– Independent Work Report Spring, 2018 –

Introduction to Web Development Using TigerMenus

Alexander Xu
Adviser: David Walker

Abstract

This document outlines a new COS 333 assignment that intends to give future students an introduction to web development in the context of building and deploying a simple web application similar to the popular student app TigerMenus, assuming no prior web development experience.

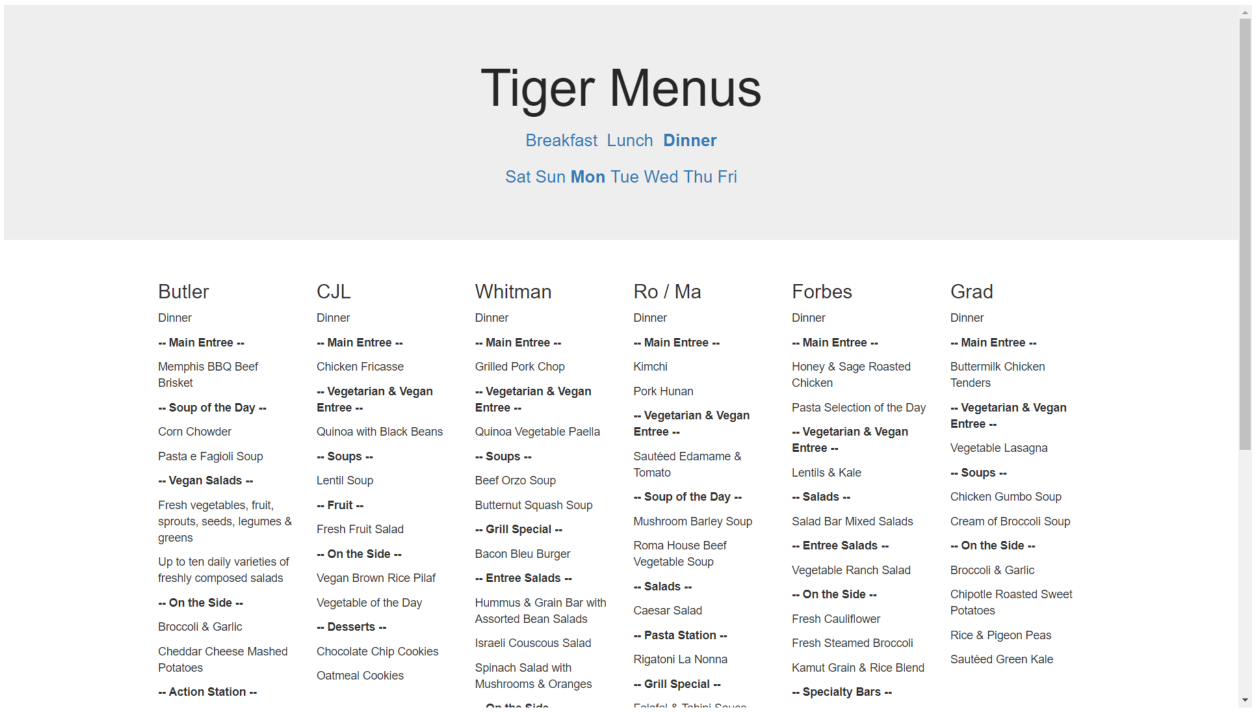
1. Introduction

For many students, COS 333: Advanced Programming Techniques (colloquially known as the "build an app" class) is their first programming intensive departmental (versus a more theoretical and math-based course like COS 340) and their first real exposure to web development.

But none of the assignments in the spring offering of COS 333 actually show how to build a full stack web application. I think an assignment that does do that would inspire students to do more on their own as side projects and could aid in their own final project for the class.

Moreover, sometimes in final projects, members are split into roles such as "front-end" or "back-end" so even after the final project is over, students still don't have a complete grasp of building a full stack application.

I think a good context to teach this would be in making a simple web application that is similar to one many students already use: the popular student app TigerMenus. TigerMenus is a convenient to use dining hall menu website that I created in mid Spring 2017 and gets 1500+ page views daily. It simply scrapes data for all the dining halls and displays it in 6 columns. It is hosted at `https://tigermenus.herokuapp.com/`. Since many students already use it, I thought students would be more engaged with the material since they could potentially contribute to TigerMenus after the assignment is done versus doing some arbitrary assignment similar to something that could be found by simply googling "mean stack tutorial" or "python flask mega tutorial."


Tiger Menus

Breakfast Lunch **Dinner**

Sat Sun **Mon** Tue Wed Thu Fri

Butler
Dinner
-- Main Entree --
Memphis BBQ Beef Brisket
-- Soup of the Day --
Corn Chowder
Pasta e Fagioli Soup
-- Vegan Salads --
Fresh vegetables, fruit, sprouts, seeds, legumes & greens
Up to ten daily varieties of freshly composed salads
-- On the Side --
Broccoli & Garlic
Cheddar Cheese Mashed Potatoes
-- Action Station --

CJL
Dinner
-- Main Entree --
Chicken Fricasse
-- Vegetarian & Vegan Entree --
Quinoa with Black Beans
-- Soups --
Lentil Soup
-- Fruit --
Fresh Fruit Salad
-- On the Side --
Vegan Brown Rice Pilaf
Vegetable of the Day
-- Desserts --
Chocolate Chip Cookies
Oatmeal Cookies

Whitman
Dinner
-- Main Entree --
Grilled Pork Chop
-- Vegetarian & Vegan Entree --
Quinoa Vegetable Paella
-- Soups --
Beef Orzo Soup
Butternut Squash Soup
-- Grill Special --
Bacon Bleu Burger
-- Entree Salads --
Hummus & Grain Bar with Assorted Bean Salads
Israeli Couscous Salad
Spinach Salad with Mushrooms & Oranges

Ro / Ma
Dinner
-- Main Entree --
Kimchi
Pork Hunan
-- Vegetarian & Vegan Entree --
Sautéed Edamame & Tomato
-- Soup of the Day --
Mushroom Barley Soup
Roma House Beef Vegetable Soup
-- Salads --
Caesar Salad
-- Pasta Station --
Rigatoni La Nonna
-- Grill Special --

Forbes
Dinner
-- Main Entree --
Honey & Sage Roasted Chicken
Pasta Selection of the Day
-- Vegetarian & Vegan Entree --
Lentils & Kale
-- Salads --
Salad Bar Mixed Salads
-- Entree Salads --
Vegetable Ranch Salad
-- On the Side --
Fresh Cauliflower
Fresh Steamed Broccoli
Kamut Grain & Rice Blend
-- Specialty Bars --

Grad
Dinner
-- Main Entree --
Buttermilk Chicken Tenders
-- Vegetarian & Vegan Entree --
Vegetable Lasagna
-- Soups --
Chicken Gumbo Soup
Cream of Broccoli Soup
-- On the Side --
Broccoli & Garlic
Chipotle Roasted Sweet Potatoes
Rice & Pigeon Peas
Sautéed Green Kale

Figure 1: How TigerMenus looks like.

Since this is a Python assignment, this would ideally be placed after the COS 333 assignment that teaches students Python.

The assignment has 3 major parts:

1. Querying a web API using the requests library (Python)
2. Using data from a pre-provided file
3. Web scraping using BeautifulSoup (Python)

These parts are meant to teach common ways of extracting data from websites. Sometimes a website will provide data in a well-formatted way through a web API. Sometimes a website will provide data as a downloadable file. In these two cases, data is already available in a convenient format like JSON. But sometimes, a website only has data in HTML meant to be read by humans with no easy way of being extracted for later use or analysis. This is where web scraping is necessary to parse the wanted information into a convenient format. We use the requests and BeautifulSoup libraries since they are the standard ways of requesting and scraping data from websites in Python.

This assignment also aims to teach:

1. Web frameworks and HTML templating using Flask (Python)
2. Basic styling using Bootstrap
3. Web hosting using Heroku

Web frameworks and HTML templating are the standard ways of transforming data in Python to human-readable webpages. We use Flask since it is a microframework well suited to building simple applications like this. It also has very little boilerplate code, so novice users will be able to understand the entire codebase of a Flask application. We use Bootstrap since it the standard way of making websites responsive and styled. We use Heroku to deploy the application since it has a free tier and is very well integrated with Python. It also has add ons that allow for easy provisioning of free PostgreSQL and MongoDB databases.

Using this technical knowledge, students will be able to imagine how to build larger scale applications.

But this assignment wants to teach more than just technical skills. During each part, I will try to avoid telling students exactly how to do something. I will instead point to documentation, sometimes hinting at useful functions. After all, this is how real software engineers learn new tools and technologies: by reading documentation. (And using Stack Overflow.)

This assignment ultimately aims to show students just how trivially easy some of parts of the web development experience are. After this assignment, students will have no excuse for not making side projects that they can talk about during interviews. Maybe some students will even submit pull requests to TigerMenus on GitHub.

2. Problem Background and Related Work

In Spring 2017, 181 students took COS 333 and filled out the beginning of semester survey. The survey asked how many programming intensive COS departmentals each student had taken or are currently taking. Responses were as follows:

```
COS326: 36   COS318: 21   COS461: 8   COS418: 1   COS320: 1
```

So therefore, of that group, at most $36+21+8+1+1=67$ students (37% of the class), had taken a programming intensive COS departmental before.

The survey also asked which web languages students were fluent in or could cope with. Responses included (ignoring any non web technologies or technologies that only got 1 response):

```
Fluent
 Python 32      Javascript 17  HTML 9         CSS 4          PHP 3
 Ruby 2
Cope:
 Python 41      HTML 24        Javascript 17  CSS 13         PHP 9
 Ruby 8         SQL 6          MySQL 3        Rails 2
```

It seems clear that for most students in the class, COS 333 will be their first exposure to web development.

Similarly, in Spring 2018, responses were (out of 108 responses):

```
  COS318: 2   COS326: 11   COS418: 3   COS435: 1   COS461: 12
```

So at most $2+11+3+1+12=29$ students (27%) had taken a COS departmental before.

Web technology familiarity responses were:

```
  Fluent
    python 29 javascript 8 html 6 css 4 react 3
  Cope
    python 35 javascript 21 html 9 sql 6 php 5 css 5 ruby 2
```

How familiar students are with web technologies seems pretty similar between 2017 and 2018.

Therefore, for many students, COS 333 is their first programming intensive departmental class and their first real exposure to web development. (Most students are sophomores.)

Therefore, these students need an assignment in that class that teaches the basics of full stack web development.

In fact, a past iteration of COS 333, previously taught in 2011, seems to fill that need. I will refer to this version of 333 as Fall 333. I will refer to the other version of 333, the one that doesn't explicitly teach web development, as Spring 333. It was taught in the Spring for the last 3 years.

Fall 333 had a completely different curriculum than the spring, with all four of its assignments focused on building a non trivial full stack web application. When it was taught again in Fall 2017, it received great reviews, a 4.71 overall vs 4.18/4.18/3.94 for Spring 333. The reviews of the assignments specifically got 4.74 vs a 3.81/3.96/3.62 for Spring 333.

It seems people like this more practical version of COS 333. In fact, it is being taught again in Fall 2018.

But the spring offering of COS 333 still doesn't have practical assignments like the fall version. So to gauge interest in an assignment like this, I ran a survey that was posted on the TigerMenus homepage that was up for about 10 days. It had two questions:

1. How hard do you think it was to make TigerMenus? [155 Responses]
2. Would you be interested if there was a tutorial that taught you how to do web scraping, web frameworks, database, web hosting, and more in the context of creating and deploying your own simple web app like TigerMenus? (No prior web dev experience assumed.) [186 Responses]

The first question had:

- 56 responses along the lines of "Hard" (36%),
- 82 responses along the lines of "Average" (53%),
- 17 responses along the line of "Trivial" (11%).

This curiously mirrors the proportion of students who never took COS 126, took COS 126, and are COS majors.

The second question had:

- 163 responses along the lines of "Yes, sounds like a great idea!" (88%),
- 10 responses along the lines of "No, this was trivial." (5%)
- 13 responses along the lines of "No, other reason." (7%).

I also ran a workshop in Fall 2017 called "Introduction to Web Development Using TigerMenus" that was attended by around 20 people. Feedback was enormously positive, so there is definitely interest.

3. Approach

The naive solution would be having the assignment be creating a simplified version of TigerMenus, which would mean scraping the Princeton Campus Dining website.

However, since the source code for TigerMenus is publicly available online through GitHub, this may not be the best idea.

So I decided to make the assignment about scraping a similar data source: Princeton eating clubs. The only 2 eating clubs with easily accessible and well formatted data were Colonial Club and Charter Club. To complement these eating clubs, the assignment will also involve using the TigerMenus API, since only two data sources isn't very exciting.

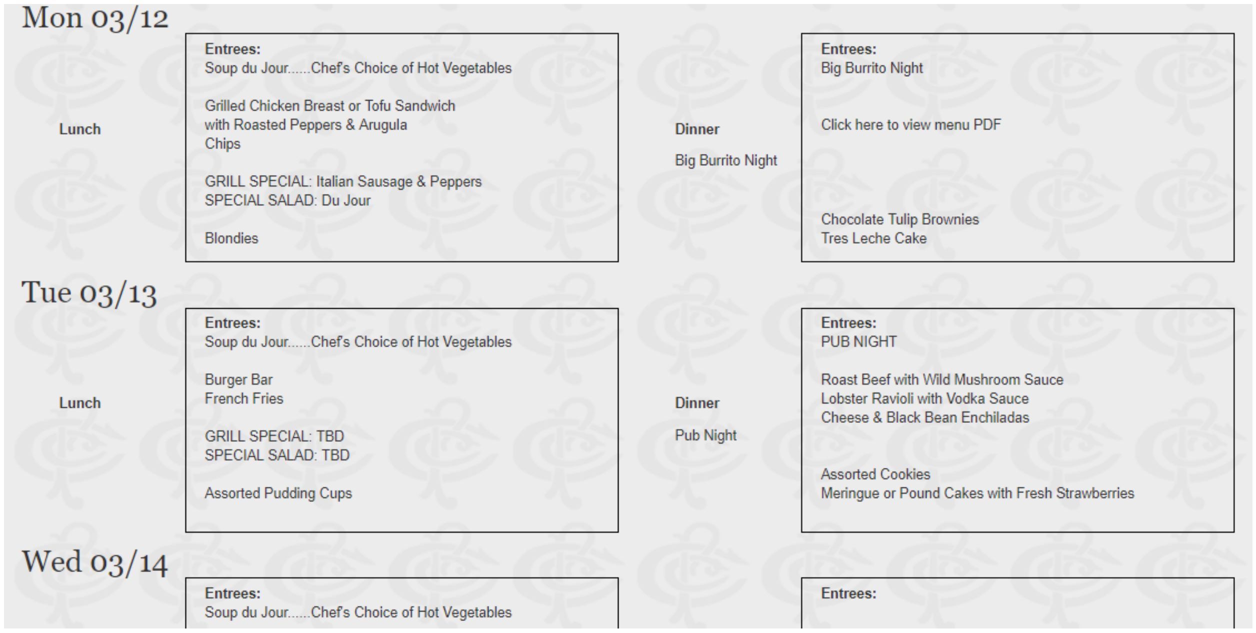
Colonial Club only had well formated data from Fall 2017. It had little data from Spring 2017 and the format changed midway in Spring 2018. So it makes sense for the assignment to focus only on Fall 2017 menus.

After extensive cleaning of the data, I got it into a format that looks like this:

```
{
  "category": "Hot Line",
  "date": "2018-02-08",
  "dishes": [
      "Chicken Steamed Dumplings / Veggie Dumplings",
      "Vegetarian Stuffed Portabellas",
      "Szechuan Meatballs / Veggie Noodles",
      "Mac n Cheese Bites / Chicken Wings",
      "Fried Calamari with Banana peppers / Fried Mozzarella sticks"
  ],
  "meal": "Dinner"
}
```

I will offer all the Colonial Club Fall 2017 menus in a Python file.

Charter Club has Fall 2017 Menu data on its own website. The HTML pages are extremely well formatted and lends itself well for scraping. I predownloaded all Fall 2017 menu HTML pages so students will not all be pinging the Charter Club server right before the deadline.



Mon 03/12

Lunch

Entrees:
Soup du Jour......Chef's Choice of Hot Vegetables

Grilled Chicken Breast or Tofu Sandwich
with Roasted Peppers & Arugula
Chips

GRILL SPECIAL: Italian Sausage & Peppers
SPECIAL SALAD: Du Jour

Blondies

Dinner
Big Burrito Night

Entrees:
Big Burrito Night

Click here to view menu PDF

Chocolate Tulip Brownies
Tres Leche Cake

Tue 03/13

Lunch

Entrees:
Soup du Jour......Chef's Choice of Hot Vegetables

Burger Bar
French Fries

GRILL SPECIAL: TBD
SPECIAL SALAD: TBD

Assorted Pudding Cups

Dinner
Pub Night

Entrees:
PUB NIGHT

Roast Beef with Wild Mushroom Sauce
Lobster Ravioli with Vodka Sauce
Cheese & Black Bean Enchiladas

Assorted Cookies
Meringue or Pound Cakes with Fresh Strawberries

Wed 03/14

Entrees:
Soup du Jour......Chef's Choice of Hot Vegetables

Entrees:

Figure 2: How well formatted for scraping Charter Club menus are.

The TigerMenus API has data in this format and contains data for Fall 2017 to present:

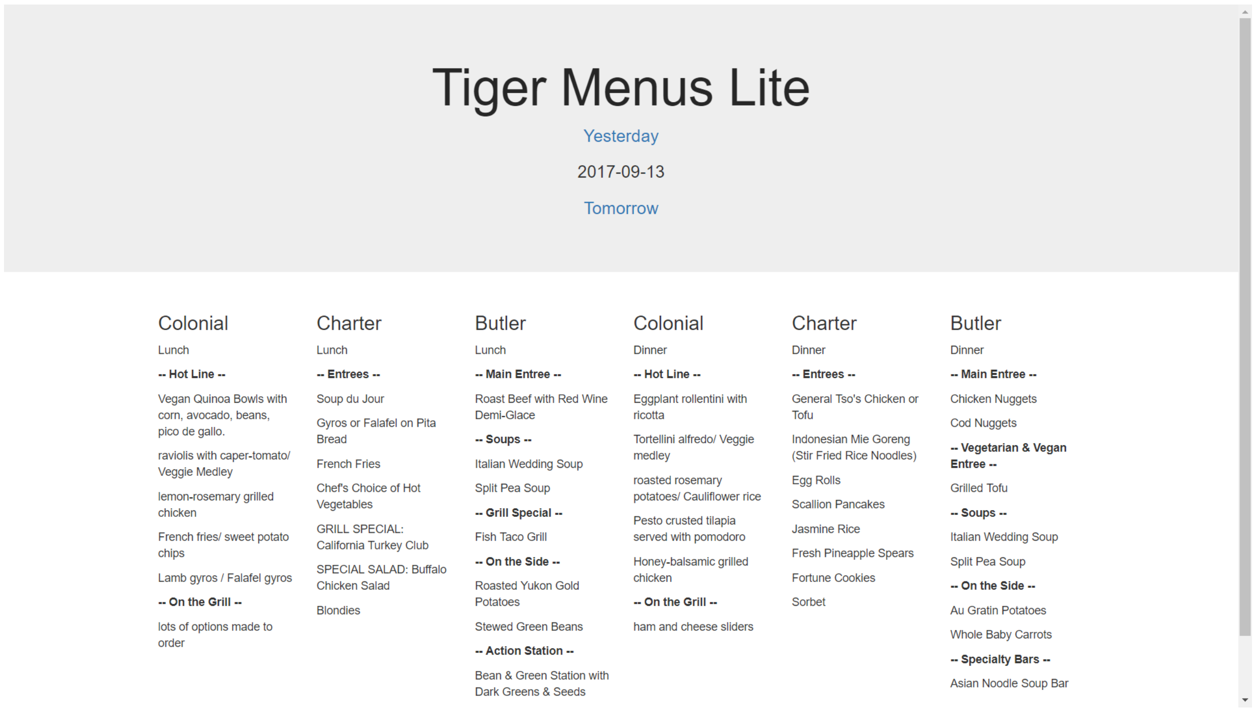
```
"date_modified": "Thu, 14 Dec 2017 00:06:08 GMT",
"dinner": [
  [
    {
      "item": "-- Main Entree --",
      "legend": "label"
    },
    {
      "item": "Peach Country Spare Ribs",
      "legend": "pork"
    },
    {
      "item": "Moroccan Lentil Soup",
      "legend": "vegan"
    },
```

I will allow students to ping the TigerMenus server for API requests so they can get experience with using a web API.

The assignment itself will involve creating a menu app that looks similar to TigerMenus but has eating club menus as well. It will only cover Fall 2017 menus.

4. Implementation

My first step was implementing the assignment's final product. It will look like this:



Tiger Menus Lite

Yesterday

2017-09-13

Tomorrow

Colonial
Lunch
-- Hot Line --
Vegan Quinoa Bowls with corn, avocado, beans, pico de gallo.
raviolis with caper-tomato/ Veggie Medley
lemon-rosemary grilled chicken
French fries/ sweet potato chips
Lamb gyros / Falafel gyros
-- On the Grill --
lots of options made to order

Charter
Lunch
-- Entrees --
Soup du Jour
Gyros or Falafel on Pita Bread
French Fries
Chef's Choice of Hot Vegetables
GRILL SPECIAL: California Turkey Club
SPECIAL SALAD: Buffalo Chicken Salad
Blondies

Butler
Lunch
-- Main Entree --
Roast Beef with Red Wine Demi-Glace
-- Soups --
Italian Wedding Soup
Split Pea Soup
-- Grill Special --
Fish Taco Grill
-- On the Side --
Roasted Yukon Gold Potatoes
Stewed Green Beans
-- Action Station --
Bean & Green Station with Dark Greens & Seeds

Colonial
Dinner
-- Hot Line --
Eggplant rollentini with ricotta
Tortellini alfredo/ Veggie medley
roasted rosemary potatoes/ Cauliflower rice
Pesto crusted tilapia served with pomodoro
Honey-balsamic grilled chicken
-- On the Grill --
ham and cheese sliders

Charter
Dinner
-- Entrees --
General Tso's Chicken or Tofu
Indonesian Mie Goreng (Stir Fried Rice Noodles)
Egg Rolls
Scallion Pancakes
Jasmine Rice
Fresh Pineapple Spears
Fortune Cookies
Sorbet

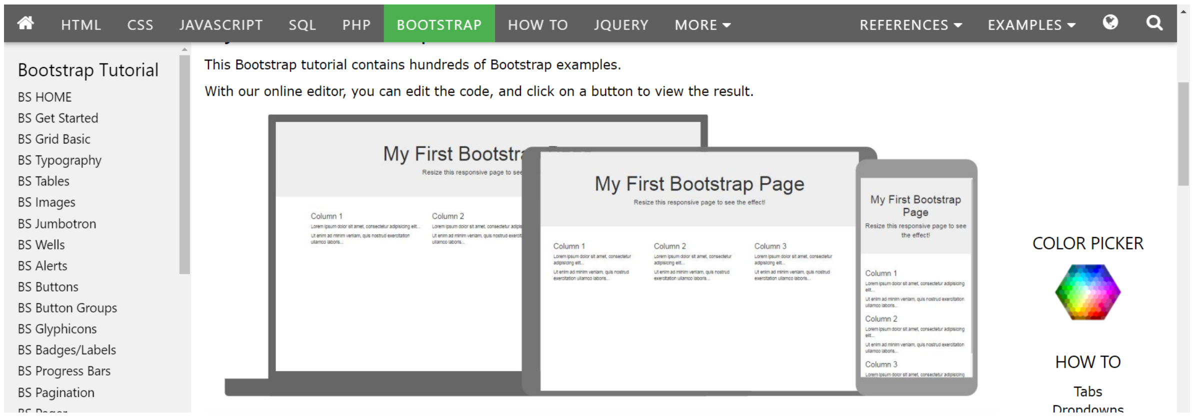
Butler
Dinner
-- Main Entree --
Chicken Nuggets
Cod Nuggets
-- Vegetarian & Vegan Entree --
Grilled Tofu
-- Soups --
Italian Wedding Soup
Split Pea Soup
-- On the Side --
Au Gratin Potatoes
Whole Baby Carrots
-- Specialty Bars --
Asian Noodle Soup Bar

Figure 3: Lunch menus on the left. Dinner on the right.

You can scroll through dates using the "Yesterday" and "Tomorrow" buttons.

4.1. Frontend

The design of the site is basically taken from one of the first results on Google for "Bootstrap"



This Bootstrap tutorial contains hundreds of Bootstrap examples.

With our online editor, you can edit the code, and click on a button to view the result.

Figure 4: W3Schools

How to create the design was as easy as following the instructions on that webpage.

First you had to import Bootstrap. In this assignment's case, doing that is simply including these lines in the HTML header:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link crossorigin="anonymous" rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
```

The first line makes the site automatically rescale for mobile screens. The second line includes the official Bootstrap CSS. This is the same for all sites powered by Bootstrap (though more complicated apps will need to import more things.) In this assignment, I included no custom CSS to keep the code simple

After importing Bootstrap, creating the design is essentially just putting title text inside a `<div class="jumbotron text-center">` element and putting column text inside a `<div class= "col-sm-2">` element.

What I really want to stress in this assignment is that making responsive, mobile-friendly designs is as simple as importing Bootstrap and googling how to do a specific thing like “Bootstrap columns.” (Or by inspecting the HTML of other websites with good design.)

4.2. Backend

To create the backend, I started from the original Python Flask code for TigerMenus. I stripped it down, removing all non-essential features in order to make the code as easy as possible to jump into. Then I adapted the code to fit its new specifications of eating club menus.

One thing I want to show students is just how simple the structure of a Flask web application is. You can create a web app with one line: `app = Flask(__name__)`. You can run a Flask application web server with one line: `app.run()` You simply define functions that correspond to website urls that return HTML code when a user accesses that url like:

```
@app.route('/hello')
def hello():
  return '<p>Hello World!</p>'
```

After writing all the scaffolding code, it made sense for the assignment to be split into these parts:

1. a `getButler()` function
2. some HTML templating code
3. a `getColonial()` function
4. a `scrapeCharter.py` script

4.3. getButler()

The getButler() function is an exercise in using the Python requests library to query a web api. The reference solution looks like this:

```
def getButler(m, d, y):
  """Get Butler menus for a specific date from the TigerMenus API

  :param int m: month
  :param int d: day
  :param int y: year

  :return: lunch and dinner lists for a specific day
  :rtype: tuple(list, list)
  """
  butlerLunch = []
  butlerDinner = []

  url = "https://tigermenus.herokuapp.com/api/{}/{}/{}".format(m, d, y)

  # STUDENT CODE BEGIN ( Ref < 10 lines)








  # STUDENT CODE END

  return butlerLunch, butlerDinner
```

Students will need to figure out the structure of a poorly documented web api (the TigerMenus API from page 7), learn how to use the requests module, and do basic list processing.

Students can test their implementation against the reference using the `test.py` script:

```
(venv) $ python test.py
Incorrect Butler 2017-09-13 lunch

Reference:                                       Yours:

Lunch
-- Main Entree --
Roast Beef with Red Wine Demi-Glace
-- Soups --
Italian Wedding Soup
Split Pea Soup
-- Grill Special --
Fish Taco Grill
-- On the Side --
Roasted Yukon Gold Potatoes
Stewed Green Beans
```

4.4. HTML Templating in app/templates/index.html

This is just an exercise in HTML Templating, teaching students how to generate HTML from Python lists. The reference solution looks like this:

```
  {% for item in list[1] %}

    <b><p>{{ item }}</p></b>
    {% else %}
    <p>{{ item }}</p>
    {% endif %}
  {% endfor %}
```

where each item is a string and if it starts with '-' make it bold. Inside the braces is basically just Python. When students re-open the site, they will see the dish categories bolded while all the regular food items are not.

4.5. getColonial()

This function is an exercise in extracting data from the pre-provided `colonial.py` file, which was shown in part on page 6.

The reference solution looks like this:

```
def getColonial(dateString):
  """Get Colonial menus for a specific date from the colonial.py file

  :param str dateString: a string in a format like 2017-09-11

  :return: lunch and dinner lists for a specific day
  :rtype: tuple(list, list)
  """
  colonialLunch = []
  colonialDinner = []

  c = [item for item in colonial if item['date'] == dateString]

  # STUDENT CODE BEGIN (Ref < 20 lines)











  # STUDENT CODE END

  return colonialLunch, colonialDinner
```

Students can test their implementation against the reference using the `test.py` script:

```
(venv) $ python test.py
Incorrect Colonial 2017-09-13 lunch

Reference:                                                     Yours:

Lunch
-- Hot Line --
Vegan Quinoa Bowls with corn, avocado, beans, pico de gallo.
raviolis with caper-tomato/ Veggie Medley
lemon-rosemary grilled chicken
French fries/ sweet potato chips
Lamb gyros / Falafel gyros
```

4.6. scrapeCharter()

This function is an exercise in learning how to use the beautifulSoup library to do webscraping.

```
def scrape(name, date):
  """Scrape a charter HTML webpage

  :param str name: a string containing a webpage filename
  :param datetime date: a datetime object representing the week’s Monday

  :return: lunch and dinner dictionaries mapping strings in the form
          "2017-09-11" to lists of food item strings
  :rtype: tuple(dict, dict)
  """
  lunch = {}
  dinner = {}

  for i in range(7):
      temp = date + timedelta(days=i)
      d = temp.strftime("%Y-%m-%d")
      lunch[d] = []
      dinner[d] = []

  # STUDENT CODE BEGIN (Ref < 30 lines)





  # STUDENT CODE END
  return lunch, dinner
```

After students read the BeautifulSoup documentation, implementing this should be fairly straightforward.

There are two things students will discover when trying to scrape:

1. There is actually duplicated data in the HTML itself that is hidden.

So students will have to be a bit more specific when scraping lest they get this when running `test_scrape.py`:

```
Incorrect Charter 2017-09-15 dinner
Reference:                     Yours:

Chef Tom's Gourmet Pizza       Entrees:
Boneless Wings                 Chef Tom's Gourmet Pizza
Sweet Potato Fries             Boneless Wings
Giant Cookies                  Sweet Potato Fries
                               Giant Cookies
                               Chef Tom's Gourmet Pizza
                               Boneless Wings
                               Sweet Potato Fries
                               Giant Cookies
```

The 2nd thing they will notice is that the required output strips streams of periods and commas, so students will need to do some regular expression splitting or else they will get this:

```
Incorrect Charter 2017-09-17 lunch
Reference:

Sausage, Ham, Muffins, Croissants
Danish, Bagels, Fresh Cut Fruit, Smoked Salmon
Cheese Blintzes with Warm Apple Compote

Yours:

Sausage,,,,Ham....Muffins....Croissants
Danish....Bagels...Fresh Cut Fruit....Smoked Salmon
Cheese Blintzes with Warm Apple Compote
```

I do this because when there are no spaces in a string, HTML can't split it across multiple lines. So columns will run into each other.

After three revisions and three people test running the assignment, this is the final assignment specification. I created YouTube helper videos that explain things already mentioned in this paper:

5. Evaluation

I had three people try out the assignment and give feedback.

To expedite the process, they attempted the assignment while I was present, to answer any questions they might have since the assignment specification was still very much under construction during each assignment run through.

5.1. ELE '19

The first one is a current student in COS 333. He completed the Butler and Colonial functions without much trouble in one hour. He got stuck on the scraping parts after another hour.

This prompted me to create the scraping helper videos.

Feedback was:

It's a fun assignment! I really hope it influences / becomes part of the curriculum.

You've done a pretty good already; it's literally barely missing anything.

5.2. ORF '19

The second student was an ORF major with no particular web development background.

His performance was similar to the above student. He got stuck with some environment set up issues, which prompted me to make the first general helper video.

He managed to complete the basic scraping portion of the assignment, without accounting for duplicated and comma separated data.

He spent three hours on the assignment.

He found it difficult to compare his output to the reference, which prompted me to create the testing scripts.

His feedback was:

So all in all it was cool to learn how to make an app that i use on a daily basis.

It was interesting to see how all the components of the site fit together and to play with the data structures involved with scraping because I had no previous software engineering experience.

5.3. ECO '20

The final student was a ECO major with prior experience in R but no formal COS classes.

Managed to complete the Butler function after a while, and is excited to complete the rest.

Thought everything was hard due to lack of experience.

Feedback was:

Speaking from the perspective of someone with almost non-existent coding background, attempting this assignment has sent me on a wild goose chase on Linux, Ubuntu, bash shell, Python, Notepad++, API, json, and many more concepts.

Those concepts were a lot to unpack; however, despite not being able to grasp many of those, I still found the assignment intellectually rewarding in terms of its practicality.

Alex's instructions were clear and helpful - the videos definitely helped! Alex was also very patient with my questions - I could tell he incorporated every feedback into developing this assignment.

Granted, Tiger Menus, to some, might be a simple web application and a simple assignment - however, I'm sure that for many this will be a valuable starting point!.

6. Summary

6.1. Conclusions

Overall, after three iterations of assignments, the assignment specification is finally complete and students seem to like it!

I learned a lot about crafting a well defined assignment specification from all my feedback.

More testing wouldn't hurt though.

But I think I have succeeded in creating a simple and fun assignment to teach web development in the context of cloning a current student app.

The assignment seems to take sufficiently technical students only a few hours to complete which is good. If this is actually incorporated in COS 333, it won't be too much of a burden on students between 5 prior assignments and the class final project.

At the very least, it can serve as a template for students to build their own more complicated Flask web applications.

6.2. Limitations

I should advertise this to people outside of COS 333 since you shouldn't have to take a COS departmental to learn about web development.

6.3. Future Work

I could add a section on CAS user authentication and database modeling.

The latest version of the code, report, and assignment specification will be updated at:

https://github.com/axu2/tiger-menus-lite

7. Acknowledgements

I'd like to thank:

Bilal Mukadam ELE '19, Jason Fu ORF '19, and Linh Tran ECO '20 for test running my assignment.

Jason Fu ORF '19, Johann Frias COS '19, Mario Liu MAE '19, Bensu Sicim COS '19 for reading my paper and presentation and providing feedback.

My adviser David Walker and my independent work seminar for their valuable feedback during this process.

And Alice Wang '19 for designing the TigerMenus logo!

Figure 5: Fin