CISC 327 GROUP-51 readme first

How to run payment feature program and test cases (Sungmoon Choi):

- Change the directory to Payment directory inside Assignment-2

```
C:\Stuff>cd C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment

C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>
```

- Create a virtual environment (this command is using Windows, the command is different depending on the operating system)

```
C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>virtualenv venv
created virtual environment CPython3.12.1.final.0-64 in 183ms
  creator CPython3Windows(dest=C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment\venv, clear=False
, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\Sungpoon\AppData\Local\pypa\virtualenv)

    added seed packages: pip==24.2
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
```

- activate the virtual environment

```
C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>venv\Scripts\activate

(venv) C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>
```

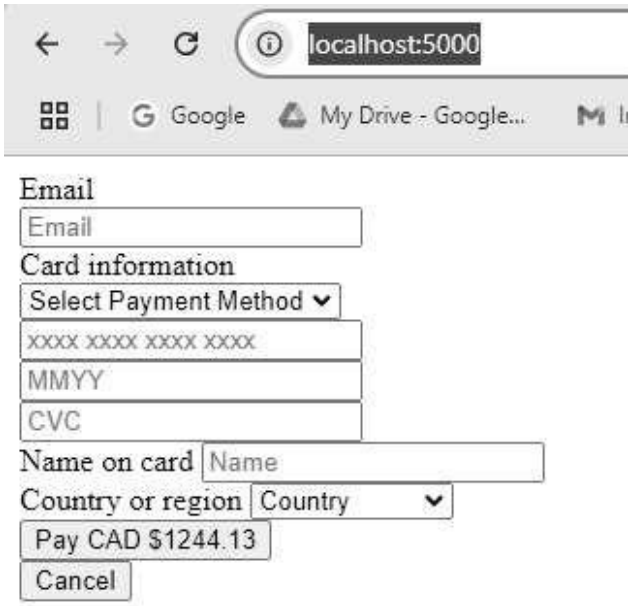
- Using pip to install the required packages

```
(venv) C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>pip install -r requirements.txt
Collecting Flask (from -r requirements.txt (line 1))
  Using cached flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from Flask->-r requirements.txt (line 1))
  Using cached werkzeug-3.0.4-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from Flask->-r requirements.txt (line 1))
  Using cached jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from Flask->-r requirements.txt (line 1))
  Using cached itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask->-r requirements.txt (line 1))
  Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask->-r requirements.txt (line 1))
  Using cached blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting colorama (from click>=8.1.3->Flask->-r requirements.txt (line 1))
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->Flask->-r requirements.txt (line 1))
  Using cached MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Using cached flask-3.0.3-py3-none-any.whl (101 kB)
Using cached blinker-1.8.2-py3-none-any.whl (9.5 kB)
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Using cached itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Using cached jinja2-3.1.4-py3-none-any.whl (133 kB)
Using cached werkzeug-3.0.4-py3-none-any.whl (227 kB)
Using cached MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl (15 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blinker, Werkzeug, Jinja2, click, Flask
Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-3.0.2 Werkzeug-3.0.4 blinker-1.8.2 click-8.1.7 colorama-0.4.6
 itsdangerous-2.2.0
```

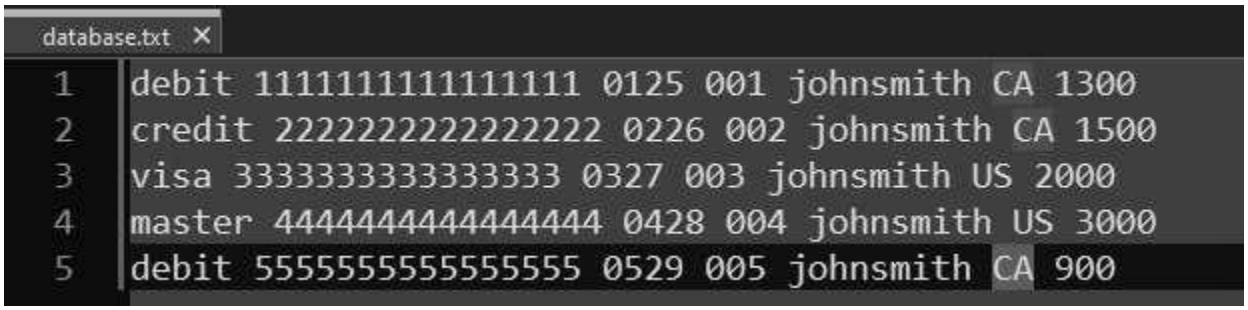
- now run app.py to start the program

```
(venv) C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 184-167-329
```

- typing http://localhost:5000/ into the browser will load the website for payment feature



- from here, you can manually test by inputting information as written in database.txt (note that name is written in lower case with no space, this is because program is made to interpret name input as such, for example, "John Smith" will be interpreted as "johnsmith")



- **or to run the test case, run the command "python -m unittest discover -s tests"**

```
(venv) C:\Users\Sungpoon\Documents\GitHub\Cisc327-GROUP51\Assignment-2\Payment>python -m unittest discover -s test
...........
----------------------------------------------------------------------
Ran 11 tests in 0.001s

OK
```

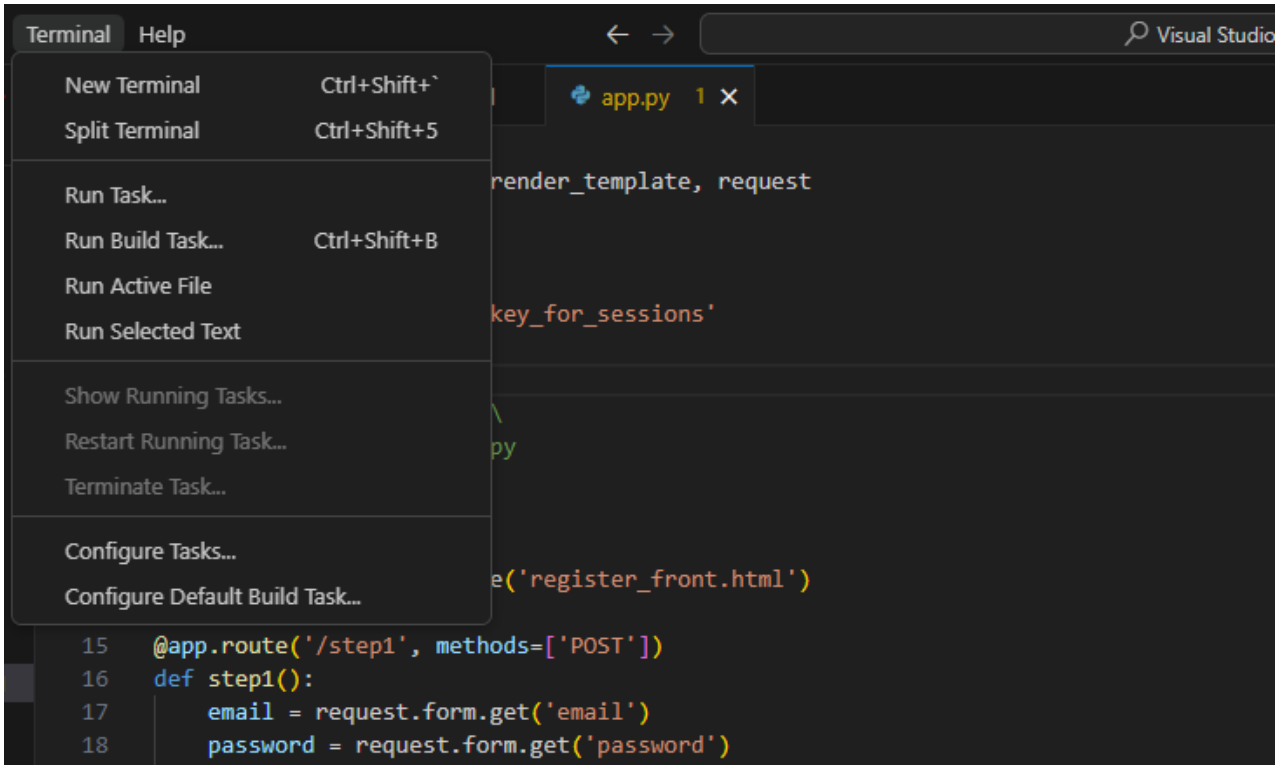
(note that 11 tests include both success and fail cases)
fail cases are tested by ensuring that no balance is returned if any of the information is incorrect, thus returning an error message on the website

Running the Registration program and test cases: Steven Guan

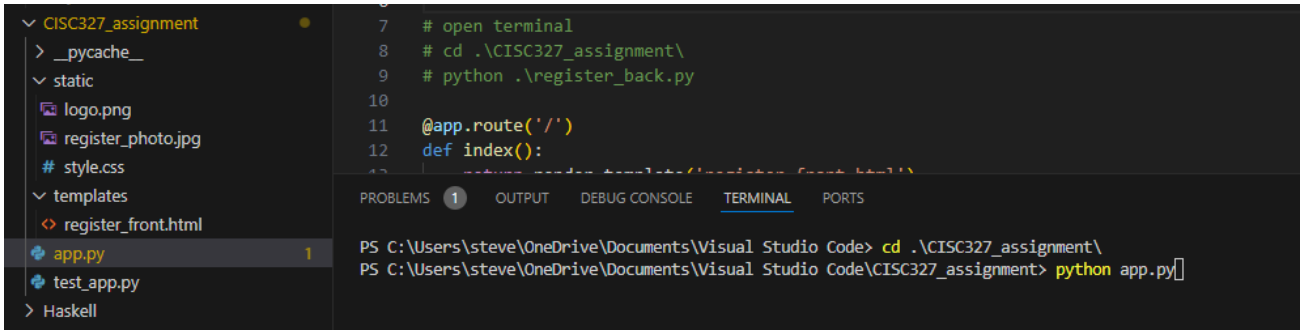
-- ENSURE YOU HAVE FLASK INSTALLED BY RUNNING (PIP INSTALL FLASK) IN THE BASH TERMINAL --

To look at the page for registration:

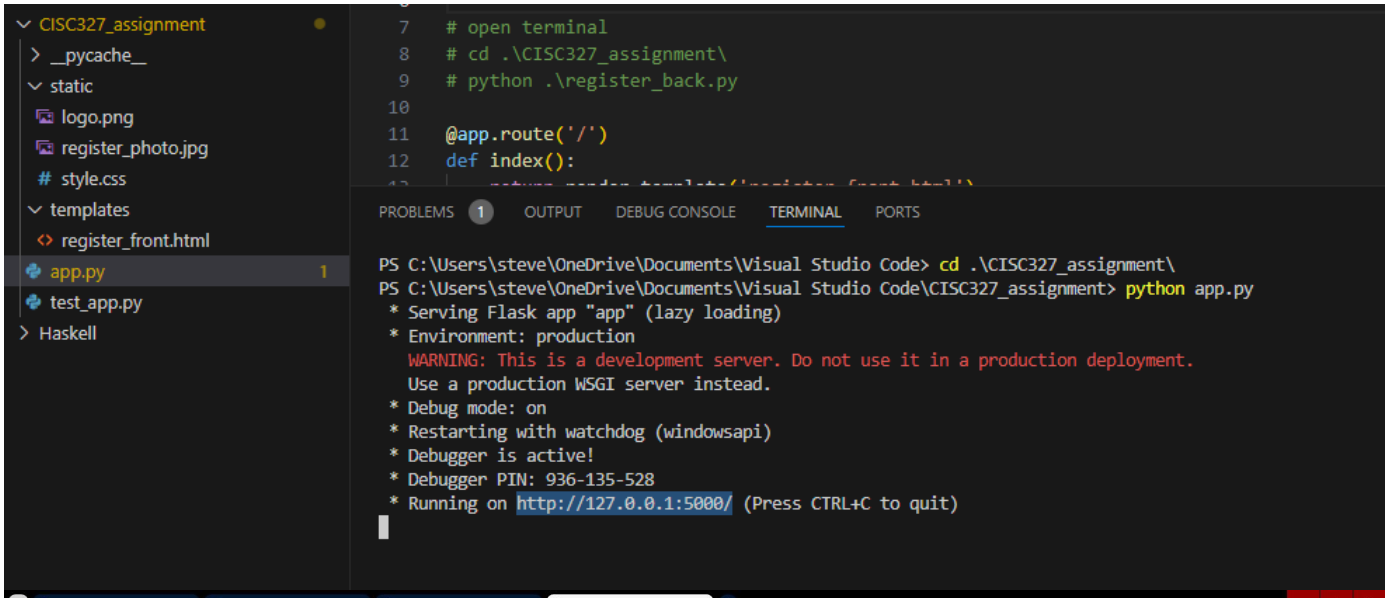
1. Start by opening a new terminal make

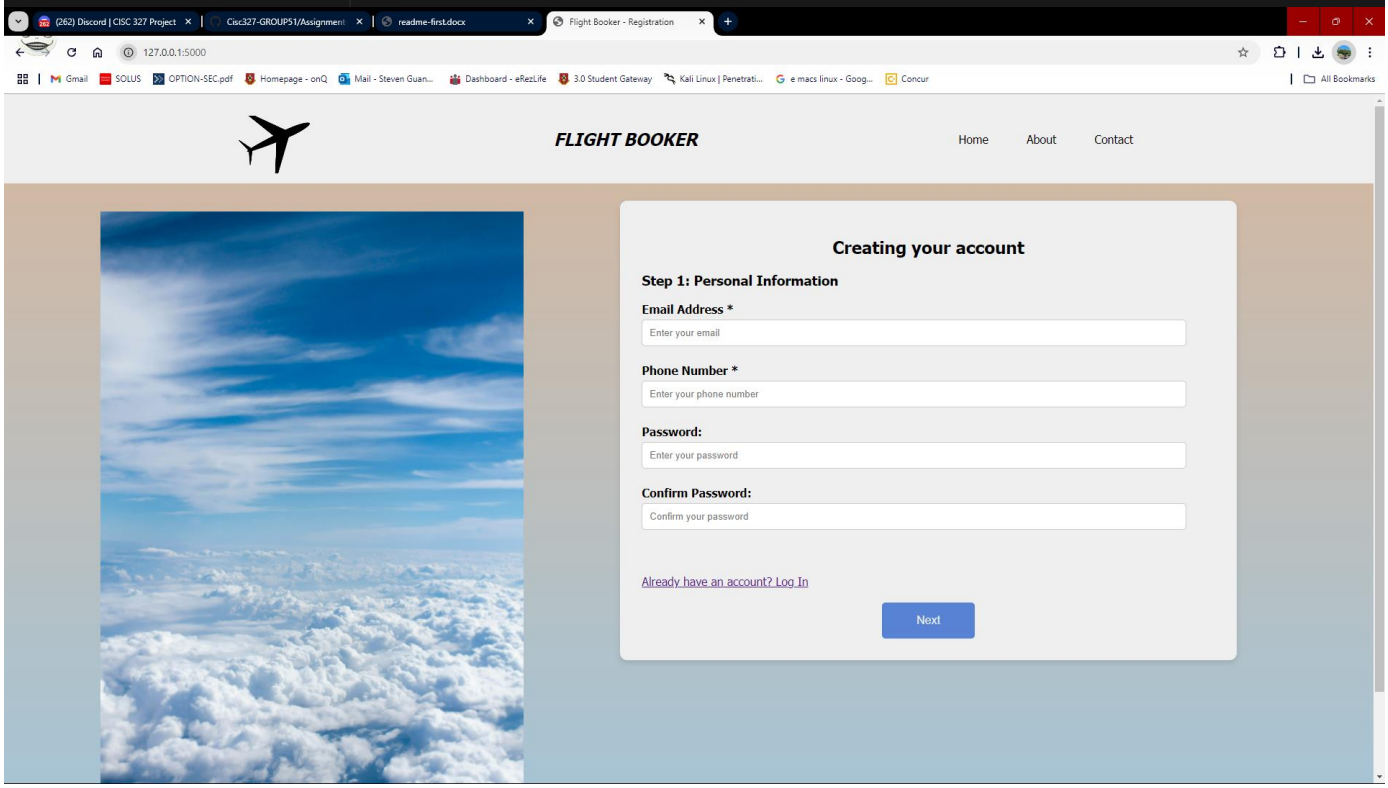


2. Ensure that the current directory is where you can access all the images, css, html and python files. Run the app.py by running ”python app.py” in the terminal.



3. Copy the url that the terminal gives you and paste it into your browser. Once you are done, exit the page by Ctrl+C on the terminal.





To run the test cases and start a test case:

1. In the same working directory as you opened the page, run “python test_app.py”

```
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Oct/2024 18:34:48] "GET / HTTP/1.1" 200 -
PS C:\Users\steve\OneDrive\Documents\Visual Studio Code\CISC327_assignment> python .\test_app.py
C:\Users\steve\anaconda3\lib\site-packages\flask\json\__init__.py:211: DeprecationWarning: Importing 'itsdangerous.json' is deprecated and wi
ad.
  rv = _json.dumps(obj, **kwargs)
........
----------------------------------------------------------------------
Ran 8 tests in 0.030s

OK
PS C:\Users\steve\OneDrive\Documents\Visual Studio Code\CISC327_assignment>
```

2. In the test_app.py file, you may alter the values of the html forms to try testing for yourself and building test cases. The logic can be found in app.py for returning 400 (failure) or 200 (success).

test_app.py | register_front.html | app.py 1

CISC327_assignment > test_app.py > FlaskTestCase > test_failed_step2_missing_1

```python
import unittest
from app import app

class FlaskTestCase(unittest.TestCase):

    # Set up the testing environment
    def setUp(self):
        self.app = app.test_client()  # Create a test client
        self.app.testing = True       # Set Flask to testing mode

    # Test if the home page loads successfully
    def test_home_page(self):
        result = self.app.get('/')  # Simulate a GET request
        self.assertEqual(result.status_code, 200)  # Check if the status code is 200 for success

    # Test for successful registration (Step 1)
    def test_successful_step1(self):
        result = self.app.post('/step1', data={
            'email': 'test@example.com',
            'password': 'password123',
            'confirmPassword': 'password123'
        })
        self.assertEqual(result.status_code, 200)  # Check if step 1 was successful
        self.assertIn(b'Step 1 successful', result.data)

    # Test for failed step 1 due to passwords that don't match
    def test_failed_step1_password(self):
        result = self.app.post('/step1', data={
            'email': 'test@example.com',
            'password': 'password_a',
            'confirmPassword': 'password_b'
        })
        self.assertEqual(result.status_code, 400)  # Expecting 400 for failed validation
        self.assertIn(b'Step 1 failed', result.data)
```

Replacing the value of ’email’, will change the input value of the form submission with the id ’email’.