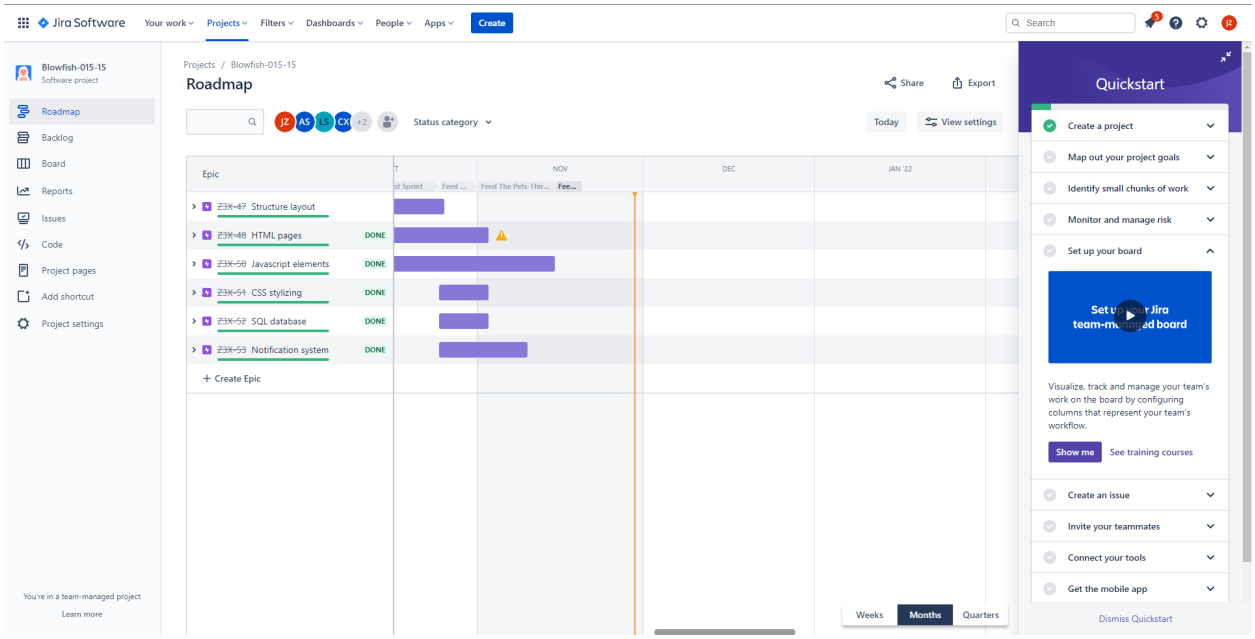
Title: Feed the pets/My Fitness Pet

Who: Catherine, Leo, Josh, Adam, Ayden

Project Description: Our project, Feed the Pets, or, My Fitness Pet (we didn't really decide on a name), is a web application designed to make it simple and easy to keep track of your pet's health. When you get to our website you are prompted to create an account, this account will hold information about what pets you have. These accounts can be shared so that you can keep track with your family to know who was the last person to feed the pets. On top of that each pet will have their own profile which will keep track of how much they need to be fed and when, it will also hold other important information about them. On our home page you will find intuitive navigation to each of the pages such as the calendar. We used Google calendar API to create a calendar that you can use to keep track of all your pets feeding times and important dates like vet appointments. We also have a notification system in place so that if you are busy you will still be reminded when it's time to feed the pets.

Project Tracker:

https://csci-3308-fall21-015-15.atlassian.net/jira/software/projects/Z3X/boards/1/roadmap

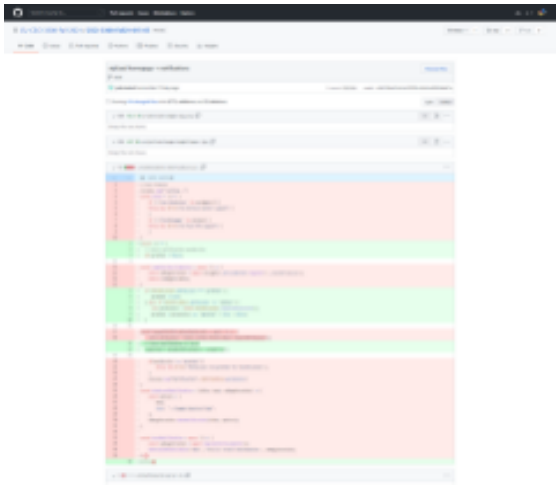


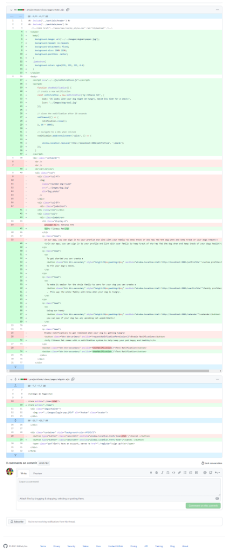
Video: https://vimeo.com/647818382/5410b04385

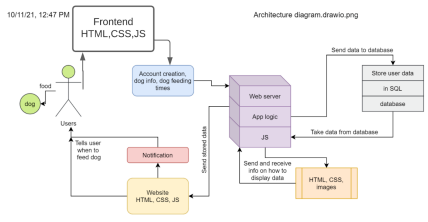
VCS: https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-015-05

Contributions:

Josh: I created the architecture diagram in order to give some structure to our web app. I used an open source diagram creation tool to make the diagram. I spent a lot of time trying to get push notifications to work with our website but ended up having to settle with local notifications because of some issues I was having with the server. I just used the built in notifications that come with nodeJS. I stylized the frontpage to make it look more professional using bootstrap and CSS.

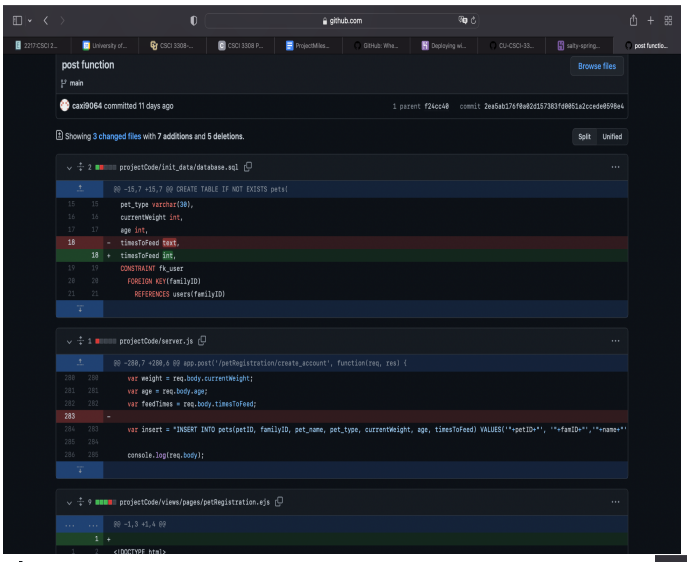


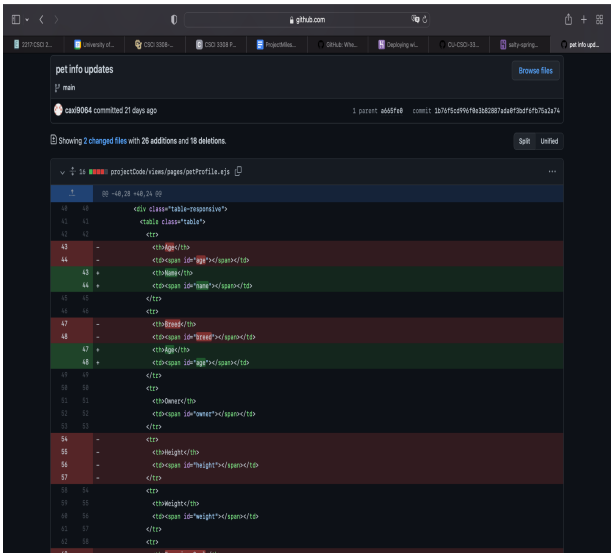


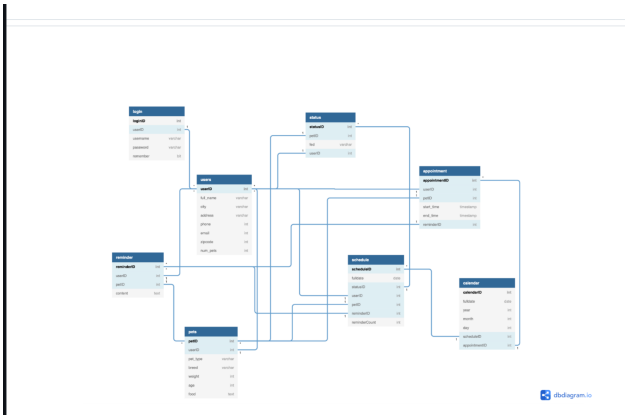


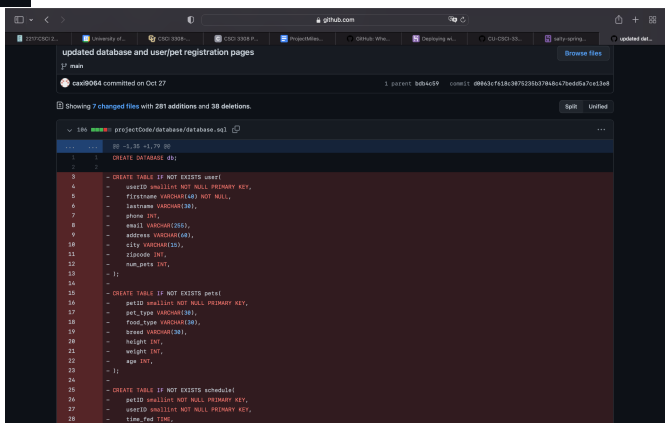
Catherine: Created database in the sql file and designed it using the db diagram app to help store the user info, pet info, and registration info. We did not need to use all the data tables displayed in the diagram and for the most part the calendar tables were not needed. Collaborated with Ayden and Leo about the backend. I added a html pet profile page to display

the pets info. I worked on getting the pet registration to connect with the database and display some information initially. There were difficulties initially with getting the proper tabs to match but Ayden was able to help with that.



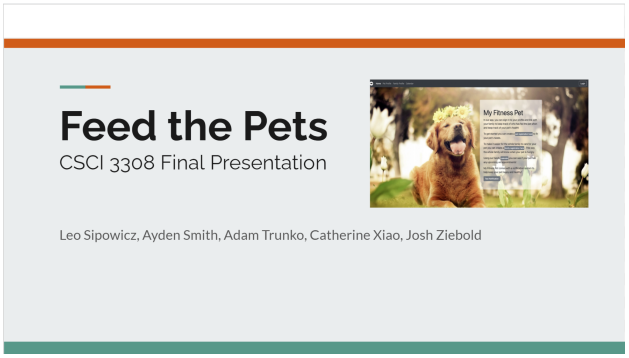


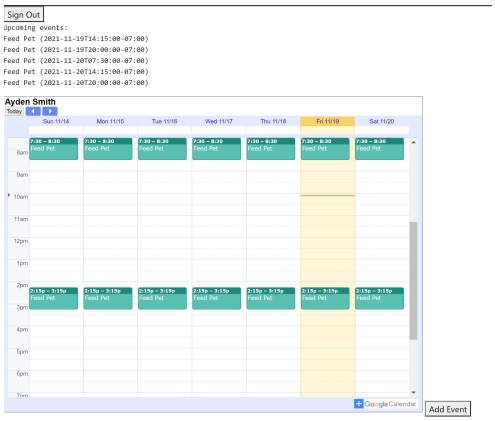


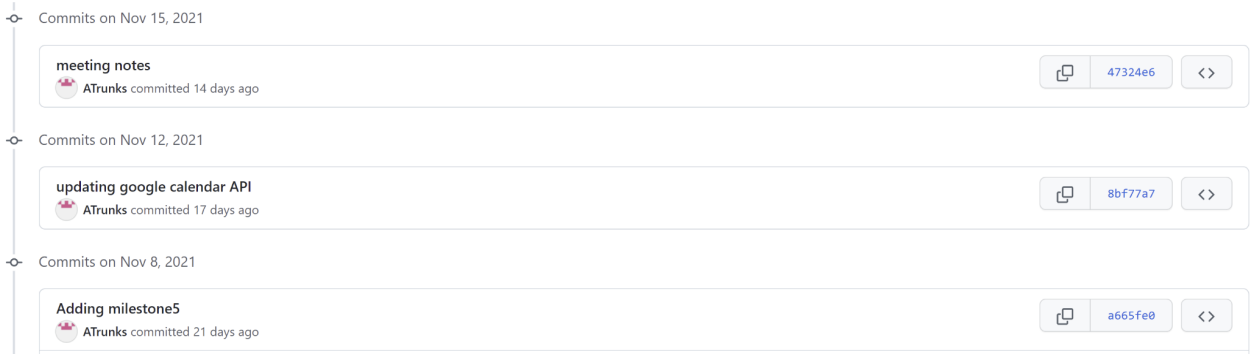


Adam: Worked on front end database work and specifically the website design. This includes ensuring the success of the login page with the required login credentials, as well as cleaning up the home page which were both HTML and then converted. Spent a lot of time trying to successfully create/implement the Google Calendar API but had some difficulties,

Ayden was able to complete it and fully implement it into our application. Created and completed the Project Presentation and helped with assigning the slides to the team members.

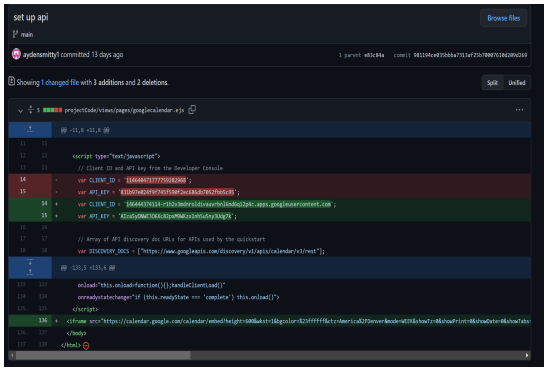


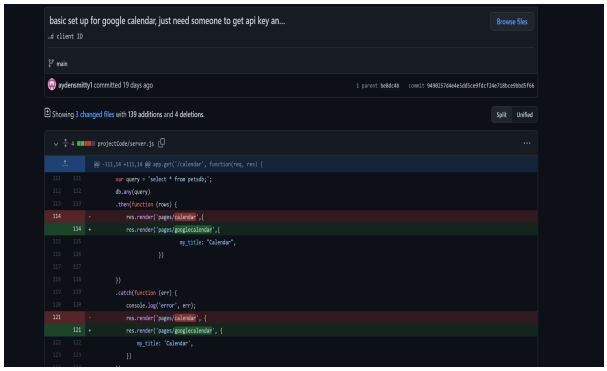


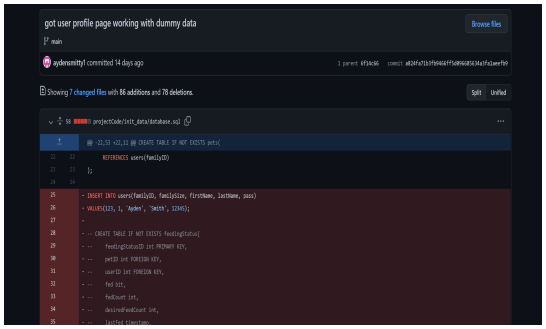


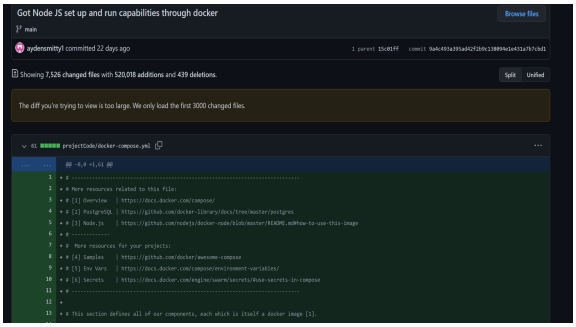
Ayden: I Worked on the Front End, Back end and tied the two together. I first incorporated some of the html we used in our labs in our projects and modified them accordingly. These pages included the login, registration, and profile pages, I then set up the google API and embedded calendar on the calendar page. Next I set up NodeJS and the docker yml file so we could run our project locally. I then worked with Leo and Cat on the postgreSQL database and collaborated to figure out what information we needed to include. I also got the user profile page tied to the database with NodeJS. After that I worked with Leo on user authentication and creation at the same time working with Cat on displaying pet info on the pet profile page.

Finally I set up and locally ran our demo.





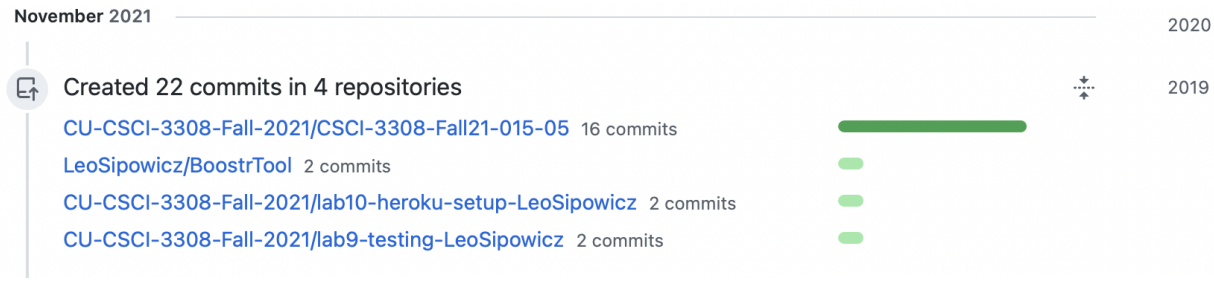




Leo: I worked on lots of different components but mostly focused on the backend and user account creation and authentication. I created the initial file structure and html layout for our app and then began work on server.js file and signin.ejs and register.ejs files. First I added functionality for users data to be stored in the database if all fields were allowed and the password had required criteria. Then I created a function in server.js to lookup the password associated with the imputed username and to check if it matched the imputed password. I also redesigned the jira to make it easier to read and understand for our whole group and made style and html changes to the apps front end.

```
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',
        })
    })
});
```

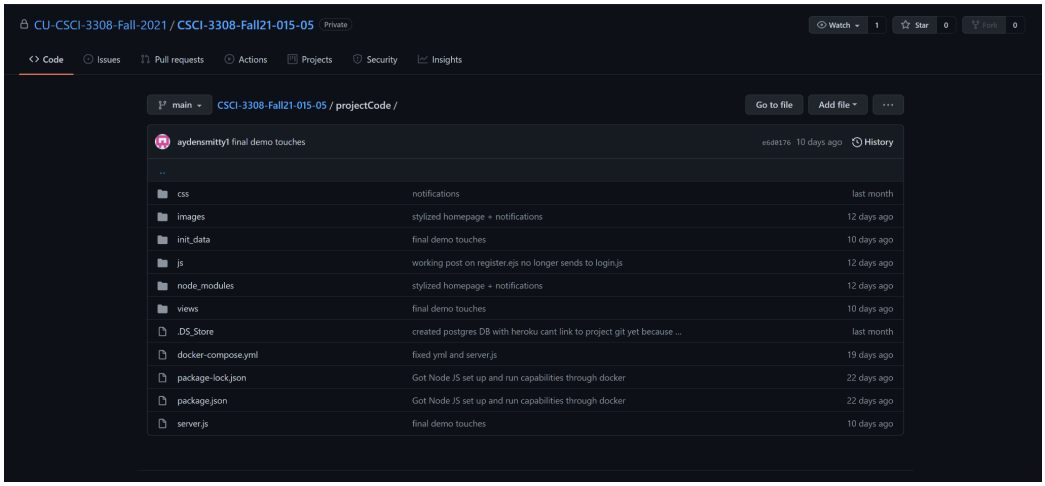
```
app.post('/signup/logintoaccount', function(req, res) {
    var EnteredEmail = req.body.Email;
    var EnteredPassword = req.body.psw;

    //  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',
            })
        })
});
```



Deployment:

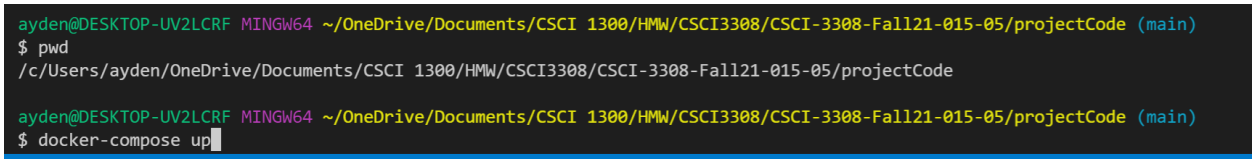
Our project is a local web application so we don't have any external access to our app. To access our app you must first pull from our github repo,

https://github.com/CU-CSCI-3308-Fall-2021/CSCI-3308-Fall21-015-05

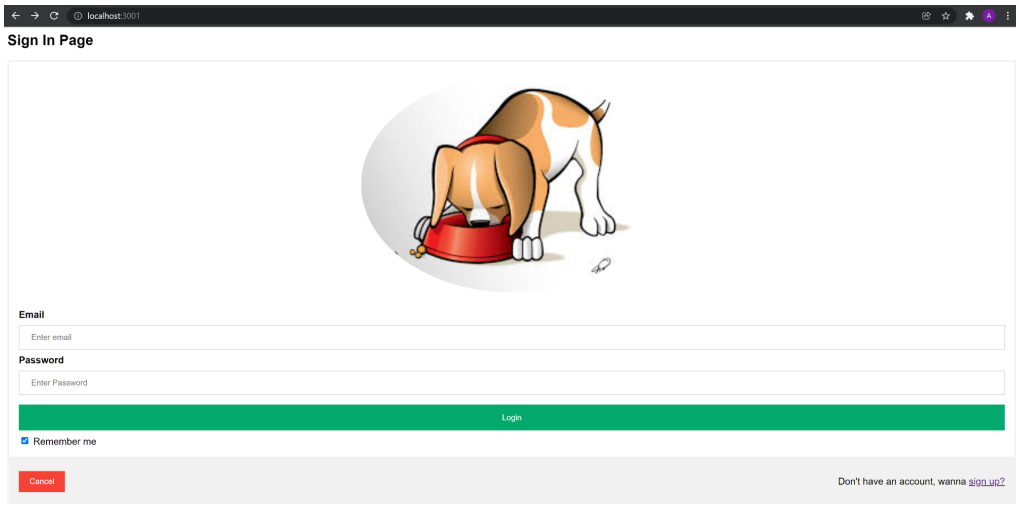


Then you must navigate to the projectCode directory.

After doing so you must run the command docker-compose up in your terminal and the init process will start and the app will be spun up



To access the working application you must finally navigate to the url http://localhost:3001/ You should encounter a scree like this:



To log into the page you may either create an account with the hyperlink in the bottom right corner of the page or you may use one of the pre-existing accounts already loaded into the database.

Preloaded creds:

Email: `1234@colorado.edu`

Password: `Password1`