## Overview of Version 1

So far, in the project, the following tasks have been completed:

1. Imported Flask library for templating and other libraries
    a. Version used is 3.0.3
2. Creation of templates using the Jinja2 template engine
    a. Reference: https://jinja.palletsprojects.com/en/3.0.x/, version – 3.1.4
    b. Created a base template – layout.html
    c. Extended base template to child templates
        i. About.html
        ii. My_works.html
        iii. Contact.html
        iv. Project_detail.html
3. Import panda data frames for creating a database
4. File structure –
    a. Portfolio_Project
        i. App (package)
            1. Init.py
            2. /static/
            3. Data_handler.py
            4. Main.py
        ii. Data (folder)
            1. Projects.csv
            2. Projects_images.csv
        iii. Documentation/
        iv. .python-version
        v. Readme.md
        vi. Render.yaml (for rendering on Render.com)
        vii. Requirements.txt
        viii. Run_app.py (starter script)
        ix. .venv (virtual env)
5. A data_handler class was created to serve as a module for performing CRUD operations to and from .csv files
    a. A default database will be created with no .csv file found in the directory.
    b. If the directory exists, the data frame will be read as dictionaries for Jinja to read and render.
    c. Images are fetched by section_titles and project_ids. Project_id serves as primary key to establish the linking of two tables.

    d. Users can add project records to the projects.csv and project_images.csv
    e. But users have to save the file and then restart the server
6. Main.py file serves as a views function for rendering templates using Jinja syntax
7. Init Python serves as the constructor for the flask object.
8. Control structures were used in each template to automate data retrieval.

```
{% block content %}
<div class = "sidebar">
    <ul>
        {% for title in section_titles %}
            {% if project[title] != '' %}
            <li><a href="#{{ title }}">{{ title.replace('_', ' ') }}</a></
            {% endif %}
        {% endfor %}
    </ul>
</div>
<div class="project-container">
    <section class="project-section">
        <h2>{{ project.title }}</h2>
        <p>{{ project.description|safe }}</p>
        <img src="{{ url_for('static', filename=project.image_path) }}" al
```

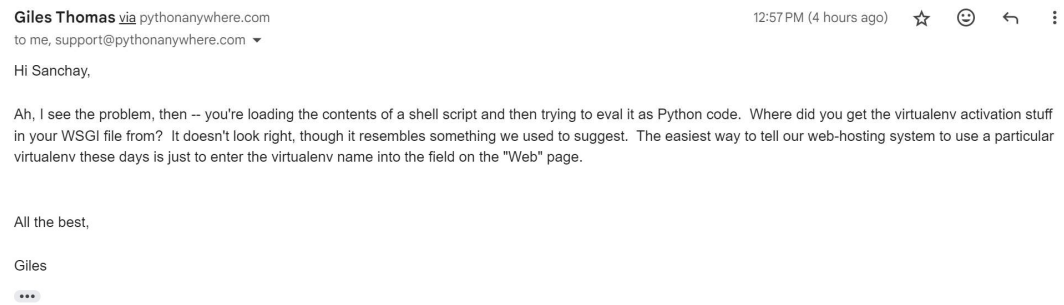
9. Document styles were implemented using CSS and Javascript.

## Issues and Resolutions

1. I faced a couple of issues with reading and retrieving data.
    a. The issues were syntactic and semantic
        i. For instance, not passing data_handler in project_detail method.
    b. Resolved circular dependencies
    c. Issues with CSS selectors and Javascript
    d. Slicing and skipping a few columns from the project.csv.
2. I added Jinja's Safe filter to prevent HTML from escaping.
3. As a first-time end-to-end coder, it was pretty challenging to cross-reference CSS statements, Jinja's syntax, python syntax, and Panda data frame functions. I kept a journal in OneNote of my learnings and cross-referenced whenever I had trouble with the code
4. **On September 21,** I tried to deploy my code in PythonAnywhere. I ran into "*Module not found,"* problems with *Venv*, and path variable issues. After cross-referencing several documents and forums, I found there could be

compatibility issues between different versions python and PythonAnywhere uses 3.10.

5. I installed *.venv* via the VSCode terminal. Reference: https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/
6. Then I installed *pyenv* to downgrade python version from 3.12 to 3.10.11. Reference: https://github.com/pyenv-win/pyenv-win/blob/master/README.md#manually-check-the-settings
7. After following these steps, I still had issues with deployment. So I created *render.yaml* and a *requirements.txt* and deployed my code on **Render.com**
8. I created a *venv* in PythonAnyWhere and connected with the support team to figure out the issue of wsgi import. The problem was the redundant activation of the *venv* shell script as a .py file and incorrect path variable in the PythonAnywhere server.



**Giles Thomas** via pythonanywhere.com 12:57 PM (4 hours ago)

to me, support@pythonanywhere.com

Hi Sanchay,

Ah, I see the problem, then -- you're loading the contents of a shell script and then trying to eval it as Python code. Where did you get the virtualenv activation stuff in your WSGI file from? It doesn't look right, though it resembles something we used to suggest. The easiest way to tell our web-hosting system to use a particular virtualenv these days is just to enter the virtualenv name into the field on the "Web" page.

All the best,

Giles

9. Now, the issue has been fixed, and the server is running well.

## Production environment

1. https://sanchay071.pythonanywhere.com/
2. https://portfolio-project-td9p.onrender.com/

## Milestones for the next few weeks

1. Adding Python scripts to handle large images
2. Refactoring data_handler to include a few more project files if need
3. Add widgets using CSS
4. Adding structure to add custom sections
5. Animations (not sure if I will have time for this)
6. Moving to SQLite for dynamic updates

With this project, I'm trying to grasp end-to-end product design, front-end, back-end, database design, and best practices. I'm quite satisfied, but I'd add more features and functions and automate the process as much as possible. Jinja templating and its syntax are so versatile, and I'm not sure how these contrast against other languages and traditional frameworks for end-to-end development. I'm looking to explore more data frames and other Python libraries.