**Monash University**

**FIT5046 – Mobile and Distributed Computing Systems**

# Practical Assignment (Server-Side) (10%)

**Due Week 5, Friday (5th of April)**

## An Overview:

- The assignment is **an individual assignment**.
- This assignment 1 (server-side) **does not have an interview**.
- This assignment **requires submitting a report** that includes all the required code, diagram and screenshots as specified and in the right order, format and structure.
  (Note: **Marks will be ONLY assigned if the code and screenshots are provided in the report**)
- Practical Assignment (client-side) will be developed based on this assignment and it will have an interview.
- The marks allocated for all the tasks in this assignment will add up to 100 marks and it is **worth 10%** of the total mark for this Unit.

**Due Date:** The submission is due on **Friday Week 5 (5th April) by 3pm.**

**Objectives and learning outcomes:**
- To identify, develop and apply different approaches and methods for building distributed and mobile computing systems;
- To select and use suitable (server-side) approaches that could support the mobile computing solution for this particular case;
- To propose and develop the server side components of a mobile distributed system for a particular 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 (server-side)** requires: 1) creating a database (Java DB database in NetBeans) that will include the specified tables and populating the tables with meaningful data; and 2) creating a RESTful web service that enables querying this data and updating it if necessary.

**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.

**Assignment 1 MUST be implemented in the NetBeans environment**.

### Report Format

You need to submit a written report which includes **the code and screenshots** per each task as requested in the following sections, **with the same order of tasks and subtasks listed here.** You need to **use**

**headings and subheadings,** and **one sentence to explain** what each code does and in which class it was added, and what the screenshot shows**.**

Your code in the report **must be the same as the code** in the NetBeans project included in the zip file.

## Task 1 – Database (20 marks):

a) Based on the requirements below, you need to create the right tables and include right attributes and data. You will use an appropriate and correct schema and decide how to implement the 1-to-many relationship and choose PKs and FKs. Each attribute should have the correct data type and an appropriate name.

- **User Table** - Information about the user including: user id, name, surname, email, DoB, height, weight, gender, address, postcode, level of activity (1 to 5), and steps per mile (1.6 km), e.g. 2200 steps. **(3 marks)**

- **Credential Table** – information about the user's credential including: username, user id, password hash, and sign-up date. In this assignment, you can use an online hash converter to convert a password to hash values but in Practical Assignment (client-side), you will hash the password using Java code and libraries. **(3 marks)**

- **Food Table** - Information about a list of food (at least 20) such as their id, name, category (e.g. drink, bread, fruit etc), calorie amount, serving unit (e.g. cup, slice, each, gram/oz etc), serving amount (this depends on its serving type), and fat (g). **(3 marks)**

- **Consumption Table** - Information about the daily food consumption including: the user (id), date, food (id), and quantity/servings of the food item for that day. **(3 marks)**

  E.g. In the Food Table, a pineapple has a serving unit of cup, and serving amount of ½ cup, here if the user consumes it twice a day, the quantity will be 2.

- **Report Table** - information about tracking and counting calories such as: the user (id), date, total calories consumed, total calories burned, total steps taken, and the set calorie goal for that day (e.g. 1400). **(3 marks)**

  If later we subtract calories burned from calories consumed, it can be compared to the user's set goal to see if the user has reached their set goal or not, and how much is the difference.

b) All the tables need to be populated with meaningful data, at least **5 records** for all the tables BUT the **Food table** should have minimum **20 records**. **(3 marks)**

You will select 20 food items from the Fat and Calorie Counter file (GP_FatCal.pdf on Moodle).

c) You need to provide **a simple ER diagram** that shows all the tables and the primary and foreign keys and the relationships between them. **(2 marks)**

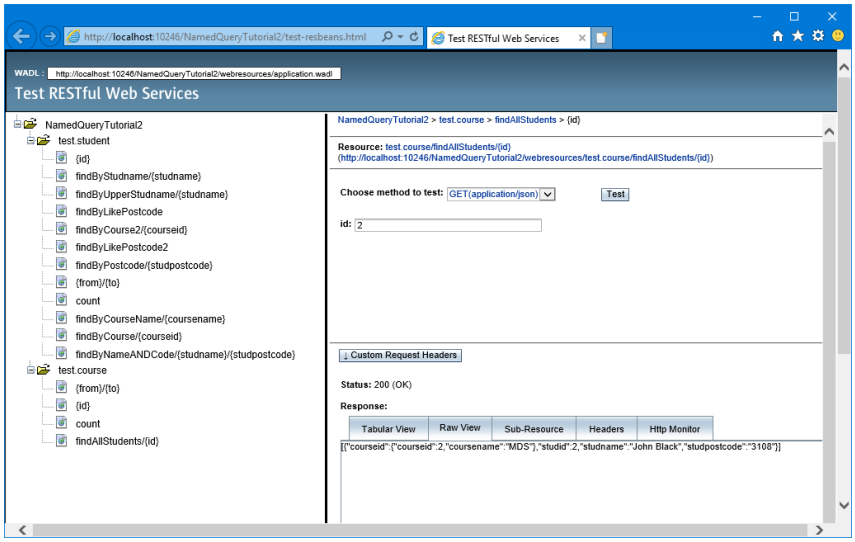
Practical Assignment (client-side) will use this database and you are allowed to make changes to it then if necessary.

You need to provide **the SQL code for creating and populating EACH table (and code for establishing FK constraints or any other constraints) and an ER diagram in your submitted report document** to receive marks for this Task. **Any missing or incomplete SQL code or an ER diagram as specified will result in mark deduction**.

## Task 2 - RESTful Web Service (5 marks)

A RESTful web service should be created based on the above-mentioned database and tested on the browser without any error. **(5 marks)**

**For this part,** you need to test the web service and **provide one screenshot showing all methods** as the figure below. On the left-hand side, you need to expand all the folders and show their contents.




---

## Task 3 – Dynamic and Static Queries (20 marks)

In this part, the web service will be extended by adding extra methods and queries using the Java Persistence Query Language (JPQL). The queries that you write will apply two different **approaches** discussed in the tutorials: one approach where you define a **static query** (NamedQuery) in Entity class and then you call it from a RESTFacade's method, and the other approach where you create **a dynamic query** directly in a RESTFacade's method.

a) You will write **additional REST methods** to query all the tables based on each attribute that the table has. E.g. if the user table has 8 attributes, you write 8 REST methods to query each attribute separately. This excludes the method to query the primary key (id) because it is automatically generated. For this task, provide all the code BUT only one screenshot for one of the REST methods for each table.
**(5 marks)**

b) You need to create a new REST method that enables querying **one table using a combination of two attributes** (any two attributes excluding the PK)**.** This should be implemented **as a DYNAMIC query**. You will decide which attributes to consider based on their usefulness in the query.
**(5 marks)**

c) You will write a new REST method that enables **querying two tables using a combination of two attributes** in the condition where **each attribute is from a different table**. The query should be **a DYNAMIC query using an IMPLICIT join**. **(5 marks)**

d) You will write a new REST method that enables **querying two tables using a combination of two attributes** in the condition where **each attribute is from a different table**. The query should be **a STATIC query using an IMPLICIT JOIN**. For this, you also need to show the code for the NamedQuery in the entity class. **(5 marks)**

For this part, you need to provide your written **code for the REST methods** and any **code in Entity classes**, and **one screenshot per each method after testing** on the browser. Any **missing code or screenshot will result in mark deduction**.

In subtask Task 3 (a), **only one screenshot** of one GET method for each table is sufficient **but provide all the code.**

## Task 4 - Calculations with REST methods (40 marks)

a) You will write a REST method that calculates the calories **burned per step** for a user. The method will accept **ONE parameter (user id)** and will **return the calories burned per step** for that user. **(10 marks)**

To do this calculation, first you need to obtain two pieces of information:
1- The user's steps per mile (1.6 km) (this was stored in the user table)
2- The user's weight in pound (lbs) (this was also stored in the user table)

You then multiply the weight of the user in pound (lbs) by 0.49 to calculate how many calories they burn per mile for casual walking.

Based on all the information you obtained, you can now compute the calories burned per each step for a user (a unique conversion factor) that will represent the user's calories burned per step.

b) Basal Metabolic Rate (BMR) is the amount of calories the body burns even when at rest. You will create a REST method to calculate BMR for a user using the following equation. The method will **accept ONE parameter (user id)** and will **return the BMR amount**. **(10 marks)**

Harris-Benedict equation[1]:
- Men: (13.75 × weight) + (5.003 × height) - (6.755 × age) + 66.5
- Women: (9.563× weight) + (1.85 × height) - (4.676 × age) + 655.1

**Note:** in the equation above, height needs to be entered in centimetres and weight in kilograms.

c) You will write another REST method that will use the following formula to calculate the total daily calories burned while at rest based on the user's BMR and their activity level stored in the database. Here, you will **call the previous BMR method** to avoid repeating the same code. The method will **accept ONE parameter (user id)** and will **return the total calories burned at rest amount**.
**(10 marks)**

1. Little/no exercise (sedentary): BMR * 1.2 = Total Calories
2. Lightly active (exercise/sports 1-3 days/week): BMR * 1.375 = Total Calories
3. Moderately active (exercise/sports 3-5 days/week): BMR * 1.55 = Total Calories
4. Very active (hard exercise/sports 6-7 days/wk): BMR * 1.725 = Total Calories
5. Extra active (very hard exercise/sports or training): BMR * 1.9 = Total Calories

d) You will add a new method to your RESTful web service. This method will accept **two parameters: user id, and a date**, and **return the total calories consumed** for the user for that day. (Note: the method should first add up total calories consumed for each food item and then calculate the total calories consumed of all the food items for that date). **(10 marks)**

[1] https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Harris-Benedict_equation.html

## Task 5 – Creating Report (15 marks)

a) You will add a new REST method that will accept **two parameters: user id** and **a date** and **return total calories consumed, total calories burned** and **remaining calorie** (based on the set goal). (Note: in this method you can query the Report Table, and do not need to redo all the calculations you did in Task 4) **(5 marks)**

b) You will add another new REST method that will **accept a user id, a starting date and an ending date** and **return the user's total calories burned, total calories consumed** and **total steps taken** for that period of time. (Note: in this method you can query the Report Table, and do not need to redo all the calculations you did in Task 4) **(10 marks)**

For Task 4 and 5, you need to provide **all the code of new REST methods,** and any code a**dded to the entity files** (and any other additional code/class you added) to receive full marks for this part. You also need to test the web service and **provide ONE screenshot** for each new method. Any **missing or incomplete code or the screenshot will result in mark deduction**.

**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 a **5% penalty per day including weekends**.

**Submission Guideline:**

A ZIP file will be uploaded to Moodle by the deadline including the following files:

1. The REPORT with **all the specified code and screenshots that were mentioned in each task in ONE word/pdf report document, NOT as separate image files.** You need **to provide a sentence to describe each screenshot and piece of code, and also use numbered headings and titles matching Assignment Tasks and parts,** and follow **the order**.
2. The **NetBeans project** including **all** the **packages** and **classes** and **files**.
3. The zip file should have this name: **FIT5046Assign1-Sem1-[studentname]-[studentid]-[tutor name].zip**

**Mark Deduction will be applied if: i) any piece of code or screenshot for each Task is missing, ii) the REPORT does not follow the above-mentioned format, or iii) if any of the NetBeans project files is missing and not included in the zip file**

**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.