PROGRESS REPORT
Testing Plan:

What needs to get Done:

Progress:

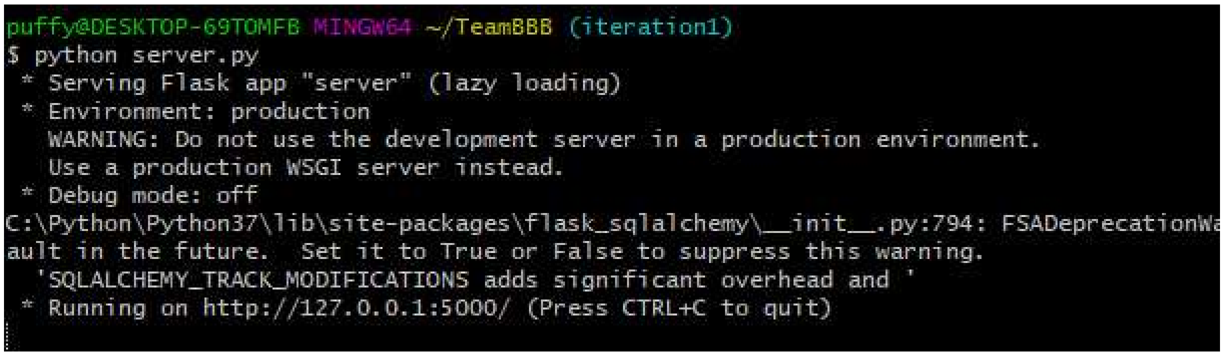
In this iteration we added our first CSS styling to our web pages. We styled our registration, login, student information, and faculty information pages. The UI design that is currently implemented with CSS is still unfinished, and will need more refinement down the road to make the UI more visually appealing. But with this iteration we added color and better spacing between elements on the web page to make the page much more readable and easy to use for the users. This implementation went fairly smooth, but we did run into trouble in some of the HTML code we wrote during the last iteration of the project. We had to rewrite some of the HTML for the login page specifically so it could work better with our CSS styling.

We also now have a class level design and a much lower level design and plan for implementation in next iterations. We figured out the different methods that are to be implemented in each class and the different fields that are required for the rest of the project.

In terms of backend we were planning to implement fully functional verification of logins and student/faculty information pages. We successfully built the login interface to check for registered account and match usernames to passwords. Then, based on if the account was registered as a student or faculty our website will send you to the respective information pages automatically as multiple users cannot have the same username, so it will check for what that username is registered as.

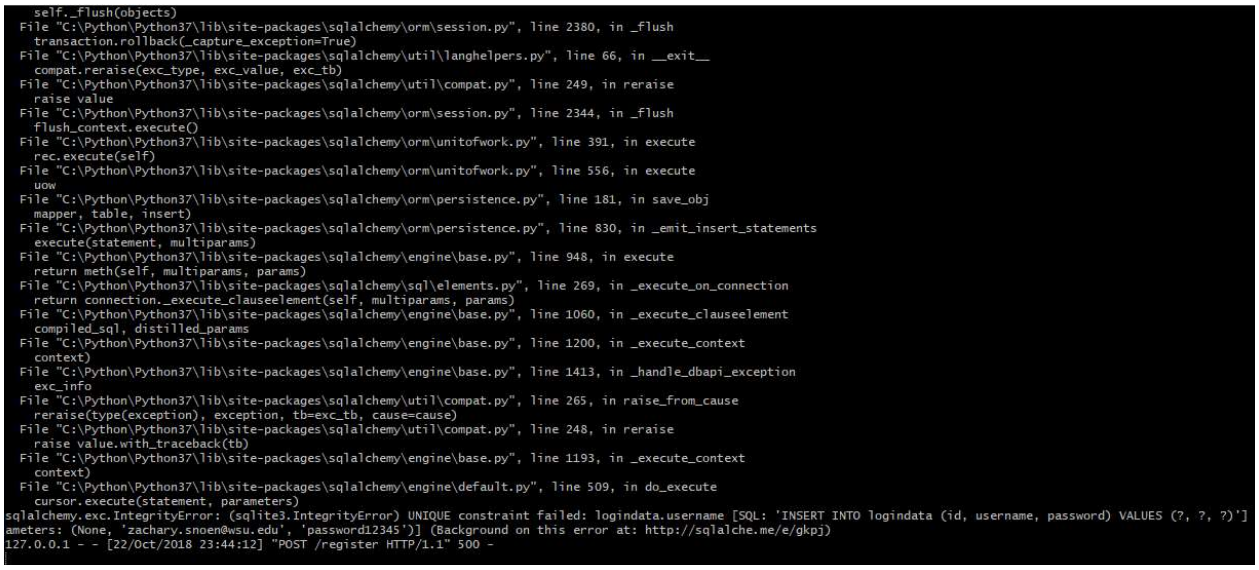
When we encountered the information pages for both student and faculty we ran into some problems. We have to keep the username variable from logging in and use that as a key to check against all users in our database to pull the correct user information and display it. So the order of operations were, logging in, checking if the entered login information was correct, sending the user to their information page, displaying information they have saved from previous sessions, and allowing them to update their information. Our problem was keeping that username variable from the login page, to the user information page, as we need it in a get request to get the information for that username. We believe we are almost at a solution but decided to work on it in iteration 3.

From the requirements document, the use case for registration or “Use Case #2” is implemented. This allows a user to enter email and password, select a student or faculty and enter that into the server. There also is logic to prevent multiple users with the same email from registering.



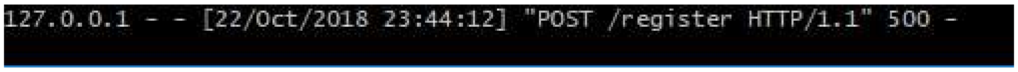
```
puffy@DESKTOP-69TOMFB MINGW64 ~/TeamBBB (iteration1)
$ python server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
C:\Python\Python37\lib\site-packages\flask_sqlalchemy\__init__.py:794: FSADeprecationWa
ault in the future.  Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Example LocalHost running (Above)



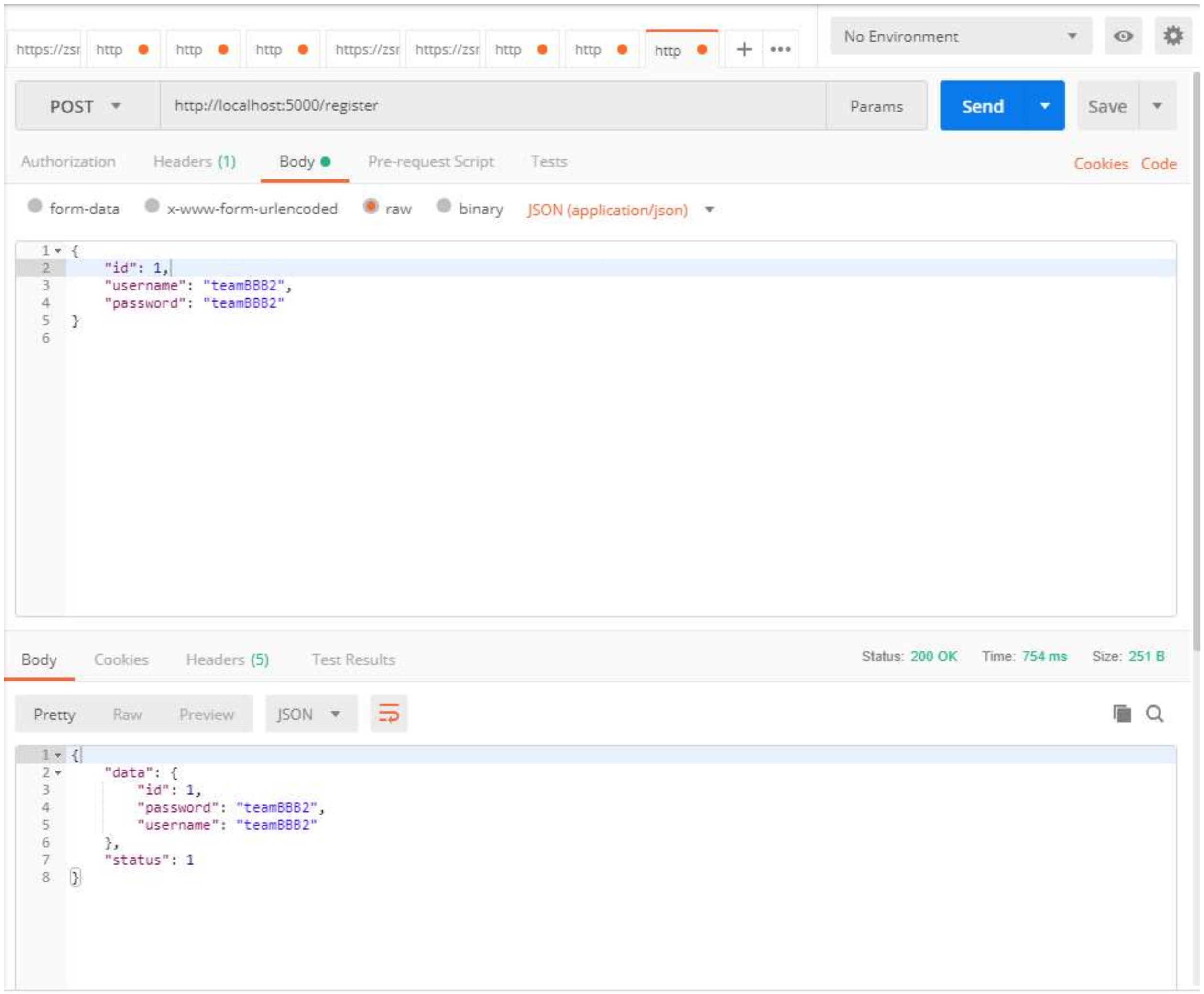
```
    self._flush(objects)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\session.py", line 2380, in _flush
    transaction.rollback(_capture_exception=True)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\util\langhelpers.py", line 66, in __exit__
    compat.reraise(exc_type, exc_value, exc_tb)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\util\compat.py", line 249, in reraise
    raise value
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\session.py", line 2344, in _flush
    flush_context.execute()
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\unitofwork.py", line 391, in execute
    rec.execute(self)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\unitofwork.py", line 556, in execute
    uow
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\persistence.py", line 181, in save_obj
    mapper, table, insert)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\orm\persistence.py", line 830, in _emit_insert_statements
    execute(statement, multiparams)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\base.py", line 948, in execute
    return meth(self, multiparams, params)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\sql\elements.py", line 269, in _execute_on_connection
    return connection._execute_clauseelement(self, multiparams, params)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\base.py", line 1060, in _execute_clauseelement
    compiled_sql, distilled_params
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\base.py", line 1200, in _execute_context
    context)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\base.py", line 1413, in _handle_dbapi_exception
    exc_info
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\util\compat.py", line 265, in raise_from_cause
    reraise(type(exception), exception, tb=exc_tb, cause=cause)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\util\compat.py", line 248, in reraise
    raise value.with_traceback(tb)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\base.py", line 1193, in _execute_context
    context)
  File "C:\Python\Python37\lib\site-packages\sqlalchemy\engine\default.py", line 509, in do_execute
    cursor.execute(statement, parameters)
sqlalchemy.exc.IntegrityError: (sqlite3.IntegrityError) UNIQUE constraint failed: logindata.username [SQL: 'INSERT INTO logindata (id, username, password) VALUES (?, ?, ?)']
ameters: (None, 'zachary.snoen@wsu.edu', 'password12345')] (Background on this error at: http://sqlalche.me/e/gkpj)
127.0.0.1 - - [22/Oct/2018 23:44:12] "POST /register HTTP/1.1" 500 -
```

Example error post request where the unique constraint on username fails, not allowing multiple users with the same email to register. (Above)



```
127.0.0.1 - - [22/Oct/2018 23:44:12] "POST /register HTTP/1.1" 500 -
```

Example successful post request (Above)



POST http://localhost:5000/register

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

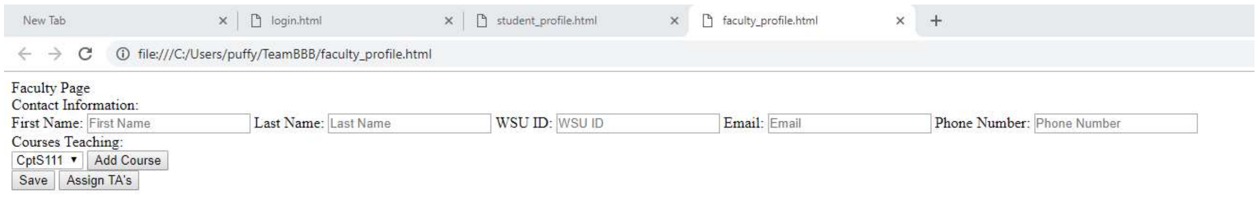
```
{
    "id": 1,
    "username": "teamBBB2",
    "password": "teamBBB2"
}
```

Status: 200 OK Time: 754 ms Size: 251 B

JSON

```
{
    "data": {
        "id": 1,
        "password": "teamBBB2",
        "username": "teamBBB2"
    },
    "status": 1
}
```

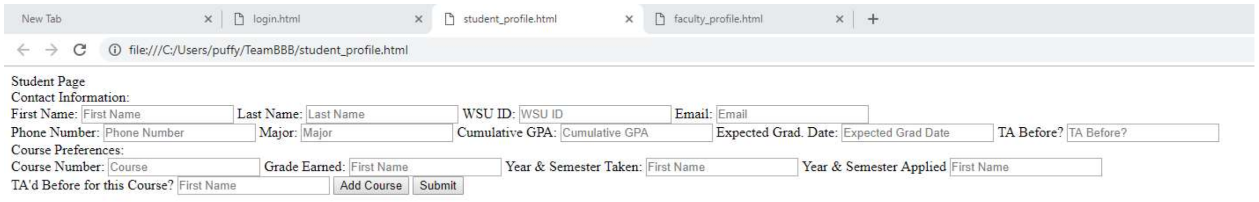
Example of sending a post request through Postman (Above)



file:///C:/Users/puffy/TeamBBB/faculty_profile.html

Faculty Page
Contact Information:
First Name: First Name Last Name: Last Name WSU ID: WSU ID Email: Email Phone Number: Phone Number
Courses Teaching:
CptS111 Add Course
Save Assign TA's

Example Faculty Information Page Design (Above)



file:///C:/Users/puffy/TeamBBB/student_profile.html

Student Page
Contact Information:
First Name: First Name Last Name: Last Name WSU ID: WSU ID Email: Email
Phone Number: Phone Number Major: Major Cumulative GPA: Cumulative GPA Expected Grad. Date: Expected Grad Date TA Before? TA Before?
Course Preferences:
Course Number: Course Grade Earned: First Name Year & Semester Taken: First Name Year & Semester Applied First Name
TA'd Before for this Course? First Name Add Course Submit

Example Student Information Page Design (Above)