Project 1 Overview

Computer Science 143 - Spring 2019

TuneSearch

In this project, you will design a (very) simple search engine to search for song lyrics. We will call this search engine *TuneSearch* for lack of creativity and take a trip back to the 1990s, a very exciting time for the World Wide Web and music if I say so myself.

Before Google debuted in 1998, there were a ton of other search engines with their own pros and cons and technical achievements such as AltaVista, Excite, Lycos, Yahoo (it was a search engine at one point), HotBot, Dogpile, Ask Jeeves, WebCrawler, Inktomi, MetaCrawler, InfoSeek and All the Web. Most of these search engines used some form of an index, inverted index, and some ranking method. In later years, song lyric sites started popping up. In this project, we will use a sample of data from LyricsFreak.com.

In this project, students will build a basic search engine for song lyrics backed by several technologies, but most importantly PostgreSQL. Just think, this project could have made you a billionaire back in the 90s.

The other technologies that will be used in this project (*though students will not need to touch most of it*) are:

1. **Nginx**, a high performance HTTP/web server. It has many other uses as a reverse proxy server, mail proxy server and a generic TCP/UDP proxy server, but we are only using the web server part.
2. **Flask**, a popular micro-platform for web and API development. Flask is a great way to prototype web projects and host small projects quickly. It is fairly lightweight.
3. **uWSGI**, is a protocol and serves as a "glue" between Nginx and Flask.
4. **Jinja2,** is a templating engine used by Flask. Developers write standard HTML files and embed special tokens where variables passed to the template should be displayed.
5. **PostgreSQL**, is an open source object-relational database management system with an emphasis on extensibility and standards compliance. It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.
6. **psycopg2**, a Python package for interacting with PostgreSQL from Python.
7. **Python 3**, the world's best programming language.

Since this is the beginning of a databases and data systems class, the focus of this project is on **writing queries** and the amount of new Python code that must be written should be minimal.

System Setup

To help students set up the uniform environment for the class project, we will be using VirtualBox to run the Linux operating system in a virtual machine. VirtualBox allows a single machine to share resources and run multiple operating systems simultaneously. You will need to download the following files

- VirtualBox binary file for your host Operating System. The latest version is 6.0.4 and we have not experienced problems with it.
- VirtualBox Image: CS143-19S-P1.ova - This is a very large file (~2.5GB), so it may take a while to download.

and follow our VirtualBox setup instruction to install VirtualBox on your own machine.
The provided virtual machine image is based on Ubuntu 18.04 LTS, PostgreSQL 10.6, Nginx 1.14, Flask 1.0.2, Python 3.6.7.

If you have access to an equivalent machine that has PostgreSQL, Nginx, Python 3, and Flask installed, you may use it instead of the virtual machine image. In this case, we do recommend setting up Python and Flask using virtualenv. However, please note that we **will not** provide support for systems other than the virtual machine image, and that your project MUST be runnable on the provided virtual machine. We will be using the virtual machine image for grading purposes, and if your submission does not work within this setup, you may get zero points. We cannot make any exceptions to your project schedule for problems incurred by using your own computing facilities.

Project

Your project is to build a simple song lyrics search engine. Your system manages all of its data at the back-end in a PostgreSQL database and provides a Web interface to the users at the front-end. While this task may sound daunting at first, you will be surprised how easy and helpful a commercial DBMS is for such a project.

To develop Project 1 all students will use the PostgreSQL DBMS and the Ubuntu Linux operating system. You will also use the Python programming language to access PostgreSQL and the Flask microplatform (powered by Nginx) to provide a Web interface. While there are many other ways to develop Web applications, and while Apache and PHP is one of the most popular method these days to implement a database-backed server for a small-to-medium scale Web site, we want to use a standard language across both CS 143 projects and also introduce students to Flask, which is very popular in the data community and lightweight. We will provide online references that will help you learn Python and PostgreSQL. We will also give you a considerable amount of real data to populate your system.

For Project 1, we will expect certain minimal functionality in your system - beyond that, the sky's the limit. Minimal functionality includes a variety of queries and searching capabilities over the data in

your system. Various integrity constraints must be monitored. Although you will use PostgreSQL, which has reasonable transaction support, multi-user issues are not a focus of the project.

The main reason for using PostgreSQL in lieu of MySQL this quarter is that it follows the ANSI SQL standard presented in the textbook much, much more closely than does MySQL. While MySQL may be more popular (currently) and in some cases easier to use, it only implements a subset of the ANSI SQL standard and even fewer features.

Project 1 consists of two parts:

Part A: Creating Schemas and Loading Data

In this part, you will get familiar with the overall project environment and learn basic Python if you do not know them yet. You will write several queries to create database tables with the correct data types, keys and constraints and then load data into these tables. **This part does not require Python,** though some students may wish to practice by writing a Python script, using the psycopg2 package, to load the data from the script instead.

Due Date: **Friday, April 19, 11:59pm**

Part B: PostgreSQL Backed Search Engine

In this part, you will write queries and some Python code that fetches search results given an initial search query, and display the results on a search page. This task involves SELECT statements, JOINs and subqueries and will provide good practice for the midterm.

Due Date: **Friday, April 26, 11:59pm**

Partners

Students may implement the project individually or in teams of two. The choice is up to each student, but please keep the following things in mind when you select your project partner:

1. An identical amount of work is expected and the same grading scale is used for individual and team projects. Faculty experience indicates that in general it is not necessarily easier or more productive to work in teams of two - it's largely a matter of personal preference and working style.
2. If you choose to work as a team, you are encouraged to make use of collaborative authoring tools for synchronizing your work and ideas, such as version control software such as a **Private** GitHub repo (private repos on GitHub are now **free**) and online document tools (e.g. Google Docs, Dropbox Paper). **Each UCLA student has free access to Google Docs and Google Drive.**
3. If you work in a team, choose your partner carefully. Teams are permitted to "divorce" at any time during the course (due to incompatibility, or any other reason), and individual students may choose to team up as the project progresses, however students from divorced teams

may not form new teams or join other teams. Put another way, if a student turns in any part of the project as part of a team, every later part of the project must be turned in individually or as part of the same team. The only exception is reforming a new partnership in the event that each partner was widowed by a partner dropping the course.

4. Both partners in a team will receive exactly the same grade for each project part turned in jointly. We will not entertain any complaints of the form "I did all the work and my partner did nothing." Choose your partner carefully!
5. If you work in a team, your work must be turned in jointly, as ONE submission. That is to say, only ONE of you two should submit your work as a team. Note that teamwork turned in as individual work will be considered as plagiarism and handled through official University channels.

Late Submission Policy

To accommodate emergencies that a student may encounter, each student (or team) has a 4-day grace period for late submission to use throughout the quarter. The grace period can be used for any part of the project in the unit of one day. For example, a student may use 1-day grace period for part 1B and 2-day grace period for part 2A. Any single project part may not be more than 2 days late. Note that the grace period can be used in the unit of one day. even if a student submits a project 12 hours late, he/she needs to use a full day grace period to avoid late penalty.

Electronic Submission of Projects

All project submission should be done electronically. **We will not accept any submissions by email, USB drive, paper printout, floppy disk etc.** Steps for submitting your project electronically is as follows:

1. Visit the online submission page on CCLE for the particular project, linked from the corresponding assignment page.
2. Make sure you have followed all directions and included any other ancillary files that are required.
3. Click Submit. You should receive a confirmation page and email with the timestamp.
4. If you need to resubmit something, just redo these directions. The submission page will notice if you are attempting to resubmit and overwrite your previous entry. Remember that only the very last submission will be graded.

Project References

Unix & VirtualBox

- Unix tutorial
- VirtualBox overview

Python3 references

- Python 3 tutorial
- Another great Python 3 tutorial
- psychopg2: PostgreSQL Access from Python 3
- Python Regular Expression Tutorial
- Python function reference