# (DRAFT) Learning Flask and SQLite by Building a Todo App

Author: Grant Achuzia (101222695)
Course: SYSC3010 - Computer Systems Development Project

## Table Of Contents



### Introduction

Throughout this tutorial, we will learn the basics of the Python Flask framework and the database engine SQLite by building a simple todo app. I find that the easiest way to learn certain programming concepts/tools is through making stuff!

Check out the github repository for the todo app here!

### Prerequisites

I'm going to assume you are familiar and comfortable programming in Python, SQL, HTML and CSS. The main goal of this project is to learn new tools (Flask and SQLite) with pre-established programming fundamentals.

### Setting Up Your Environment

In a directory of your choosing, make a project folder called `todo-app`, and create a Python virtual environment within that project folder where you can type the command `pip install Flask`.

```
C:directory/of/your/choosing/todo-app
C:directory/of/your/choosing/todo-app python -m venv venv
C:directory/of/your/choosing/todo-app/venv/Scripts activate
(venv) C:directory/of/your/choosing/todo-app pip install Flask
```

Our project will contain the libraries needed for the todo app within this venv, so don't deactivate it!

### Creating the Flask Application

In the `todo-app` we're going to make an `app.py`, `index.html`, and `index.css` files. The index.html file should be placed ina folder called `templates`, and the `index.css` file in a folder called `static`. Flask uses these folders for static and dynamic content, respectively.

```
todo-app/
│
├── app.py
├── templates/
│   └── index.html
└── static/
    └── index.css
```

The `app.py` file defines the web application we want to build. We import modules from the Flask framework:
`Flask`: Creates an instance of the web application.
`render_template`: Renders HTML templates which allows us to dynamically generate HTML.
`request`: An object that contians info about incoming requests (e.g. form data, query parameters). `url_for`: Generates URLs based on endpoint names in our applciation.
`redirect`: Redirects the user to a different endpoint or URL within the app.

The `database` module contains functions used to interact with our SQLite databse (more about that here).

#### Routes

The file has serval "routes" which represent different URL endpoints that use the `@app.route` decorator. We'll go over the home/index page route as an overview.

```
from flask import Flask, render_template, request, url_for, redirect
from database import connect_to_db, get_db

app = Flask(__name__)


@app.route("/", methods = ["POST", "GET"])
def index():
    db = get_db()
    tasks_getter = db.execute("select * from todolist")
    all_db_tasks = tasks_getter.fetchall()
    return render_template("index.html", all_db_tasks = all_db_tasks)
```

#### UI Connectivity

### Creating the SQLite Database

### Testing the App

### Conclusion

### Resources

Project Github repo: GAchuzia/yag-todo
Flask Docs: Quickstart
MDN Web Docs