id
request_status
Workout Request
phone
address
user_id
zipcode
reg_id
activity_type
Registered User
send, cancel, accept, reject
Friend Requests
View
Block
friend_id
request_status
from_reg_id
to_reg_id
block_to_reg_id
block_from_reg_id
block_time
block_id
block_date
Events
Creste & Delete
Blocked Users
event_id
to_date
reg_id
from_date
description
end_time
start_time
have
buddy_pass
gym_zipcode
share_pass
log_id
reg_id
gym_loc
activty_info
membership_start_date
gym_name
History Logs
is_workout_request
membership_expiry_date
reg_id
Gym Memebrship
is_friend_request
gym_id
is_buddy_pass_request
Clubs
is_block_user
is_club_request
club_id
open_to_all
reg_id
description
have
Buddy Pass
amount
bp_id
date_used
reg_id
to_reg_id
gym_id

4. Database Model:



5. DBMS:

The database chosen to develop the project is MySQL since it is well known RDBMS, easy to use and many GUI tools are available for the development and maintenance of the MySQL database.

- ***Media storage:***
  - The media used in the FItHub app will be images of the profile pictures and club pictures.
  - They will be stored in the **file system** and the format of image will be JPEG/ JPG/ PNG.

- ***Search/filter architecture and implementation:***

  - Search Algorithm:
    - The input to the search algorithm shall be provided from the search bar.
    - The user will input only usernames to search from the search bar
    - The input will be looked into the database
    - The output for the search algorithm will be the list of usernames that fully / partially matches the user input in the ascending order

      The DB fields that will be searched here is :
    - Table: Account
      - → username

  - Filter Algorithm:
    - The input to the filter algorithm shall be provided by clicking on search button from the filters section
    - The user will be given the option to search based on the location, indoor/outdoor activities, workout type by selecting the appropriate filters.
    - These filtered inputs shall be looked into the database. These fields shall also be indexed in the database for quick search results.

- The output for the search algorithm will be the list of items that match the user's filter input. The information in each item fetched from the database will have the username, location, zip code, activity_type, workout_type.

  The db terms that will be searched here is :

- Table: Account
  - username
- Table: Registered User
  - location
  - zipcode
  - activity_type
  - workout_type

5. High Level APIs and Main Algorithms

a. API.js File

i. Register

1. Post request:
    a. When a user registers to our web application, we would receive a post request with the user's username, email, along with their password. We will check to see if the user is valid, which will then correspond with the database if it is successful. Else, it will throw an error.

ii. Login

1. Post request:
    a. When a user logs into our web application, we would receive a post request with the user's username and password. The backend will interact with the database to ensure that the information entered exists within our database. If the database has the correct info that was inputed, it will not throw an error.

iii. Logout

1. Post request
    a. When a user logs out of our web application, we would receive a post request. Session will be destroyed in the database resulting in them being logged out.

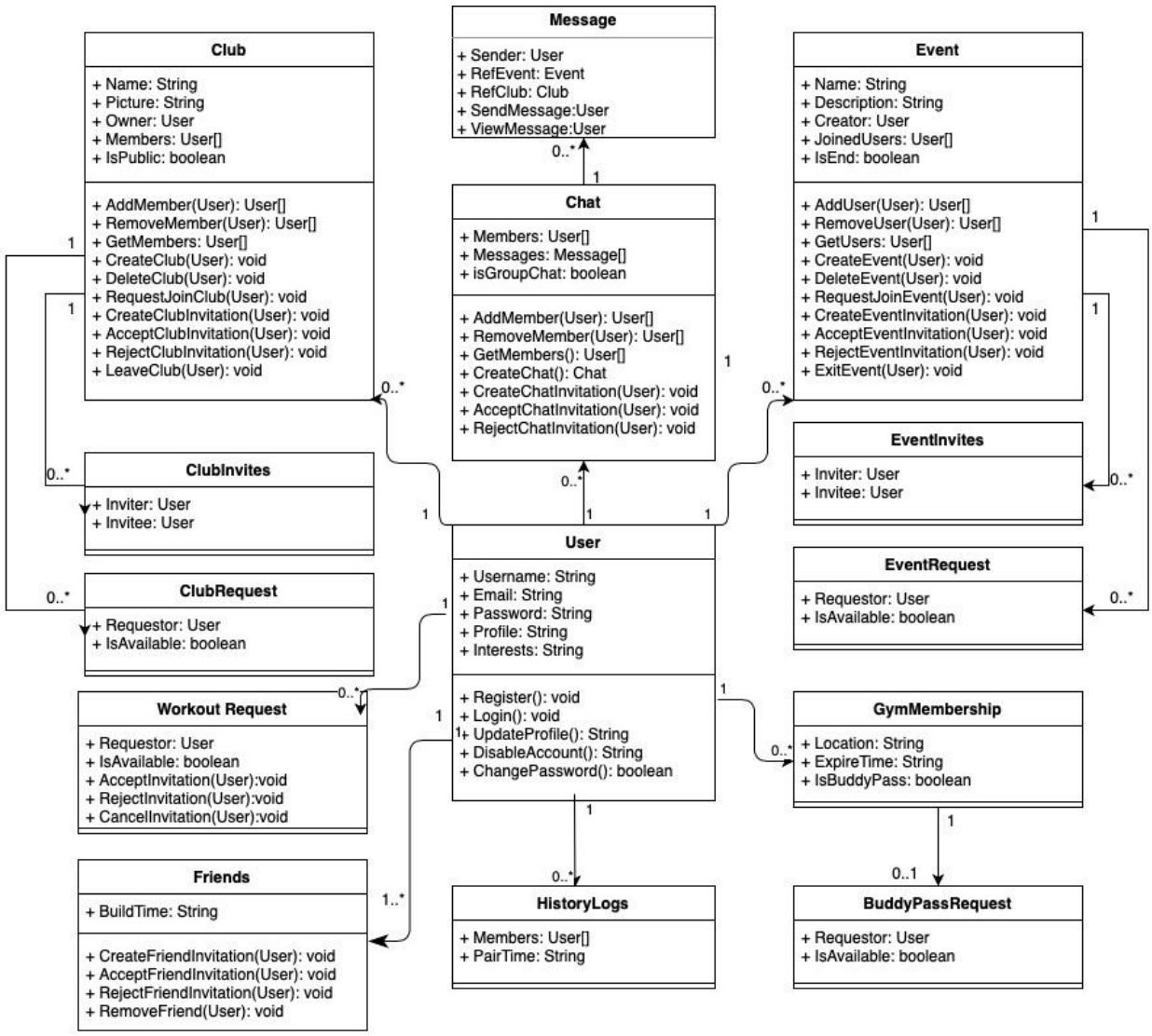
iv. Posting Events

1. Post request:
    a. The backend will receive a post request. It will communicate with the database and store the information.

v. Search by Filter

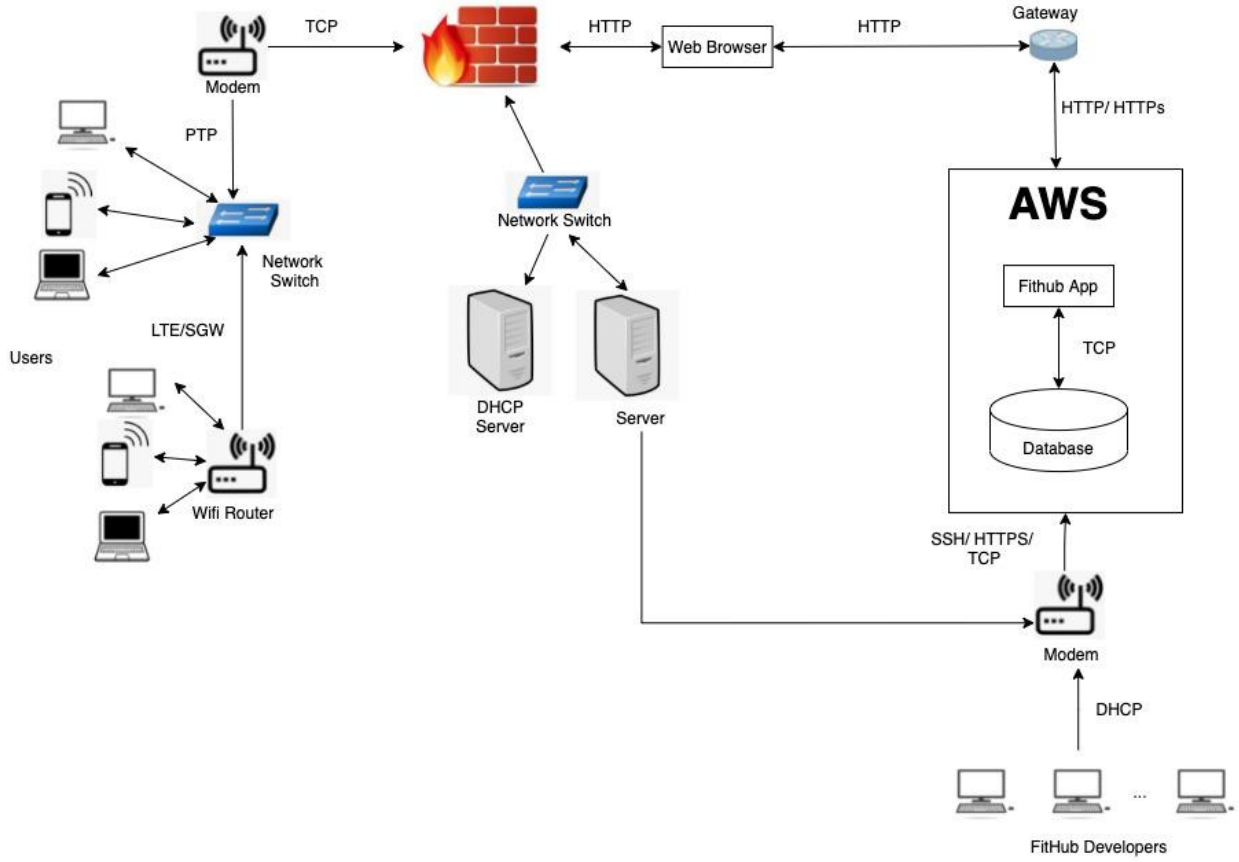
1. Get request
    a. When users search by filter, the backend will receive a Get request. If the filter is in the database, it will send data back. If it does not, it throws an error

6. High Level UML Diagrams

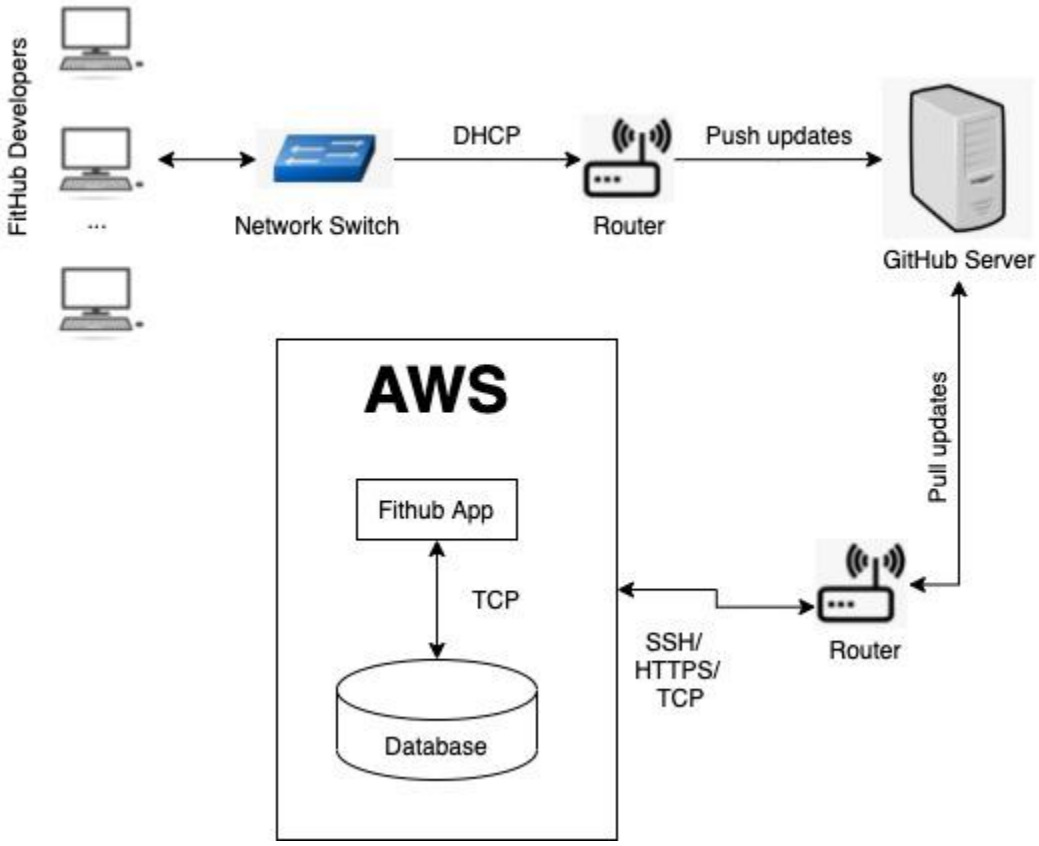


7. High Level Application Network and Deployment Diagrams

Application Network Diagram:



Deployment Diagram:



8. Identify actual key risks for your project at this time

1. No Legal/Content Risks

2. Skill risk: New Tech

   We are using frameworks/technologies that are new to some members. Those members will have to learn the frameworks. If they fall behind they will have a harder time contributing and understanding the project.

   ➔ Solution:

   We have already decided what frameworks we will be using so that everyone has time to freshen up on them. We are also splitting work in a way so that more experienced members tackle the more technical tasks and we will have many meetings and discussions to make sure everyone is on track.

3. Schedule risk: Time constraints

   We are working with tight deadlines and our goal as a team is to create an exceptional project. To achieve this goal we are meeting constantly throughout the week which has been difficult or stressful for some members.

   ➔ Solution:

   Our meetings now have a detailed schedule of topics that we need to cover so that we can get through them quickly. We will also be having less impromptu meetings and instead plan out meetings based on how we are doing, so that everyone has enough of a notice.

4. Teamwork risk: Team member contribution

   Some members are difficult to contact through email and discord. We also have a problem with participation and attendance in team meetings.

   ➔ Solution:

   We will look for alternative methods of contacting team members like getting cell phone numbers in case they aren't responding back within an acceptable time frame. We also keep track of meeting attendance and work contributed so we can talk with any team members that are falling behind.

5. Technical risk: Map Integration

   We are interested in having a map on the website that allows users to see the nearby gyms or the events that are happening. The simplest solution seems

to be to use the Google maps API, but that will require learning it and possibly paying for access.

➔ Solution:

We are learning more about how to integrate Google Maps and the various options and services they provide. We are also discussing alternatives such as linking to Google Maps search results or using a different service.

9. Project management

- ➔ For every milestone received, the tasks are created and assigned to the team members
- ➔ The team meets every monday to plan and divide the weekly tasks
- ➔ There is an internal deadline set each of the tasks
- ➔ There are team meetings conducted on frequent basis in the week in order to track the ongoing progress, resolve any queries, discuss the concepts taught in class and how to implement them in our project
- ➔ Also, the team members are available on discord for any instant meeting, updates, announcements, queries and to ensure everyone is up to date
- ➔ The team uses Github inorder to manage the development of code for the project. There are two branches created 1. Master and 2. Develop. The users take the pull of the develop branch, write down their code and then push the changes to develop branch. These changes are merged with the master branch by the github master and then the server instance is restarted.

- ➔ For task management purposes, we use Asana. It helps us to keep track of who is doing a task, and what is the progress of it.
- ➔ For team meetings, we use zoom and discord.

10. Detailed list of contributions (this section must be done by the team lead)

Team Member	Contribution
Vidhi Vora ***(Team Lead)***	*As a Team Lead:* ‣ Ensuring all the requirements are met and adhering to the M2 guidelines ‣ Assigning and supervising the task progress, organizing regular team meetings and the agenda ‣ Assigning internal deadlines, resolving querries, discussing the concepts for the M2 *As a Team Member (Milestone 2):* Full Contribution to following sections: ‣ Project management ‣ Detailed list of contributions Partial Contribution to following sections: ‣ Prioritized Functional Requirements ‣ High level database architecture and organization ‣ High Level Application Network and Deployment Diagrams *Vertical Prototype:* ‣ Setting up the node express (api.js) ‣ Connecting to the database
Johnson Nguyen ***(Backend Lead)***	*Milestone 2:* Full Contribution to following sections: ‣ High Level APIs and Main Algorithms Partial Contribution to following sections: ‣ - *Vertical Prototype:* ‣ Developing and inserting data into the database for the registration page
Roberto Simental ***(Frontend Lead)***	*Milestone 2:*

	Full Contribution to following sections: ‣ UI Mockups and Storyboards ‣ Identify actual key risks for your project at this Partial Contribution to following sections: ‣ - *Vertical Prototype:* ‣ -
Michael Satumba ***(Frontend Member)***	*Milestone 2:* Full Contribution to following sections: ‣ - Partial Contribution to following sections: ‣ Data definitions ‣ UI Mockups and Storyboards *Vertical Prototype:* ‣ Developing the home page and the registration page
Eduardo Hernandez ***(Backend Member)***	*Milestone 2:* Full Contribution to following sections: ‣ - Partial Contribution to following sections: ‣ Data definitions ‣ High level database architecture and organization *Vertical Prototype:* ‣ Developing and inserting data into the database for the registration page
Zhinan Zhao ***(Backend Member)***	*Milestone 2:* Full Contribution to following sections: ‣ High Level UML Diagrams Partial Contribution to following sections: ‣ High Level Application Network and Deployment Diagrams *Vertical Prototype:*

	‣ -
Ziming Wang ***(Frontend Member)***	*Milestone 2:* Partial Contribution to following sections: ‣ Prioritized Functional Requirements *Vertical Prototype:* ‣ -