Project Design

Designing Aspects the Project

Team Member :

Li Pei : lip@andrew , Tongyun Lu : tongyunl@andrew , Weiting Zhai : wzhai@andrew

1. Designing Screens

Our application supports **two resolutions**: **540 * 960 pixels** (Samsung Galaxy S4 zoom with 4.3 inches screen) and **1200 * 1920 pixels** (Google Nexus 7 with 7 inches screen). All the screens of this application can be viewed in portrait or landscape orientation without distortion. And all android resources are identified by their id in Java source code, including strings, pictures, styles and other resources with id number. These could be checked by viewing xml files. And all activities' .xml file are named in the form of "PackageName_ActivityName.xml".

2. Designing Presentation Tier

All activities can be reached by **intent** for this unit. Each activity has the java file in package ui and corresponding xml file in layout folder. The statements have been setup to read and write values from screen. User can register, login, edit user information, choose activity and nutrition by entering text or choose item from spinner. All buttons can be clicked and make corresponding action.

3. Designing Content Provider

In this application , a **content provider** is needed to help implementing the web service below

1. Store users information for the use of the other users.
2. Update user's information for the use of the other users.
3. Read other user's information for some activities' use.

To complete this mission, we need to design a user database and an intermediate server to handle all this request from client android device and work as a bridge between client and database.

First, we start with a **MySQL database** design for content store and provider. This MySQL database is deployed in a PC and data stored in it is not transient, which is not working like SQLite database on android device.

Then we will see what to store for designing the schema of database. For our project, the core storage is the user's information, which contains username, password, weight ,height,rival,goal and current consumption,etc. For all this information, only username is unique. Furthermore, username and password cannot be null. We considered the normalization and extensibility of database, and design a one table database TEAMUPFATDOWN, with one table USERS with several rows in it. One user is one row in the database, the primary key is user ID and the username and password is set to NOT NULL. This table is extensible and easily manipulated because you could add more information for user just to add one column to the table. The UML notation for database is shown like :

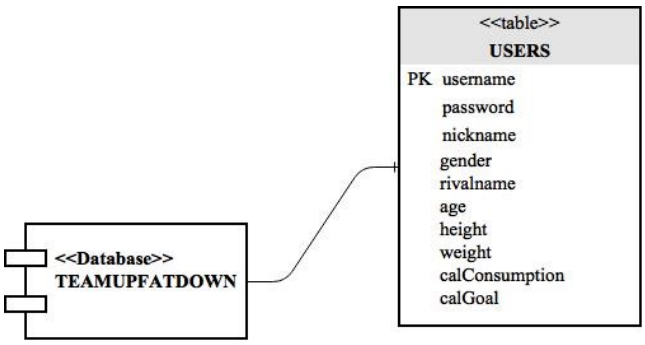

Image 3.1 Database Schema

After designing the database, we have to consider how to manipulate it with **CRUD** method. Start with the simplest operation, **DELETE**. **DELETE** is operated by one line query to delete a row in table according to username, but in our application, delete method is not exposed because we do not provide delete user selection for android users.

The delete method is used when we actually want to **CREATE** or **UPDATE** a person in database. In our app, the only activity to **CREATE** a person is when you register, but when you update, we also treat you as a new user, but with same username and password as yourself in the database. With these design of project, we combine **CREATE** and **UPDATE** together as one approach, in which you **DELETE** a user with username in database and insert a new row in database. So for **CREATE**, there is no user in database with the username, everything is handled by server exception and then user is insert to database. For **UPDATE**, there will be a user in database with same username, we delete it first and insert the updated user into database.

For **READ**, we have to consider getting one row and getting all rows in the database scenarios. When you want to read a user's information, you only have to read one row from database, but when you are working on the TEAM activity, you have to get all the rows from database. There will be exception in this process, like when you want to get one user, the person may not be existing.

After analyze the **CRUD** method for database, we could deploy a java class to handle the communication between server and database. We designed three methods for database manipulation class, DatabaseIO. First is to add one to database, which work as described before, delete one row in database according to username and then insert a new row in database. Second is get one row from database table and third is get all rows in the database table.

After designing the database access method, we could build the server to handle request from server and bridge database and client. In this project, due to the fact that communication information from database and server are simple information, we use socket with input and output stream to handle communication. The server socket worked iteratively and handle the session in one thread for each client request, and kill the thread after handling one request. There are three method for a server socket communication handler, open connection, close connection and handle session. For client end, there will be a corresponding thread running client. For each request from user, it start a new thread and communicate with server to handle the session.

Then the last design is about client end user type. User object in our project have lots of attribute fields and method, but only part of it is used for communication, so in this project we design a DatabasePerson class, which is a brief version of Person with limited scope of information focused on communication. When we want to make communication, we transform user into DatabasePerson, and after communication we rebuild user from database person, which simplify the communication process and hide some user information for client to server.

To implement encapsulation, this DatabasePerson is totally hide from user. It is only used in the proxyClient class, and transform from and to user class Person when you want to communicate with server.

The total design diagram is shown below

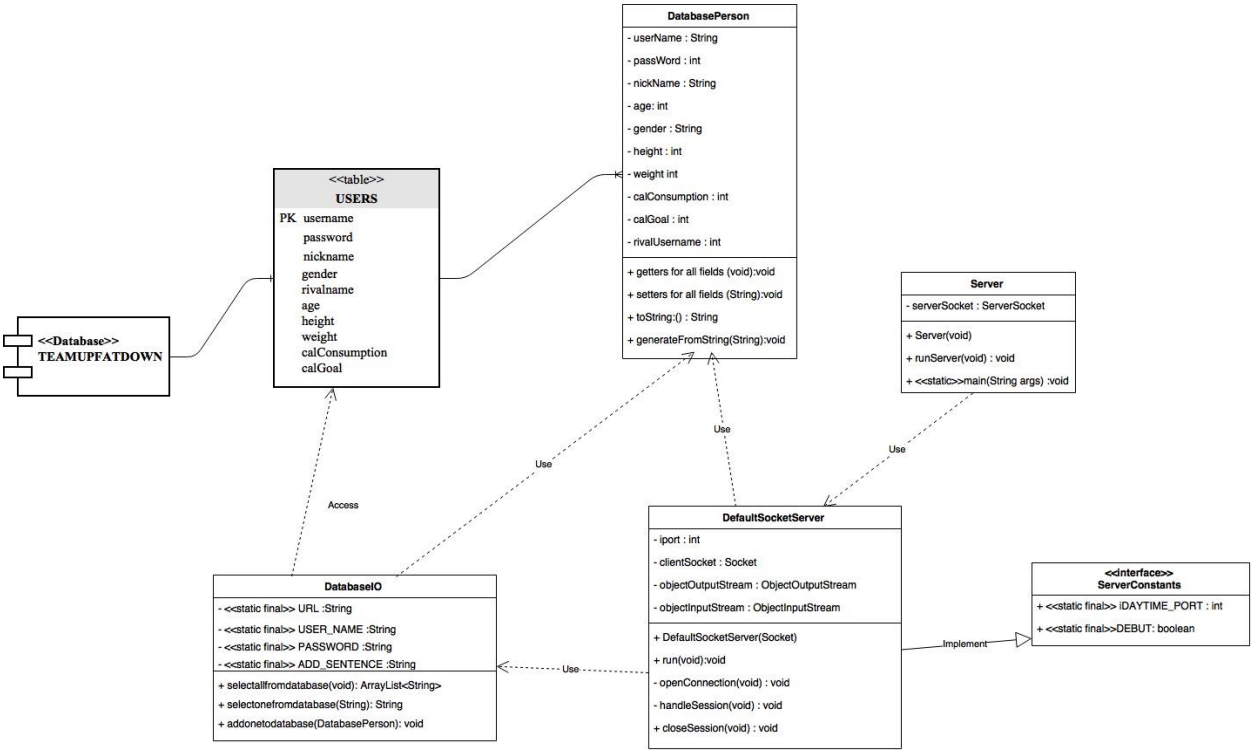

Image 3.2 Server End Design UML

4. Designing Application Tier

We create an abstract class Activity to represent all activities which user may do everyday, and make some concrete classes to extend the Activity abstract class, such as Swimming, Soccer and Basketball. Each kind of activity contains name, duration or distance which could be used to calculate calories consumption. An abstract class Nutrition is created to represent all nutritions user may ingest everyday, and several classes would extend it. Each kind of nutrition contains name, weight which could be used to get intake of calories.

In our application, all users of this application would be considered as a team. We create a class Person to represent the user. In this class, there are username, password, some personal information, arraylist of activity, arraylist of nutrition, array list of friend and rival which the type is Friend. The Friend Interface is used to encapsulate the Friend class when user want to access information about other users. All other users information can be retrieved from database, but the user can only get part of information by the protect of Friend interface, which achieve the design goal: information hiding.

A local SQLite database is also implemented for saving some information of user. It used for the change page and store the information of user when it want to change, the time user update the photo and the path for store it. It helps when you want reenter the app at change page to show you the information and page set last time.

5. Implementing Service for Project

In our project, we implement several services for user.

First is a remote service described in the Content Provider section, in which we designed a backend database to save information for client and a server to handle request and transfer data between database and client. For some activities in the project, the remote service work background to provide content support your social action, like “duel” and write message.

Second is a local music playing service. When you enter main page and menu page, there will be background music, you could choose to stop them by gesture reminded by toast , but if you do not stop them, it will playing music continuously to provide you with a more music for training or exercise.

Third is a web mail service. When you want to write a message to your friend, there will pop up a email service for writing email to the user you choose, which is handy for your social intent.

6. Exception Handling

Activity exception handling is important to handle runtime issues in project. In our project, main cause of exception is about user input. Here is all the exception issues we considered in our project for current design.

In user page, if user input username not exist, or username did not match the password, there will be exception to handle the issue and tell user about the issue.

In the registration page, if you input the username already exist in database, your username is not legal as email address, your password is not long enough, or your re-entered password is not matching your firstly input password, an exception will throw out and user could read the exception in a toast.

In the Activity page, you could choose your activity through a spinner, and input the duration time or distance for you training. The input type has limited that you could input integer only, and there is a handler to make sure your input is in a rational range.

Similar in Nutrition page, which you select take-in nutrition in a spinner, but input the quantity or calorie. The range and type of input is also handled in a way that limit the input.

In the edit user page, you will input your nickname, age, weight, height and calorie goal. Exception will occur when you input something illegal. Like weight and height for impossible number. That will be handled and a toast will notify the user.