Questions for tony:

1 We want to allow the user to de-select the activity buttons when choosing “high” “medium” ”low” etc. Looking for guidance as to how to do that.

2 Also allow the user to add other activities in activity modals and save.

3 How do we add icons (to change skiier image) to the database and allow user to change and update their icon?

4 How do we make the activities/activity level the user selects show up on their profile page? Do we need to write a function?

5 We also need to work on schedule feature, allow user to edit and save schedule.

6 Sign up verification. (make sure there are no duplicates in database), (make sure passwords match)

Plans for tomorrow:

- Add external SQL database for Natasha!

-Discuss questions above.

**I come up with some new plans for the following week after Friday meeting.**

Now, I think it is better to implement some basic requirement and test for that rather than design much more requirements. So we can do like this:

1 list a requirement table to tell what to implement, and what to test, Also we should talk with Renata about what website will be like at the end of March

2 Natasha can do more on front end, verification and testing. Matthew can focus on back end. For me I will set up the environment on Amazon Server

3 There is setting below is for Matthew that how to set up Wifi by computer

Re1: the color for selected button, another color for de-selected button

Re2: do as you say is ok

Re3: if the picture is not uploaded by user, it not necessary to store that in database,

Just like what you did in front end development. If uploaded by user, store the filename, fileType(jpg,png…),fileUrl(c:/…) at least

Re4: the function is sportService.findSportLevelByUserId()

Re5: we’d better to basic function like searching function first. Suppose that if you are user, maybe you will prefer to search what you like, then add it into schedule

Re6: verification have two part, one for front end verification by Javascript, the other for back end verification by java, you can start from front end

## Service structure

Service is for business logic, DAO is for SQL

\*Service.java is interface, \*ServiceImpl.java is to implement this interface

\*Repository.java is interface, \* Repository.xml is to implement this interface

\*Query.java is interface, \* Query.xml is to implement this interface

Criteria is the paramater object for DAO function

Entity is the return object for DAO function

\*Respository.xml contains very simple SQL, it is better to:

Findone(id) select \* from \* where id #{id}

FindAll() select \* from \*

FindByCriteria(criteria) select \* from \* where criteria1=#{ criteria1} and ….

Insert(object)

Update(object)

Delete(object)

\*Query.xml contains complex SQL and you can write yourself if need

findSportByUserId() select …………………

Critera.java/

Other parameter

\*\*\*Service.java(\*\*\*ServiceImpl.java)

\*\*\*Repository.java(\*\*\*Repository.xml)

\*\*\*Query.java(\*\*\*Query.xml)

Entity.java

\*Service is in package com.adnature.\*.service

\*ServiceImpl is in package com.adnature.\*.service.impl

\*Repository.java is in package java->com.adnature.domain\*.repository

\*Repository.xml is in package resource->com.adnature.domain\*.repository

\*Query.java is in package java->com.adnature.domain\*.query

\*Query.xml is in package resource->com.adnature.domain\*.query

criteria is in package com.adnature.domain\*.criteria

entity is in package com.adnature.domain\*.entity

## Demo:

#### Service interface

public List<String> findSportLevelByUserId(String userId);

#### Service impl:

**public** List<String> findSportLevelByUserId(String userId){

List<String> level =**new** ArrayList<String>();

List<Sport> sports= sportQuery.findSportByUserId(userId);

**for**(Sport sport:sports){

**if**(level.contains(sport.getFitnessLevel())){

level.add(sport.getFitnessLevel());

}

}

**return** level;

}

Parameter whatever you want

Return whatever you want

Just call SQL by sportQuery.findSportByUserId(userId)

#### DAO interface:

List<**Sport**> findSportByUserId(@Param("**userId**") **String** userId);

#### DAO impl:

<select id=*"findSportByUserId"* parameterType=***"String"*** resultType=***"Sport"***>

SELECT

The same name as entity so that they can mapped to variable in entity,

Here is capital word

S.ID AS **ID,**

S.NAME AS **NAME,**

S.FITNESS\_LEVEL AS FITNESS\_LEVEL

FROM SPORT S , SPORT\_USER SU

<where>

AND S.ID = SU.SPORT\_ID

<if test=*"userId!=null"*>

AND SU.USER\_ID = **#{userId}**

</if>

</where>

</select>

ParameterType is related to interface Parameter type, same Parameter name

ResultType is related to interface return type

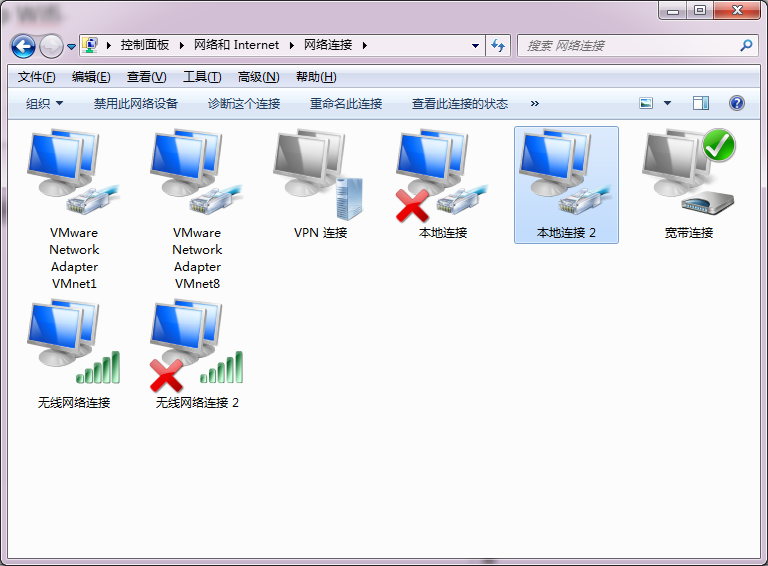
#### Setup Wifi

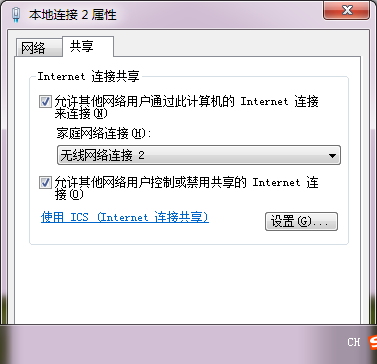
Firstly, if Natasha can visit the page by “Matthew’s IP:8080/index.htm”, Matthew do not need to do what I say below, if it not working Matthew should setup a Wifi by your computer for Matthew to visit

1 The command for creating a new wifi:

netsh wlan set hostednetwork mode=allow ssid=Tony2 key=TonyZhang1991

2





3 The command for starting a the wifi:

netsh wlan start hostednetwork

4 The command for find a the ip on your computer, for example 192.168.137.1

ipconfig

5 other can visit your page by 192.168.137.1:8080/index.htm