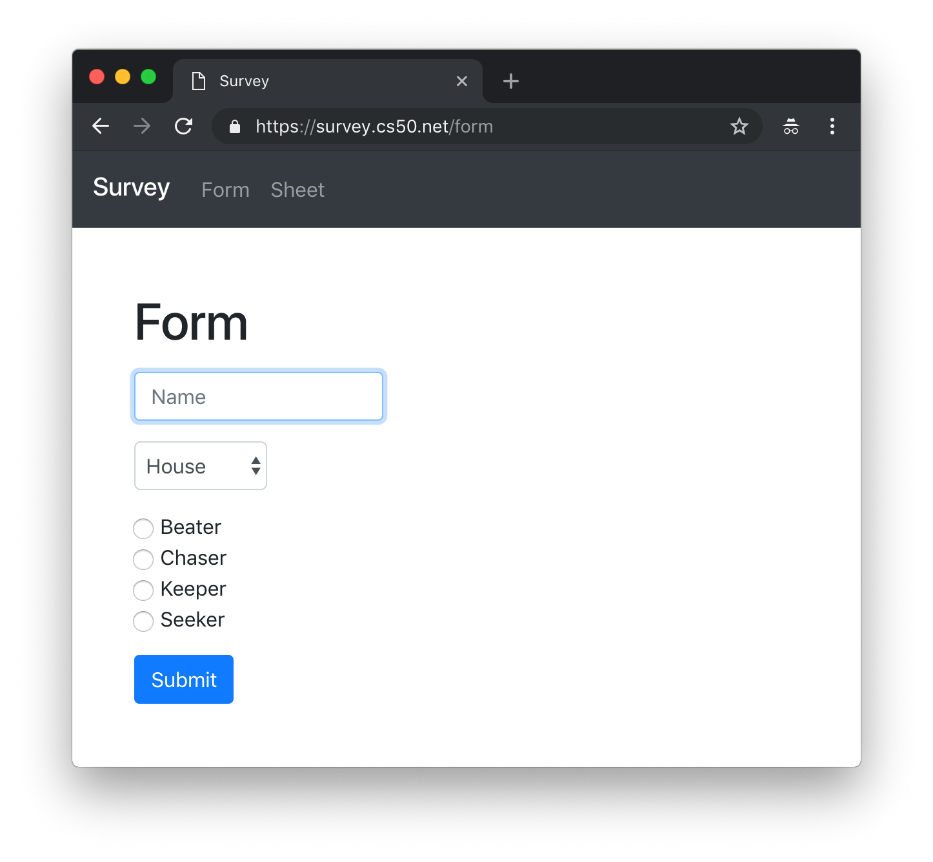
Survey

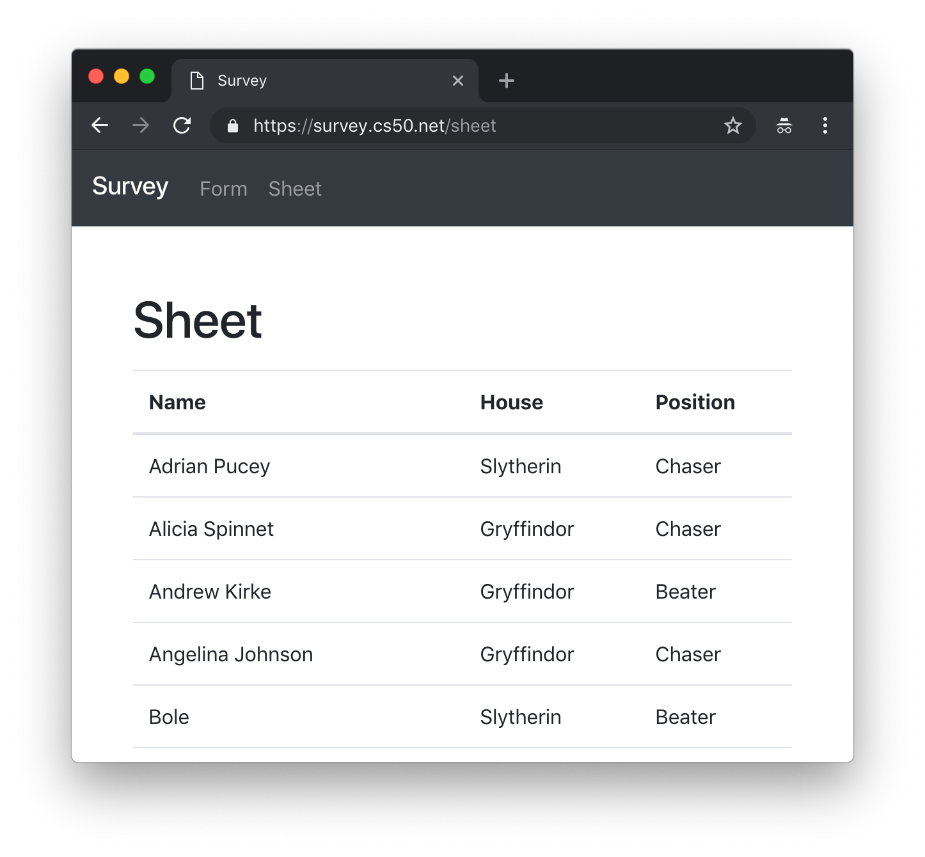
[tl;dr](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#tldr)

Implement a web app that enables a user to

* fill out a form, a la Google Forms, the results of which are saved to a comma-separated-value (CSV) file on the server, and
* view a table of all of the submissions received, a la Google Sheets,

a la the below.





[Introduction](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#introduction)

<https://youtu.be/0zIpv_dbaSQ>

[Getting Started](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#getting-started)

Here’s how to download this problem’s distribution into your own CS50 IDE. Log into [CS50 IDE](https://ide.cs50.io/) and then, in a terminal window, execute each of the below.

1. Execute cd to ensure that you’re in ~/ (aka your home folder).
2. Execute mkdir pset7 to make (i.e., create) a directory called pset7 in your ~/ directory, if you haven’t already done so.
3. Execute cd pset7 to change into (i.e., open) that directory.
4. Execute wget <https://cdn.cs50.net/2018/fall/psets/7/survey/survey.zip> to download a (compressed) ZIP file with this problem’s distribution.
5. Execute unzip survey.zip to uncompress that file.
6. Execute rm survey.zip followed by yes or y to delete that ZIP file.
7. Execute ls. You should see a directory called survey, which was inside of that ZIP file.
8. Execute cd survey to change into that directory.
9. Execute ls. You should see this problem’s distribution inside.

[Understanding](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#understanding)

[application.py](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#application-py)

This file represents your web app’s "controller," all of the logic that implements its functionality. Atop the file are a few imports of libraries followed by some configuration of Flask, per the comments therein. Below that are declarations of four routes, two of which are for you to do!

[templates/layout.html](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#templateslayout-html)

This file represents your web app’s layout, an HTML structure that all of your views will share.

[templates/form.html](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#templatesform-html)

In this file will live your very own form, only the skeleton of which we’ve written for you.

[templates/error.html](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#templateserror-html)

In this file is a template for any messages you might like to display to the user in the event of some error.

[static/styles.css](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#staticstyles-css)

In this file will be any of your own CSS properties for any or all of your app’s views.

[Specification](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#specification)

* Complete the implementation of templates/form.html in such a way that the form therein contains not only a button but also
  + one or more text fields of any type,
  + one or more checkboxes or two or more radio buttons,
  + one or more select menus, each with two or more options, and
  + zero or more other inputs of any type.

Style that form using [Bootstrap](http://getbootstrap.com/docs/4.1/components/forms/) so that it looks nicer than it would with raw HTML alone.

Add to that file some JavaScript code that prevents users from submitting the form if they have not provided values for one or more fields, alerting the user accordingly, as via [alert](https://www.w3schools.com/jsref/met_win_alert.asp) or [Bootstrap](http://getbootstrap.com/docs/4.1/components/forms/#validation).

* Complete the implementation of post\_form in such a way that it
  + validates a form’s submission, alerting users with a message via error.html if they have not provided values for one or more fields, just in case your JavaScript code let something through (or was disabled),
  + writes the form’s values to a new row in survey.csv using csv.writer or csv.DictWriter, and
  + redirects the user to /sheet.
* Complete the implementation of get\_sheet in such a way that it
  + reads past submissions from survey.csv using csv.reader or csv.DictReader and
  + displays those submissions in an HTML table via a new template.

Style that table using [Bootstrap](http://getbootstrap.com/docs/4.1/content/tables/) so that it looks nicer than it would with raw HTML alone.

Optionally enhance the table with JavaScript, as via [DataTables](https://datatables.net/examples/styling/bootstrap4).

Provided you meet these specifications, you’re welcome to alter the aesthetics of your app however you’d like, as via [Bootstrap](http://getbootstrap.com/docs/4.1/) or your own CSS and HTML.

[References](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#references)

* [Bootstrap](http://getbootstrap.com/docs/4.1/)
  + [Forms](https://getbootstrap.com/docs/4.1/components/forms/)
  + [Tables](https://getbootstrap.com/docs/4.1/content/tables/)
* [csv](https://docs.python.org/3/library/csv.html)
  + [csv.writer](https://docs.python.org/3/library/csv.html#csv.writer)
  + [csv.DictWriter](https://docs.python.org/3/library/csv.html#csv.DictWriter)
* [DataTables](https://datatables.net/examples/styling/bootstrap4)
  + [Options](https://datatables.net/reference/option/)
* [HTML Forms](https://www.w3schools.com/html/html_forms.asp)
  + [HTML Form Elements](https://www.w3schools.com/html/html_form_elements.asp)
  + [HTML Input Types](https://www.w3schools.com/html/html_form_input_types.asp)
  + [HTML Input Attributes](https://www.w3schools.com/html/html_form_attributes.asp)

[Testing](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#testing)

[Correctness](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#correctness)

Afraid there’s no check50 for this problem; it’s incumbent upon you to write and test your code using the testing and debugging strategies we have discussed throughout the course. As in past problems where you have not had access to check50, know that your correctness score on this problem will be based on whether you meet the requirements of the specification as outlined above, whether your code is free of bugs, and whether your HTML is well-formed and valid. To ensure that your pages are, you can use [the W3Schools HTML Validator](https://validator.w3.org/#validate_by_input) service, copying and pasting your HTML directly into the provided text box. Take care to eliminate any warnings or errors suggested by the validator before submitting!

[Style](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#style)

style50 application.py

Afraid style50 does not support HTML files, and so it is incumbent upon you to indent and align your HTML tags cleanly, as the per the examples shown in prior weeks' lectures. Know also that you can create an HTML comment with:

<!-- Comment goes here -->

but commenting your HTML code is not as imperative as it is when commenting code in, say, C, Python, or JavaScript.

[Staff’s Solution](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#staffs-solution)

<https://survey.cs50.net/>

It is reasonable to view its HTML and CSS.

[How to Submit](https://docs.cs50.net/2019/x/psets/7/survey/survey.html#how-to-submit)

Execute the below from within your ~/pset7/survey directory, logging in with your GitHub username and password when prompted. For security, you’ll see asterisks (\*) instead of the actual characters in your password.

submit50 cs50/problems/2019/x/survey