Duck Feeding – Crowdsourcing

Submitted by Avinash Gazula (avinashgazula@gmail.com)

Requirements

The assignment requires the development of the application that collects crowdsourcing information related to the feeding habits of ducks around the world. The web application is used to submit these data points and to view the submitted data

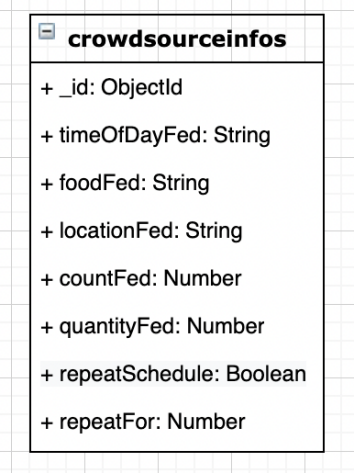
Technologies Used

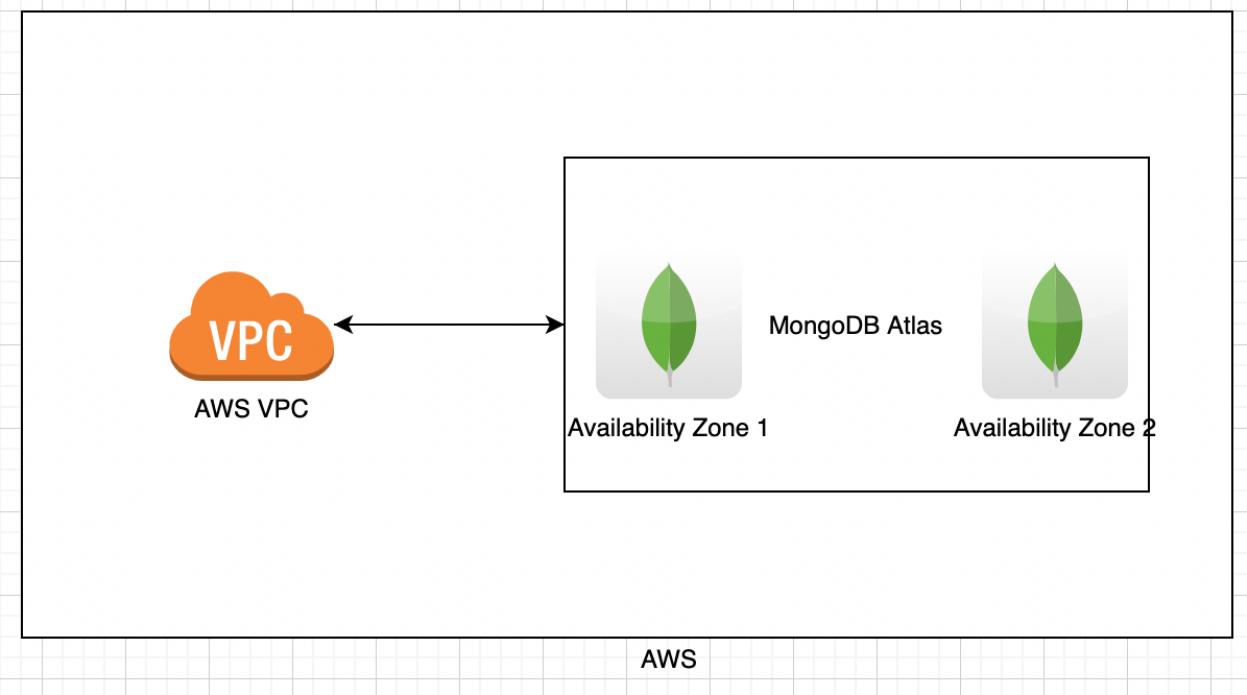
Client: React.js is used to develop the client-side application. React and Angular are the most popular JavaScript development frameworks. React was chosen over Angular because of its Virtual DOM which improves performance and speeds up the development process as it does not have to re-compile on every render. Bootstrap is used for styling and to make the application fully responsive

Server: Node.js along with express is used to develop the server side of the application because of its scalability and performance along with extensive third-party developer community

Database: A NoSQL MongoDB database is used because the requirements would work perfectly with a document database and it allows the application to handle data of all sorts in a scalable way

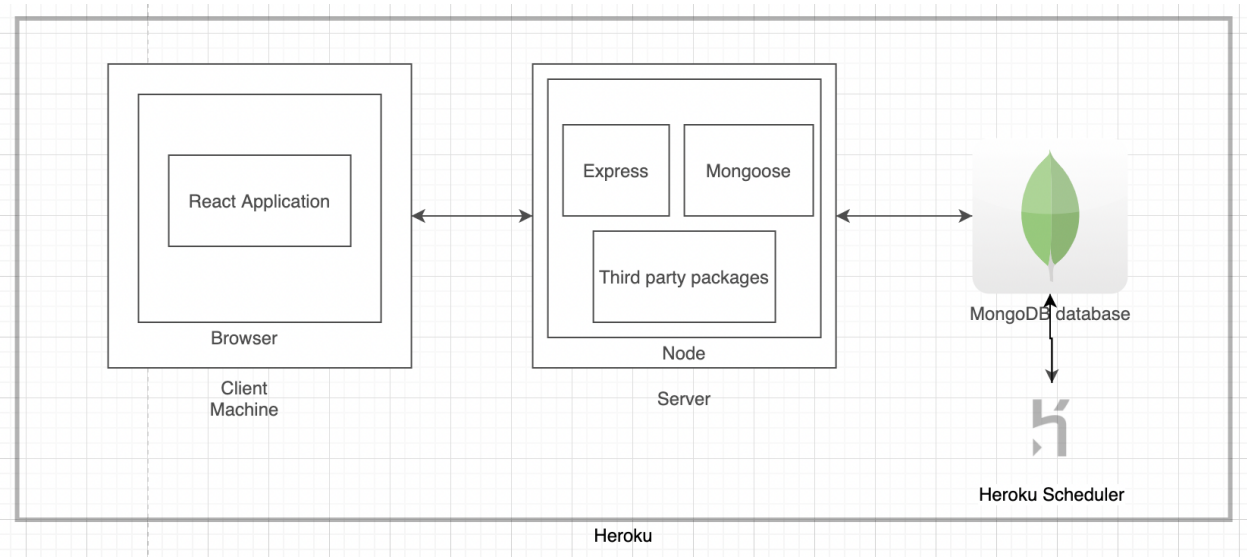
Database Model





The MongoDB clusters are hosted on multiple availability zones on AWS and a Virtual Private Cloud (VPC) is used to access these resources and MongoDB Atlas is used to monitor the cluster

High Level Component Diagram and Approach



The react client application sends API requests to the Node Server which uses express and other third-party libraries. Mongoose is the MongoDB driver used to connect to the database. Axios is used to send API requests to the server. There are react components that allow submitting data which then send a POST request to the server, which connects to the MongoDB instance using a mongoose driver. The component to view responses sends a GET request to the server which then connects to the database to retrieve all the responses and then displays them. React Context API is used to manage context variables across components.
The scheduling feature is implemented in multiple ways, while entering the responses, an option is asked to add entries for multiple days, or a scheduler job is run on Heroku that checks all entries that should be repeated and re-enters them every day. The job runs at 12 AM UTC every night to prevent multiple entries to be run every day.

Testing

Unit tests are written for all routes on the server side and tests are written for all component on the client side

Security

Security features such as rate limiting, and API authentication are implemented to prevent abuse. The rate limiter is applied to all API requests but not to the client side so the website is always accessible. All API requests are authenticated using json web token. The client has a bearer token which is used to authenticate the requests, so the server only accepts requests only from the client.

Time Taken

Total Time taken: ~5 hours
Time for server: -1.5 hours
Time for client: ~2 hours
Time for styling, deployment and documentation: ~1.5 hours

Access

Repository: https://github.com/avinashgazula/duck-crowdsourcing.git
Heroku Production URL: https://duck-feeding-crowdsourcing.herokuapp.com/