Lab 3 Handout: The ORM Magic and The Service Layer

Due: 29 December 2021

1 Objectives

- Understand *dependency inversion*.
- Map a class to a database table using SQLAlchemy's ORM (object-relational mapper).
- Implement a service layer for a user to read an article.
- Practice Test-driven development (TDD).

2 Task description

In this lab, you are going to understand how to keep the domain model *pure* by following the principle of *dependency inversion* - let the infrastructure depend on the domain model, but not the other way around.

Also, you are going to implement a service layer in `services.py` for EnglishPal, which provides a core service called `read`. This service would choose a suitable article for a user to read. The function `read` takes as input the following four arguments and returns an article ID if the user has been successfully assigned with an article to read.

- `user`: a `User` object. The class `User` is defined in `model.py`. `User` has an important method called `read_article`.
- `user_repo`: a `UserRepository` object. The class `UserRepository` is defined in `repository.py`.
- `article_repo`: an `ArticleRepository` object. The class `ArticleRepository` is defined in `repository.py`.
- `session`: an SQLAlchemy session object.

The function `read(user, user_repo, article_repo, session)` raises an `UnknownUser` exception if `user` does not have a correct user name or a correct password, or raises a `NoArticleMatched` exception if no article in the article repository, i.e., `article_repo`, has a difficulty level matching the user's vocabulary level. We say that an article's difficulty level, L_a, matches a user's vocabulary level, L_u, iff $L_a > L_u$. If more than one article satisfies $L_a > L_u$, then the one with the smallest L_a is chosen.

An article's difficulty level is recorded in the `level` field in the database table `articles`. A user's vocabulary level is defined as the average value of top n most difficult words in the user's list of new words (recorded in the database table `newwords`), where n is either 3 or the number of new words belonging to that user in the table `newwords`, whichever is smaller.

For simplicity, we only consider the words in the following dictionary, where the values represent these words' difficulty levels.

```
d = {'starbucks':5, 'luckin':4, 'secondcup':4, 'costa':3, 'timhortons':3,
     'frappuccino':6}
```

Download the following starter code to get started:

- `orm.py`
- `model.py`
- `repository.py`

- `services.py`
- `conftest.py`
- `test_services.py`

You must complete `orm.py` and `services.py` such that running the following command could make all the five test cases defined in `test_services.py` pass: `pytest -v -s test_services.py`. You must not modify anything in `test_services.py`.

Hint: `conftest.py` helps `pytest` set things up while running `test_services.py`. You may want to learn what a fixture is and how to use it.

3 Requirements

- Do the lab in a group. The group must be the same as your course project group.
- Your job is to complete the downloaded python source code `orm.py` and `services.py` such that running the command `pytest -v -s test_services.py` shows no error.
- Do not write any raw SQL statements in `services.py` or `model.py`.
- Submit by the due data a lab report prepared using Read the Docs. Your lab report must follow the structure described in How to Write a Computer Science Lab Report.
- Your lab report must contain the following content:
  1. The modified `orm.py`.
  2. The modified `services.py`.
  3. A detailed explanation for why your modification works.
  4. An answer to this question: Does your function `read` in `services.py` follow the Single Responsibility Principle (SRP) principle? Why or why not?
- Submit your lab report (in PDF format) through LRR. Do not forget to include your group information. Do not miss any group member's name.