Monash University

FIT5046 – Mobile and Distributed Computing Systems

Practical Assignment Phase 2 (Client-Side) (30%)

Due Week 10, Friday 3pm, (17th of May)

An Overview:

- The assignment is **an individual assignment**.
- The marks allocated for all the tasks in this assignment will add up to 100 marks and it is **worth 30%** of the total mark for this Unit.

Due Dates & Demonstrations/Interviews:

- The submission is due on **Friday Week 10 (17th May) by 3pm.**
- **Demos/interviews** will be held in the Tutorials and Consultation times **in Week 11 and 12** (you should book a time with your tutor)

Objectives and learning outcomes:

- To identify, develop and apply different approaches and methods for building distributed and mobile computing systems;
- To consider, evaluate several models and approaches and select suitable mobile computing solution (Android) to this particular case;
- To propose and develop a mobile distributed system (including both server and client sides) for this problem domain.

Calorie Tracker Application

The practical assignments (server-side and client-side) aim towards building a personalised fitness application that will keep track of what you eat, your daily calorie intake and calories burned, and provide you with useful information and reports. It will also allow you to set goals and inform you every day if you have met your goals. The final application will retrieve the information from public web APIs and the RESTful web service that you will create in Assignment 1.

Practical Assignment (client-side) will involve creating an android application that will interact with public web APIs and the REST web service (created in Assignment 1) to query and retrieve information. The application will receive user input, query web services, retrieve and process data, and create useful reports (different types of graphs and charts) about the user's burned and consumed calories.

The assignment MUST be implemented **in Android Studio**.

Task 1 – Invoking public web APIs (15):

You will invoke and consume two public APIs:

- **Google Custom Search** to get an image of the food item and useful information
- The **FatSecret** APIs or **National Nutrient Database** APIs can be used to provide calorie and nutrition information for food items.

  The APIs' web sites:
  http://ndb.nal.usda.gov/ndb/api/doc
  http://platform.fatsecret.com/api/

You need first to sign in and then make yourself familiar with the APIs mentioned above by reading their online documentation to be able to get a key and use them (find out about the parameters you need to pass and the response you will receive and how to parse it to get the right information). This is part of this task.

You will use the FatSecret or NDB API for making queries about the new food items that does not exist in your database, created in Phase 1.

Google

a) You will make queries to Google APIs to retrieve useful information about a food item (the new food items entered by the user). In your query, you need to make the best use of Google API by using right parameters in the query and adding related sites in the custom search engine. The returned results should be filtered by removing any irrelevant information and parts, hyperlinks and characters like '…'. **(5 marks)**

b) You will also use the Google search to find and display at least one relevant image of the food item. **(3 marks)**

National Nutrient Database or FatSecret APIs

c) You will make queries from the FatSecret or National Nutrient Database APIs using a keyword (a food item name). You need to use right parameters in your query to improve the results and then parse the results properly to retrieve the useful parts such as calorie and fat information plus a few other interesting facts. The details about the screens are provided under Task 6. **(7 marks)**

Task 2 Android Client of RESTful WS (5 marks):

a) The android application client will connect to the server and consume the RESTful web service created in Phase1. You need to make http connections using HttpURLConnection to invoke all the methods required for completing tasks in this assignment. **(2.5 mark)**

b) Accessing data and executing all the queries from the server side (and the web service) should be achieved using the AsyncTask approach. **(2.5 marks)**

Task 3 Android SQLite Database and Services (10 marks):

In this application, in addition to the data that is stored on the server side and can be queried through the web service, you will also store data about the user locally on the mobile phone in one table:

a) The daily steps table will store the steps entered by the user in a day at different times**.** This table will be queried later to provide a total number of steps per day. **(2 marks)**

b) Your code logic should make sure at the end of the day (11:59 pm), the following data is written (added) to the backend database (to the report table) of RESTful web services (by calling the POST method). You will use Android Services to achieve this. The POST method will add a new record to the report table at the end of the day with the following information:

- The user calorie set goal (this will be entered daily by the user in the home screen)
- the total calories consumed for that day (based on REST calculation methods),
- the total number of steps (based on the daily steps table's data)
- the total calories burned for that day (based on REST calculation methods)

(4 marks)

c) Then you need to delete all the existing data entries in the SQLite table. This means the table always stores the data for the current day. **(2 marks)**

Here, we assume the app works 24/7 without any disconnection.

During your interview, since we cannot wait for 11:59pm to test this part, you need to add a small button to one screen of the app (any screen) so when the button is clicked, the existing data for the current date will be written to the REST database, and then clear the SQLite table. Using this button will also allow you to test your code. **(2 marks)**

Task 4 Login and Sign up Screen (10 marks):

a) The login screen enables existing users to login, and allows the new user to sign up (Use only ONE screen here). The login screen will require a username and a password. This data will be verified with the data stored in the server-side database during the login. You also need to follow the password design guidelines (refer to Lecture 7). **(2 marks)**

b) The passwords should not be sent in the clear text to the server-side database. First, you need to hash the password on the client side using the right libraries in Java. **(2 marks)**

c) The Sign up form will enable the new user to enter their information (i.e. first name, surname, email, DoB, height, weight, gender, address, postcode, level of activity (1 to 5), and steps per mile, user name and password). In this screen, use a radio button for the gender, and use spinners for the level of activity (and any other data like postcode). For DoB, use a date picker (a calendar widget to pick a date). Data entry validation and error messages should be implemented where necessary. You need to make sure that the user name and email do not exist already, and ask the user to re-enter it if there is a duplicate. Your form should follow the mobile form guidelines (refer to Lecture 7). The details of the new user should be added to the user table at the server side database using the POST method. **(6 marks)**

Task 5 Home Screen - Dashboard (10 marks):

a) The home screen – this page should display "Calorie Tracker" as the app title, a picture related to fitness, and the current date and time. The main page should also welcome the user by their first name.

(2 mark)

b) This screen enables the user to enter a calorie goal for that day (facilitate data entry and avoid incorrect data entry by using an appropriate UI input control). If the user has already set the goal, this screen should allow the user to edit the goal. **(2 marks)**

c) The main page will use navigation drawer and fragments to navigate to other screens. **(6 marks)**

Task 6 My Daily Diet Screen (10 marks):

a) My Daily Diet screen will have two lists. The first list will allow the user to select a food category such as Drink, Meal, Meat, Snack, Bread, Cake, Fruit, Vegies, and Other. The second list will be populated automatically, from the food table in the server-side database by invoking REST methods, with the food items which are under the selected category. **(5 marks)**

b) The screen should also allow the user to enter a new food item that doesn't exist in the second list. This new food item will be added to Food table using the POST method. **(5 marks)**

The details of the new food item (i.e. calorie and fat amount) will be retrieved from public APIs (FatSecret or National Nutrient Database) and will be displayed. The selected food item's picture and other related information will be displayed by invoking Google. (APIs' marks already allocated in Task 1)

Task 7 Steps Screen (5 marks):

a) The Steps screen**:** this screen enables the user to enter the steps they have taken several times a day (you need to facilitate data entry and avoid incorrect data entry by using an appropriate UI input control). The screen will allow displaying all the entries and their time of entry. The data can be edited. **(5 marks)**

The number of the steps entered each time during the day will be stored locally (in the SQLite database), and at the end of day, the total number of all steps will be added to the server-side table as described in Task 3.

Task 8 Calorie Tracker Screen (5 marks):

a) The Calorie Tracker screen will show the set goal (already entered by the user), the total steps taken, and the total number of consumed and burned calories for that day up to that time. To calculate the consumed and burned calories you need to invoke REST calculation methods using the total steps taken, all the food eaten, and other related information. The total number of steps plus calories burned while at rest will be used to calculate calories burned for a user through invoking the REST calculation method for steps. (Note: You have the option of including the contents of this screen and/or the steps screen into the home/dashboard screen) **(5 marks)**

Task 9 Report Screen (15 marks):

The Report screen: this screen will enable the user to select and generate two types of reports. Our assumption here is that the user will use the Calorie Tracker Screen to view every day data, and use the reports for the past/historical data.

a) Pie chart: this section/screen will include a date picker to allow the user to select a date. Then the user's fitness report is displayed using a pie chart that shows the total calories consumed, the total calories burned (including calories burned at rest and for steps taken), and the remaining calorie (after compared to the calorie set goal). This will require querying the report table calling the right REST method. The labels and percentages should be shown on the pie chart. **(7 marks)**

b) Bar graph: a bar graph will show the total calories consumed, and the total calories burned (at rest and for steps taken) per day for a certain period. The screen will allow the user to enter a starting date and an ending date to create the report for that period of time. In the bar graph, you need to differentiate between calories consumed and burned (using different colours). If you use a line graph for this task, you could achieve only the half of the mark. **(8 marks)**

Task 10 Map Screen (15 marks):

a) The map screen will show the user's home location. You need to programmatically convert the location of the user based on their address and postcode into latitude and longitude values (using either Google Geocoding API or Android built-in libraries like Geocoder). Then use the latitude and longitude values for displaying the location on the map. **(8 marks)**

b) The map screen will also show the nearest parks to the user within the radius of 5 kilometres. **(5 marks)**

c) When the user taps on a park marker on the map, a new screen/view or widget should appear with the park information, e.g. the name**.** **(1 mark)**

d) The marker for the user and the marks showing nearest parks should have a different colour**.** **(1 mark)**

You can use MapQuest or Google maps to complete this task.

Additional Marking Criteria

Only completing a task does not yield a HD mark. There **are 5 other criteria** that will be considered for marking each task

Additional marking criteria are described below. Not meeting any of these criteria can result in mark deduction:

1. **Full functionality of all the operations (during the interview),**
2. **During answering interview questions, student's deep understanding of their code and the program logic (during the interview)**
3. **Handling all the exceptions and user data entry validation (and preventing program crashes)**
4. **Following coding standards, proper and meaningful naming of variables and methods, and providing minimal but useful code comments where appropriate.**
5. **Following good design guidelines for the graphical interface such as consistency (e.g. colour, font), alignment, balance, using the right contrast, avoiding clutter, visibility of objects.**

Submission Guideline:
A ZIP file will be uploaded to Moodle by the deadline including the following files **(any missing file in the zip file will result in mark deduction up to 2 marks from your total marks):**

1. You need to provide **screenshots of all your screens in ONE word document** in a proper order with a title (NOT each as a separate file).
2. The **Netbeans and Andriod projects** including **all the packages and classes and files**.
3. The zip file should have this name: **FIT5046AssignAndriod-[studentsurnname]-[studentid]-[tutor name].zip**

Late Submission:
Late Assignments or extensions will not be accepted unless you submit a special consideration form and provide valid documentation such as a medical certificate prior to the submission deadline (NOT after). Otherwise, there will be **5% penalty per day including the weekends**.

PLEASE NOTE.

Before submitting your assignment, please **make sure that you haven't breached the University plagiarism and cheating policy**. It is the student's responsibility to make themselves familiar with the contents of these documents.

Please also note the following from the Plagiarism Procedures of Monash, available at http://www.policy.monash.edu/policy-bank/academic/education/conduct/plagiarism-procedures.html

Plagiarism occurs when students fail to acknowledge that the ideas of others are being used. Specifically it occurs when:

- other people’s work and/or ideas are paraphrased and presented without a reference;
- **other students’ work is copied or partly copied**;
- **other people’s designs, codes** or images are **presented as the student’s own work**;
- Lecture notes are reproduced without due acknowledgement.