HANOI UNIVERSITY
Faculty of Information Technology

61FIT3SS1
Python Programming - Fall 2021
Final Project

For the final project, you will create a Flask web application to manage a todo list. Your app will have the following main features:

1. Create an item in the todo list
2. Show all item in the todo list
3. Update the items with status as "Done" or "Doing"
4. Delete the items from the list

Part 1. (5 points):

Create a Python Flask project for the todo app. At this point, the data is a list of dictionary (see Appendix 1 for an example of todo list data). Your project should have the main app.py file and a html template for the application interface (see Appendix 2 for an example of the html template). Your app should have four main features that are described above, for example:

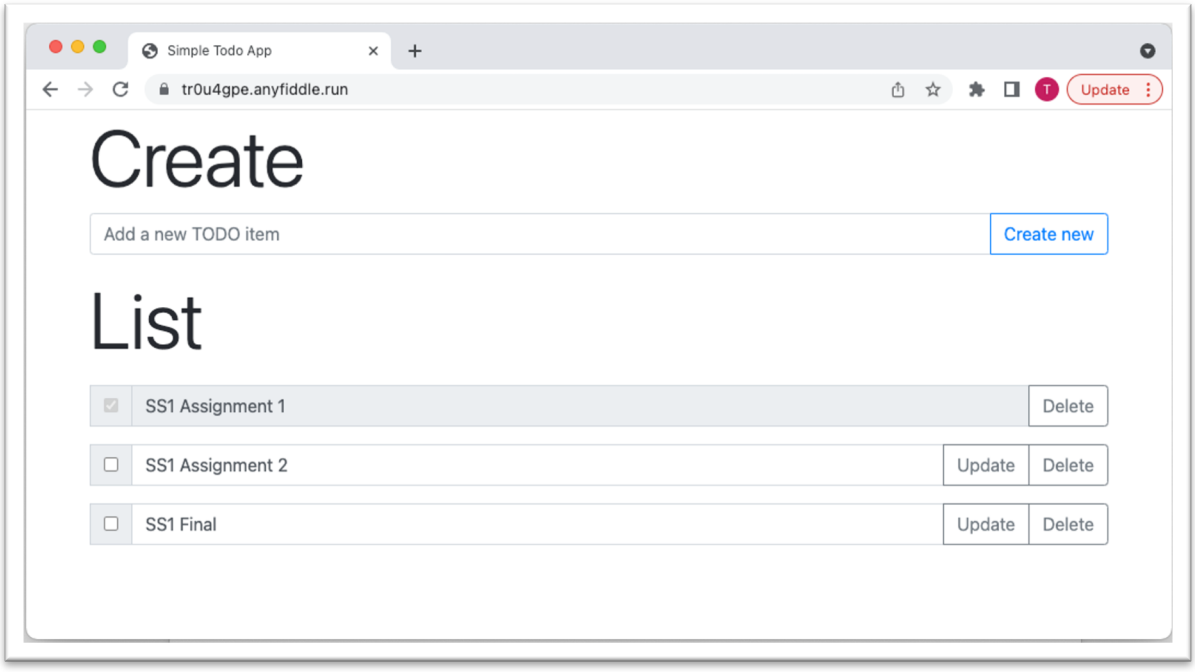

Figure 1: Todo list app web interface. List all items in the todo list. Each item has its description (an input text) and status (a checkbox). User can delete both "Done" (checked) or "Doing" (unchecked) items, but only "Doing" item can be updated

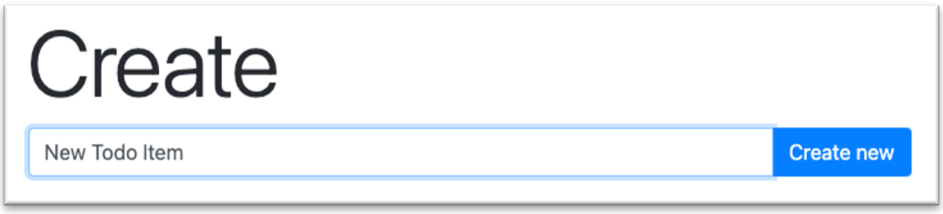

Figure 2: Create a new todo item. The status of the new item is "Doing"

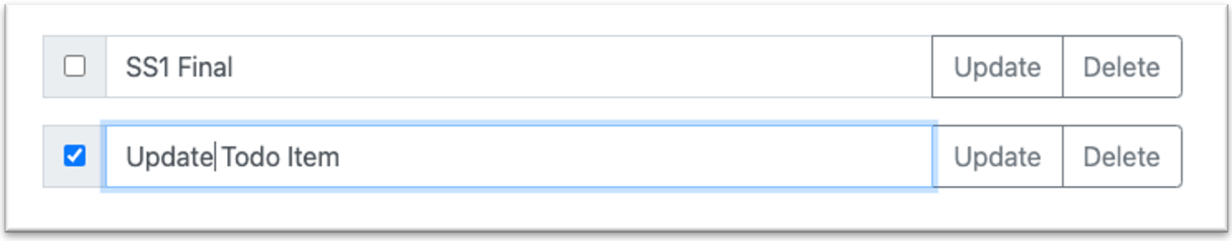

Figure 3: Update an item. User can change the item's description or tick the checkbox for changing the item's status from "Doing" (unchecked) to "Done" (checked)

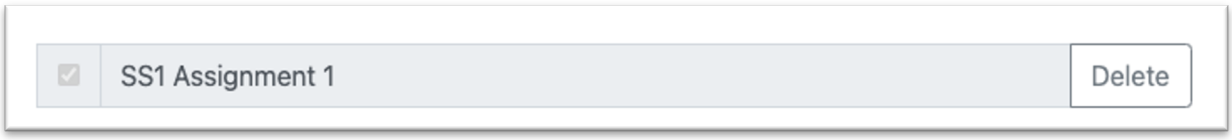

Figure 4: Delete an item

Part 2 (3 points):

Upgrade your project to use a database management system to store todo list data. This time, your data will be stored in a table (see Appendix 3 for an example of todo table). You can use SQLite or any other database management system supported by Flask.

Part 3 (2 points):

Upgrade your project to provide three REST API to manage the todo list. The API end point (URI) are:

- /api/list (GET method): returns all items in JSON format
- /api/add (POST method): add new item to the list
- /api/update?id=<Item ID> (GET method): update an item's status. If the current status is "Doing" then the new status will be "Done" and vice versa.
- /api/delete?id=<Item ID> (GET method): delete an item from the list

An example of a fronend app can be downloaded from:
https://drive.google.com/file/d/16cmmxTdNYGYk2RMX-HEh4JwjH5d-lphi/view?usp=sharing

Figure 5: An example of todo frontend app

Submission Instruction

Please create a new folder named 61fit3ss1_finalproject_<your student ID>. You have to copy the following files to the folder:

- part1.zip : contains your Flask project for Part 1
- part2.zip: contains your Flask project for Part 2
- part3.zip: contains your Flask project for Part 3

Please compress your folder into a .zip file and submit this file via MS Teams.

Appendix 1: An example of todo list data

```
todoList = [
    {'id' : 1, 'description' : 'SS1 Assignment 1', 'status' : 'Done'},
    {'id' : 2, 'description' : 'SS1 Assignment 2', 'status' : 'Doing'},
    {'id' : 3, 'description' : 'SS1 Final', 'status' : 'Doing'}
]
```

Appendix 2: An example html template

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Todo App</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
        <h2 class="display-3">Create</h2>
        <form method="POST" action="/add">
            <div class="input-group mb-3">
                <input class="form-control" type="text" name="itemDescription" placeholder="Add a new TODO item">
                <div class="input-group-append" id="button-addon">
                    <button class="btn btn-outline-primary" type="submit" value="update">Create new</button>
                </div>
            </div>
        </form>
        <h2 class="display-3">List</h2>
        {% for item in todoList: %}
            <p>
                <form method="POST" action="/edit">
                    <div class="input-group mb-3">
                      <input type="hidden" name="itemID" value="{{item['id']}}">
                      {% if item['status'] == "Done": %}
                        <div class="input-group-prepend">
                            <div class="input-group-text">
                                <input type="checkbox" name="itemStatus" checked disabled>
                            </div>
                        </div>
                            <input class="form-control" type="text" name="itemDescription"
value="{{item['description']}}" readonly>
                        <div class="input-group-append" id="button-addon2">
                            <button class="btn btn-outline-secondary" type="submit" name="clickBtn"
value="delete">Delete</button>
                        </div>
                          {% else %}
```

```
                        <div class="input-group-prepend">
                            <div class="input-group-text">
                                <input type="checkbox" name="itemStatus" value="{{item['status']}}">
                            </div>
                        </div>
                            <input class="form-control" type="text" name="itemDescription"
value="{{item['description']}}">
                        <div class="input-group-append" id="button-addon2">
                            <button class="btn btn-outline-secondary" type="submit" name="clickBtn"
value="update">Update</button>
                            <button class="btn btn-outline-secondary" type="submit" name="clickBtn"
value="delete">Delete</button>
                        </div>
                          {% endif %}
                    </div>
                </form>
            </p>
        {% endfor %}
    </div>
  </body>
</html>
```

Appendix 3: An example of todo table

```
TODO table:
    - id : INTEGER PRIMARY KEY
    - description: TEXT NOT NULL
    - status: TEXT NOT NULL

CREATE TABLE todo (id INTEGER PRIMARY KEY, description TEXT NOT NULL, status TEXT
NOT NULL)

INSERT INTO todo (description, status) VALUES ('SS1 Assignment 1', 'Done')
INSERT INTO todo (description, status) VALUES ('SS1 Assignment 2', 'Doing')
INSERT INTO todo (description, status) VALUES ('SS1 Final', 'Doing')
```