# **SI 364: Midterm assignment**

**Total points:** 1500

**Deadline:** October 29, 2017 at 11:59 PM

Your task for the midterm assignment is to write a Flask application.  
  
You should create a Git and a GitHub repository for it and push all the files and directories required for it to the GitHub repository you choose.

You should then submit the link to the GitHub repository to the Midterm Assignment assignment on Canvas.  
  
**Below, there is a list of requirements. For full credit on this assignment, you must:**

* Fulfill each of these requirements
  + (larger ones, like "do you have at least 3 routes that work", will be more points than smaller ones, like "did you set a cookie in any one of the routes", but everything is worth at least some points)
* NOT make a direct copy of anything we have done in class (though it is OK if it is similar or uses the same data sources),
  + And not use any route names or direct template copies of material we have used in lecture or section
  + You may use functions or view methods you have used in previous HW that you came up with *on your own* in a HW Part 2
* Submit code for a Flask application that runs (no points will be given for code that does not run, so test it out)
* Start from scratch (we are not giving you any starter files for the first time -- this application is all yours!)
* In the Canvas assignment, write at least 1 sentence about what your application is supposed to do, and **if we should not go to the main '/' route, TELL US WHAT ROUTE TO ACCESS in the text you submit.**

You may make your project as simple or as complex as you want -- but make sure it's relatively simple, because you should be able to complete this in only a week and a half.

You may discuss your plan with others and with instructors, but you may not work together on writing code and you may not use or copy code from anyone else to complete this assignment. (If you rely on complex code that others have written, e.g. an external library, that is fine.) You may use any code you've seen or written before as inspiration and may work on something quite similar, but you may not use code identical to code we have worked on in class or code you submitted for a free part of HW2 or HW3 (though you may reuse some of that, as long as you add to it or change it somewhat).

By submitting this assignment, you are asserting that you have complied with the academic honesty requirements laid out in the syllabus and for the University of Michigan at large, and have not shared your work with anyone else or used work directly from anyone else.

**Requirements for your Flask application:**

* At least 3 different routes.
  + At least one of your routes must employ a set\_cookie and use the make\_response function, so a cookie is set in that request.
    - (The cookie data can be whatever you want, including, as shown in the Grinberg book, {"answer":42} -- this is practice for later on.)
* At least 1 form, written with WTForms, with a correct/reasonable action and method
  + To which data can be entered and submitted
  + Submitting data via the form should cause a visible result -- e.g. entering a Pokemon name and then seeing a list of Pokemon whose names are similar to the Pokemon you searched (that'd be pretty complex, but doable)
* At least 2 dynamic links appearing inside template file(s) -- links to other page(s) within the application) that send data through the URL *(similar to links to info about e.g. Solange and Kendrick Lamar you saw in HW3 -- but for whatever data you want)*
* At least 1 image saved in a static/ directory and rendered on a page
* At least 3 different templates, rendered using render\_template, and included must be at least:
  + 1 Jinja templating language for loop (e.g. to show a list of things)
  + 1 Jinja templating conditional statement (e.g. to only show something in a certain case)
  + These 3 templates must *not* include custom error message templates (see below)
* At least 2 custom error message templates / error handlers, as you saw in section a few weeks ago
  + At least one should be a 404 page
  + The other can be to handle any type of error you like, but it must be a genuinely possible error code (consider errors 400, 500 -- check out the material you looked at in section for more!)

**Other tools that are not required which you may want to consider using:**

* Redirects
* The **url\_for** function
* Template extensions

**Strongly recommended:**

* Look through the code we've done in past sections, lectures, and HW, and write out in English what each of your routes will do.
* Print out this page, or make a copy of it, and check off each bullet-point as you decide where you're going to put that requirement in your code -- and again once you've written the code and know that it runs!