Design Documentation

Alex(Zheng) Li
University of Victoria

1.0 Approach to the Problem

The very first step for me to approach to this problem is to read and understand all requirements. The project is a data collecting website that needs the database support so the next step for me is to write down what data needs to be collected. The user wants to collect duck data and can be analyzed for research study. I have an idea in my mind that the website can help users to visualize the collected data and a form where users can submit their data. I draw a low fidelity entity diagram for database with to help me figure out the layout and how many tables I need for this project. In addition, I draw a simple work flow diagram to help me layout the communication between front end, server, and database. I used technology that I am familiar with to implement this project, and the final products is deployed to Heroku.

2.0 Technology

2.1 Database

The database I used is MySQL because I have already set up MySQL server and workbench in my local machine. I also have experience with MySQL. I did a database course with MySQL at Uvic.

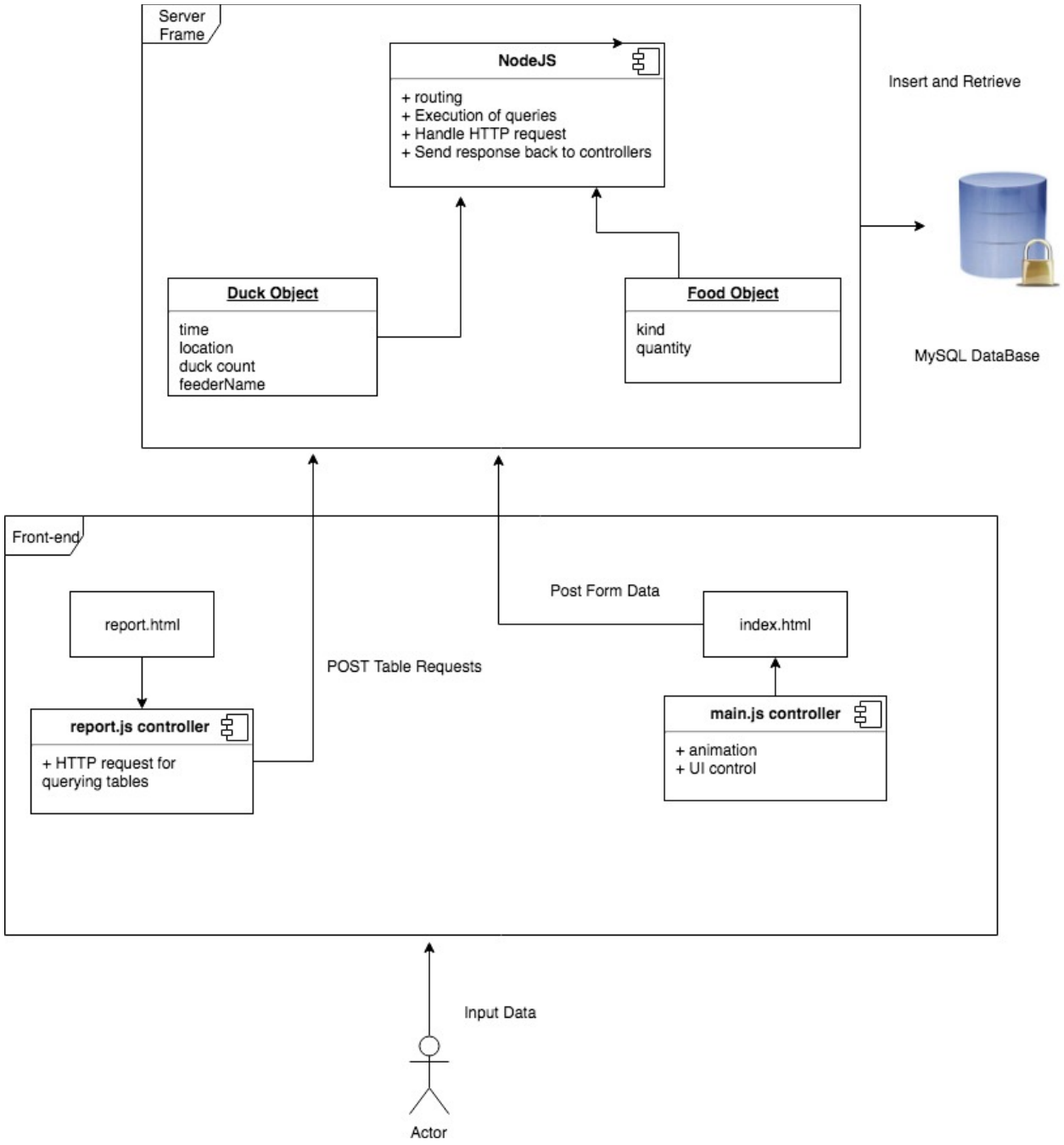
2.2 Server

I used NodeJS as my server side and ExpressJS as web application framework. NodeJS provides good usability to connect to MySQL database. NPM manager can get MySQL module that can execute queries. ExpressJS is easy to use for creating my API. It declares URLS to push and get data from database by http router. The front-end post request can also be handled and referenced by corresponding endpoint. I choose these two instead of using PHP for server side is because these technologies have great usability, and easy for me to set up.

2.3 Front-End

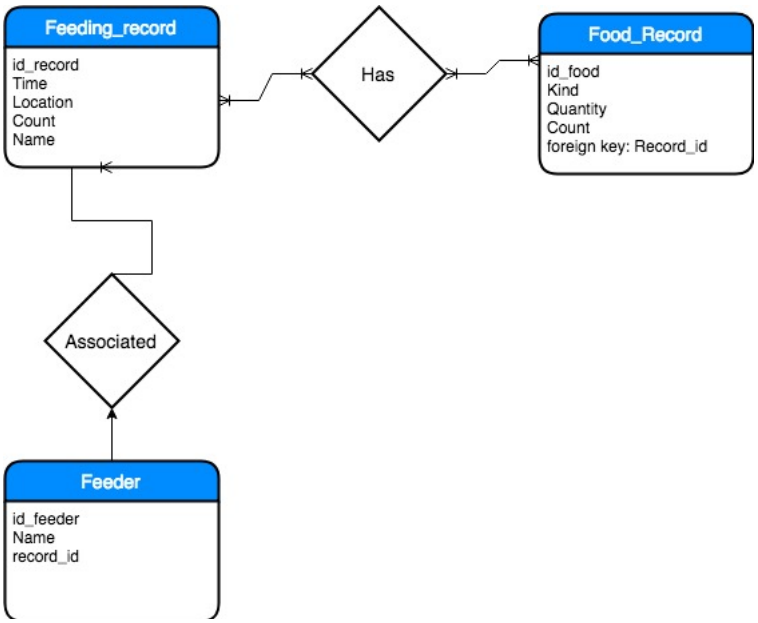
The technology I used for front-end are HTML5, CSS and bootstrap CSS framework since I think a form data collection page does not need view engine to parse data so I want to keep it simple.

3.0 Component Diagram



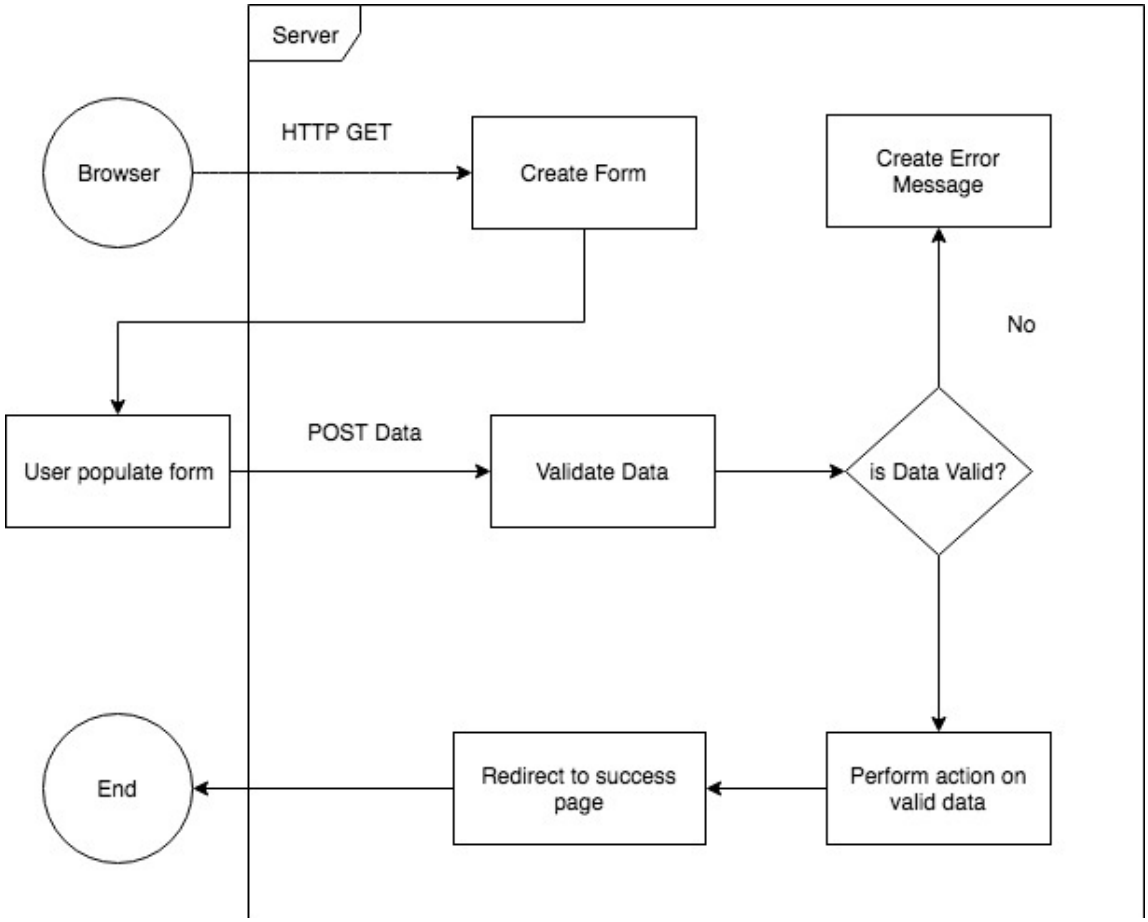
The front end consists of two pages, report.html and index.html. Both pages have corresponding controllers to handles users' request. When users click the submit button at index.html page, it sends all data by POST HTTP request to nodeJS server. The server will parse the data by body-parser and insert the data into MySQL database. If users want to retrieve data from database, the front end will also trigger a POST command to server by specific endpoint. Then the nodeJS server will query the data according to different endpoint. The data will be converted into JSON object and pass to controller. The controller will read the data, generate a table and display to the view.

5.0 Database Model Diagram



The database schema consists of three tables initially. Feeding_record table has one-to-one relationship with Food_Record. The foreign key Record_id in Food_Record table can reference to Feeding_record so that these two tables can be joined together. The feeder table has one to many relationship with feeding_record. However, I did not build this table for actual implementation since I cannot come up the reasoning why I create this table. Instead, I add the name attribute for feeding_record table. I can query the person's feeding record easily. The design of database is the part that I had troubles with because I am not entirely sure if it is a good design.

5.0 Work Flow Diagram



6.0 Worklog

Duration	Activities
August 14 4 hours	Set up NodeJS Server and Database Schema. Finish the view for index page, report page, its controller.
August 15 4.5 hours	Finish up all API that can generate tables. Fix UI bugs and routing bugs
August 16 3 hours	Deploy the application to Heroku, change configuration of database for Heroku, and fix disconnection issue with Database