Title: Powerwash: The way for better EDA

Who: Jacob Kurry (UWorldJK), Tate Middleton (tmid1/tmid3), Riley Sackett (rileysackett), Abiral Tuladhar (AbiralTr), David Velarde (DavidJVelarde)

Project Description:

Powerwash is a user-friendly web-based exploratory data analysis and data cleaning program designed to simplify data preparation. Powerwash allows users to upload datasets in .csv format and choose from a variety of cleaning and analysis options. These features include handling missing values, removing duplicates, standardizing formats, and filtering data. Once users select their desired options, Powerwash performs the operations and presents the results in a visually appealing manner. The platform supports multiple types of graphical outputs, such as bar charts, scatter plots, histograms, and box plots, enabling users to easily interpret patterns and insights within their data.

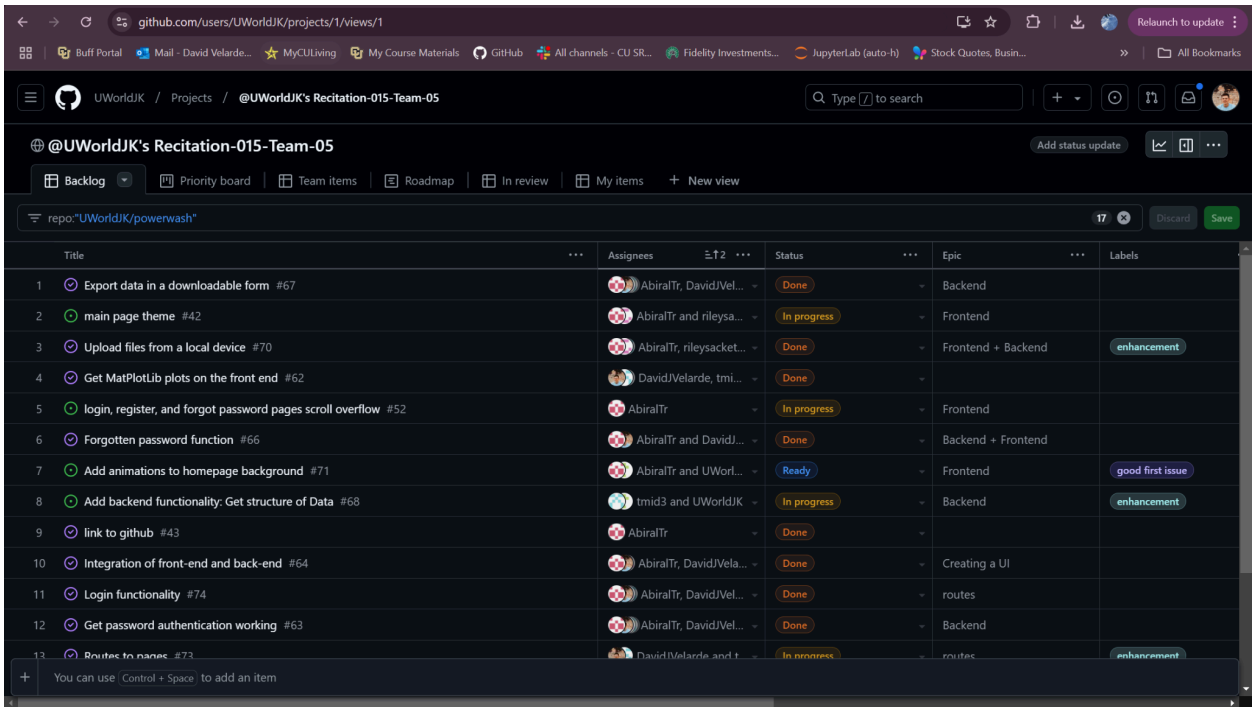
The version control was entirely handled using Github, and code writing and editing was done using VSCode and VIM. The main backend functionality of the program was developed using python's pandas library for data manipulation, and utilized flask for web integration. The frontend functionality was developed using handlebars files to format and hold all the observable components of the project, with bootstrap themes for styling. Node.JS was utilized to seamlessly connect the front and backend portions of the project. Additionally, PostgreSQL was used for the databases for the project, which hold users login information, and data uploaded to the program. All testing was done through Mocha and Chai.

Finally, Render was used as our deployment environment, allowing our team to publish completed versions of the project to the public.

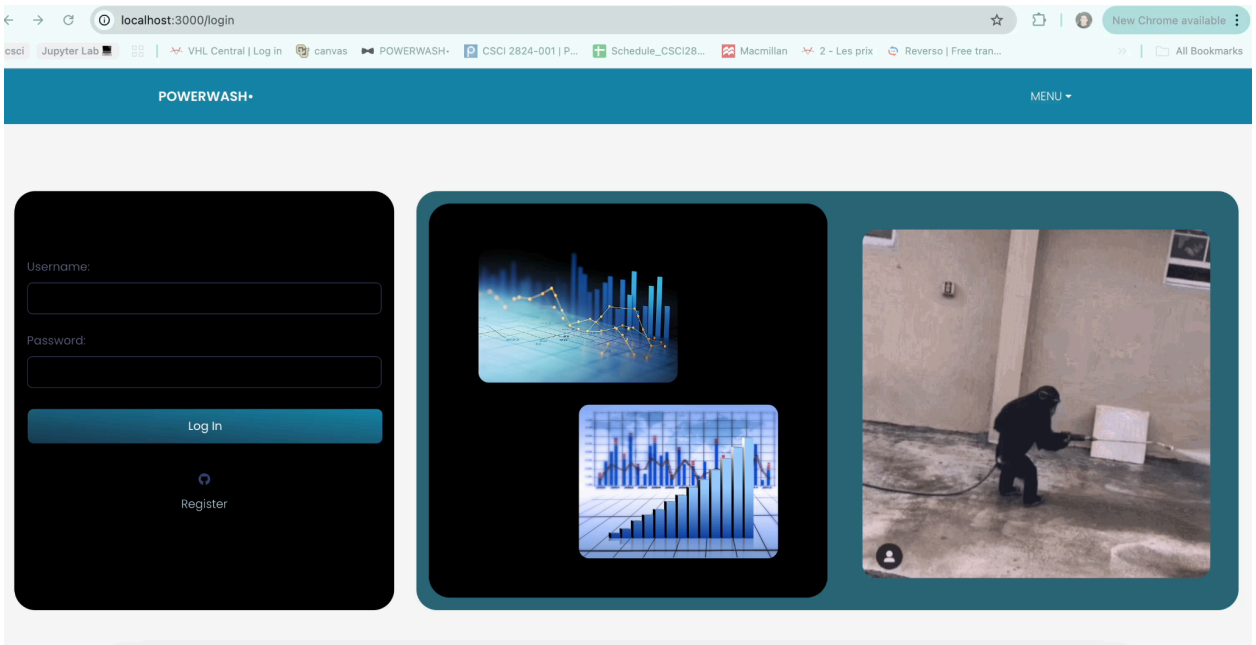
Project Tracker

Link to your Project Tracker (for instructor & TAs): https://github.com/users/UWorldJK/projects/1

Screenshot showing your project in your project tracker:



Video:

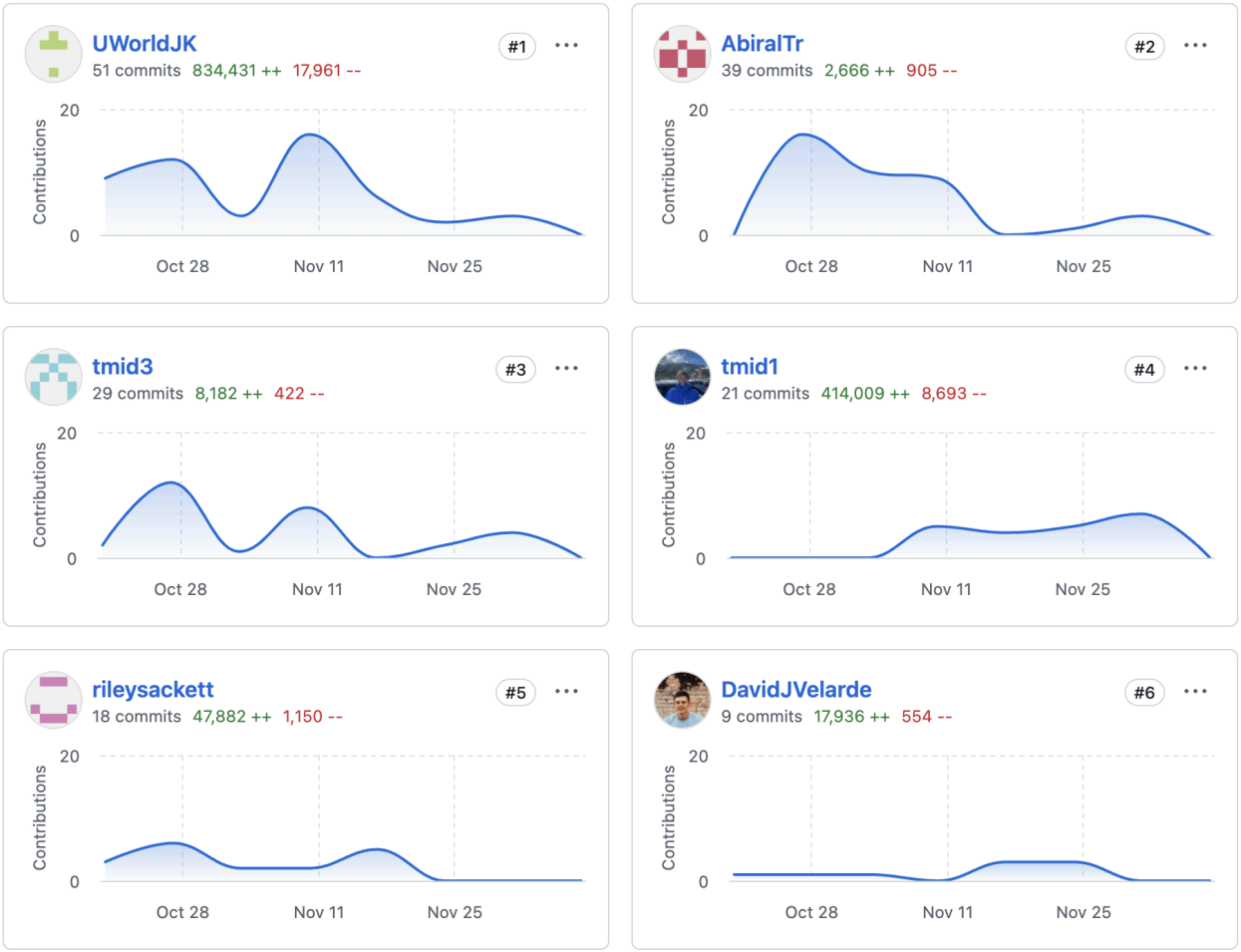


This video demonstrates the use of our finished product.

VCS:

https://github.com/UWorldJK/powerwash

Contributions:



(UWorldJK is Jacob Kurry, AbiralTr is Abiral Tuladhar, and tmid3 and tmid1 are Tate Middleton)

Tate: In the project, I floated around the full tech stack, but focused mainly on the backend side of things. This included: Javascript functions and routes in .hbs pages allowing for backend fetching, Python functions and Flask routes, and many of the data cleaning functions.

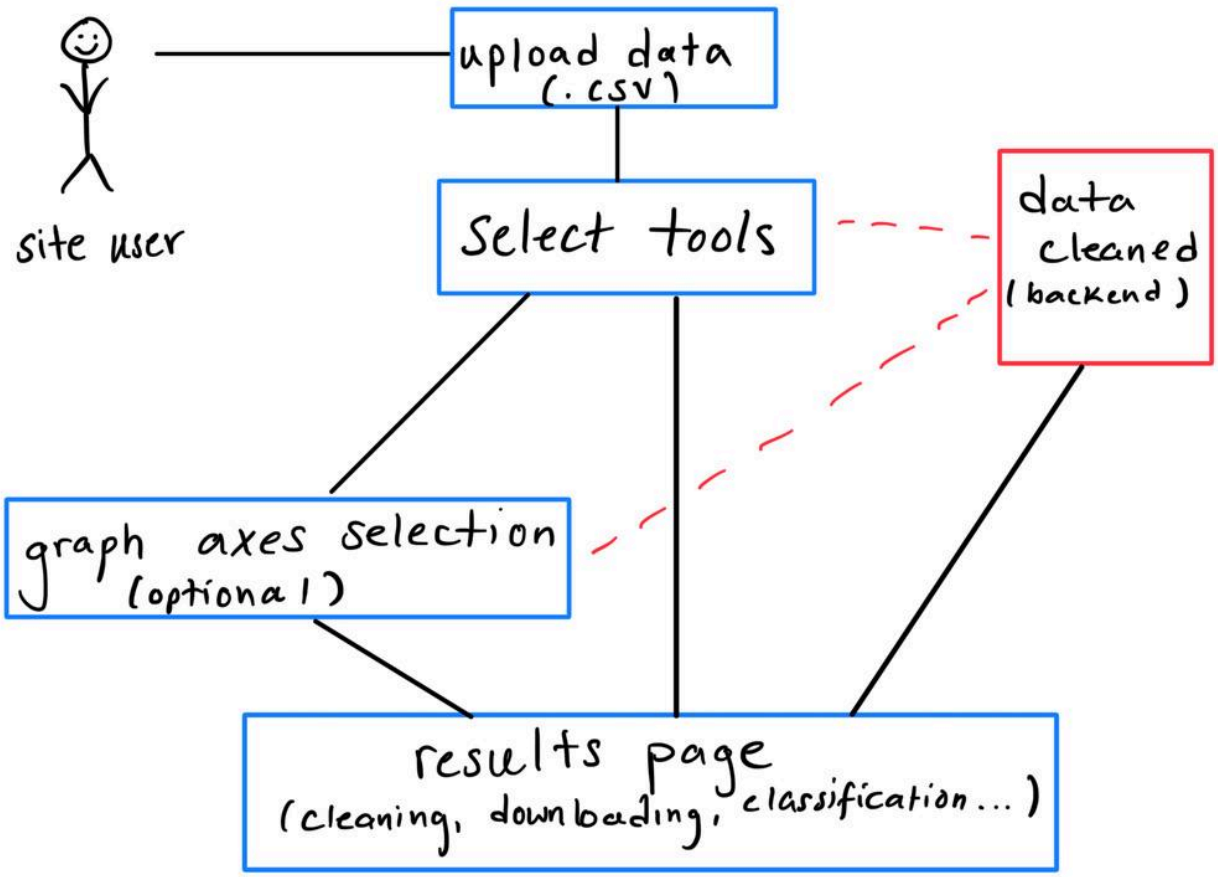
David: In the project, I worked mainly on our backend of the project, I also tweaked some front-end functionalities to work with the backend portion I was implementing. Routes within the index.js, frontend partials.

Jacob: In the project, I worked mainly on the backend and DevOps side of things. This meant jumping around between Flask, Node, Docker, and all the .env files trying to get our different services working together. I also got to play around with some of our automatic data science algorithms. I played a big role in coming up with the overall architecture of the program.

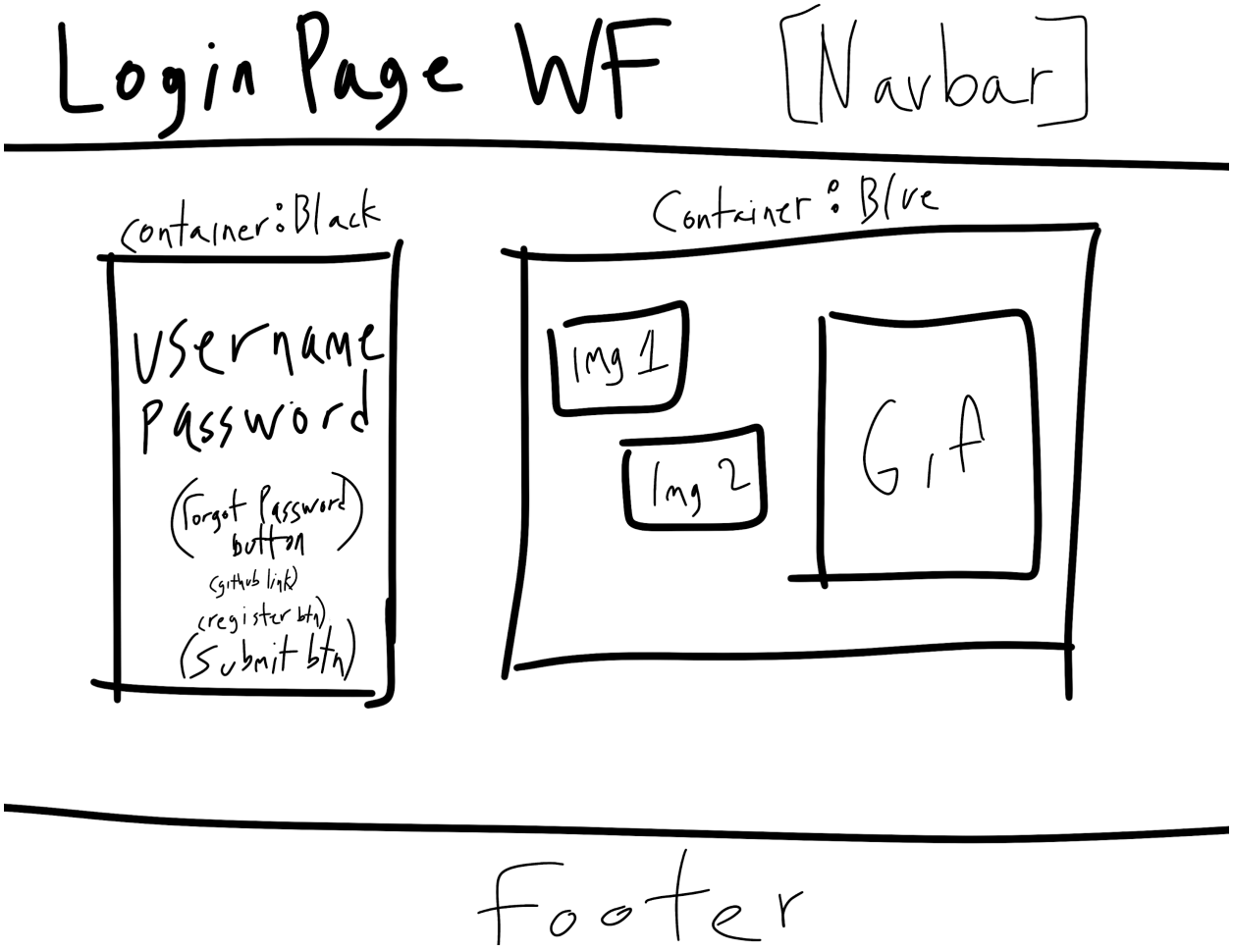
Riley: In the project I worked on the front end. Things I worked on included: Handlebars pages for the main structure of the project, utilizing bootstrap libraries, implementing data entry with forms and routes with buttons.

Abiral: In this project, I mainly worked on the frontend, trying to give form to whatever our team mates on the back end were visualizing. The creation of pages with handlebars and the NodeJS component were where I spent the most of my time. The pages I worked on include: the login page, register page, forgotten password page, profile page, about page, as well the chip selection visuals.

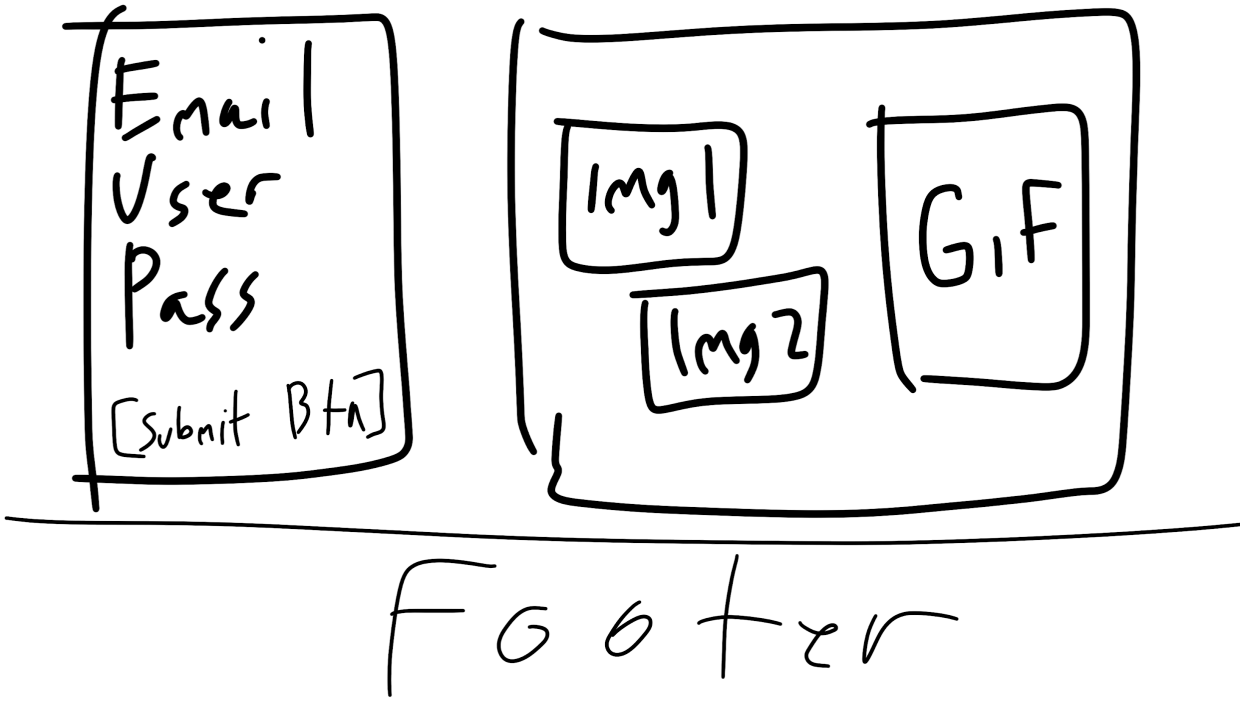
Use Case Diagram:



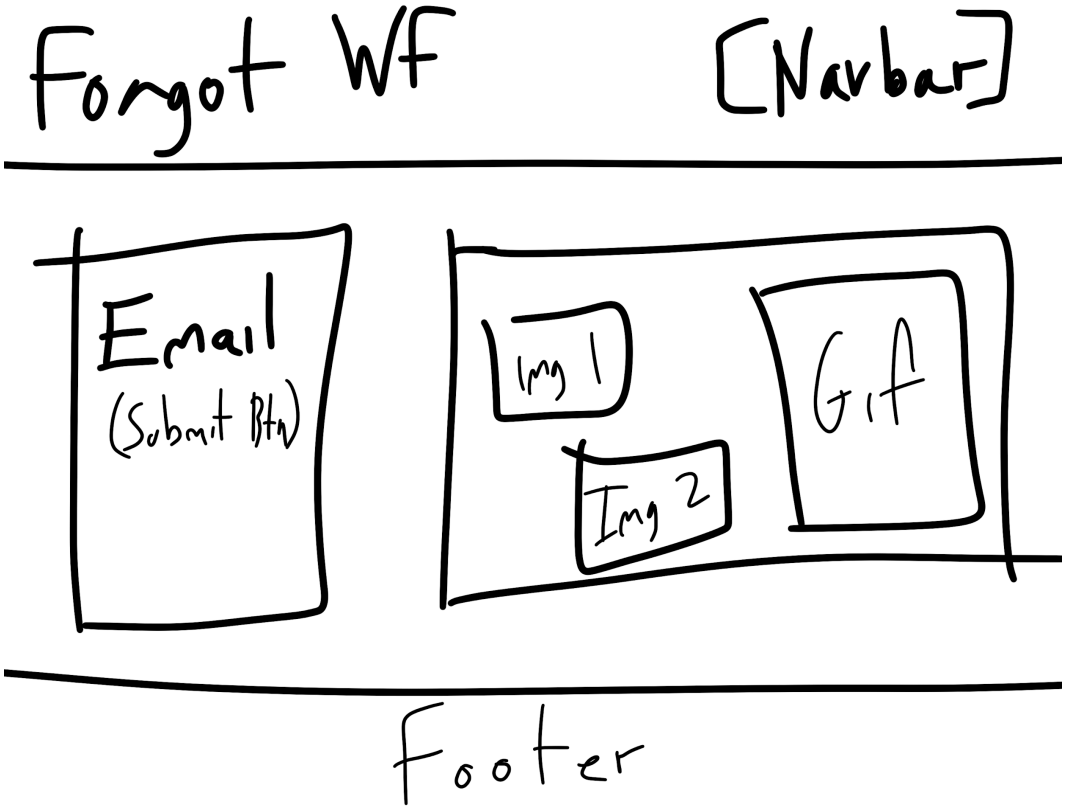
Wireframes:

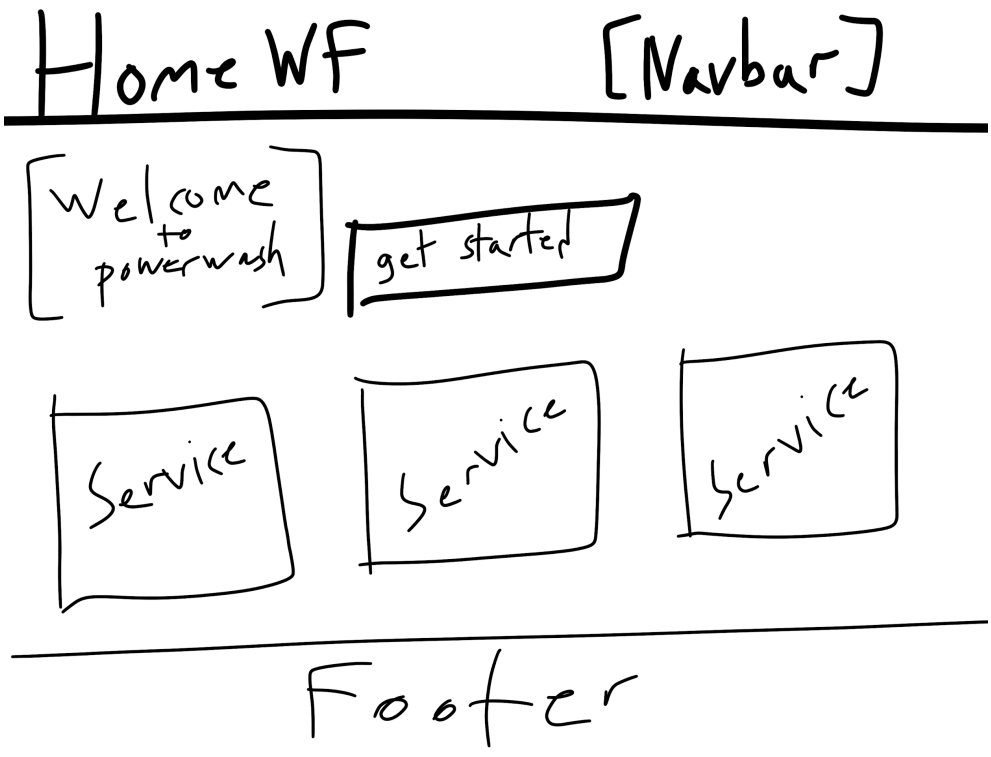

Login Page WF
[Navbar]
container: Black
Username
Password
(Forgot Password button)
(github link)
(register btn)
(Submit btn)
Container: Blue
Img 1
Img 2
Gif
Footer

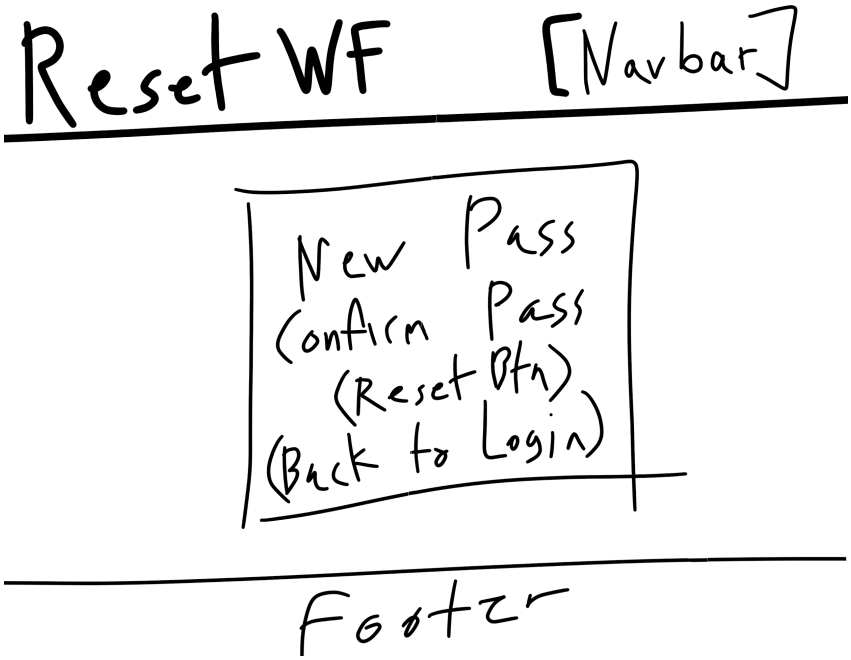
Register WF [Navbar]

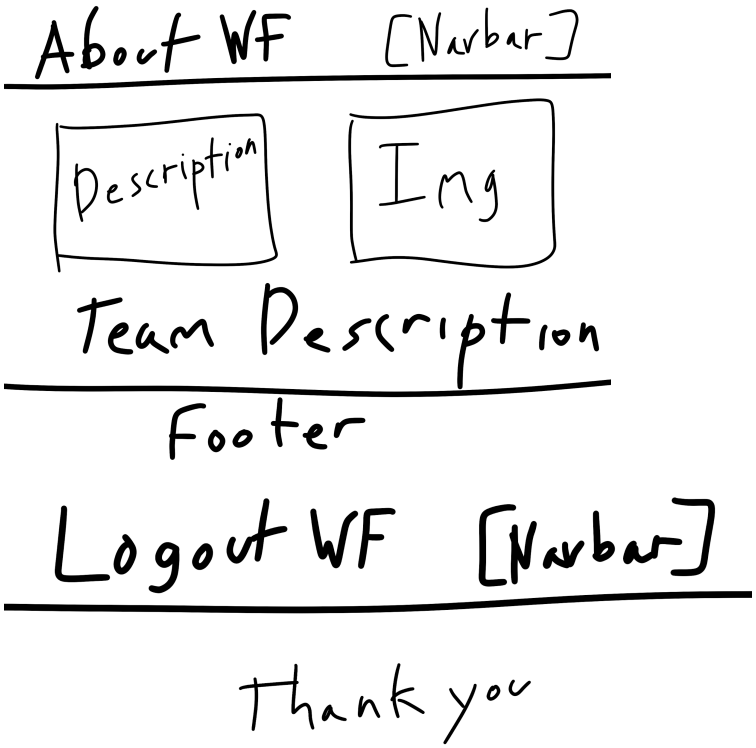


Footer


Forgot WF
[Navbar]
Email
(Submit Btn)
Img 1
Img 2
Gif
Footer


Home WF
[Navbar]
Welcome to powerwash
get started
Service
Service
Service
Footer


Reset WF
[Navbar]
New Pass
Confirm Pass
(Reset Btn)
(Back to Login)
Footer


About WF [Navbar]
Description
Img
Team Description
Footer
Logout WF [Navbar]
thank you

Test results:

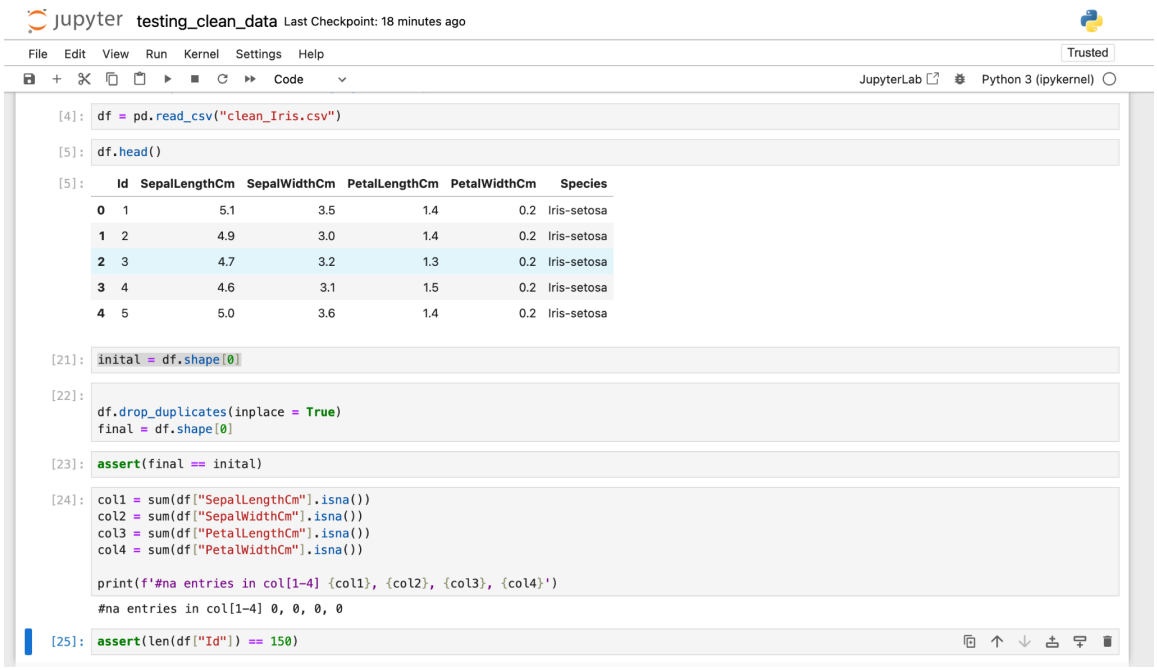
The **first feature** that we wanted to test on our UAT plan was the types of files that our app would accept as valid submissions. We implemented this in our main.py Flask application which checks if the file ends in the .csv format. If the file is either not in the CSV format or some kind of error occurs when we try to instantiate our Cleaner class we return that to the user:

```
@app.route("/upload", methods=["POST"])
def getFile():
    global data
    global fileName
    global granularity
    global choice
    global classify

    file = request.files["file"]
    #checks
    fileName = file.filename
    print("FILE NAME", fileName)
    if fileName == "":
        return jsonify({"error": "No selected file"}), 400
    if file and fileName.endswith(".csv"):
        try:

            file_content = file.stream.read().decode("utf-8")
            # print("File Content:\n", file_content)  # Display file content
```

Our **second feature** was designed to verify that username, password, and email address are accurately imputed into the database. To verify this test feature we created unit tests using Mocha and Chai. This test works by inputting a test username, password and email address into the db and then creating a query that, in theory, would return the username and password given the unit test was successful, we then use an assert statement to verify that our query returned the correct information. Additionally, it tests that the correct page is rendered when the user logs in.

```
web-1  |
web-1  | > test
web-1  | > mocha src/test/server.spec.js
web-1  |
web-1  | Server is listening on port 3000
web-1  |
web-1  |
web-1  |   Testing Add User with all Fields
web-1  | REGISTERING
web-1  | Database connection successful
web-1  | loggin in
web-1  |     ✓ Positive : /register. Checking all fields (121ms)
web-1  |
web-1  |   Testing Add User w/o all fields
web-1  |     ✓ Negative : /register. Checking invalid fields
web-1  |
web-1  |   Testing Render
web-1  |     ✓ test "/login" route should render with an html response
web-1  |
web-1  |   Testing Render Negative
web-1  |     ✓ test "a fake render" route should not render with an html response
web-1  |
web-1  |
```

Our **third** UAT feature requires a slightly different framework in order to verify that the data was cleaned directly. To do this testing I read the cleaned csv that was output to us via the Powerwash/result page and read it into a testing Jupyter notebook that I created. I then performed operations on this data set in order to verify that it was cleaned correctly. For instance, I found the shape of the dataset and then dropped all the duplicate rows and checked the shape again. I then used an assert statement to verify that these two numbers were the same. Additionally, I checked that our Cleaner class correctly removed all N/A entries from the columns. I once again used an assert statement to do this. While this is currently designed to just check one dataset it would ideally be part of a CI/CD pipeline that is tested every time we implement a new feature. My final check was that the granularity that was given from the /results page was indeed the granularity of the dataset. See image below.



Deployment:

https://powerwash.onrender.com/