

# Project Development Report

## Project Title and Group Members

Rehab Revive by Team orangeOS, including Alexis-Rachelle Ramelb, Wilneris Carrion-Colon and Fletcher MacDonald.

## Introduction

Medical clinics are an essential service that many individuals utilize in their daily lives, yet many patients in Hawai'i struggle to find clear and specific information about healthcare providers in one centralized place. This project aims to create an easy-to-navigate website that centralizes essential information on healthcare services, with a particular focus on sports medicine for those seeking specialized care. In this Project Development Report you will get an insight into the development of our website including, **API's**, **Design**, **Challenges Faced** and **Future Improvements**.

## API Development

Within our project we have a total of 6 API's; 4 GET, 1 POST and 1 DELETE.

GET:

The three GET API retrieve the data from the analyze\_data.py and serve the information to specific html files to be displayed. The three API routes are:

```
'/ortho_data' ->(Returns orthopedic clinic data.)  
'/add-clinic-page' ->(Serves the HTML page where users can add a new clinic.)  
'/medical-imaging_data' ->(Returns medical imaging service data.)  
'/pt_data' ->(Returns physical therapy (PT) data.)
```

DELETE:

This API deletes all data related to a specific clinic based on the name you provide in the URL using Postman. It removes the clinic's information from the json file, so it will no longer appear when you retrieve all clinics using the GET API. The DELETE API route is:

```
'/delete/<clinic_name>' ->(Deletes all data related to the specified clinic name.)
```

Thus, for example, in Postman, if you do:



it will delete Kuakini Plaza Imaging and all its information off the medical imaging clinics page, as well as in the json file.

## POST:

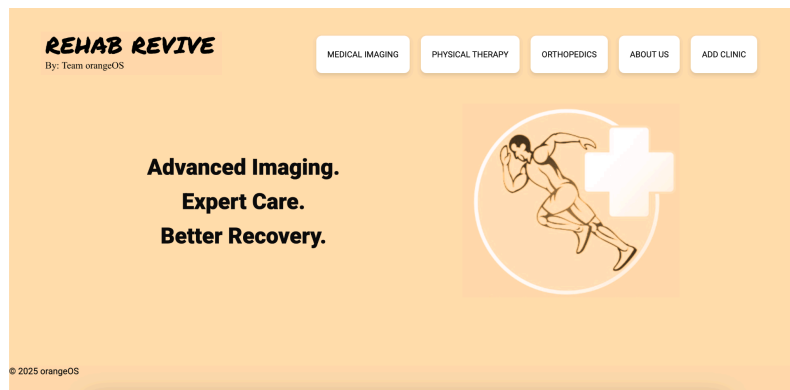
The POST API adds all the data for a new clinic that you input through the form on the Add Clinic Page. When you submit the form, it sends a POST request to `http://127.0.0.1:5555/add_clinic_page`, and the clinic's information gets added to the specified JSON file, in which the clinic will now be stored. So when you go to display all the clinics using the other GET API, it will now be displayed on the website and included in the JSON file. The API route is:

```
'/add-clinic-page' ->(POST request containing JSON data for a new clinic.)\
```

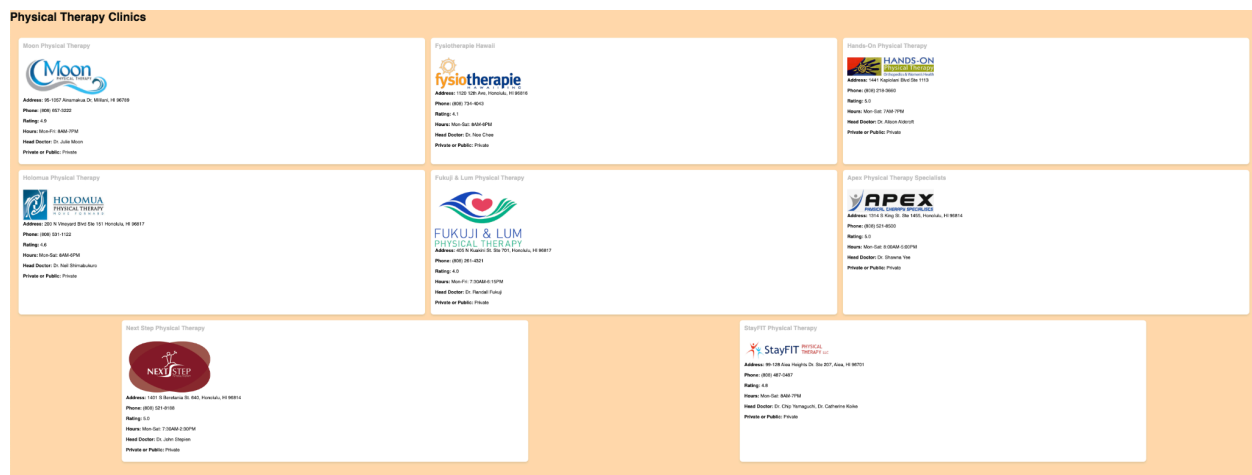
## Front-end

(Note: All Screenshots are reduced view size to show all information)

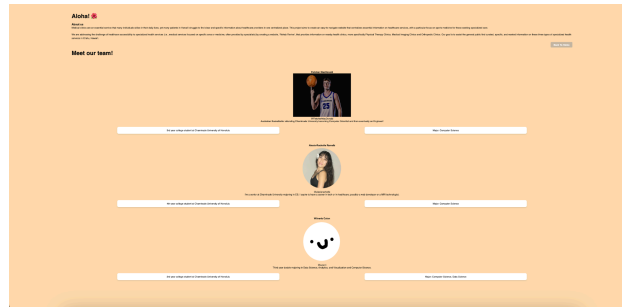
Homepage:



Clinic Pages (All clinic pages are the same layout just with their own information):



About Us Page:



Add Clinic Page:

## **Database**

Our website uses a JSON file instead of a database to store clinic information. A Python function reads the file, and Flask (API) sends the data to the HTML page to display it. When a user deletes or adds a clinic it is added to the JSON file. The json file consists of multiple keys that you see on the clinic pages.

## **Challenges**

Some of the main challenges we faced during this project were; organizing working on our sections of work and not clashing on github leading to multiple branches (we got better as it went on though), trouble with keys and making sure the html files were getting the right key names from the analyze\_data.py so the loop of the html could actually print out all the information and images, general css formatting issues (overlapping). The loops within the html to output the clinic information worked really well when we figured it out. The one css file also referenced the whole webpage so it was easy to have a similar look throughout the whole website. The main thing we learnt was how to reference the information we wanted to input on the web page, whether it be images, information etc. And outputting them to our html using Flask instead of manually writing all the information which is inefficient.

## **Future Improvements**

Future improvements that could be implemented could include; a proper database (SQL), people are able to add clinics by using clinics websites link, restrictions of what people can add, more advanced css styling.

### **Conclusion**

Overall, our project successfully created a functional and user-friendly website to display healthcare clinic information using JSON files and Flask. We learned a lot about API integration, frontend design, and working as a team. We found that this project made us much more knowledgeable on web page development.