The Duck Tracker – FreshWorks Studio Selection Test

By: Nilay Sondagar

Date: September 16, 2018

The Assignment

To create a web application for crowdsourcing data about ducks from around the world. The data points that the scientist needs to collect are the following:

- What time the ducks are fed
- What food the ducks are fed
- Where the ducks are fed
- How many ducks were fed
- How much food the ducks were fed

The collected data also needs to be presented in a user-friendly way to allow the scientist to evaluate the submitted data.

The Game Plan

Since the crowdsourced data was to be short, text-based points, I felt that the best way to collect the data was through the use of an HTML form. In addition to the form, the web application would need several other pages:

- **An introduction page:** Contains background on the why the data needed to be collected, and a short message from the scientist.
- **A form page:** Contains a form with fields for submitting the necessary data with error checking enabled on each field to ensure the submitted data was usable and valid.
- **A verification page:** Contains a short message confirming the user's form submission, as well as a link to see the submitted data.
- **A data viewing page:** Contains the submitted data in a table format, making the submissions easy to view and analyze.

Since the data was being crowdsourced, it was possible that some of the data that was submitted may be incorrect or invalid, which meant it was also important to include the option to delete entries. Finally, the application needed to be designed in a way that it was easily accessible on desktop and mobile.

Utilized Technologies

The web application was composed of several languages and services, which can be broken down into a three-tiered system.

Backend Stack

The backend of the web application consisted of an SQL database, using the PostgreSQL database system. I chose PostgreSQL as my database of choice because it happens to be one of the most common SQL databases, as well as one frequently used at FreshWorks Studio. PostgreSQL was also already installed as an add-on on my personal server, and it had well documented PHP commands which made it the obvious choice. This was my first time creating and using an SQL database from scratch!

Mid-tier (Backend Data Processing)

All of the data processing and webpage creation was done through PHP. The data submitted through the HTML form was sent to my PHP script, where it was then validated and entered into the PostgreSQL database. The fields were individually validated by stripping away unnecessary symbols, and checking for the correct type of input. For example, any fields where a numerical answer was expected, the PHP script would throw an error if any non-numerical values were submitted. If an error was thrown, the form page would be shown to the user with the appropriate error messages. During this process, all valid fields would keep their previously entered values, so the user only had to change the fields with invalid values.

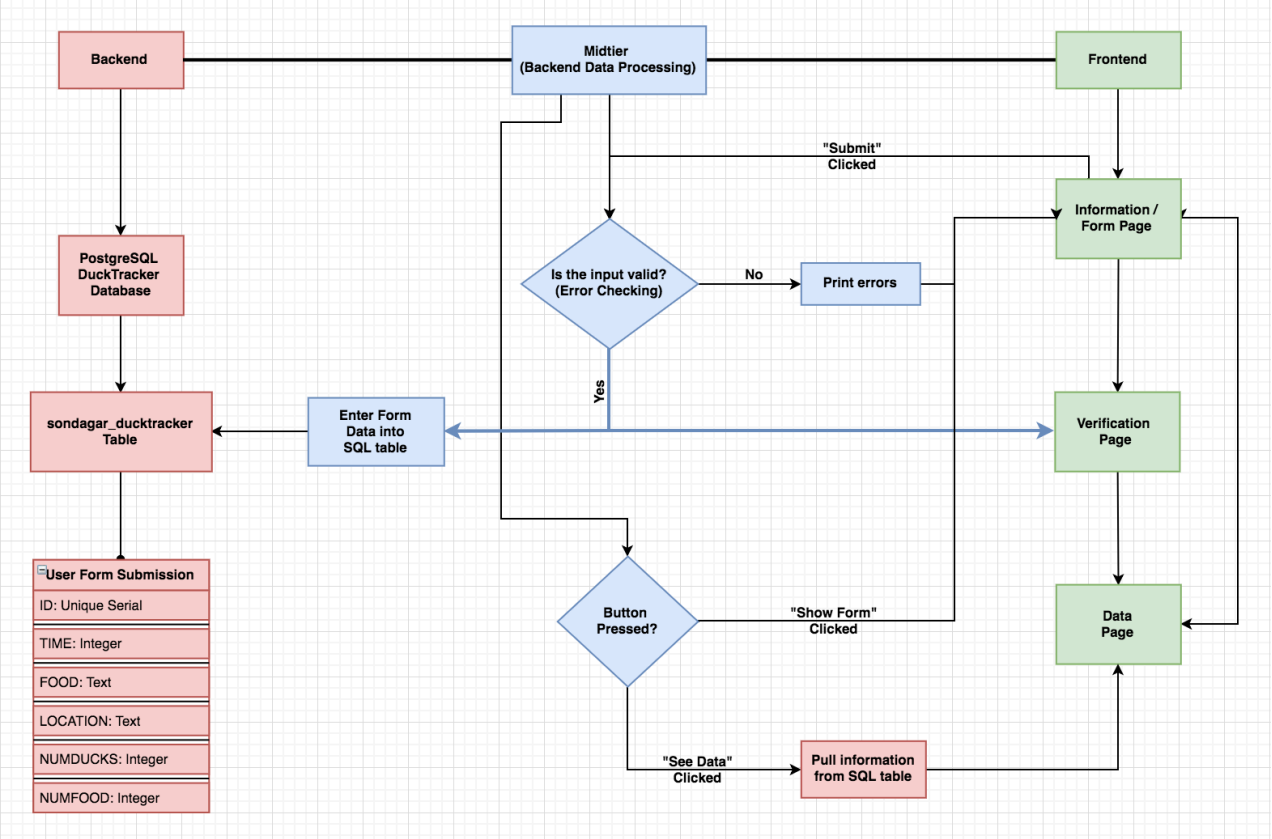
The data page was generated by PHP, used SQL queries to gather the information, and HTML to print out the data into a table. This table also included a “Delete” checkbox to allow submissions to be removed from the database. Under normal circumstances, that option would only be available to the scientist using a private login. Unfortunately, due to the time restraints on this assignment, this option is currently accessible to anyone who is viewing the page.

All navigation between pages (form page, validation page and data pages) were also handled by PHP, using GET requests submitted on button presses to determine what page should be shown. I chose to use PHP because it’s one of the languages that I have the most experience with, as well as the language I have made similar form-based websites with in the past.

Frontend (User Interface)

The user interface was created using pure HTML, CSS, and JavaScript. All the styles and dynamic resizing were handled by CSS, using relative lengths like *vh* and *vw* (viewport height & viewport width), and using media queries to determine the orientation of the viewing device. JavaScript was used to slide the form page in and out of hiding, and to close the form page when an orientation change was detected, to prevent the form page from sliding in the incorrect directions. I chose these languages because they are the ones I have the most experience with, and they allowed me to do everything required for this application.

High-Level Component Diagram



Database Model Diagram

The database is relatively simple, as it stores 5 snippets of user submitted data along with a unique row identifier in each row.

DuckTracker Database → **sondagar_ducktracker Table**

User Form Submission		
Identifier	**Type**	**Description**
ID	Serial Unique	Unique row identifier
TIME	Text	Time ducks are fed
FOOD	Text	Food ducks are fed
LOCATION	Text	Where the ducks are fed
NUMDUCKS	Integer	Number of ducks fed
NUMFOOD	Integer	Amount of food the ducks were fed (lb)

Time Commitment

I used the full 10 hours approved for this assignment to complete and test all aspects of the web application. However, due to the time restriction, there was one bug that I was unable to fix, as I discovered it at the end of my testing. When using the website on Android OS, the page attempts to resize itself when the virtual keyboard pops up, which makes the page appear to jump. The elements do resize themselves properly, however, it can be quite distracting.

Links

Git Repository: https://github.com/nilaysondagar/FreshWorks-Duck-WebApp/

Live Website: http://nilay.sondagar.com/DuckTracker/

Final Words

I very much enjoyed this assignment, as it gave me an opportunity to try out some new design elements I had been wanting to implement (like the slide-out card). It also allowed me to learn how to set up and use SQL databases. Thank you for taking the time to look over my selection test. I hope it was up to the FreshWorks standard, and I look forward to hearing from you soon!