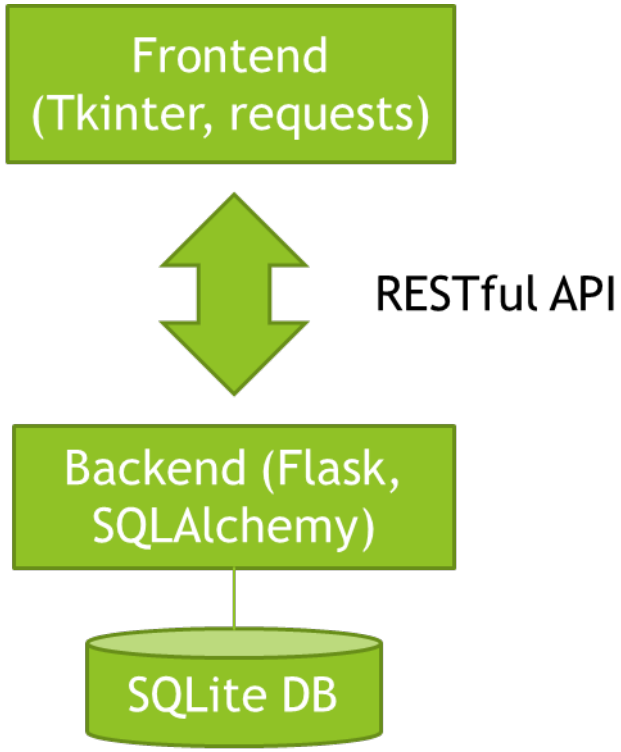
ACIT 2515 – Final Exam Practice Question

This practice question can be used as preparation for the final exam. **The question on the final exam will have different objects, different requirements and a specific marking breakdown**. However, the general pattern for the application will be the same – i.e., backend RESTful API/SQLite database and frontend Tkinter GUI application.

For this practice question, you will be creating a backend and frontend application following the general pattern:



The application manages student records for a school.

A co-worker who no longer works at your company started building it, but did not complete it. It is your task to get it fully working.

The backend consists of:

- A SQLite database consisting of a single a single table called “students”
- An incomplete SQLAlchemy declarative for the student records from the database (student.py)

- A stubbed out StudentManager class but with a working unit test (student_manager.py and student_manager_test.py)
- A stubbed out RESTful API that should use the StudentManager class to add, delete and get all student records (student_api.py)

The frontend only has a main application with a root window (student_gui.py).

It is your task to complete the implementation of both the backend and frontend applications into a fully functional and usable system.

The high-level requirements include:

- User is able to view all, add and delete student records through the GUI.
- A student record consists of a timestamp, first_name, last_name and username. The timestamp is displayed in the list of student record in the GUI (i.e., View All). However, it is only set by the system when the student record is first added (i.e., it is not set through the GUI).
- The GUI uses Tkinter windows, widgets and layouts and the requests package for calls to the backend RESTful API.
- The GUI must use the RESTful API to get all, add and delete the student records.
- RESTful API is implemented using Flask.
- The RESTful APIs must return 200 as the status code on success and 400 for bad input data.
- The StudentManager class provides methods to add, delete and get all student records and interacts with the database using a SQLAlchemy session and a declaratives (Student class).
- The StudentManager class passes all the provided unit tests.

Instructions:

- Create a new Python project
- Make sure you install the Flask, SQLAlchemy and requests packages
- Install the code from the zipfile provided with this practice question into your project. Files include:
    - base.py
    - create_tables.py
    - student.py
    - student_api.py
    - student_gui.py
    - student_manager.py
    - test_student_manager.py
- Create the SQLite database using the create_tables.py script
- Implement the code and make sure the following work:

- Frontend – View All, Add and Delete Students. Integrated with the backend APIs.
- Backend APIs:
  - GET /students/all
  - POST /students
  - DELETE /students/<int:id>
- Unit Tests – test_student_manager.py