CS 530: Developing User Interfaces

Assignment 1

Goals

This assignment asks you to build simple but realistic web site for "Drexel Bikes", a fictional organization (derived from Drexel Bike Share) in which a student can view a catalog of bikes and rent a bike. The assignment allows you to practice several aspects of full-stack web development, including front-end coding using HTML, CSS, JavaScript, and jQuery, and back-end coding using Flask and Python.

Assignment

The implementation of our "Drexel Bikes" web site uses Flask as the back-end web server. In Assignment 0, you set up your local machine to ensure that Python and Flask are correctly installed on your system. In this assignment, we will build on a skeleton site and flesh out several pieces of our desired web site.

To begin, please download the ZIP package of files attached to this assignment. The unpacked top-level directory **a1** should be put within the **cs530** directory from Assignment 0; this way, this assignment can share the course-level libraries (e.g., jQuery, Bootstrap) that we are using throughout the course.

The current **a1** directory contains the basic "Goat Pasture" web site we used as an example in class. You need to modify this site to reflect the specifications below.

Implementation Setup

As we saw in Assignment 0, you can start the Flask web server by running the **run.py** file in the **a1** directory:

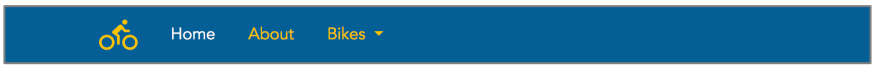
```
> python3 run.py
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 ...
```

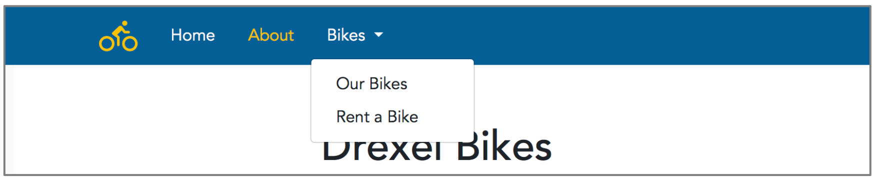
Again, this command starts a Flask web server on a local address and port number (http://127.0.0.1:8080/), such that you can point any web browser to this address and you will see the current version of your web site. In general, Flask allows for changes to the component files without restarting the server. However, if you generate a severe Python error from which the server cannot restart, you may need to manually restart the server by running the same command again.

To get you started, your downloaded package contains two template files in the **templates** directory. The **base.html** template is the base template that will be shared by all pages across the web site. Here, you should add any content that should appear on all pages: for example, the page header (<head>) that loads jQuery and Bootstrap, and the main navigation bar. The **index.html** template includes the content specific to the home (index) page of the web site. Please see the lecture notes for more details about these templates. You will be modifying **index.html** and creating additional pages that use **base.html** as the base template.

Navigation Bar

Your first task is to create the navigation bar needed for the site. The first image below shows the desired navigation bar, and the second image shows the navigation bar with the "Bikes" menu when expanded.



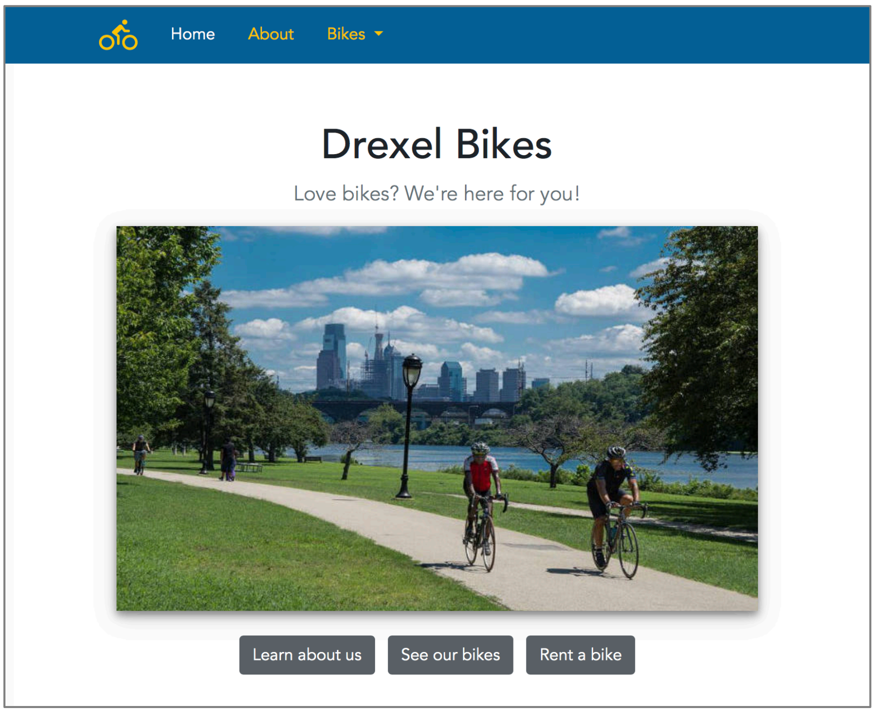


You can start with the given navigation bar and modify it to look like the desired nav bar above. The bike icon and all other images used in this assignment are provided for you in the **a1/public/img** directory.

We will use official Drexel colors to style our navigation bar, which are provided here: https://drexel.edu/identity/drexel/color/ . The background is the dark complementary blue (PMS 7691C), and the foreground text is the primary yellow (PMS 7548C). To get the hex version of these colors, you can use the developer tools on your browser to examine this page and extract the necessary colors. Using these colors, you should style the navigation bar by editing the file **a1/public/css/main.css** and adding CSS specifications that change the header bar. (Please do NOT style it by adding specifications such as a "style" attribute to your HTML elements; this approach does not generalize well.)

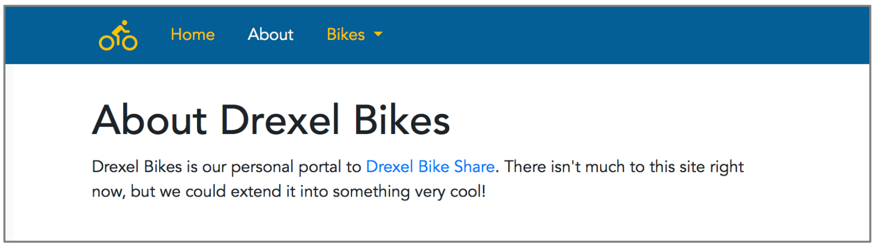
Home Page

You can now create the home page's main content area as a Bootstrap jumbotron, just like the examples seen in class. The page should display a jumbotron with the title "Drexel Bikes", subtext "Love bikes? We're here for you!", a main image, and three buttons ("Learn about us" which links to the "About" page, "See our bikes" which links to the "Our Bikes" page, and "Rent a bike" which links to the "Rent a Bike" page). Note that the main image is provided for you in the **a1/public/img** directory. The resulting page should adjust the size of the image responsively based on the window size; you should get this "for free" by simply using the given template.



"About" Page

The about page simply displays basic HTML information about the site. The text to display appears below. Note that the text includes a link to the Drexel Bike Share site: https://drexel.edu/campusservices/studentCenters/creeseCenter/BikeShare/

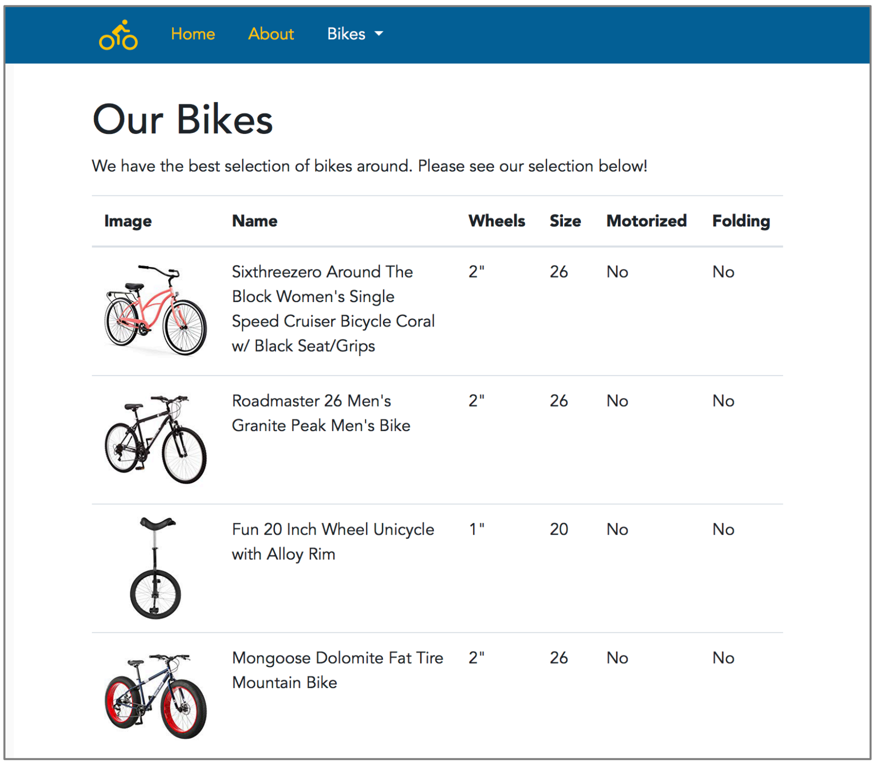


About Drexel Bikes

Drexel Bikes is our personal portal to Drexel Bike Share. There isn't much to this site right now, but we could extend it into something very cool!

"Our Bikes" Page

The next page to develop, "Our Bikes", uses Flask templates and dynamic data to populate its contents. The end result should look as follows:



Our Bikes

We have the best selection of bikes around. Please see our selection below!

Image	Name	Wheels	Size	Motorized	Folding
	Sixthreezero Around The Block Women's Single Speed Cruiser Bicycle Coral w/ Black Seat/Grips	2"	26	No	No
	Roadmaster 26 Men's Granite Peak Men's Bike	2"	26	No	No
	Fun 20 Inch Wheel Unicycle with Alloy Rim	1 "	20	No	No
	Mongoose Dolomite Fat Tire Mountain Bike	2"	26	No	No

To start this page, create a template **bikes.html** with the header and text content you see above. Then, create a table with a header row containing the header cells ("Image", "Name", etc.). This table should include Bootstrap's **table** class (i.e., <table class="table">) to get Bootstrap's nice styling of tables.

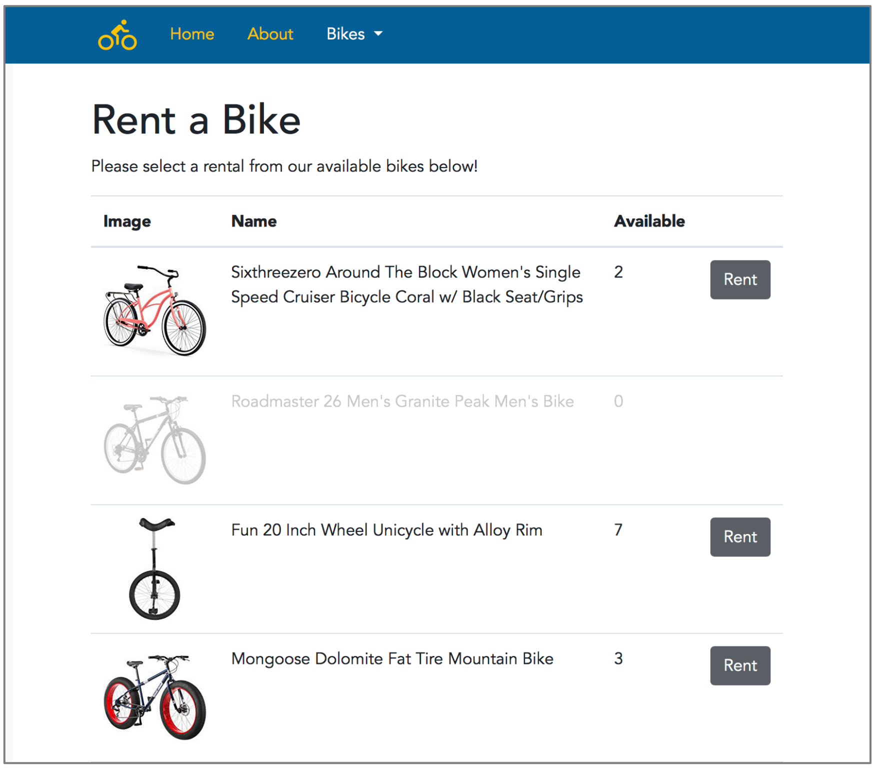
The body of the table (<tbody>) should NOT be included in the HTML template. Instead, you should use Flask's template system to generate the body from data. First, you should modify **run.py** to include the data, namely by copying the contents of **data.txt** into **run.py**. Next, you should create a new function within **run.py** that renders a template for the route "/bikes"; the call to **render_template()** should have an additional parameter that passes the data to the template. Finally, you should use the passed data to dynamically generate the table shown above, using the "{% for … %}" pattern within the template

As one final step, note that the images cannot be rendered at their full size, but instead should be limited in size. To accomplish this, as usual, you should modify the **main.css** file such that images within the table have a maximum width of 100px.

To complete this page, you will likely find that you need to refer to the documentation pages for Bootstrap, jQuery, and/or Flask. That's ok! In fact, this is good practice for typical full-stack development, where you will often be bouncing between languages and frameworks as you determine the best way to pass data between the front end and back end.

"Rent a Bike" Page

The final page, "Rent a Bike", generates dynamic content in a different way, namely using JavaScript to populate the page table and handle events on the page. The end result should look as follows (only the top part of the page is shown):



Rent a Bike

Please select a rental from our available bikes below!

Image	Name	Available	
	Sixthreezero Around The Block Women's Single Speed Cruiser Bicycle Coral w/ Black Seat/Grips	2	Rent
	Roadmaster 26 Men's Granite Peak Men's Bike	0	
	Fun 20 Inch Wheel Unicycle with Alloy Rim	7	Rent
	Mongoose Dolomite Fat Tire Mountain Bike	3	Rent

Create a template **rent.html** that contains the basic header and content above the table, then add a table header and empty body like you did for the "Our Bikes" page. For this page, however, you should NOT make any additions to **run.py**. Instead, you should add a <script> element to your **rent.html** file and, within this element, add JavaScript to generate the table and handle events.

To build the table, again copy the data from **data.txt** and use it to populate a JavaScript variable. Then, write code that iterates over each bike and, using jQuery and JavaScript, adds table rows with the content shown above: bike image, bike name, number available, and a "Rent" button. If the availability is 0, you should add a class **unavailable** to the entire table row, and then in your **main.css** file, give this class an opacity of 0.25; you should also skip adding the "Rent" button, since this function would be disabled.

When the "Rent" button is clicked, the code should decrement that bike's availability by 1. You can use jQuery to access this element and then change it to the updated number. Once the availability goes down to 0, you should add the **unavailable** row class and remove the "Rent" button as described above. (For a real web site, the system would need to communicate these changes to a back-end database, but we are ignoring such issues in this assignment.)

Documentation

It is expected that all code written for this assignment is properly commented where needed, especially in places where you have made particular choices about data structures and/or algorithms to employ. Also, please add the following identification header to every file you create:

```
    # Your Name, Your Email
    # CS530: DUI, Assignment [#]
```

(or whatever commenting syntax is appropriate for the file at hand).

Submission

Please submit your files as an attachment for the assignment on Blackboard Learn. Please use a compression utility to compress your files into a single ZIP file (NOT RAR, nor any other compression format). The final ZIP file must be submitted electronically using Blackboard—do not email your assignment to a TA or instructor! If you are having difficulty with your Blackboard account, you are responsible for resolving these problems with a TA or someone from IRT before the assignment it due. If you have any doubts, complete your work early so that someone can help you.