Flask DND Stat Sheets

Daniel Smalley\*

Ira A. Fulton School of Engineering and Technology, Brigham Young University

Clyde Engineering Bldg, Campus Dr, Provo, UT 84604

\*smalley@byu.edu

**Abstract:** In this paper I give a description of the code used to create a first attempt at a DND stat sheet. Specifically, I focus on the Google API component.

1. Introduction

This project describes a web application for tracking student growth via stat sheets. Stat sheets are useful in putting together teams for large quests and for tracking growth and deficiencies as a student progresses. It also provides a mechanism for recognizing and rewarding achievement.

The web app uses Google Sheets as a database and the Google API to communicate to a Flask python web application hosted on Pythonanywhere as shown in Figure 1.

|  |
| --- |
|  |
| Figure 1. Overview of web app |

1. Methods

I am including notes for the key software components: Google API, Flask, and venv.

*Google API Notes:*

The web app uses Google Sheets as a database and the Google API to communicate to a Flask python web application hosted on Pythonanywhere. The best tutorials for setting up google API are:

<https://www.youtube.com/watch?v=7I2s81TsCnc>

<https://www.youtube.com/watch?v=cnPlKLEGR7E&t=21s>

The file structure:

Flask2 (project directory)

|\_\_\_\_ stats\_main.py

requirements.txt

spreadsheetexample-272021-….json

My\_Goog\_Pack

|\_\_\_\_\_\_\_\_\_ \_\_init\_\_.py

goog\_funcs\_r2.py

templates

|\_\_\_\_\_\_\_\_\_\_\_\_ spec.html

fenv…

UBUNTU

*Google API*

1. Setup a Google Sheet, give sheet a name
2. Go to Google Console, create a project
3. Enable Sheets API
4. Enable Google Drive API
5. Create Credentials (service account)
6. Create Key (JSON)
   1. Copy into proj. directory
   2. Fill in goog\_func heading info
7. Get client email
   1. Share google sheet with client email

*Command Line*

1. In cmd line project directory:
   1. python –m venv fenv
   2. source fenv/bin/activate
   3. pip install gspread
   4. pip install oauth2client
   5. pip install flask
2. In hello.py

from flask import Flask

from My\_Goog\_Pack import goog\_funcs as goog

from markupsafe import escape

from flask import render\_template

print(goog.get\_goog\_data())

lol=goog.get\_goog\_data()

topics=lol[0]

def find\_user(username,list\_of\_lists):

user\_list=[]

for elem in list\_of\_lists:

if elem[0] == username:

user\_list=elem

return user\_list

def convert\_to\_dict(my\_list, keys):

my\_dict={}

for index in range(len(keys)):

my\_dict.update({keys[index]:my\_list[index]})

print(my\_dict)

return my\_dict

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Bello, World!'

@app.route('/index')

def compare():

return render\_template('compare.html',topics=topics, lol=lol[1:])

@app.route('/user/<username>')

def show\_user\_profile(username):

my\_list=find\_user(username,lol)

if my\_list:

my\_dict=convert\_to\_dict(my\_list,topics)

else:

my\_dict={}

return render\_template('spec.html',my\_dict=my\_dict)

1. In templates/spec.html

<!DOCTYPE html>

<html>

<body>

{% if my\_dict %}

<h1>DND SPEC SHEET FOR: {{my\_dict.username}}</h1>

<p>Stats:</p>

<ul>

<li>code level: {{my\_dict.code\_level}} </li>

<li>art level: {{my\_dict.art\_level}} </li>

<li>fab level: {{my\_dict.fab\_level}} </li>

</ul>

{% else %}

<h1>Sorry, no such user</h1>

{% endif %}

</body>

</html>

1. export FLASK\_APP=hello.py
2. export FLASK\_ENV=development
3. flask run

WINDOWS

*Google API*

1. Setup a Google Sheet, give sheet a name
2. Go to Google Console, create a project
3. Enable Sheets API
4. Enable Google Drive API
5. Create Credentials (service account)
6. Create Key (JSON)
   1. Copy into proj. directory
7. Get client email
   1. Share google sheet with client email

*Command Line*

1. In cmd line project directory:
   1. python –m venv fenv
   2. fenv/Scripts/activate
   3. pip install gspread
   4. pip install oauth2client
   5. pip install flask
   6. pip install pprint (optional)
2. In stat\_main.py

from flask import Flask

from My\_Goog\_Pack import goog\_funcs as goog

from markupsafe import escape

from flask import render\_template

print(goog.get\_goog\_data())

lol=goog.get\_goog\_data()

topics=lol[0]

def find\_user(username,list\_of\_lists):

user\_list=[]

for elem in list\_of\_lists:

if elem[0] == username:

user\_list=elem

return user\_list

def convert\_to\_dict(my\_list, keys):

my\_dict={}

for index in range(len(keys)):

my\_dict.update({keys[index]:my\_list[index]})

print(my\_dict)

return my\_dict

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Bello, World!'

@app.route('/index')

def compare():

return render\_template('compare.html',topics=topics, lol=lol[1:])

@app.route('/user/<username>')

def show\_user\_profile(username):

my\_list=find\_user(username,lol)

if my\_list:

my\_dict=convert\_to\_dict(my\_list,topics)

else:

my\_dict={}

return render\_template('spec.html',my\_dict=my\_dict)

1. In templates/spec.html

<!DOCTYPE html>

<html>

<body>

{% if my\_dict %}

<h1>DND SPEC SHEET FOR: {{my\_dict.username}}</h1>

<p>Stats:</p>

<ul>

<li>code level: {{my\_dict.code\_level}} </li>

<li>art level: {{my\_dict.art\_level}} </li>

<li>fab level: {{my\_dict.fab\_level}} </li>

</ul>

{% else %}

<h1>Sorry, no such user</h1>

{% endif %}

</body>

</html>

1. set FLASK\_APP=hello.py
2. set FLASK\_ENV=development
3. flask run
4. Results

I can get the stats for any individual student in the database with user/ and the student name like this:

<http://127.0.0.1:5000/user/daniel> or

<http://127.0.0.1:5000/user/dylan>

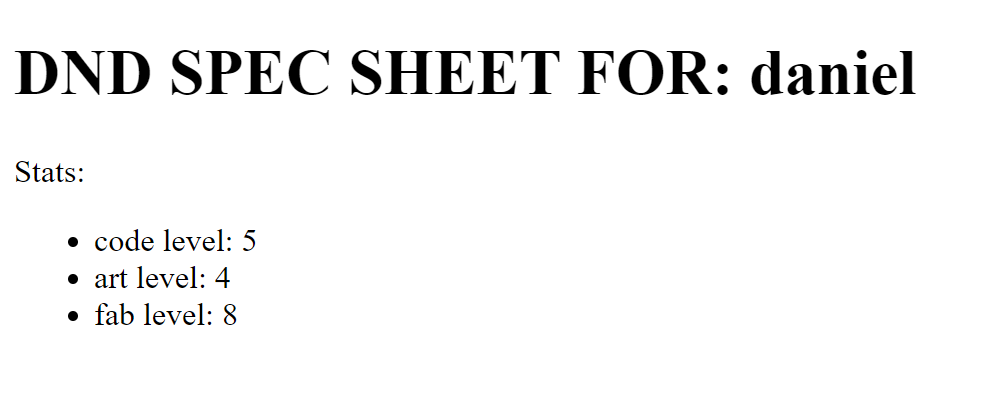


Figure 2. Screenshot of stat page

1. Conclusion

The Flask app works well, it especially nice that I can dynamically make pages by inputing student names directly into the url. The next step will be to make a comparison page and revise the input form.

References

Google API:

<https://www.youtube.com/watch?v=7I2s81TsCnc>

<https://www.youtube.com/watch?v=cnPlKLEGR7E&t=21s>

Flask official documentation is the best reference:

<https://flask.palletsprojects.com/en/1.1.x/quickstart/#routing>

<https://flask.palletsprojects.com/en/1.1.x/quickstart/#rendering-templates>

Appendix A: Difficulties

Remember to set flask variables and use flask run command to run the application

URL variables ended up being the best way to make up dynamic pages.