I used this site as a source for where I learned how to use flask and apply it
https://stackabuse.com/building-a-todo-app-with-flask-in-python/

I first had to install. Curl. You can do it in a command line such as bash, command prompt, or windows powershell.

Then I had to create a virtual environment using the command
Py -m venv

After that I activated it using the command

source "c:/Users/Bishajit Lodh/venv/Scripts/activate"

Then I installed flask inside the virtual environment

```
pip install Flask
```

And then I created app.py and typed these commands into it. And I placed it in the venv directory which I added more to later

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'
```

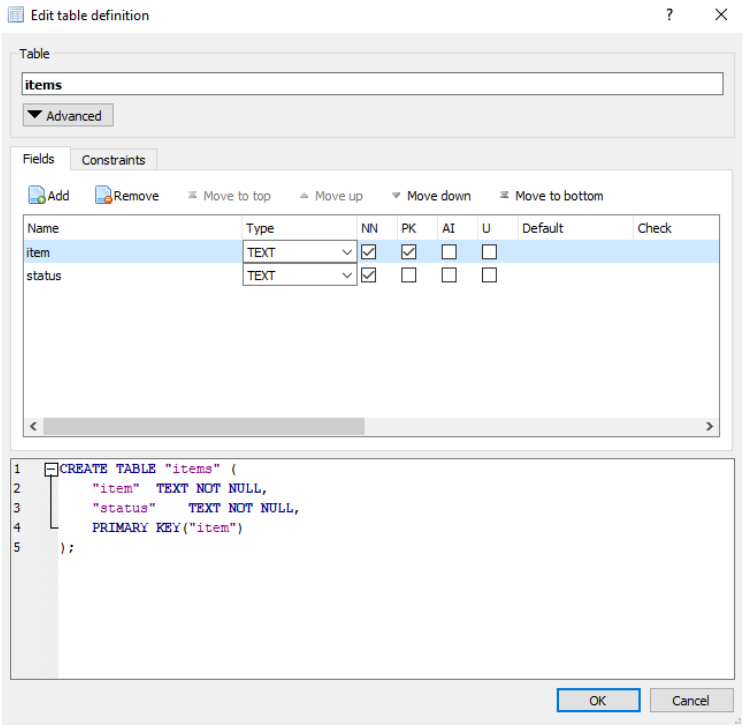
After that I created a

```
FLASK_APP=main.py flask run
```

```
Running on http://127.0.0.1:5000/
```

```
curl -X GET http://127.0.0.1:5000/
```

Created todo.db



Save it

You can Open Database path todo.db to see all the latter commands you use with curl.

After creating the database I created helper.py which continuously added as I went through the instructions

After I added all the functions and items the instruction told me to do these are the commands I ran.

'First command
curl -X POST http://127.0.0.1:5000/item/new -d "{\"item\": \"Setting up Flask\"}" -H 'Content-Type: application/json'

>Expected outcome: {"Setting up Flask": "Not Started"}

Add one item to the list

Second commands
curl -X POST http://127.0.0.1:5000/item/new -d '{"item": "Implement POST endpoint"}' -H 'Content-Type: application/json'

>Expected outcome: {"Implement POST endpoint": "Not Started"}

Adds a second one to the list

Third Command
curl -X GET http://127.0.0.1:5000/item/all

>Expected outcome: json {"count": 2, "items": [["Setting up Flask", "Not Started"], ["Implement POST endpoint", "Not Started"]]}

Gets all items

Fourth Command
curl -X GET http://127.0.0.1:5000/item/status?name=Setting+up+Flask

>Expected outcome: {"status": "Not Started"}

Fifth Command
curl -X PUT http://127.0.0.1:5000/item/update -d '{"item": "Setting up Flask", "status": "Completed"}' -H 'Content-Type: application/json'

>Expected outcome: {"Setting up Flask": "Completed"}

Sixth Command
curl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Setting up Flask"}' -H 'Content-Type: application/json'

>Expected outcome: {"item": " item that was deleted"}

7th Command

curl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Implement POST endpoint"}' -H 'Content-Type: application/json'

>Expected outcome: {"item": " item that was deleted"}

New functions I added to my to-do app are get_all_item and get_all_status. The function get_all_item does exactly as it says by listing all the item values. And the other function get_all_status gets all the status values.

```python
@app.route("/item/all_item")
def get_all_len():
    # Get items from the helper
    res_data = helper.get_all_item()

    # Return response
    response = Response(json.dumps(res_data), mimetype="application/json")
    return response

@app.route("/item/all_status")
def get_all_status():
    # Get items from the helper
    res_data = helper.get_all_status()

    # Return response
    response = Response(json.dumps(res_data), mimetype="application/json")
    return response
```