IOLab HW4: Web Frameworks, Python, Flask

Due: Wednesday, November 27, 12pm

Part A: As we learned in 202, anything can be considered a resource. On the web, this abstraction is a guiding principle behind REST APIs. From Wikipedia:

> An important concept in REST is the existence of **resources** (sources of specific information), each of which is referenced with **a global identifier** (e.g., a URI in HTTP). In order to manipulate these resources, components of the network (user agents and origin servers) communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources (the actual documents conveying the information).
>
> Any number of connectors (e.g., clients, servers, caches, tunnels, etc.) can mediate the request, but each does so without "seeing past" its own request …Thus, **an application can interact with a resource by knowing two things: the identifier of the resource and the action required**—it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML, or JSON document of some kind, although it may be an image, plain text, or any other content.

Imagine you are building a web-based organizing system according to this principle. Consider:

1. What resource(s) would your system organize?
2. What interactions would the system support?
3. What routes (URL paths and HTTP methods) correspond to each of these actions?

Deliverable: Create a table that outlines the structure of resources in your app. For example, if I was building a recipe organizing system, I might have the following table:

Path	Method	Parameters	Returns	Description
/	GET	None	HTML	Goes to the home page
/recipes	GET	keyword	JSON	Returns list of recipes, optionally filtered by keyword
/recipe/<id>	GET	None	HTML	Detail page for a single recipe
/recipes	POST	title, ingredients, steps, difficulty, image_file	JSON	Creates a new recipe. Returns JSON success message.
/recipe/<id>	DELETE	None	JSON	Deletes the recipe with the specified id. Returns JSON success message.

Send us the table as a PDF or Word doc.

Part B: Build a back end for the to-do list you created for HW 1 and 2. See here for instructions: https://github.com/krushton/iolab-homework4

Deliverable: Send us a zip file of your Flask app directory.