Project 1 - Back-End APIs

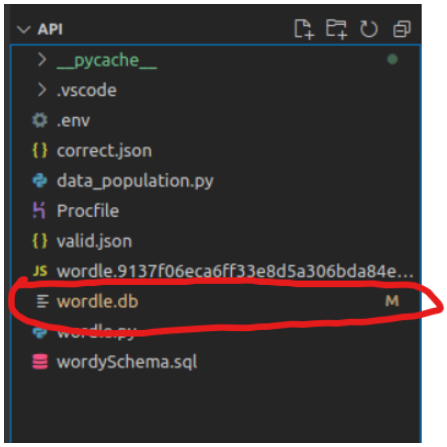
Members: Mayur Kolhe, Ajinkya Bhalero, Alex Chen, Nolan O'Donnell

Due Date: 10/22/2022

Initialization:

To run the application, we must first initialize our database:

1. Check that wordle.db does not exist in the project hierarchy already. If it does, delete it.



1.9 You may use the __init__.sh script to initialize it. I modified it to create the correct DB at the correct path. This would populate DB with all necessary data (correct and valid from the url, along with some other data that is necessary for the application to run correctly). If you choose this way, you may jump to the Starting app section.

Or you may:

2. Make sure you are in the correct directory, then in a new terminal, run the command:
sqlite3 wordle.db
to create the db.

```
:~/Documents/Github/Wordle449/quart/api$ sqlite3 wordle.db
```

3. Read the wordySchema.sql into the wordle.db by calling
.read wordySchema.sql

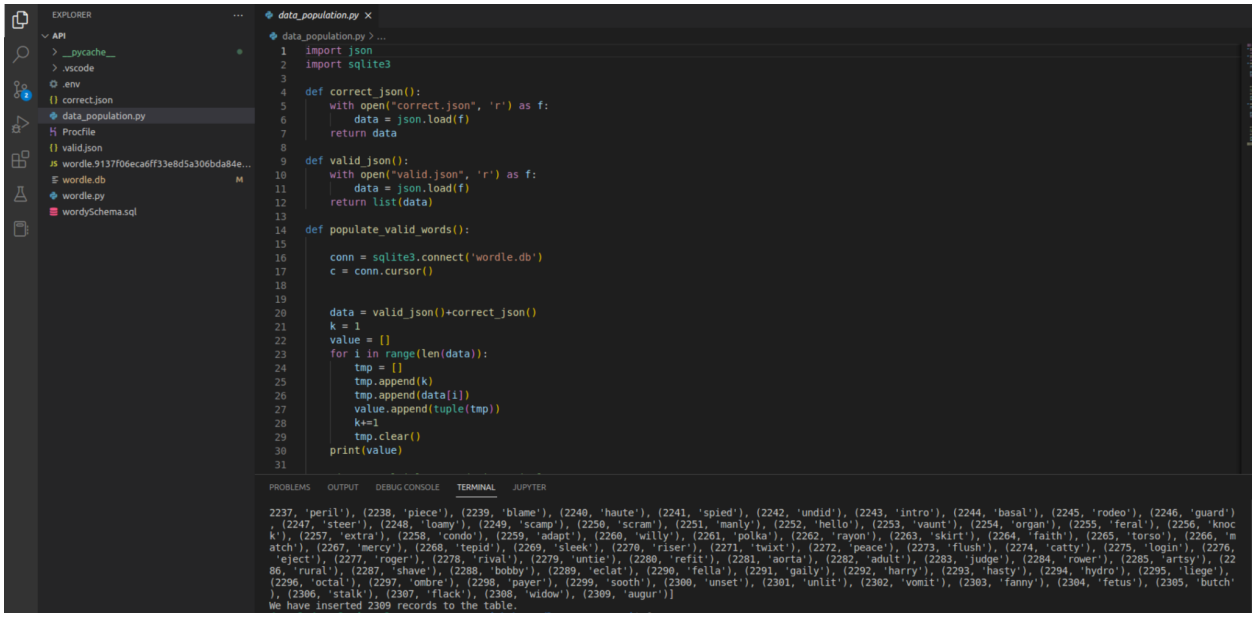
```
sqlite> .read wordySchema.sql
```

4. Exit the sqlite database by calling .exit

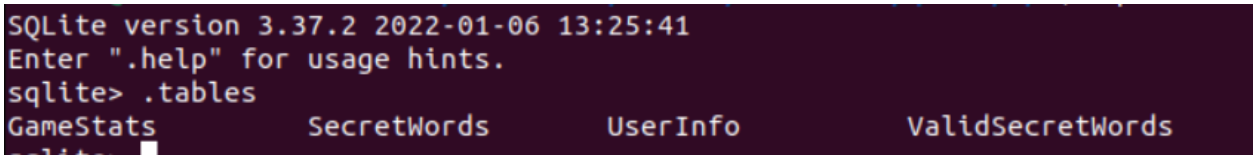
```
sqlite> .exit
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$
```

To populate our database, we must run data_population.py in our project hierarchy one time in order to load correct.json and valid.json in the db.

1. Run data_population.py. Verify that the terminal shows the values being populated into the db.



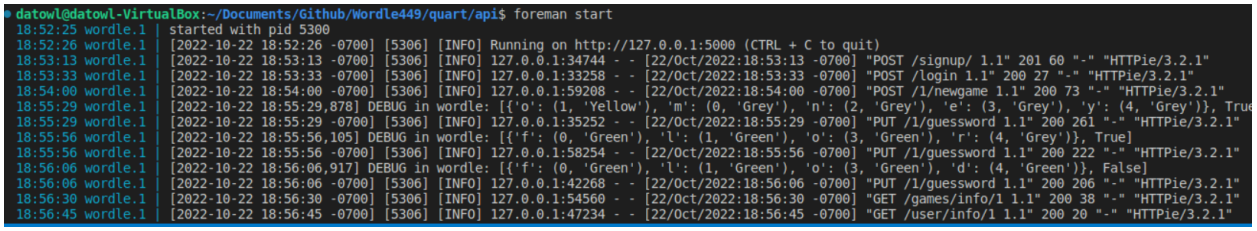
2. Go back to our terminal to verify that the db has been populated. Use the command sqlite3 wordle.db to open the database.
3. Call .tables and verify that values now exist in the database



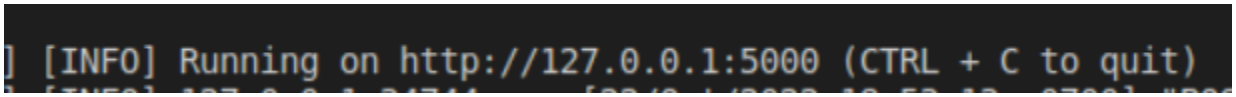
Starting App:

After the database has been populated, the service needs to start.

1. Call foreman start in the terminal to allow for the service to make request to the server



2. With our procfile, we are specifying our quart/hypercorn server that we are binding to port 5000 to make requests to our service.



Example Execution and Functionality:

Step 1: Sign up with POST Request:

Syntax: http POST http://127.0.0.1:5100/signup/ user_name=<example> user_password= <example>

Example request: http POST http://127.0.0.1:5100/signup/ user_name=nolan_odonnell user_password=123

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http POST http://127.0.0.1:5000/signup/ user_name=nolan_odonnell user_password=123
HTTP/1.1 201
content-length: 69
content-type: application/json
date: Sun, 23 Oct 2022 04:15:47 GMT
server: hypercorn-h11

{
    "user_id": "Your User_id is 2",
    "user_name": "nolan_odonnell"
}
```

Step 2: Login Post Request:

Syntax: http POST http://127.0.0.1:5000/login user_name=<> user_password=<>

Example Request: http POST http://127.0.0.1:5000/login user_name=nolan_odonnell user_password=123

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http POST http://127.0.0.1:5000/login user_name=nolan_odonnell user_password=123
HTTP/1.1 200
content-length: 27
content-type: application/json
date: Sun, 23 Oct 2022 04:15:57 GMT
server: hypercorn-h11

{
    "authorization": true
}
```

POST Request Continue Example (error check)

Example: login failed due to an incorrect password

http://127.0.0.1:5100/login user_name=nolan_odonnell user_password=123122

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http POST http://127.0.0.1:5000/login user_name=nolan user_password=1231222
HTTP/1.1 401
content-length: 50
content-type: application/json
date: Sun, 23 Oct 2022 04:28:31 GMT
server: hypercorn-h11

{
    "authorization": "Failed-Incorrect Password"
}
```

Step 3: Create New Game - where the user_id must be specified
http POST http://127.0.0.1:5000/<user_id>/newgame
Example: http POST http://127.0.0.1:5000/2/newgame

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http POST http://127.0.0.1:5000/2/newgame
HTTP/1.1 200
content-length: 73
content-type: application/json
date: Sun, 23 Oct 2022 04:29:14 GMT
server: hypercorn-h11

{
    "New Game Started": "success",
    "game_id": "Your new game id is 8"
}
```

Step 4: New Move Functionality: Put Request
For this request, the user must specify their move entry (word they'd like to guess) and the attempt number they are on.
Syntax: http PUT http://127.0.0.1:5100/<game_id>/guessword entry=<guess_word> attempt_number=<current_attempt>
Example Request: http PUT http://127.0.0.1:5100/8/guessword entry=money attempt_number=1

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=money attempt_number=1
HTTP/1.1 200
content-length: 263
content-type: application/json
date: Sun, 23 Oct 2022 04:32:30 GMT
server: hypercorn-h11

{
    "1st attempt": "5 Attempts Remaining",
    "word": {
        "e": [
            3,
            "Grey"
        ],
        "m": [
            0,
            "Grey"
        ],
        "n": [
            2,
            "Yellow"
        ],
        "o": [
            1,
            "Yellow"
        ],
        "y": [
            4,
            "Grey"
        ]
    }
}
```

The game will reiterate this same check until the user has guessed the name or has used up all their guesses.

http PUT http://127.0.0.1:5100/8/guessword entry=bread attempt_number=2

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=bread attempt_number=2
HTTP/1.1 200
content-length: 259
content-type: application/json
date: Sun, 23 Oct 2022 04:34:23 GMT
server: hypercorn-h11

{
    "2nd attempt": "4 Attempts Remaining",
    "word": {
        "a": [
            3,
            "Grey"
        ],
        "b": [
            0,
            "Grey"
        ],
        "d": [
            4,
            "Grey"
        ],
        "e": [
            2,
            "Grey"
        ],
        "r": [
            1,
            "Grey"
        ]
    }
}
```

With correct positions of letters, the color string will change
http PUT http://127.0.0.1:5100/8/guessword entry=clair attempt_number=3

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=clair attempt_number=3
HTTP/1.1 200
content-length: 261
content-type: application/json
date: Sun, 23 Oct 2022 04:35:52 GMT
server: hypercorn-h11

{
    "3rd attempt": "3 Attempts Remaining",
    "word": {
        "a": [
            2,
            "Grey"
        ],
        "c": [
            0,
            "Green"
        ],
        "i": [
            3,
            "Grey"
        ],
        "l": [
            1,
            "Green"
        ],
        "r": [
            4,
            "Grey"
        ]
    }
}
```

Error checking involves preventing the user from putting in incorrect values in:
http PUT http://127.0.0.1:5100/6/guessmove entry=aaaaa attempt_number=4

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=aaaaa attempt_number=4
HTTP/1.1 200
content-length: 35
content-type: application/json
date: Sun, 23 Oct 2022 04:36:48 GMT
server: hypercorn-h11

{
    "attempt": "Not a valid word"
}
```

- It will not allow 4th attempt 2 time we can only hit 1 attempt 1 time.

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=atoms attempt_number=4
HTTP/1.1 200
content-length: 25
content-type: application/json
date: Sun, 23 Oct 2022 04:37:54 GMT
server: hypercorn-h11

{
    "attempt": "failed"
}
```

On 6th attempt, check the win condition for the game
http PUT http://127.0.0.1:5100/6/guessmove entry=clown attempt_number=6

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/8/guessword entry=clown attempt_number=6
HTTP/1.1 200
content-length: 247
content-type: application/json
date: Sun, 23 Oct 2022 04:38:53 GMT
server: hypercorn-h11

{
    "6th attempt": "won",
    "word": {
        "c": [
            0,
            "Green"
        ],
        "l": [
            1,
            "Green"
        ],
        "n": [
            4,
            "Green"
        ],
        "o": [
            2,
            "Green"
        ],
        "w": [
            3,
            "Green"
        ]
    }
}
```

Step 6:
Game outcome determination: win or loss based on guesses/word matching.
Syntax: http GET http://127.0.0.1:5000/games/info/<game_id>
For this previous example, we used game_id=8 on the nolan_odonnell account. This is what is shown:

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http GET http://127.0.0.1:5000/games/info/8
HTTP/1.1 200
content-length: 38
content-type: application/json
date: Sun, 23 Oct 2022 04:43:51 GMT
server: hypercorn-h11

{
    "Message": "You Won the game!!!"
}
```

Step 7: Total Wins/Losses/Ties
We can see the total wins, losses, and ties for our player:
http GET http://127.0.0.1:5000/user/info/<user_id>
Example: http GET http://127.0.0.1:5000/user/info/2

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http GET http://127.0.0.1:5000/user/info/2
HTTP/1.1 200
content-length: 20
content-type: application/json
date: Sun, 23 Oct 2022 04:41:16 GMT
server: hypercorn-h11

{
    "Total_Win": 2
}
```

Testing:
Testing was done via black-box testing (see screenshots below). We wanted to test the specific functionality of the database and the schema that was being used, so we tested to see if we created a new user if we would be able to track their Wordle game progress.

Process:

1. Create a new user account

```
http POST http://127.0.0.1:5000/signup/ user_name=nolan user_password=123
```

2. Validate the user account creation reached the database

```
sqlite> select * from UserInfo;
1|nolan|123
```

3. Log in with the new user account

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http POST http://127.0.0.1:5000/login user_name=nolan user_password=123
HTTP/1.1 200
content-length: 27
content-type: application/json
date: Sun, 23 Oct 2022 03:30:40 GMT
server: hypercorn-h11

{
    "authorization": true
}
```

4. Play through world games (win/loss/in progress) - Making sure to update the guess and attempt number with each iteration.

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/2/guessword entry=money attempt_number=1
HTTP/1.1 200
content-length: 261
content-type: application/json
date: Sun, 23 Oct 2022 01:57:43 GMT
server: hypercorn-h11

{
    "1st attempt": "5 Attempts Remaining",
    "word": {
        "e": [
            3,
            "Yellow"
        ],
        "m": [
            0,
            "Grey"
        ],
        "n": [
            2,
            "Grey"
        ],
        "o": [
            1,
            "Grey"
        ],
        "y": [
            4,
            "Grey"
        ]
    }
}
```

5. Error check input

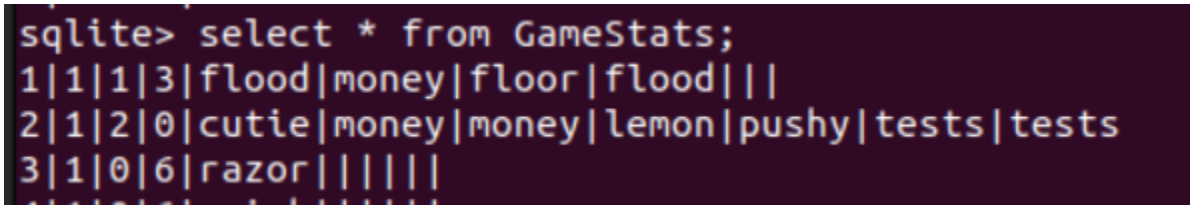
```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/2/guessword entry=aaaaa attempt_number=5
HTTP/1.1 200
content-length: 35
content-type: application/json
date: Sun, 23 Oct 2022 01:58:42 GMT
server: hypercorn-h11

{
    "attempt": "Not a valid word"
}
```

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http PUT http://127.0.0.1:5000/2/guessword entry=yes attempt_number=3
HTTP/1.1 200
content-length: 35
content-type: application/json
date: Sun, 23 Oct 2022 01:58:08 GMT
server: hypercorn-h11

{
    "attempt": "Not a valid word"
}
```

6. Verify that the account is tracking the wordle game progress in the DB

```
datowl@datowl-VirtualBox:~/Documents/Github/Wordle449/quart/api$ http GET http://127.0.0.1:5000/user/info/1
HTTP/1.1 200
content-length: 59
content-type: application/json
date: Sun, 23 Oct 2022 03:28:54 GMT
server: hypercorn-h11

{
    "In_Progress": 1,
    "Total_Lost": 1,
    "Total_Win": 1
}
```

After a valid game session was played on a new account, we verified the replayability for the accounts and how new games were being tracked. Each new game would have a new game id being tracked in the database. The screenshot found above shows that 3 separate sessions have been played, while the screenshot below shows all 3 games and how they were played out:



The first column is the game id, the 2nd column is the user_id, the third column corresponds to the game progress (in progress = 0, win = 1, loss = 2), the fourth column is the number of guesses, and the fifth column is the current word that needs to be guessed. The remaining 6 columns are the guesses the user has made.