CPSC 449 - Web Back-End Engineering

Project 1 - Fall 2023

In this project, you will work with a team to design and implement a back-end Web Service to manage course enrollment and waiting lists with functionality similar to TitanOnline.

Project Teams

This project must be completed in a team of five or six. The instructor will assign teams for this project in Canvas.

See the following sections of the Canvas documentation for instructions on group submission:

- How do I submit an assignment on behalf of a group?

Tasks

Define the API

Create a RESTful service that exposes the following resources and operations:

Classes

Each class is assigned a *department*, *course code*, *section number*, *name*, and *instructor*. The system also tracks the *current* and *maximum* enrollment for each class.

The API should allow students to:

- List available classes
- Attempt to enroll in a class
- Drop a class

Instructors should be able to:

- View current enrollment for their classes
- View students who have dropped the class
- Drop students administratively (e.g. if they do not show up to class)

The registrar should be able to:

- Add new classes and sections
- Remove existing sections
- Change the instructor for a section

- Freeze automatic enrollment from waiting lists (e.g. during the second week of classes)

Waiting lists

If a student attempts to enroll in a class that is currently full, they should automatically be added to the next spot in the waiting list. The maximum list size is 15, and students may not be on more than 3 waiting lists. Waiting lists are ordered by the date a student was added to the list.

The API should allow students to:

- View their current position on the waiting list
- Remove themselves from a waiting list

Instructors should be able to:

- VIew the current waiting list for the course

Design the database

Define an SQL schema to store the resource information in the database. Note that the tables in the database do not necessarily map directly to resources — retrieving a resource and representing it in JSON may require data from several tables to be joined together.

Informally, your database schema should be in approximately third normal form. In particular, data items should be atomic: this means, for example, that lists of words should be stored as separate rows, not as a single column. For a review of database normalization, see Thomas H. Grayson's lecture note Review: Database Design Rules of Thumb from MIT Course 11.521.

In addition to defining the schema, pre-populate the database with sample courses, enrollments, and waiting lists. You may do this directly in SQL, using the SQLite Command Shell to import CSV or other formats, or from Python code.

Implement the service

Use FastAPI to define endpoints and representations for the resources above. Each endpoint should make appropriate use of HTTP methods, status codes, and headers.

Your APIs should follow principles of RESTful design, with the exception that all input and output representations should be in JSON format with the `Content-Type:` header field set to `application/json`.

Your service should be stateless, with each request to a resource identifying all necessary information to perform the operation. For more details, see Chapter 5 of the textbook.

Provide a Procfile to start the service, and test it with Foreman.

Platform

Your project must run successfully on Tuffix 2022.Provide the source code and any other necessary files (for example, data or shell scripts). You will also need to provide either a plain-text README file or documentation in PDF format describing how to use your project.

Note: this is a back-end project only. While your browser may be able to display JSON objects returned in response to an HTTP GET request, you do not need to implement a front-end or other user interface for testing API methods. You may assume that endpoints will be tested with an HTTP client program such as HTTPie or curl, or with the interactive API docs generated by FastAPI.

Libraries and tools

This project must be implemented in Python using the FastAPI framework and ancillary tools such as Foreman and the Command Line Shell For SQLite. Database code must use the `sqlite3` module from the Python Standard Library; you may not use an ORM library such as SQLAlchemy or Peewee.

Documenting your results

Document your API, input and output formats, and any other ancillary material such as programs or scripts in a report in PDF format.

At the beginning of your report, include the following:

- The names of each member of the team who participated in the project
- The course number, section, and semester
- The project number

For each task:

- Identify what is to be done.
- Document the tools to be used.
- Describe your results, including definitions, diagrams, screenshots, or code as appropriate.
- Comment on the results, including references you consulted, variations you considered, or issues you encountered.

Submission

Submit your work as a tarball (`.tar.gz`, `.tgz`, `.tar.Z`, `.tar.bz2`, or `.tar.xz`) file through Canvas by the due date. Only one submission is required for a team.

Grading

The project will be evaluated on the following five-point scale, inspired by the general rubric used by Professor Michael Ekstrand at Boise State University:

Exemplary (5 points)

The project is a success. All requirements met. The quality of the work is high.

Basically Correct (4 points)

The project is an overall success, but some requirements are not met completely, or the quality of the work is inconsistent.

Solid Start (3 points)

The project is mostly finished, but some requirements are missing, or the quality of the work does not yet meet professional standards.

Serious Issues (2 points)

The project has fundamental issues in its implementation or quality.

Did Something (1 point)

The project was started but has not been completed enough to assess its quality fairly or is on the wrong track.

Did Nothing (0 points)

The project was not submitted, contained work belonging to someone else, or was of such low quality that there is nothing to assess.