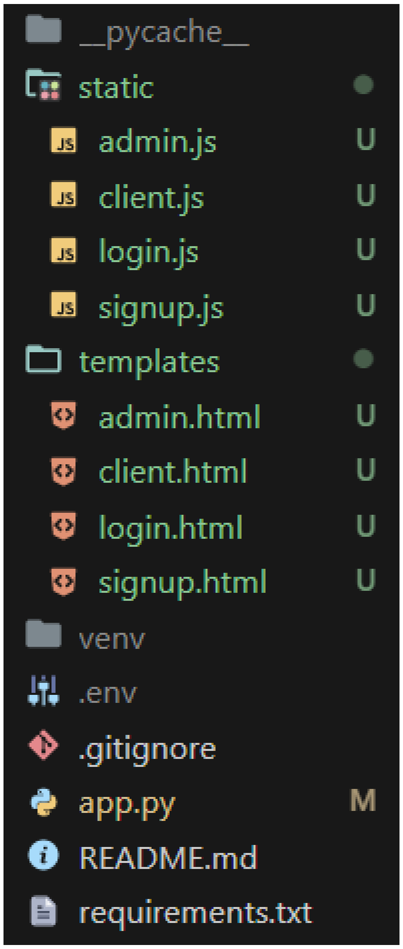
Database Design Project 4 Short Report
Siwon Kim, Hyunjung Lee

- An initial version of web three-tier architecture
  - Interface - FE - HTML, CSS, JS
  - Application logic - Server - Flask
  - Database - MongoDB

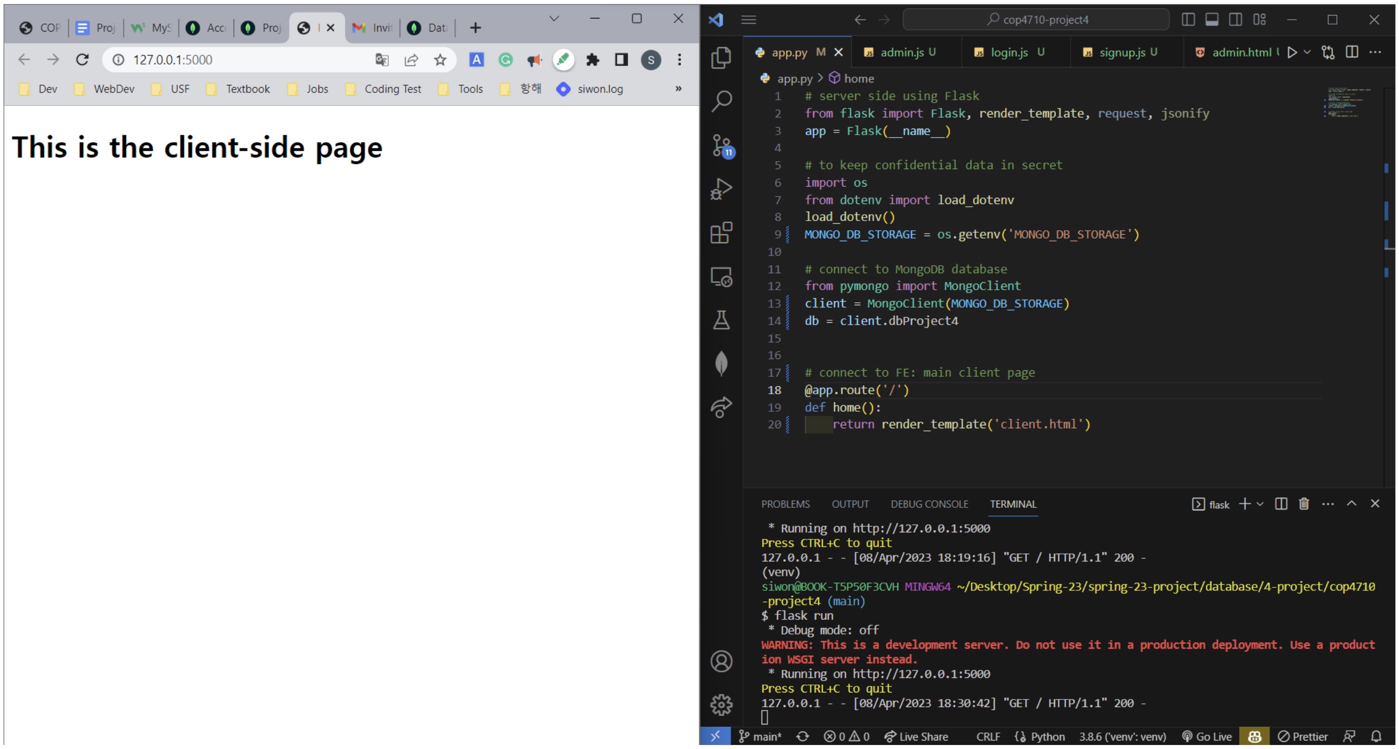
- Functionalities
  - A database that consists of at least three relations, and you should load the tables with some data (you need to either make up some data or borrow data from other sources with permission from the data owner)
    - Admin page to view all dataset on database (if needed)
  - At least eight different types of queries (in terms of SQL statements) should be supported by your system. You must have some queries that can modify the content of your database
    - When bringing some dataset - SELECT
    - When modifying some dataset - UPDATE
    - When deleting some dataset - DELETE
    - When adding some dataset - INSERT
    - When first creating the table - CREATE TABLE
  - A text-based (or web-based) interface with at least three different appearances (or web pages)
    - Login/Register page
    - Client page - to place an order
    - Admin page - to manage clients, orders, employers, etc
  - Utilization of JDBC/ODBC (or other communication protocols) to connect to the database and send in queries
    - We are going to utilize Flask for the server side
  - Create users for your system, assign userID/password to them
    - We are going to utilize JWT to keep the password tokenized
  - Create views for your database, assign different privileges to different users
    - The normal users can only purchase the products.
    - The privileged users can view the employers and department information.
  - Using stored procedures/functions to process application logic on the database side

        - By using stored procedures instead of MySQL to manage the admin side database
    - Using client-side scripts (e.g., JavaScript) to process application logic
        - We are going to utilize JavaScript
    - Any other features that are relevant to the system
        - User-side
            - User can place an order
            - User can sign up/sign in
            - Some users can view the employers/department information page
            - User can update their information
        - Admin-side
            - Admin can manage products
            - Admin can view both client side page and admin page
            - Admin can manage the customer itself and customer’s order
- Directory Structure
    - 

```
__pycache__
static              ●
  admin.js          U
  client.js         U
  login.js          U
  signup.js         U
templates           ●
  admin.html        U
  client.html       U
  login.html        U
  signup.html       U
venv
.env
.gitignore
app.py              M
README.md
requirements.txt
```

    - Under app.py, all server-related functions will be written, including MongoDB management code

    - Under templates, all HTML files will be included
    - Under static, all CSS and JavaScript files will be included
- Our working status



This is the client-side page

```python
# server side using Flask
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

# to keep confidential data in secret
import os
from dotenv import load_dotenv
load_dotenv()
MONGO_DB_STORAGE = os.getenv('MONGO_DB_STORAGE')

# connect to MongoDB database
from pymongo import MongoClient
client = MongoClient(MONGO_DB_STORAGE)
db = client.dbProject4


# connect to FE: main client page
@app.route('/')
def home():
    return render_template('client.html')
```

```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [08/Apr/2023 18:19:16] "GET / HTTP/1.1" 200 -
(venv)
siwon@BOOK-T5P50F3CVH MINGW64 ~/Desktop/Spring-23/spring-23-project/database/4-project/cop4710
-project4 (main)
$ flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [08/Apr/2023 18:30:42] "GET / HTTP/1.1" 200 -
```

    - We connected the client-side html page and Flask server.
    - We connected the Flask server and the MongoDB database.