School of Information Systems
Academic Year 2018/19 Term 2

SMT203: Smart City Systems and Management

Assignment 02 (Pairwork)

Due Date: Mon 25 Mar 2019 @ 1000 hrs

INSTRUCTIONS

1. This assignment is to be done **in pairs**; only one person in the pair needs to submit the assignment via eLearn.
2. You should submit a total of
   a. **three Python files** (which are named app.py, models.py and manage.py); and
   b. **1 ER diagram** (in a separate .doc or .pdf file).

   Use the skeleton Python files provided.

   You are **not** expected to make any changes to the manages.py file.
3. Your solution should be uploaded to eLearn by the stipulated deadline.
4. The late submission penalty is as follows:
   - submit <= 24 hrs late: 20% penalty (i.e., maximum grade is 80%)
   - submit <= 48 hrs late: 50% penalty (i.e., maximum grade is 50%)
   - submit > 48 hrs late: 100% penalty (i.e., maximum grade is 0%)

GRADING CRITERIA

1. This assignment will constitute 10% of your final grade.
2. Marks will be awarded based on the *logic*, *completeness* and *efficiency* of your solution. This includes your database model design and implementation, as well as API design and implementation.

OBJECTIVES

The objectives of the assignment are as follows:

1. Familiarity of Object Relational Mapping (ORM) in Python, using SQLAlchemy – which allows classes (in Python) to be mapped to the database.
2. Familiarity with creation of APIs to perform CRUD operations on the database.
3. Familiarity with ORM queries.

OVERVIEW

You are required to develop a Flask web application that uses PostgreSQL as the database, and SQLAlchemy as the ORM. The Flask web application is a simplified version of the National Steps Challenge (NSC) application, which promotes healthy and active living among Singapore residents.

In the NSC application, each person can clock a number of steps on a daily basis – which allows them to earn points (that can subsequently be exchanged for rewards – such as vouchers).

Besides the main (individual) steps challenge, each person may participate in additional challenges (see Figure 1) – such as thematic challenges, community challenge, corporate challenge and youth challenge – to win more prizes.

Figure 1 Different challenges that each person can participate in, from the NSC app

REQUIRED FUNCTIONALITIES

In particular, your Flask application should include the following functionalities:

1. **Models (classes)** for: (i) `User`; (ii) `Daily_Steps_Record`; and (iii) `Challenge`.

   Each `Challenge` object should *minimally* have a title, start date (indicating when the challenge begins) and end date (indicating when the challenge ends). You are expected to design the (attributes of) `User` and `Daily_Steps_Record` classes.

   Each user can have zero or one record in the `Daily_Steps_Record` for each day, i.e., a user can have one record on 24 Dec 2019, and 0 records on 25 Dec 2019. However, it is not possible for the user to have two or more records on 26 Dec 2019.

   In addition, each user may choose to participate in zero or more challenges; similarly, each challenge can be participated in by more than one user.

2. **Constructor** `__init__` methods for *each* of the above classes (`User`, `Daily_Steps_Record`, and `Challenge`.

3. **Serialize** `serialize()` methods for *each* of the above classes (`User`, `Daily_Steps_Record`, and `Challenge`.

4. **POST (Create)** APIs for *each* of the above classes (`User`, `Daily_Steps_Record`, and `Challenge`.

   Users may be created during the onboarding process, i.e., when they 'sign up' for an account. The daily steps record is created whenever the user clocks a number of steps for a particular day. The challenge record is typically created by the administrator(s) of the application.

5. **GET (Read)** APIs for *each* of the above classes (`User`, `Daily_Steps_Record`, and `Challenge`.

FAQS

You are highly encouraged to ask assignment related questions in the eLearn discussion forum, instead of contacting the instructors individually.

You may also find the below FAQs helpful.

1. Must I only have three (3) classes (`User`, `Daily_Steps_Record`, and `Challenge`)?

   Ans: No, you can have more than three classes in your solution, as you deem fit. However, the three above-mentioned classes *must* be present in your solution.

2. Must I write the `__init__`, `__repr__` and `serialize()` methods for each class?

   Ans: Refer to Required Functionalities #2 and #3. Only the `__init__` and `serialize()` methods *must* be included in your solution, for the three classes (`User`, `Daily_Steps_Record`, and `Challenge`). You are allowed to have more than these three classes, and you are also free to implement additional methods for any of the classes, as you deem fit.

3. What are the attributes to be included in each class/model?

   Ans: This is part of your (graded) class/model design.

4. What type of relationships should exist between the classes?

   Ans: This is part of your (graded) class/model design. You may refer to Required Functionalities #1 to determine the appropriate relationship(s) between the classes, if any.

5. What are the parameters to be used in each of the API requests (POST, GET and UPDATE)?

   Ans: This is part of your (graded) API design.

6. Can I make changes to the method parameters?

   Ans: Yes – you definitely need to make changes to some of the method parameters for your code to work.

7. Can I make changes to the method names that have been provided in the skeletal codes?

   Ans: Yes, but preferably not. You can include additional methods if you wish to.

8. Can I make changes to the API URLs that have been provided in the skeletal codes?

   Ans: Yes, but preferably not. You can include additional APIs if you wish to.

REFERENCES

1. Flask
2. PostgreSQL
3. SQLAlchemy

~ END ~