# **Fraud Detection App Deployment**

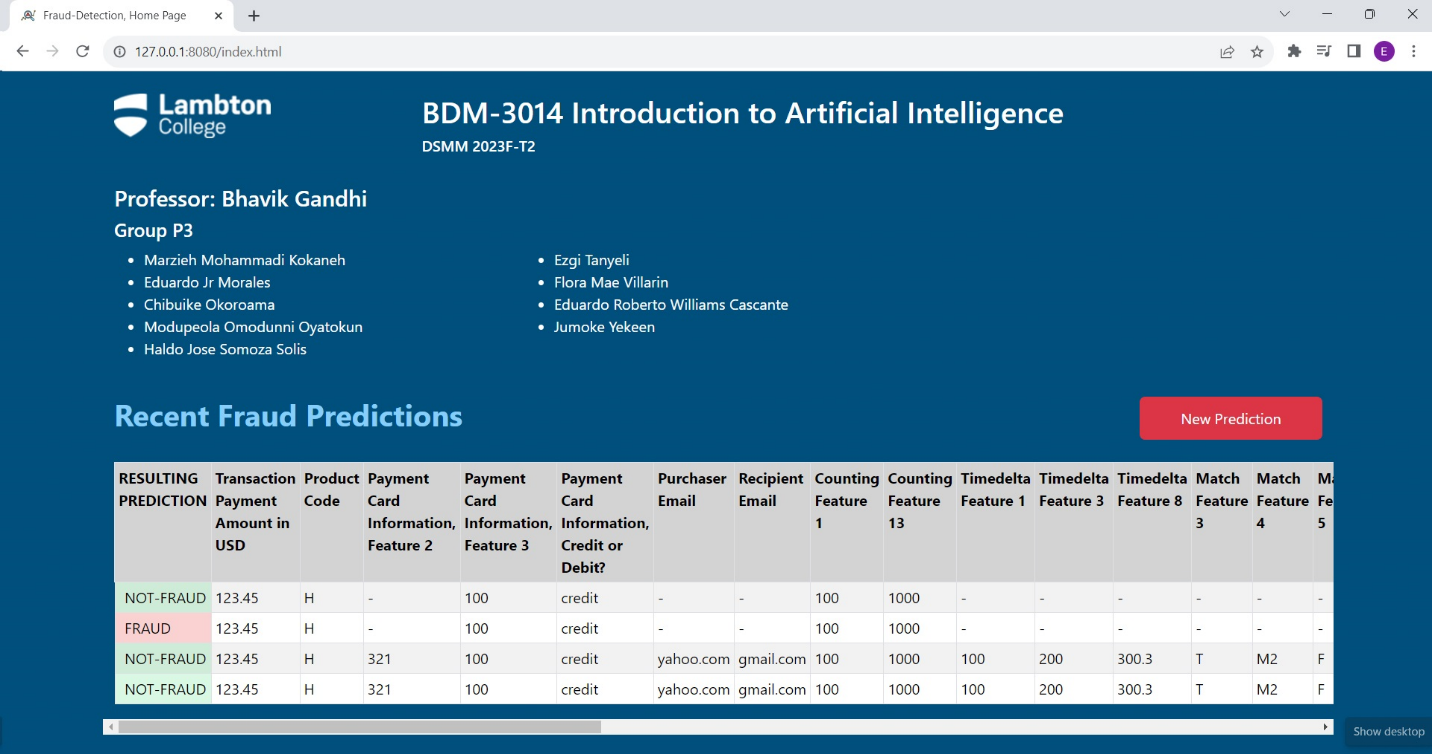
## **Frontend Component**

For the user interface we decided to build a customized website since it offers more flexibility to interact with the backend tier. This website consists of two pages, but it involves html, js, css, csv and json components.

Website URL: <https://fraud-detection-demo-5b6679a4c9d0.herokuapp.com>

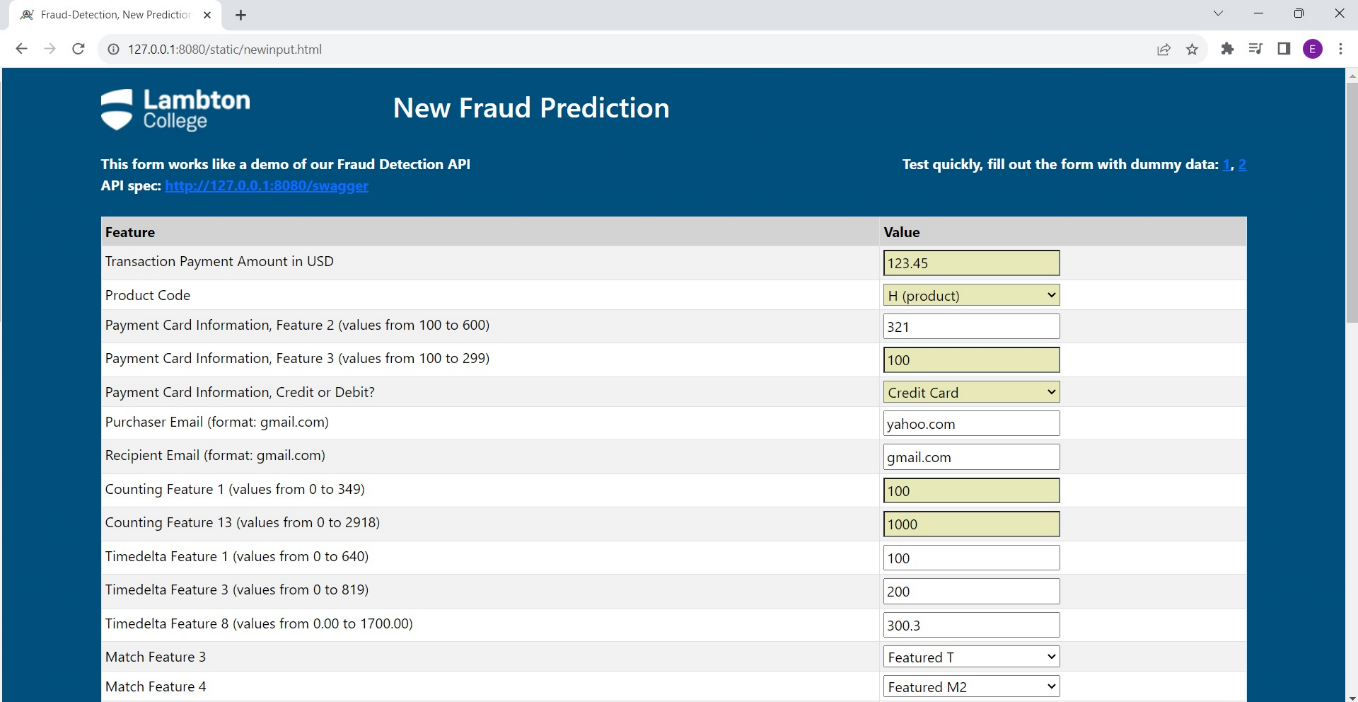
**Main Site Page**

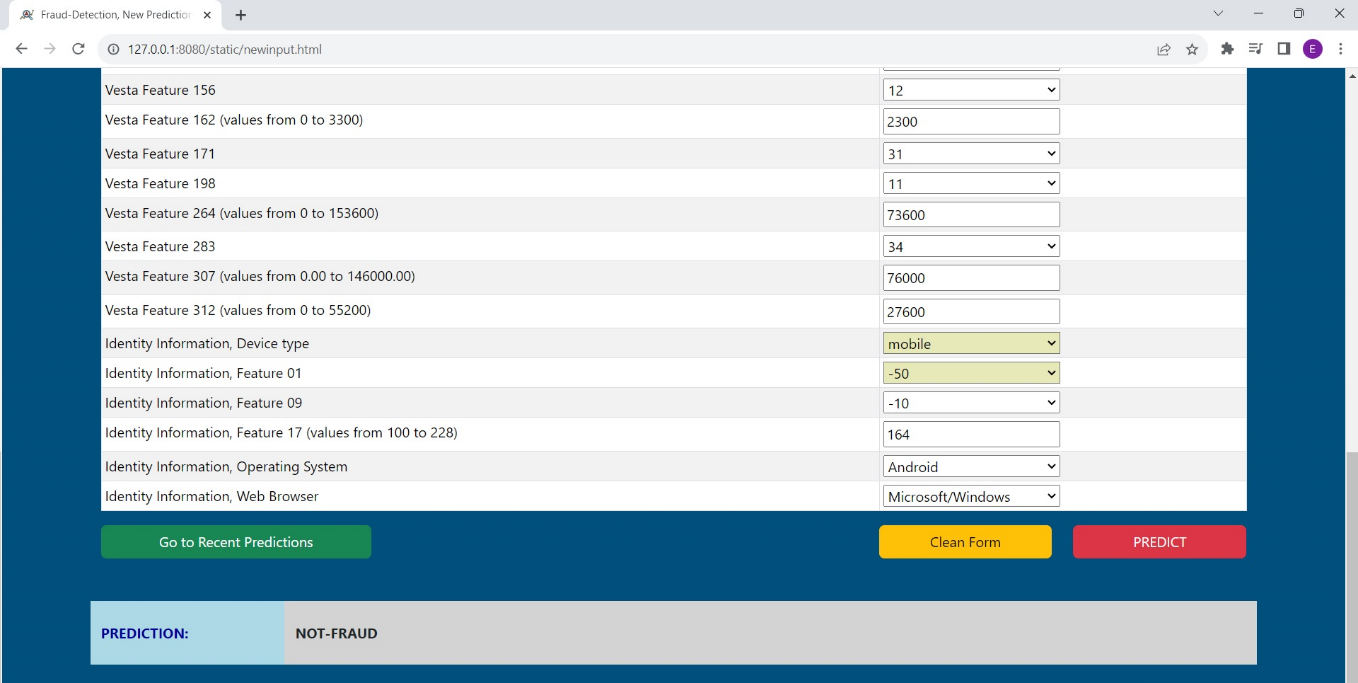
The main page contains information about the project and the work team members. In addition, it shows recent fraud predictions which are read from a csv file. Also, it contains a button to make a new prediction.



**New Fraud Prediction Page**

This page contains a form through which parameters (features) can be entered in order to perform a new prediction. When the page is loading, it reads a json file which has the configuration of the fields to build the form, therefore the form is automatically built depending on this configuration file. It means that we can add or remove fields depending on model requirements, without making changes to the html or java script code.





At the bottom, the page has some buttons for going back to the main page, cleaning the form fields, and making the new prediction. In addition, it has a label to show the prediction result.

**JSON Config File**

This file contains the fields configuration for building the form. In this file we must define all properties for each field, some of them are read from the html file and others are read from the orchestrator (python program). Among these properties are name of feature, type (text, numeric, list, datetime), maximum and minimum allowed values, information related to encoding, and so on.

|  |
| --- |

## **Backend Component**

For the backend layer we are combining some technologies, such as: Heroku, Flask, and Python.

**Heroku Platform**

Our solution is deployed over a cloud platform called Heroku, which offers PaaS, and additionally supports Flask and Python.

**Flask Framework**

In order for the frontend to invoke the models, which make the predictions, and receive the results, we have published an API Rest service. We have developed two python programs by using the Flask framework: app.py and predition.py.

App program is the orchestrator which contains the API Rest definition for receiving and responding to the frontend requests. In addition, it calls the prediction program.

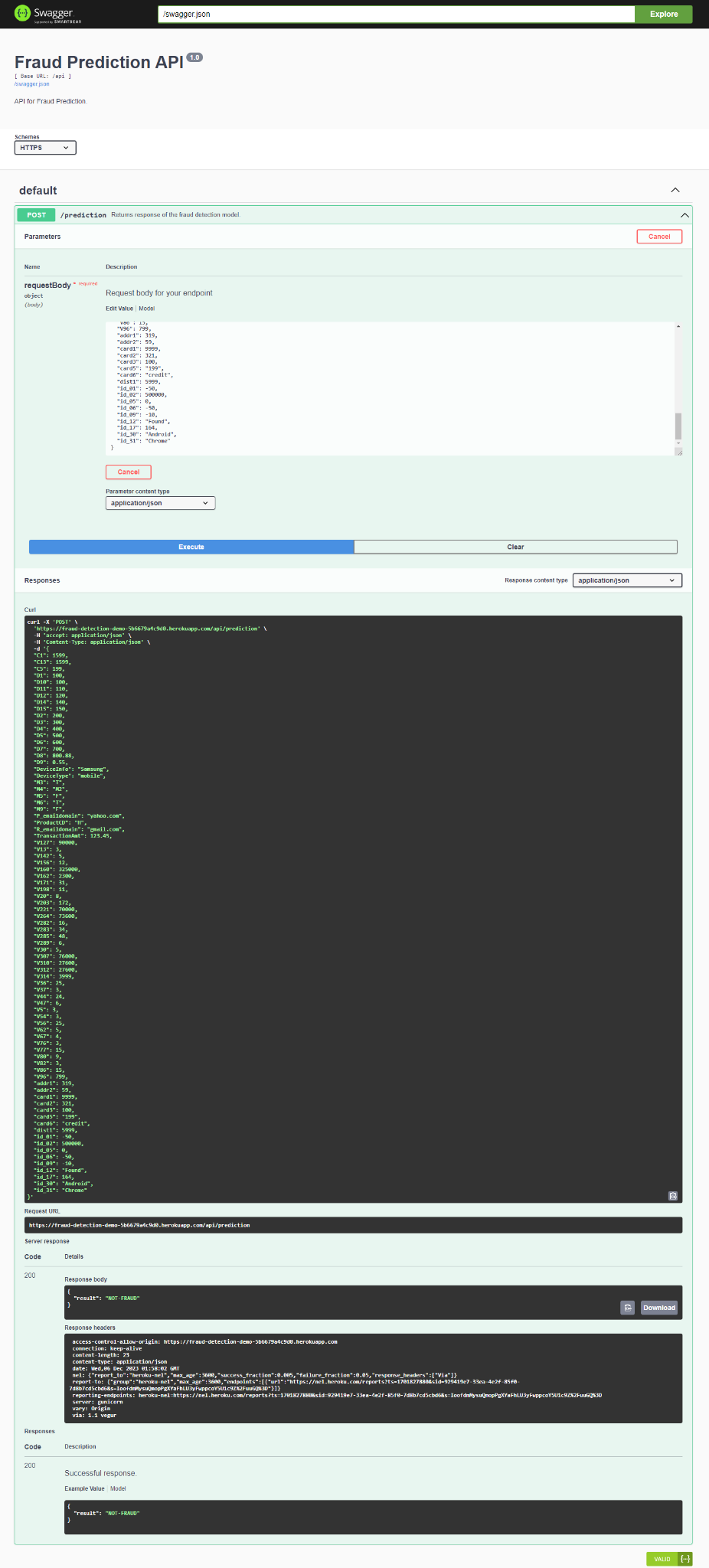
Prediction program validates the input parameters, encodes them, and finally sends them to the models, which perform the prediction as such, respond, and save the transaction information in the csv file. Models are compiled as pickle objects.

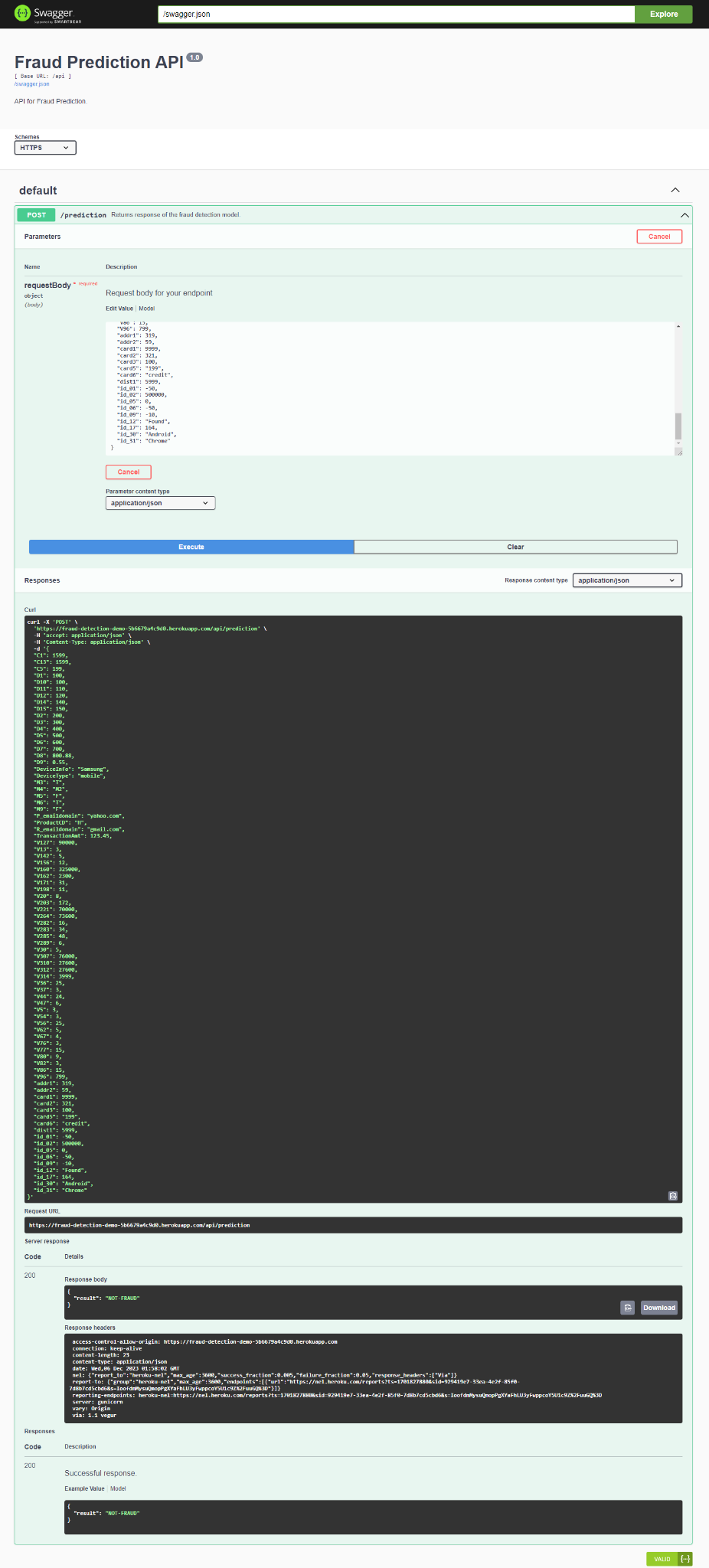
**API Rest Service**

One API REST is published into the same website, and their URL and specification are show below:

<https://fraud-detection-demo-5b6679a4c9d0.herokuapp.com/api/prediction>

<https://fraud-detection-demo-5b6679a4c9d0.herokuapp.com/swagger>





## **Project Work Table**

| Tasks | Subtasks | Status |
| --- | --- | --- |
| Features Engineering | Feature separation | Done |
|  | Standardization | Done |
|  | One Hot encoding | Done |
| Cleaning | Check for data type conversions | Done |
|  | Check for missing values | Done |
|  | Handle missing values | Done |
| Dimension Reduction | Handle data imbalance | Done |
|  | Perform dimension reduction | Done |
| Build Model and Evaluation | Developing first version of Model | Done |
|  | Testing | Done |
|  | Cross validation | Done |
|  | Implement Bagged or Boosted Ensemble | Done |
|  | Hyperparameter Tuning | Done |
|  | Testing of the (draft) Model | Done |
| Model Threshold Adjustment | Calculate optimal threshold for models | Done |
|  | Apply and evaluate the result | Done |
| Deployment of Model | Flask app development | Done |
|  | Html app development | Done |
|  | Server habilitation (Heroku) | Done |
|  | Publishing and configuration | Done |
|  | Site testing | Done/Pending |
|  | Style Part (CSS, Boostrap) | Done |
|  | Deployment documentation | Done |
| Source Versioning | Create an account on GitHub | Done |
|  | Create master, test, and dev branches | Done |
|  | Manage branches access | Done |
|  | GitHub dev branch commits | Done/Pending |
|  | GitHub test and master branch pull requests | Done/Pending |