# FC712 Jan 21 Project

Mingye Wang

I hereby declare that all code in this project is my own original work, unless specifically marked.

(This document is converted by pandoc using the mergedoc.sh here.)

# Course webform

This directory holds task1 of the project.

## The structure

There is no heavy use of any algorithm in this project. The layout is, in fact, basically identical to the sample project, with the fields simply changed. In this case I decide to use the Flask-Bootstrap4 package just to be lazy.

I originally thought about collecting multiple courses at once, but doing so sounds more complicated and will make the data output less flexible.

As with the sample, the project only has one entry point in the form of the / handler. In case of a GET, the template is rendered without anything provided, while in the POST case validation and file writing is done.

A CSV format is used for easier machine manipulation such as in Microsoft Excel. Note that the Unix dialect of the CSV file format is used due to my familiarity with it, so I guess it’s more of “for easier manipulation by R and whatever statitians use these days” rather than in Excel.

## Testing

Testing is manually done with the idea of maximizing coverage – in the code I have written. In other words, the following cases are tested:

* Plain GET.
* POST with invalid data.
* POST with valid data.

In each case, the “smoke test” is first conducted by inspecting the browser output. If they look fine, I move on to checking the CSV.

## Further work

It would be really nice to add an Idempotency-Key, so that refreshes do not result in a resubmit. The package [flask-idempotent](https://pypi.org/project/Flask-Idempotent/) provides an almost ideal solution, as it does not require an additional HTTP header. However, its dependency on Redis means I cannot really trust the testing environment to do the job. Cooking up an OrderedDict-based datastore and capping it onto the main function as a “caching” decorator is interesting but possibly a bad idea given the deadline.

## References

1. *Flask-Bootstrap4 Documentation* (c. 2018) <https://flask-bootstrap4.readthedocs.io/en/latest/>)
2. Portions of the Flask-Boostrap4 source code, for figuring out what the template thing is all about.

# TODO, not MVC, not even GUI

Mingye “Too tired to Microsoft Word” Wang, 2021.

Doing GUI is too much of a drain on my sanity.

I don’t see much point in doing MVC on CLI, since users are not going to want to have an automatic “repaint” anyways. Drawing with TUI might be less sanity-consuming than a Tkinter GUI, but then I’d have to ship ncurses.

Wasn’t ed(1) the “standard editor”? Yeah, let’s do that.

## The core

This project centers around the TODO list as defined in the model. We first define the Item as having four component: the UUID, the title, the body, and a boolean flag for whether it is checked.

The TODOList is then defined as containing a dict of Items mapping from the UUID (as an optimization assuming a fully-featured GUI), a list of action “hooks” again serving the now-abandoned MVC structure, and finally a string key specifying what file to load or save from.

As with project #1, the standard RFC CSV format (dialect='unix') is used for saving.

## Interaction

The CLI is a loop using input to read commands. For basic quoting ability, such as for writing titles with spaces (add "Reheat pizza at 3pm"), the builtin shlex library is used to parse the input into a list.

The design of the CLI is nowhere near good. Things that heavily impact usability include:

* Lack of tab completion. I know it’s somewhere in the builtin readline module.
* Awful help messages. Can be better written.
* No multi-named commands (e.g. abbreviating help as ?) due to the way help is implemented. Can be solved by using a separate ALIASES dictionary.
* search result are not highlighting matches.

Most commands are wrappers around what’s available in the TODOList object. The exceptions are:

* search, which performs a linear search;
* edit, which dumps an entry to a text file, calls the system’s text editor, and then reads back when done. Probably the only thing I am proud of here.

## References

1. *TodoMVC*, <https://todomvc.com/>. The original intention was to mimic it in Python, but I ended up only roughly taking the Model design.
2. *Python 3.9.6 documentation*, specifically the itertools module for the nth function, the re module for usage, and the tempfile module for a replacement of mktemp.