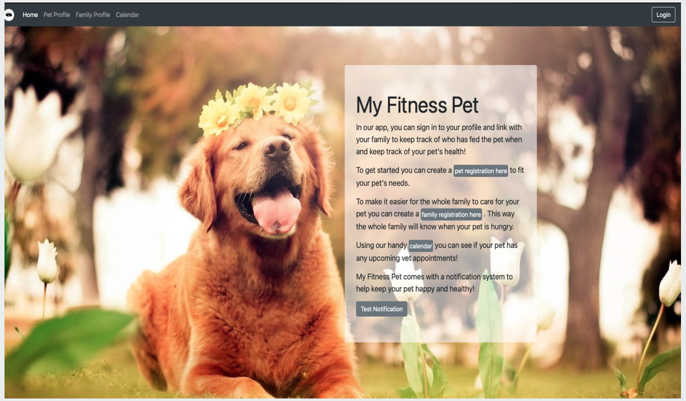
Feed the Pets

CSCI 3308 Final Presentation



My Fitness Pet

In our app, you can sign in to your profile and link with your family to keep track of who has fed the pet when and keep track of your pet's health!

To get started you can create a pet registration here to fit your pet's needs.

To make it easier for the whole family to care for your pet you can create a family registration here . This way the whole family will know when your pet is hungry.

Using our handy calendar you can see if your pet has any upcoming vet appointments!

My Fitness Pet comes with a notification system to help keep your pet happy and healthy!

Leo Sipowicz, Ayden Smith, Adam Trunko, Catherine Xiao, Josh Ziebold

Background

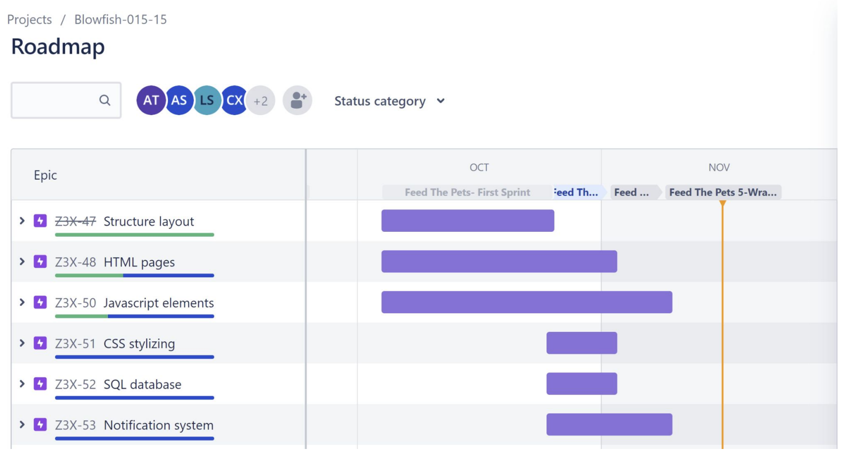
Feed the Pets is a database application that allows users to register a pet, keep track of his or her dietary scheduling, and receive notifications.

Tools Overview

- Features:
    - User registration / login
    - Feeding notifications
    - Organization calendar / history
    - Pet /user profile page
- Tools
    - Jira, Github, PostgreSQL, VS Code, Docker, NodeJS, Google Calendar API
- Methodologies
    - Agile, Peer Code Reviews, Pair Programming,

Application Tools - Jira Software

- Used as a Project Tracker (Agile)
- Team organization software
    - What was completed
    - What needed to be done
    - Who was completing what task
    - Roadmap (project calendar)



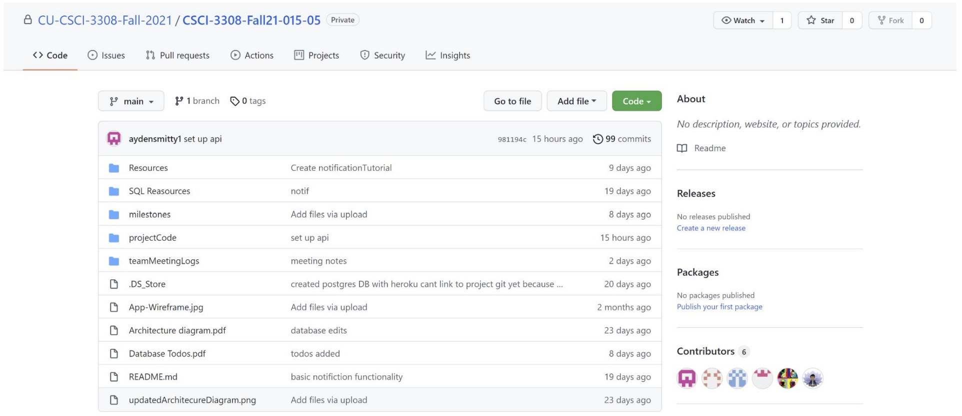
Projects / Blowfish-015-15

Roadmap

Epic
~~Z3X-47~~ Structure layout
Z3X-48 HTML pages
Z3X-50 Javascript elements
Z3X-51 CSS stylizing
Z3X-52 SQL database
Z3X-53 Notification system

Application Tools - Github ☆☆☆☆☆

- VCS Repository (Peer code review)
- Used to store project updates
- Includes all of the team's commits
    - Milestones
    - Code
    - Resources



CU-CSCI-3308-Fall-2021 / CSCI-3308-Fall21-015-05 Private

1 branch 0 tags

aydensmitty1 set up api 981194c 15 hours ago 99 commits

Resources	Create notificationTutorial	9 days ago
SQL Reasources	notif	19 days ago
milestones	Add files via upload	8 days ago
projectCode	set up api	15 hours ago
teamMeetingLogs	meeting notes	2 days ago
.DS_Store	created postgres DB with heroku cant link to project git yet because ...	20 days ago
App-Wireframe.jpg	Add files via upload	2 months ago
Architecture diagram.pdf	database edits	23 days ago
Database Todos.pdf	todos added	8 days ago
README.md	basic notifiction functionality	19 days ago
updatedArchitecureDiagram.png	Add files via upload	23 days ago

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 6

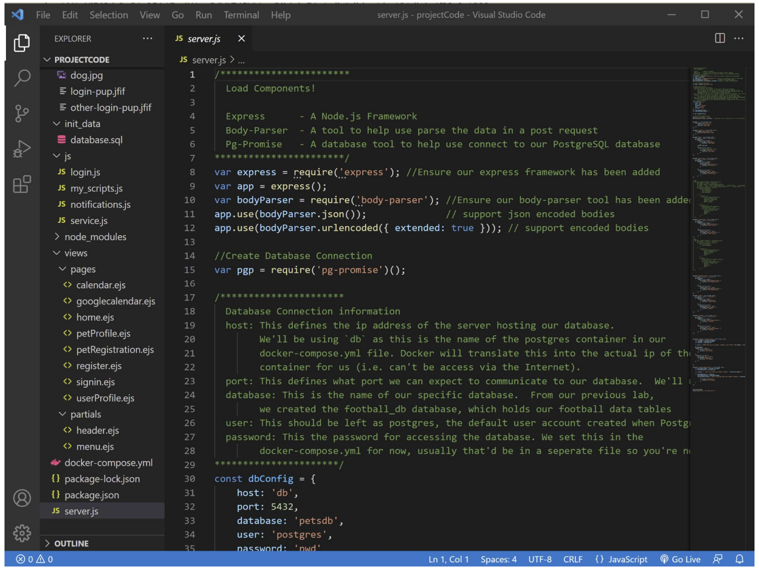
Application Tools - PostgreSQL

- Database management system
- User data
- Pets data
- Linked with familyID

Application Tools - VS Code

- Coding environment (Pair programming)
- Used to create the application
    - HTML, CSS, ejs, docker, database

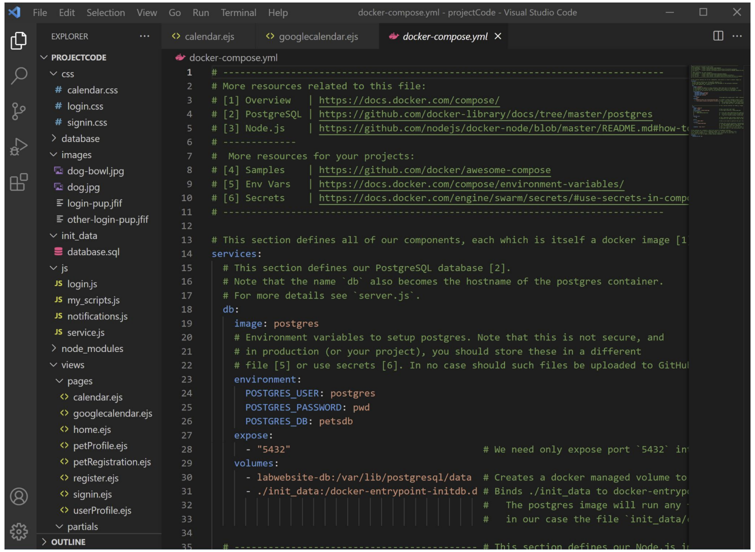
Visual Studio Code



```
/**********************
  Load Components!

  Express       - A Node.js Framework
  Body-Parser   - A tool to help use parse the data in a post request
  Pg-Promise    - A database tool to help use connect to our PostgreSQL database
**********************/
var express = require('express'); //Ensure our express framework has been added
var app = express();
var bodyParser = require('body-parser'); //Ensure our body-parser tool has been adde
app.use(bodyParser.json());              // support json encoded bodies
app.use(bodyParser.urlencoded({ extended: true })); // support encoded bodies

//Create Database Connection
var pgp = require('pg-promise')();

/**********************
  Database Connection information
  host: This defines the ip address of the server hosting our database.
        We'll be using `db` as this is the name of the postgres container in our
        docker-compose.yml file. Docker will translate this into the actual ip of th
        container for us (i.e. can't be access via the Internet).
  port: This defines what port we can expect to communicate to our database.  We'll
  database: This is the name of our specific database.  From our previous lab,
        we created the football_db database, which holds our football data tables
  user: This should be left as postgres, the default user account created when Postg
  password: This the password for accessing the database. We set this in the
        docker-compose.yml for now, usually that'd be in a seperate file so you're n
**********************/
const dbConfig = {
    host: 'db',
    port: 5432,
    database: 'petsdb',
    user: 'postgres',
    password: 'pwd'
```

Application Tools - Docker ☆☆☆

- Deployment environment
- Stores the SQL database



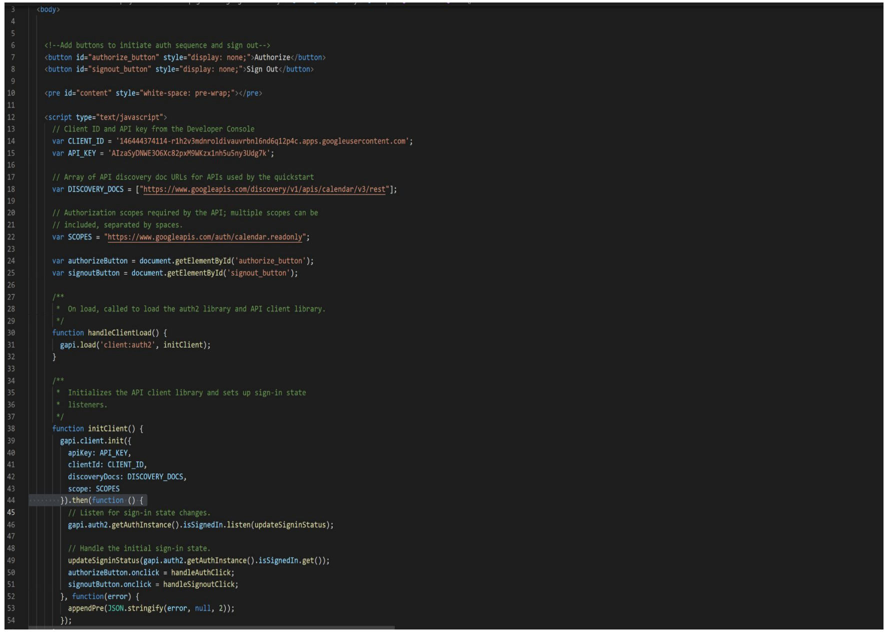
docker-compose.yml

```yaml
# ----------------------------------------------------------------------------
# More resources related to this file:
# [1] Overview   | https://docs.docker.com/compose/
# [2] PostgreSQL | https://github.com/docker-library/docs/tree/master/postgres
# [3] Node.js    | https://github.com/nodejs/docker-node/blob/master/README.md#how-t
# ------------
#  More resources for your projects:
# [4] Samples    | https://github.com/docker/awesome-compose
# [5] Env Vars   | https://docs.docker.com/compose/environment-variables/
# [6] Secrets    | https://docs.docker.com/engine/swarm/secrets/#use-secrets-in-comp
# ----------------------------------------------------------------------------

# This section defines all of our components, each which is itself a docker image [1
services:
  # This section defines our PostgreSQL database [2].
  # Note that the name `db` also becomes the hostname of the postgres container.
  # For more details see `server.js`.
  db:
    image: postgres
    # Environment variables to setup postgres. Note that this is not secure, and
    # in production (or your project), you should store these in a different
    # file [5] or use secrets [6]. In no case should such files be uploaded to GitHu
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: pwd
      POSTGRES_DB: petsdb
    expose:
      - "5432"                                 # We need only expose port `5432` in
    volumes:
      - labwebsite-db:/var/lib/postgresql/data  # Creates a docker managed volume to
      - ./init_data:/docker-entrypoint-initdb.d # Binds ./init_data to docker-entryp
                                                #   The postgres image will run any
                                                #   in our case the file `init_data/

  # ----------------------------------------- # This section defines our Node.js i
```

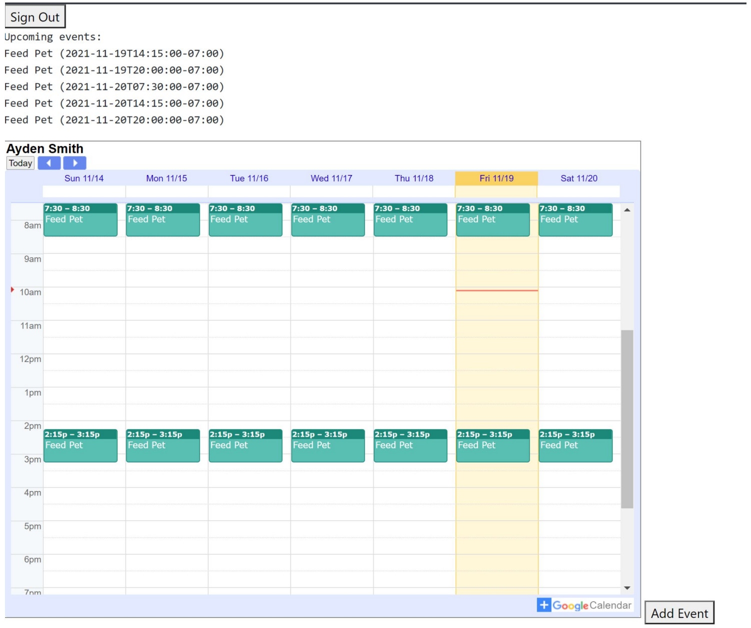
Application Tools - Node JS

- Framework
- Stores the SQL database and connect the backend to frontend

Application Tools - Google Calendar API ☆☆☆



```html
<body>


  <!--Add buttons to initiate auth sequence and sign out-->
  <button id="authorize_button" style="display: none;">Authorize</button>
  <button id="signout_button" style="display: none;">Sign Out</button>

  <pre id="content" style="white-space: pre-wrap;"></pre>

  <script type="text/javascript">
    // Client ID and API key from the Developer Console
    var CLIENT_ID = '146444374114-r1h2v3mdnroldivauvrbnl6nd6q12p4c.apps.googleusercontent.com';
    var API_KEY = 'AIzaSyDNWE3O6Xc82pxM9WKzx1nh5u5ny3Udg7k';

    // Array of API discovery doc URLs for APIs used by the quickstart
    var DISCOVERY_DOCS = ["https://www.googleapis.com/discovery/v1/apis/calendar/v3/rest"];

    // Authorization scopes required by the API; multiple scopes can be
    // included, separated by spaces.
    var SCOPES = "https://www.googleapis.com/auth/calendar.readonly";

    var authorizeButton = document.getElementById('authorize_button');
    var signoutButton = document.getElementById('signout_button');

    /**
     *  On load, called to load the auth2 library and API client library.
     */
    function handleClientLoad() {
      gapi.load('client:auth2', initClient);
    }

    /**
     *  Initializes the API client library and sets up sign-in state
     *  listeners.
     */
    function initClient() {
      gapi.client.init({
        apiKey: API_KEY,
        clientId: CLIENT_ID,
        discoveryDocs: DISCOVERY_DOCS,
        scope: SCOPES
      }).then(function () {
        // Listen for sign-in state changes.
        gapi.auth2.getAuthInstance().isSignedIn.listen(updateSigninStatus);

        // Handle the initial sign-in state.
        updateSigninStatus(gapi.auth2.getAuthInstance().isSignedIn.get());
        authorizeButton.onclick = handleAuthClick;
        signoutButton.onclick = handleSignoutClick;
      }, function(error) {
        appendPre(JSON.stringify(error, null, 2));
      });
```

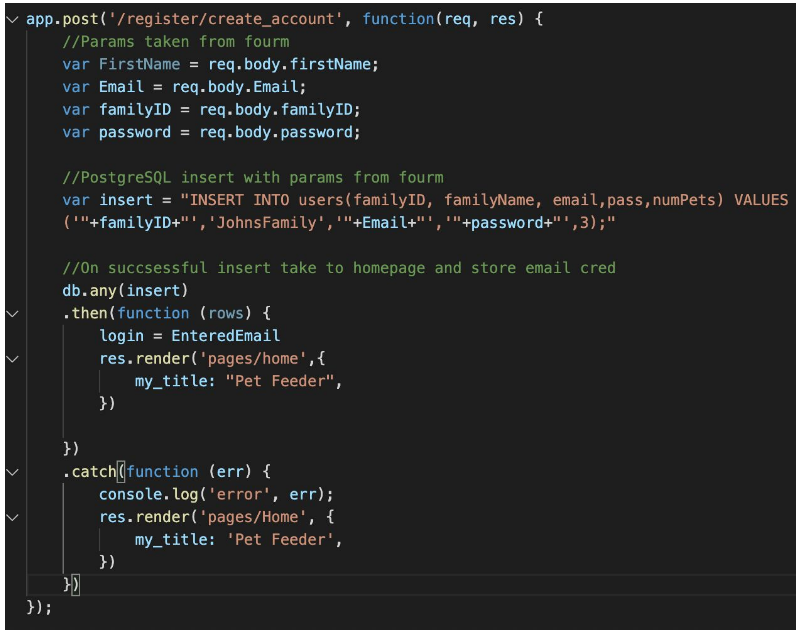


```
Upcoming events:
Feed Pet (2021-11-19T14:15:00-07:00)
Feed Pet (2021-11-19T20:00:00-07:00)
Feed Pet (2021-11-20T07:30:00-07:00)
Feed Pet (2021-11-20T14:15:00-07:00)
Feed Pet (2021-11-20T20:00:00-07:00)
```

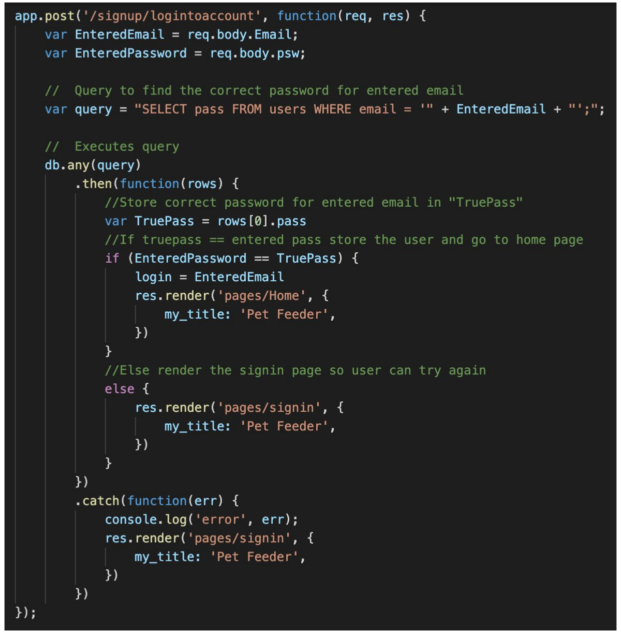
Ayden Smith

	Sun 11/14	Mon 11/15	Tue 11/16	Wed 11/17	Thu 11/18	Fri 11/19	Sat 11/20
8am	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet	7:30 – 8:30 Feed Pet
9am							
10am							
11am							
12pm							
1pm							
2pm	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet	2:15p – 3:15p Feed Pet
3pm							
4pm							
5pm							
6pm							
7pm							

User Authentication



```
app.post('/register/create_account', function(req, res) {
    //Params taken from fourm
    var FirstName = req.body.firstName;
    var Email = req.body.Email;
    var familyID = req.body.familyID;
    var password = req.body.password;

    //PostgreSQL insert with params from fourm
    var insert = "INSERT INTO users(familyID, familyName, email,pass,numPets) VALUES
    ('"+familyID+"','JohnsFamily','"+Email+"','"+password+"',3);"

    //On succsessful insert take to homepage and store email cred
    db.any(insert)
    .then(function (rows) {
        login = EnteredEmail
        res.render('pages/home',{
            my_title: "Pet Feeder",
        })

    })
    .catch(function (err) {
        console.log('error', err);
        res.render('pages/Home', {
            my_title: 'Pet Feeder',
        })
    })
});
```

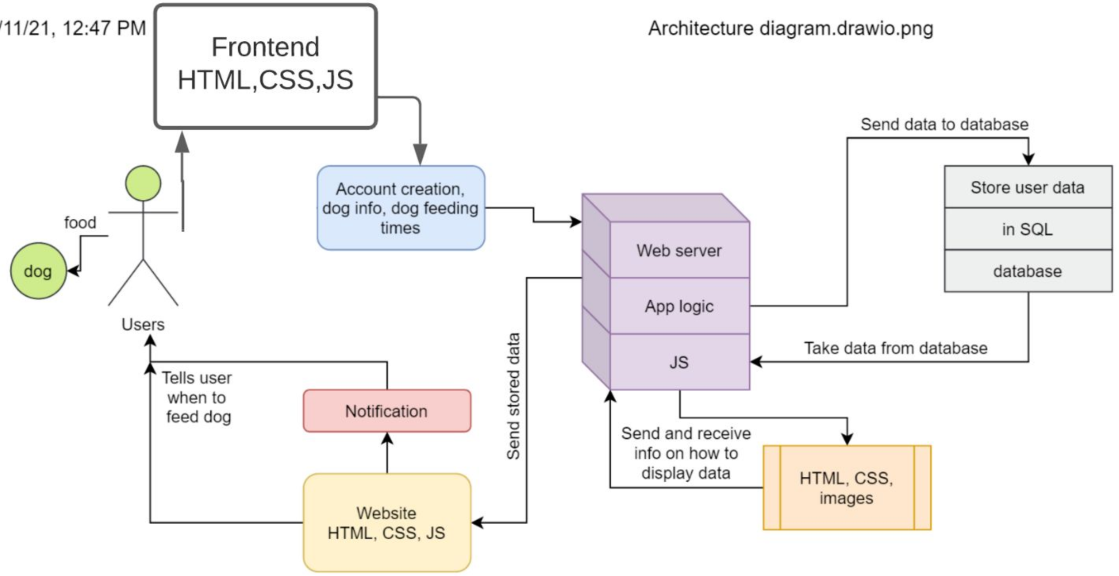


```
app.post('/signup/logintoaccount', function(req, res) {
    var EnteredEmail = req.body.Email;
    var EnteredPassword = req.body.psw;

    //  Query to find the correct password for entered email
    var query = "SELECT pass FROM users WHERE email = '" + EnteredEmail + "';";

    //  Executes query
    db.any(query)
        .then(function(rows) {
            //Store correct password for entered email in "TruePass"
            var TruePass = rows[0].pass
            //If truepass == entered pass store the user and go to home page
            if (EnteredPassword == TruePass) {
                login = EnteredEmail
                res.render('pages/Home', {
                    my_title: 'Pet Feeder',
                })
            }
            //Else render the signin page so user can try again
            else {
                res.render('pages/signin', {
                    my_title: 'Pet Feeder',
                })
            }
        })
        .catch(function(err) {
            console.log('error', err);
            res.render('pages/signin', {
                my_title: 'Pet Feeder',
            })
        })
});
```

Architecture Diagram

10/11/21, 12:47 PM

Architecture diagram.drawio.png



Challenges

1) We had some trouble integrating the postgreSQL database with the frontend
2) Implementing the Google Calendar API was a challenge
    a) Internal website URL wasn’t working, had to implement API using quickstart JS
3) Making sure login and user info is secure/web security threats
4) Implementing push notifications proved to be much harder than initially expected and we ended up having to settle for local notifications for the time being.
    a) Resolved the challenges with google calendar API by doing the quickstart JS demo and it appears in our web app now. This enables the user to add notifications instead of using push notifications and is more convenient. We were able to get our database working and registration with user input working

Demo

http://localhost:3001/

In Case of Emergency: https://vimeo.com/647818382/5410b04385

Questions?