CIS 445 Project 1 - Python Web Application

Matt Gates, Tanner Graham, Cade McPartlin

**Goal**

Our main goals for this project were to learn the basics of programming in Python, how to create a web API in Python, and how to integrate a database in Python. To accomplish these goals, we decided on creating a webpage that displayed the player information of four major league baseball teams. The team and player information we stored in MongoDB Atlas, a cloud-based database.

**Predicted Lessons Learned**

**Front End**

Since I had more experience in back end programming, I wanted to test myself to see if I could create a nice GUI for this web application. By taking on the front end of this application, I was hoping to learn more about how HTML, CSS, and JavaScript work, while also learning the Bootstrap framework, because it is a really powerful framework for quickly developing GUIs. I also hoped to increase my efficiency when it came to HTML escaped code, and dynamically generating data that I am sent from the back end.

**Back End - Python Code**

I have been told by multiple people that Python is a very good programming language to know. I had no experience with Python prior to the start of this project. My goal for this project was to learn the basics of how to program in Python. I would then also learn how to use the Flask library to create a web server in Python along with pulling information from a database.

**Database**

As I had never done any integration of applications or code before I figured it would be an interesting challenge to take charge on creating the database. I hoped that creating this database would teach me more about integrating a server address and managing data to then be queried at a later point. Lastly, it would help me figure out what different web servers and databases may be used for in other projects.

**Issues**

**Front End**

I (Cade) was tasked with designing and creating the front end for this web application. When initially designing the front end, it seemed like a straight forward design. We wanted a home page where a user could select the MLB team they wanted to see the roster for, and then we would display a table with all of the players of that team, along with some of their statistics.

The issues I had when creating the front end was with dynamically generating the information. We wanted our front end to be dynamic so if our data changed, then the front end would not need to be fixed. To dynamically generate the data, we used HTML templates, which is a feature in Flask. These templates allow you to HTML escape Python code so that you can pass in data into the HTML page to be loaded.

The problem I ran into was that I was using Python to generate this data, but I also needed to pass this data from Python to JavaScript, because we wanted player images to popover the player’s name when the mouse hovered over them.

**Back End - Python Code**

For this project I(Tanner) was assigned to write the Python code on the back end that would create the web server, query the necessary data from the database, and then pass it to the html to be displayed on the website. We decided on using the Flask library as our web API.

At first the main issues I had were learning and getting familiar with the Python syntax as it is much different from the other languages I have worked with(C++ & Java). Also learning the functions and commands of the Flask library was another resource I had to familiarize myself with. Once I got more into writing the code some an issue that arose was how to pass the information from the database to the html. We had to decide what would be the most efficient and easiest way to break down the data that needed to be sent to the html. Another issue that I came upon was what type of data structure to use to store the information and pass it to the website.

**Database**

For this project I (Matt) was tasked with getting a database running, populating it with data, and then beginning the connection between Python and MongoDB using PyMongo. The first issue I had was that whatever database I had had to be accessed by anyone at any time. This meant that it would must be a service through a company or I would have to host it on my own PC.

Once I had gotten the database up and running I then needed to populate it with data and choose string labels that would be easily identifiable in the code for clarity sake. This didn’t turn out to be much of an issue even though I went ahead and labelled certain elements “Player Name” or something similar, which could have presented itself as an issue with the spacing. Lastly, there was the connection between MongoDB and Python with PyMongo. This proved to be a rather finnicky task with certain modules and terminal/command prompt commands not working.

**Solutions**

**Front End**

The issue I described above in the issues section took me some time to figure out and required me to write a jQuery function so that whenever a new record was added to the table, it also had a popover event added to it. Then I had to HTML escape the image URL into the popover tag that was created with each player so that information would be stored per player.

Overall, this issue that I faced was the most time consuming part of creating the front end, but I got more hands on experience using multiple languages at once, and I learned how to make those languages interact with each other on a basic level.

**Back End - Python Code**

To resolve the issues I was having with the Python syntax I had to do some research on w3schools.com and watch some tutorials on YouTube. For the Flask functions and commands I was able to find a guide to Flask at flask.pocoo.org. When trying to figure out the best way to break down the data we deduced that it would be best to store each MLB team’s information in one data structure. Then we passed that single data structure to the html and in the html we pulled each player from the data structure. When determining which data structure to use to store the data in the two options we had were using either a dictionary or a list. I decided to use a list as the data structure to store each team’s player information in.

**Database**

To resolve the issue of a dedicated host for the database I signed up for MongoDB Atlas under their free/trial server. With this we are allowed up to 100 connections at any given time along with 512MB of storage (we’ve used a measly 64.2KB). Cade was able to point me in the direction of using MongoDB and from there I was able to determine this was the best option for our needs. Once I had that up and running I was able to populate it with our data sets. We decided to do four teams of 25 players. Each of these data sets included their name, age, position, height, weight, jersey number, batting method, throwing arm, team name, and a photo link. These are all stored on the database as JSON values. Lastly, we had to connect the database using PyMongo. This was more of a group effort between all of us in solving it. We all put together bits and pieces to get it working and in the end was a pretty simple set of code to establish a connection.

**Lessons Learned**

**Front End**

My lessons learned when it came to the front end design were mostly what I expected. I wanted to learn more about the Bootstrap framework, and I got a little taste of it when designing the GUI. I used a few Bootstrap elements such as the jumbotron, navbar, and the table is actually a Bootstrap table. I found that Bootstrap made it very easy to create these elements because all I needed to do was apply a specific class to the element I wanted to modify and Bootstrap took care of the rest.

I also learned more about HTML escaping, and how different scripting languages can interact with each other on common ground. It is interesting that both Python and JavaScript can work together within HTML and CSS to create a web application. This was really useful for when I wanted to use the Bootstrap popover element. I passed data from Python to a JavaScript event handler, which in turn passed HTML to Bootstrap to generate the popover that displays when a user hovers over a player’s name.

**Back End - Python Code**

The biggest lesson I learned while working on this project was the basics of how to program in Python. This was my main goal for this project I am definitely accomplished it as I was able to really familiarize myself with the Python language. While the syntax is much different than C++ and Java the logic was pretty much the same. It was very interesting seeing the differences between an interpreter language and a compiler language.

Another lesson I learned was how to create a web server using the Flask library. I used the Flask library functions to create routes to a website and pass information to the website html. Also I was able to learn how to access a database in Python and pull information from the database using queries.

**Database**

My lessons learned were about as expected. I learned that sometimes it’s easier to use a web service to setup a database as opposed to hosting your own. I’ll want to learn this eventually but for now it may be a bit ahead of me. I also learned to manage data and establish a connection between two services. Integrating the Python code with Mongo was very exciting because I could see on the most basic level a connection between two platforms and a transfer of data from one to the other.

I also learned a lot about Python, which I had never coded in beforehand. It was definitely a bit confusing at first not having to end a line with a semicolon and spacing being such a key component. Most of my experience is using Perl or Java so it was eye opening to what other types of languages are like and how they each have a specialty that they’re best at.

**Application of Knowledge**

As a team we believe that the most important part of this project that we can apply to future projects was the integration between the MongoDB database, the Python code, and the GUI. This process posed a challenge to us in the beginning but now we have a better understanding of how to do this. It is important to know how to do this because most projects require some sort of integration.

Works Cited

Api.mongodb.com. (2018). *PyMongo 3.7.2 Documentation — PyMongo 3.7.2 documentation*. [online] Available at: https://api.mongodb.com/python/current/ [Accessed 15 Oct. 2018].

Flask.pocoo.org. (2018). *Welcome to Flask — Flask 1.0.2 documentation*. [online] Available at: http://flask.pocoo.org/docs/1.0/ [Accessed 15 Oct. 2018].

"Python Tutorial", *W3schools.com*, 2018. [Online]. Available: https://www.w3schools.com/python/default.asp. [Accessed: 15- Oct- 2018].

W3schools.com. (2018). *Bootstrap 4 Tutorial*. [online] Available at: https://www.w3schools.com/bootstrap4/default.asp [Accessed 15 Oct. 2018].

Docs.atlas.mongodb.com. (2018). *Connect via Driver — MongoDB Atlas*. [online] Available at: https://docs.atlas.mongodb.com/driver-connection/#python-driver-example [Accessed 15 Oct. 2018].