Project 1: Books

Objectives

- Become more comfortable with Python.
- Gain experience with Flask.
- Learn to use SQL to interact with databases.

Overview

In this project, you'll build a book review website. Users will be able to register for your website and then log in using their username and password. Once they log in, they will be able to search for books, leave reviews for individual books, and see the reviews made by other people. You'll also use the a third-party API by Goodreads, another book review website, to pull in ratings from a broader audience. Finally, users will be able to query for book details and book reviews programmatically via your website's API.

Getting Started

PostgreSQL

For this project, you'll need to set up a PostgreSQL database to use with our application. It's possible to set up PostgreSQL locally on your own computer, but for this project, we'll use a database hosted by Heroku, an online web hosting service.

1. Navigate to https://www.heroku.com/, and create an account if you don't already have one.
2. On Heroku's Dashboard, click "New" and choose "Create new app."
3. Give your app a name, and click "Create app."
4. On your app's "Overview" page, click the "Configure Add-ons" button.
5. In the "Add-ons" section of the page, type in and select "Heroku Postgres."
6. Choose the "Hobby Dev - Free" plan, which will give you access to a free PostgreSQL database that will support up to 10,000 rows of data. Click "Provision."
7. Now, click the "Heroku Postgres :: Database" link.
8. You should now be on your database's overview page. Click on "Settings", and then "View Credentials." This is the information you'll need to log into your database. You can access the database via Adminer, filling in the server (the "Host" in the credentials list), your username (the

"User"), your password, and the name of the database, all of which you can find on the Heroku credentials page.

Alternatively, if you install PostgreSQL on your own computer, you should be able to run `psql URI` on the command line, where the `URI` is the link provided in the Heroku credentials list.

Python and Flask

1. First, make sure you install a copy of Python. For this course, you should be using Python version 3.6 or higher.
2. You'll also need to install `pip`. If you downloaded Python from Python's website, you likely already have `pip` installed (you can check by running `pip` in a terminal window). If you don't have it installed, be sure to install it before moving on!

To try running your first Flask application:

1. Download the `project1` distribution directory from https://cdn.cs50.net/web/2019/x/projects/1/project1.zip and unzip it.
2. In a terminal window, navigate into your `project1` directory.
3. Run `pip3 install -r requirements.txt` in your terminal window to make sure that all of the necessary Python packages (Flask and SQLAlchemy, for instance) are installed.
4. Set the environment variable `FLASK_APP` to be `application.py`. On a Mac or on Linux, the command to do this is `export FLASK_APP=application.py`. On Windows, the command is instead `set FLASK_APP=application.py`. You may optionally want to set the environment variable `FLASK_DEBUG` to `1`, which will activate Flask's debugger and will automatically reload your web application whenever you save a change to a file.
5. Set the environment variable `DATABASE_URL` to be the URI of your database, which you should be able to see from the credentials page on Heroku.
6. Run `flask run` to start up your Flask application.
7. If you navigate to the URL provided by `flask`, you should see the text `"Project 1: TODO"`!

Goodreads API

Goodreads is a popular book review website, and we'll be using their API in this project to get access to their review data for individual books.

1. Go to https://www.goodreads.com/api and sign up for a Goodreads account if you don't already have one.
2. Navigate to https://www.goodreads.com/api/keys and apply for an API key. For "Application name" and "Company name" feel free to just write "project1," and no need to include an application URL, callback URL, or support URL.

3. You should then see your API key. (For this project, we'll care only about the "key", not the "secret".)
4. You can now use that API key to make requests to the Goodreads API, documented here. In particular, Python code like the below

```
import requests
res = requests.get("https://www.goodreads.com/book/review_counts.json", params={"key": "KEY",
print(res.json())
```

where `KEY` is your API key, will give you the review and rating data for the book with the provided ISBN number. In particular, you might see something like this dictionary:

```
{'books': [{
                'id': 29207858,
                'isbn': '1632168146',
                'isbn13': '9781632168146',
                'ratings_count': 0,
                'reviews_count': 1,
                'text_reviews_count': 0,
                'work_ratings_count': 26,
                'work_reviews_count': 113,
                'work_text_reviews_count': 10,
                'average_rating': '4.04'
            }]
}
```

Note that `work_ratings_count` here is the number of ratings that this particular book has received, and `average_rating` is the book's average score out of 5.

Requirements

Alright, it's time to actually build your web application! Here are the requirements:

- **Registration**: Users should be able to register for your website, providing (at minimum) a username and password.
- **Login**: Users, once registered, should be able to log in to your website with their username and password.
- **Logout**: Logged in users should be able to log out of the site.
- **Import**: Provided for you in this project is a file called `books.csv`, which is a spreadsheet in CSV format of 5000 different books. Each one has an ISBN number, a title, an author, and a publication year. In a Python file called `import.py` separate from your web application, write a

program that will take the books and import them into your PostgreSQL database. You will first need to decide what table(s) to create, what columns those tables should have, and how they should relate to one another. Run this program by running `python3 import.py` to import the books into your database, and submit this program with the rest of your project code.

- **Search**: Once a user has logged in, they should be taken to a page where they can search for a book. Users should be able to type in the ISBN number of a book, the title of a book, or the author of a book. After performing the search, your website should display a list of possible matching results, or some sort of message if there were no matches. If the user typed in only part of a title, ISBN, or author name, your search page should find matches for those as well!
- **Book Page**: When users click on a book from the results of the search page, they should be taken to a book page, with details about the book: its title, author, publication year, ISBN number, and any reviews that users have left for the book on your website.
- **Review Submission**: On the book page, users should be able to submit a review: consisting of a rating on a scale of 1 to 5, as well as a text component to the review where the user can write their opinion about a book. Users should not be able to submit multiple reviews for the same book.
- **Goodreads Review Data**: On your book page, you should also display (if available) the average rating and number of ratings the work has received from Goodreads.
- **API Access**: If users make a GET request to your website's `/api/<isbn>` route, where `<isbn>` is an ISBN number, your website should return a JSON response containing the book's title, author, publication date, ISBN number, review count, and average score. The resulting JSON should follow the format:

```
{
    "title": "Memory",
    "author": "Doug Lloyd",
    "year": 2015,
    "isbn": "1632168146",
    "review_count": 28,
    "average_score": 5.0
}
```

If the requested ISBN number isn't in your database, your website should return a 404 error.

- You should be using raw SQL commands (as via SQLAlchemy's `execute` method) in order to make database queries. You should not use the SQLAlchemy ORM (if familiar with it) for this project.
- In `README.md`, include a short writeup describing your project, what's contained in each file, and (optionally) any other additional information the staff should know about your project.
- If you've added any Python packages that need to be installed in order to run your web application, be sure to add them to `requirements.txt`!

Beyond these requirements, the design, look, and feel of the website are up to you! You're also welcome to add additional features to your website, so long as you meet the requirements laid out in the above specification!

Hints

- At minimum, you'll probably want at least one table to keep track of users, one table to keep track of books, and one table to keep track of reviews. But you're not limited to just these tables, if you think others would be helpful!
- In terms of how to "log a user in," recall that you can store information inside of the `session`, which can store different values for different users. In particular, if each user has an `id`, then you could store that `id` in the session (e.g., in `session["user_id"]`) to keep track of which user is currently logged in.

FAQs

For the API, do the JSON keys need to be in order?

Any order is fine!

`AttributeError: 'NoneType' object has no attribute '_instantiate_plugins'`

Make sure that you've set your `DATABASE_URL` environment variable before running `flask run`!

How to Submit

1. Using Git, push your work to `https://github.com/submit50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `web50/projects/2019/x/1` or, if you've installed `submit50`, execute

   ```
   submit50 web50/projects/2019/x/1
   ```

   instead.

2. Record a 1- to 5-minute screencast in which you demonstrate your app's functionality and/or walk viewers through your code. Upload that video to YouTube (as unlisted or public, but not private) or somewhere else.
3. Submit this form.