TDT4145 Project 2021 DB2

Simen Omholt-Jensen, Anna Zhang, Vebjørn Steinsholt

March 24, 2021

Demo

For a video demonstration of some of the implemented use cases, please go to
`https://github.com/simenojensen/TDT4145_Project/`.
Repository made public after the deadline.

Modules

- `main.py`
  - This module prompts the user for their MySQL login information. A database called `TDT4145ProjectGroup131` is created and filled with data from the `.csv` files in the `data` folder. Program is then run by prompting the user for either (1) log in as student, (2) log in as instructor, or (3) quit.
- `utils.py`
  - This module contains code that creates the `TDT4145ProjectGroup131` database, creates the tables of the `TDT4145ProjectGroup131` database, and fills the database with data found in the `.csv` files in the `data` folder.
  - **Functions:**
    - `create_database()`
      - Helper function that executes the MySQL query for database creation.
    - `setup_database()`
      - Drops the `TDT4145ProjectGroup131` database if already exists, then creates the database and executes the table initialization statements.
    - `insert_data()`
      - Reads the `.csv` files from the `data` folder and inserts the data into the existing `TDT4145ProjectGroup131` database.
- `tables.py`
  - This module contains code which defines the MySQL tables, their fields and their constraints used to setup the `TDT4145ProjectGroup131` database. The database name is stored in the string `DB_NAME` variable. The tables are stored in a dict named `TABLES`.
- `piazza_user.py`
  - This module contains code for the classes implementing the functionality for the required use cases. The `PiazzaUser` class is a super class to the `Instructor` and `Student` classes. The `PiazzaUser` contains functionality shared between the `Instructor` and `Student` classes such as logging in, creating posts such as threads or replies, and keyword search. The `Instructor` class includes functionality for viewing statistics.

Classes

- `PiazzaUser` (super class):
  Handles functionality shared among the `Student` and `Instructor` sub classes.
  - **Methods:**
    - `login()`
      - Handles both `Student` and `Instructor` login functionality by prompting the user for an `useremail` and a `userpassword`. The method then verifies the input by checking if they exist in the `Login` MySQL table.
      - On successful verification, the method then stores information about the user (`UserID`) and the course (`CourseID`) which are used for later use cases.
    - `create_post()`
      - Provides a text-based menu interface prompting the user for which type of post to create. User is prompted for (1) create a thread, (2) create a reply, or (3) go back to the action menu.
    - `create_thread()`
      - Creates a thread post by prompting the user for post content, folder, and tag. The input is then inserted in the MySQL tables `Post`, `Thread`, `Tags`, and `ThreadInFolder` as explained in DB1.
    - `create_reply()`
      - Creates a reply post by prompting the user for a post id and post content. A post is then inserted into the MySQL table `Post`, and the MySQL table `Thread` is updated to reflect that the thread post has received a reply.
    - `search_keyword()`
      - Prompts the user for a keyword to search. The program then searches for matches in (1) the post content of posts, (2) thread tags, (3) folder names, and (4) user names. The method then prints the `PostID` for posts related to the keyword search matches.
    - `close()`
      - Closes the MySQL connection.

- `Student` (sub class):
  Provides a text-based menu interface for allowable `Student` actions.
  - Methods:
    - `action_menu()`
      - User is prompted for the following options (1) create a post, (2) search for a keyword, and (3) log out.

- `Instructor` (sub class):
  Provides a text-based menu interface for allowable `Instructor` actions. Also includes functionality to view user statistics.
  - Methods:
    - `action_menu()`
      - User is prompted for the following options (1) create a post, (2) search for a keyword, (3) view statistics, and (4) log out.
    - `view_statistics()`
      - Methods prints a table consisting of user names, the number of threads read by those users, and the number of posts created by those users. The table is sorted on highest read posting numbers.

Use Cases

All use cases were implemented within a fixed course.

1. A student or an instructor can log in by checking that their user email and user password exists in the MySQL `Login` table. This is implemented by either instantiating a `Student` or an `Instructor` sub class. The `__init__` function of the super class `PiazzaUser` is then run, which handles the log in functionality. This is implemented in the `PiazzaUser` method `login()`.
2. A student or an instructor can create an initial post, which is then considered a thread. The thread includes post content, a folder (e.g. "Exam"), and a tag (e.g. "Question"). The function for creating a thread is either called from the `Student.action_menu()` method, or the `Instructor.action_menu()` method depending on whether the user is logged in as a student or instructor. The method that implements the functionality for creating a thread is implemented in the `PiazzaUser.create_thread()` method.
3. A student or an instructor can reply to a post. The reply contains a post id to the post to reply to and some post content. The function for creating a reply is either called from the `Student.action_menu()` method, or the `Instructor.action_menu()` method depending on whether the user is logged in as a student or instructor. The method that implements the functionality for creating a reply is implemented in the `PiazzaUser.create_reply()` method.
4. A student or an instructor can search for a keyword and receive a list of post ids of posts that are related to the keyword. A related post id is a post related to a match in (1) the post content of posts, (2) the thread tags, (3) folder names, and/or (4) user names. The function for keyword search is either called from the `Student.action_menu()` method, or the `Instructor.action_menu()` method depending on whether the user is logged in as a student or instructor. The method that implements the functionality for keyword search is implemented in the `PiazzaUser.search_keyword()` method.
5. An instructor can view user statistics. This statistics view includes user names for a course, their number of threads read, and number of posts created. This functionality is unique for an `Instructor`, and is thus implemented in the `Instructor.view_statistics()` method.