

## Databases Group Project

Bonita Davis, Katherine Wirskey, Lucas Li, Davis Lynn

### Database Schema:

- **User Table**

username VARCHAR(40) NOT NULL,  
social\_media VARCHAR(40) NOT NULL,  
first\_name VARCHAR(40),  
last\_name VARCHAR(40),  
country\_birth VARCHAR(40),  
country\_residence VARCHAR(40),  
age INT,  
gender VARCHAR(10),  
verified BOOLEAN,  
PRIMARY KEY (username, social\_media)

\*Holds all user data, in the HTML it only requires a username and social media to be inputted, which is why the not null tag is applied only to the primary key

- **Post Table:**

username VARCHAR(40) NOT NULL,  
social\_media VARCHAR(40) NOT NULL,  
time\_posted DATETIME NOT NULL,  
text TEXT,  
city VARCHAR(40),  
state VARCHAR(40),  
country VARCHAR(40),  
num\_likes INT,  
num\_dislikes INT,  
multimedia BOOLEAN,  
is\_repost BOOLEAN,  
orig\_user VARCHAR(40),  
orig\_social\_media VARCHAR(40),  
orig\_time\_posted DATETIME,  
PRIMARY KEY (username, social\_media, time\_posted),  
FOREIGN KEY (username, social\_media) REFERENCES user(username, social\_media),  
FOREIGN KEY (orig\_user, orig\_social\_media) REFERENCES user(username, social\_media)

\*Holds all post data, connects to users table as a foreign key. Furthermore, if is\_repost bool is true, the orig\_user, orig\_time\_posted, and orig\_social\_media reference back to the original post within the same post table.

- **Project Table:**

project\_name VARCHAR(40) NOT NULL PRIMARY KEY,  
project\_manager VARCHAR(40),  
institute VARCHAR(40),  
field\_names LONGTEXT,  
start\_date DATETIME,  
end\_date DATETIME

\*Holds all high level project data, as well as all fields related to the project. These fields are held as a CSV String, which become part of the primary key for the below table.

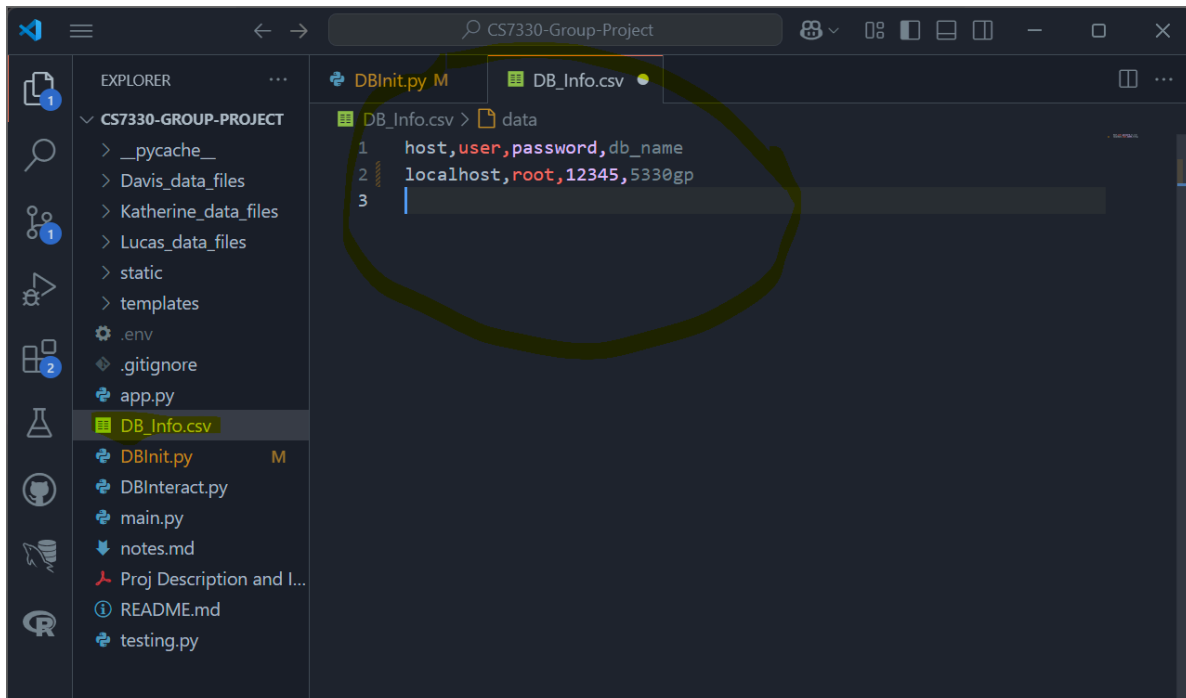
- **Project Data Table:**

project\_name VARCHAR(40) NOT NULL,  
post\_username VARCHAR(40) NOT NULL,  
post\_social\_media VARCHAR(40) NOT NULL,  
post\_time\_posted DATETIME NOT NULL,  
field VARCHAR(40) NOT NULL,  
result VARCHAR(40),  
PRIMARY KEY (project\_name, post\_username, post\_social\_media, post\_time\_posted, field),  
FOREIGN KEY (project\_name) REFERENCES Project(project\_name),  
FOREIGN KEY (post\_username, post\_social\_media, post\_time\_posted) REFERENCES Post(username, social\_media, time\_posted)

\*Holds all posts contained in the project, as well as their fields and results. Each row has a post and project associated with it, as well as a field and a potential result. The rows in this table reference both project and post tables, pulling data from both. If results are not entered at creation, a user can go and update the results for each field afterwards.

## User Manual:

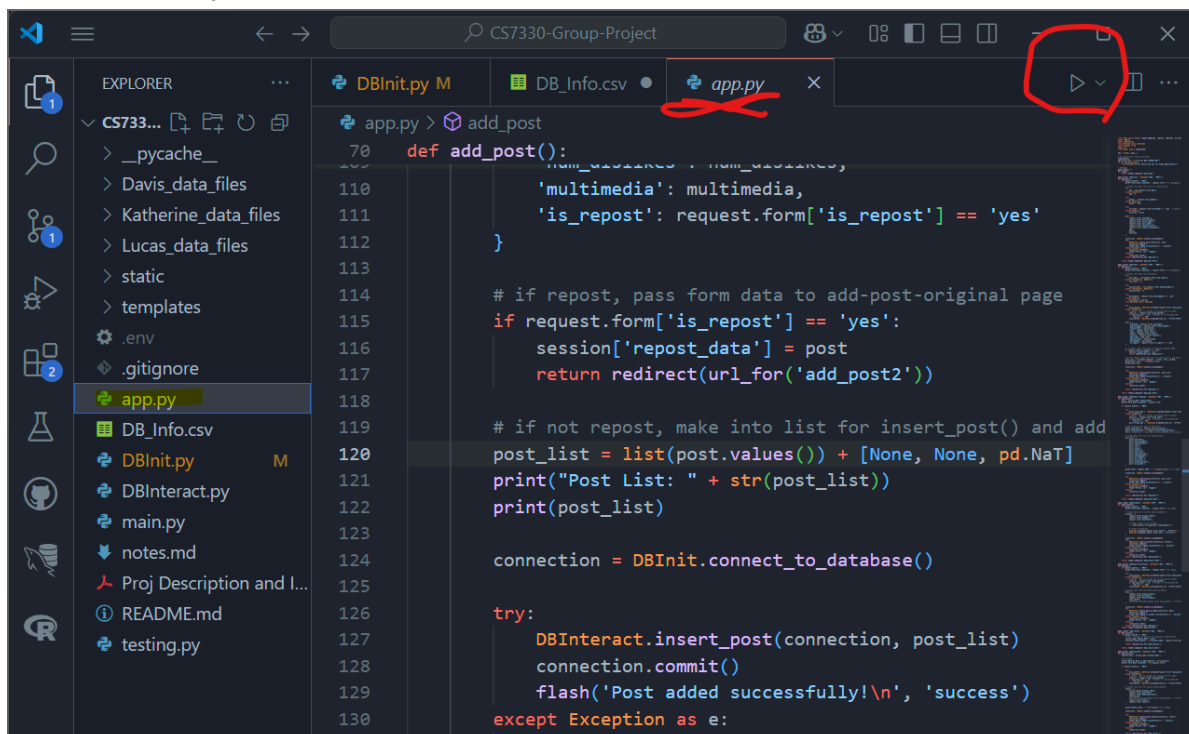
1. Insert DB connection info. You can use the DB\_Info.csv file to put in details. There is a key provided at the top of the file, do not remove the key. **Do we want to mention the .env file too?**



The screenshot shows the VS Code interface with the 'CS7330-Group-Project' workspace. The Explorer sidebar on the left lists files including `DB_Info.csv`. The main editor displays the contents of `DB_Info.csv`, which has a header row `host,user,password,db_name` and a data row `localhost,root,12345,5330gp`. A yellow circle highlights the data row.

```
1 host,user,password,db_name
2 localhost,root,12345,5330gp
3
```

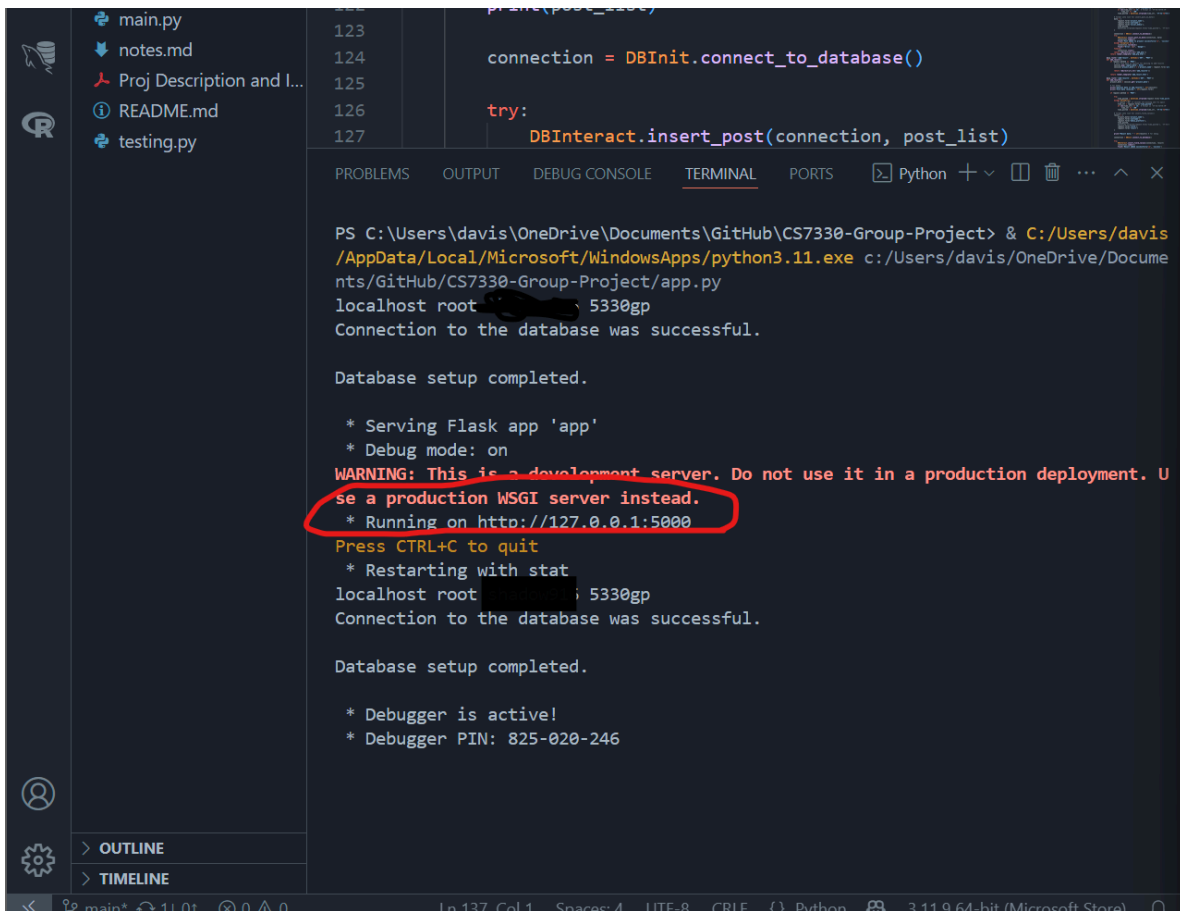
2. Run `app.py` file (note that Flask must be installed)



The screenshot shows the VS Code interface with the `app.py` file open. The Explorer sidebar on the left lists files including `app.py`. The main editor displays the code for the `add_post` function. A red circle highlights the Run button (a play icon) in the top right corner of the editor window.

```
70 def add_post():
110     'multimedia': multimedia,
111     'is_repost': request.form['is_repost'] == 'yes'
112 }
113
114 # if repost, pass form data to add-post-original page
115 if request.form['is_repost'] == 'yes':
116     session['repost_data'] = post
117     return redirect(url_for('add_post2'))
118
119 # if not repost, make into list for insert_post() and add
120 post_list = list(post.values()) + [None, None, pd.NaT]
121 print("Post List: " + str(post_list))
122 print(post_list)
123
124 connection = DBInit.connect_to_database()
125
126 try:
127     DBInteract.insert_post(connection, post_list)
128     connection.commit()
129     flash('Post added successfully!\n', 'success')
130 except Exception as e:
```

3. Look in terminal for an IP address, open the IP address in your web browser.



```
123
124 connection = DBInit.connect_to_database()
125
126 try:
127     DBInteract.insert_post(connection, post_list)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... ^ X

PS C:\Users\davis\OneDrive\Documents\GitHub\CS7330-Group-Project> & C:/Users/davis
/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/davis/OneDrive/Docume
nts/GitHub/CS7330-Group-Project/app.py
localhost root 5330gp
Connection to the database was successful.

Database setup completed.

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. U
se a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
localhost root 5330gp
Connection to the database was successful.

Database setup completed.

* Debugger is active!
* Debugger PIN: 825-020-246
```

4. You can now add data to the database through the user interface. We suggest you interact with the database in the following order:
- Add user(s) to the database. Each user's username and social media platform combination must be unique. All other fields are optional, but note that age requires a positive integer value.
  - Add project(s) to the database. Each project name must be unique. All fields are required, and the start date must come before the end date. You must associate at least one field with a project, and you can add as many as you like using the "Add More Fields" button.
  - Add post(s) to the database. Each post's username, social media platform, and time posted must be unique. Note that the user (i.e., username and social media) platform must already exist in the database. You must also enter text for the post and specify whether or not the post is a repost, but all other fields are optional.
    - If you indicate that a post is a repost, you will be taken to a new page where you will link the repost to the original post by specifying the original post's username, social media platform, and time posted. Note that a repost cannot be posted at a date prior to the original post.

- d. Associate post(s) to project(s). Enter in the name of an existing project and the username, social media platform, and time posted of an existing post. All inputs are required. Note that you cannot associate a post that has been posted after a project's end date with the project.
  - e. Enter result(s) for field(s). Enter in the name of an existing project. Enter in the name of a field associated with that project and the username, social media platform, and time posted of a post associated with that project. Then, enter the value for that field. All inputs are required.
- 5. Once you have added data to the database, you can query the data. The user interface supports two types of queries:
  - a. Query posts: You can query the posts by any combination of (1) social media platform, (2) date range of time posted, (3) username, (4) user first/last name combination. If there are posts that fit your criteria, you will see the text of the post, social media platform, username, time posted, and experiments the post is associated with.
  - b. Query projects: You can input the name of a project to see a list of posts associated with the project and any field values that have been entered for these posts. Additionally, there will be a list of each field associated with the project along with the percentage of posts within the project that contain a value for each field.