Furry-Minder Final Report

Sarah Jorissen, Jonathon Hoffman, Laura Lopez

Austin Peay State University

This is a student final report regarding the Furry-Minder semester project for Dr. Nicholson's CSCI 4805, Senior Capstone course.

Introduction:

Each day of our lives, we are faced with filling up our day with tasks that we don't want to accomplish. We constantly have to force ourselves to do them or get something out of it to get the job done. Due to our human nature, we need the incentive to feel accomplished after completing our tasks. To assist in this constant issue, Furry-Minder provides a solution.

The pet helps incentivize the user to stay productive and on task by tying the pet's care to a to-do list. If tasks are finished on time, the pet will be fed & cared for and continue to grow. However, neglecting your tasks also means neglecting your pet, eventually leading to its death and starting all over again. Furry-Minder's creation was inspired by the popular 90's toy, Tamagotchi. Similar to Tamagotchi pets, Furry-Minder's purpose is to incentivize its users to complete their desired tasks on time while concurrently having an interactive pet to guide them along the way. Furry-Minder provides user stimulus as their pet will deteriorate as they fail to complete tasks. If tasks are neglected for too long, the pet will also be neglected, leading to its **death**. The user will then have to be assigned a new baby pet to start over again.

On the flip side, as users complete their tasks, their pet will be given health which allows them to grow. All pets will begin as a baby pet and progress to an adult. Further incentives to complete tasks include the unlocking of items to customize their pet as well as pet color changes to maximize user motivation of use.

Furry-Minder ideally targets users that are predominantly procrastinators and need an extra reward, or in some cases punishment, to get the job done. Furry-Minder is also for those that are nostalgic about the once popular and raving Tamagotchi. This may include animal lovers who cannot stand to let a pet suffer, even if it may be electronic. However, Furry-Minder is set apart from typical reminder or calendar systems and Tamagotchi due to its mix of innovative stimulus and practicality. The truth of the matter is: there has been nothing that compares to the combined functionality of Furry-Minder out in the real world, until now.

Technology:

In this section, the technologies that were used throughout the life-cycle of the project will be discussed.

Browser

The primary browser that Furry-Minder was developed for was Google Chrome. The team decided that Google Chrome was the best choice because everyone on the team was most familiar with it. Due to familiarity with the browser this technology didn't hinder the project for any of the members and also made development easier because it worked well with the other technologies used throughout the project's life cycle.

Jonathon - I think that the browser was one of the easiest decisions we made because everyone on the team was most familiar with the technology. I believe that Chrome added some slight difficulty because of how Chrome reacts to React (funny right?).

Laura - I like Chrome. Except my IDE kept defaulting to MS Edge and I was too lazy to go into my settings to change it. So MS Edge kept opening when I ran our app and I manually opened up Chrome to the local host. I really like making things harder for myself. :)

Sarah - I use Firefox mainly, but Chrome is probably the most popular browser out there now, so it seemed like the most obvious choice for building our project for. Of course, I still had a habit of testing things in Firefox during development, but we always made sure to check it would run properly in Chrome as well.

Operating System

The team also thought it best that they all work using the same operating system, however, Furry-Minder was developed with team members using Windows and Ubuntu. The main operating system was Windows, mainly because the majority of the team was most familiar with that OS compared to Ubuntu.

There was one issue that became quite noticeable around the midway mark of the project and that was how we were formatting our files in the IDE we chose. Due to the different OS being used by the team, whenever files were pulled from Git, the formatting would be reset and the members of the team would have to reformat the files in order for them to be usable. In this case, it was quite an easy fix because the IDE would reformat after saving any changes to the file.

Jonathon - This is the same deal as the browser, I was kind of worried about whether or not using different operating systems would affect the development of our project, which it kind of did. We had to include some weird ESLINT code into each of our IDE's in order for the formatting to not be affected across each of our environments after pulling from GIT.

Laura - The use of the operating system went smoothly besides the ESLINT issue we had that acted weird on Windows but appeared fine on Ubuntu. Jonathon and I had to both disable it as we were both running on Windows but after we did that it was all fine.

Sarah - Unfortunately, I was unsuccessful in convincing the others to switch to a superior operating system, but thankfully since we were all using WebStorm, we were working in a similar-enough of an environment where it likely wouldn't be too much of a problem.

IDE

Speaking of IDE's, the team decided on using WebStorm IDE by JetBrains. This IDE was chosen because one of the team members was very acclimated to the IDE and felt it would be the best for the type of development that the application would need.

WebStorm is a quality JavaScript IDE that works on multiple operating systems and comes with many useful features that not only help simplify and streamline development but also smooths out some of the issues that can come with multiple people working on one project. Its Git interface greatly helped with committing files, as well as pushing and pulling from GitHub without having to worry about using different interfaces due to our different operating systems. The most helpful feature for the team, though, was probably JetBrain's "code with me" feature, which allowed us to be able to connect to each other's devices and work collaboratively, even when not meeting in-person. This greatly helped when trying to work together to solve an issue someone had stumbled across.

Another important bonus is that all JetBrains products are free for college students to use, ensuring that this was a choice that all members of the team had access to.

Jonathon - I didn't find using this IDE too difficult, rather it was more about learning where everything was. After about a week or two of using WebStorm, I learned the environment and didn't have much difficulty in terms of using it. I feel like this IDE made development a lot easier compared to if we used a different IDE, the only reason I will say this is because of WebStorm having a built-in GitHub service.

Laura - I had never used WebStorm before, but I'm glad I know what it is now. It was extremely easy to use and had some great features, including the built-in GitHub service. It reminds me a lot of Visual Studios, which is my personal favorite IDE. However, I'm sure Sarah is throwing up at me for mentioning a Microsoft product being my favorite.

Sarah - Visual Sudio is *fine*, I suppose. But ever since I'd been introduced to JetBrains' IDEs earlier in my college career, I tried to use their products as much as possible, and recommended them to others. (Sadly, JetBrains has yet to contact me for any sponsorship deals.) It runs well on any OS (that I've encountered) and comes with a variety of useful features that helps with a lot of the more tedious aspects of coding.

Languages

The languages the team decided to use were HTML, CSS, JavaScript, and TypeScript because we were developing a single-page web-based application and those languages are essential. HTML and CSS are self-explanatory since they are necessary for any web application. Since our overall UI was relatively simple, we wrote much of the overall structure ourselves and relied on MUI for more complicated features such as the modals and buttons.

Jonathon - I haven't had to code HTML, CSS, or JavaScript for quite some time so I had to relearn some aspects of the language. I feel like after about two weeks I was on track with the team and I also got to learn a few cool tricks from my other team members! I feel like these languages, even though I wasn't familiar at first, made development easy compared to if we used any other language.

Laura - Learning to code in these three languages is challenging at times only because I had no previous experience with them. I had to do a lot of outside research and rely on my teammates for consulting about the best way to do things. Although some things are still difficult to read or interpret, I'm glad I got the experience with all of these.

Sarah - I was probably the one who was most experienced in these. I'd always been experimenting with web design in various forms, even small ways like updating my old MySpace page, so I came into this project already familiar with the basics.

Library

A library that was used throughout the project's life cycle and was vital to the application's development was the React library. The team chose React because the library helps streamline the creation of a React project and comes with various useful features built-in. It will help ease the process of starting and working on a React app for a team that is mostly new to React.

React allows us to break down the different aspects of our application into their own components. Each component can handle its own state as needed, and external information it needs can be passed to it via its props.

Another benefit of using React is that it is extremely popular among web developers, making it easier for us to find helpful resources and answers to questions outside of its documentation.

Jonathon - I had no previous experience with React, and it was quite difficult to wrap my brain around some of the concepts it uses. I found it quite difficult to learn how React implements HTML elements and CSS and how it breaks everything down into components. I was asking questions non-stop to Sarah because she had way more experience using the library than me or Laura. I am actually quite glad about what I learned about React and can say with confidence that I will continue to work with the library to try to become more aware and knowledgeable of all of its features.

Laura - Since I had no previous experience with JavaScript, I was able to learn React on a clean slate, without comparing the use of JavaScript without using React. What I felt could have been a little bit easier with React was learning how to maneuver the MUI components and handle each of their states.

Sarah - I'd only had the chance to work on React briefly on a project I'd started over the summer, but I was intrigued by it and the ways it can help you break down your app into smaller components. It's definitely a tool I hope to continue to be able to use in the future.

API

The team decided that it was best not to create the calendar component from scratch and to use an API called FullCalendar to help push the development process along. Jonathon was the main member of the Furry-Minder team that used the FullyCalendar API throughout the development of the project. The main reason the team chose FullCalendar was due to it having the features we need already implemented in the base version of the API. Jonathon had quite a few issues during the beginning stages of the project's life cycle due to the API and its lack of quality documentation. He was able to fully implement the calendar and the features required for the project, it just took longer than expected.

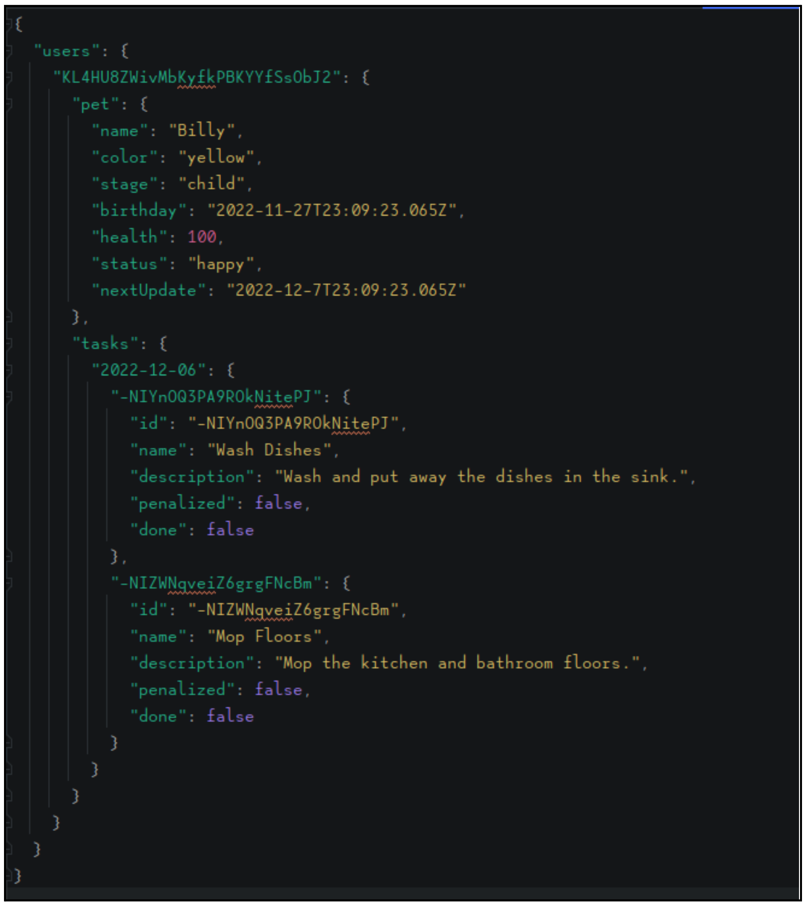
Jonathon - The FullCalendar API was honestly like reading Hieroglyphics due to how poorly things are documented. It took me about two weeks to figure out how to properly implement the API into the project and an additional week just to get dummy data (from a JSON file) to be read into the calendar. After struggling with the API I learned how to manipulate it to do what my team needed and was able to implement all of its features within a decent time frame. I would say that I have developed a solid understanding of how to use this API, but the developers are constantly updating and changing it so I would have to consistently read up on it to stay up-to-date.

As mentioned previously, the team heavily relied on MUI components from React in order to implement a variety of our features. The DatePicker component uses the AdapterJS API.

Laura - It wasn't too difficult to implement this as there were several online resources that showed how to use this API when implementing the DatePicker. It's suspected that the use of this API may have been one of the causes for some of the bugs we had in the Edit Task component because of the timezone issues.

Database

Figure 1: Example data as JSON



```
{
  "users": {
    "KL4HU8ZWivMbKyfkPBKYYfSsObJ2": {
      "pet": {
        "name": "Billy",
        "color": "yellow",
        "stage": "child",
        "birthday": "2022-11-27T23:09:23.065Z",
        "health": 100,
        "status": "happy",
        "nextUpdate": "2022-12-7T23:09:23.065Z"
      },
      "tasks": {
        "2022-12-06": {
          "-NIYnOQ3PA9ROkNitePJ": {
            "id": "-NIYnOQ3PA9ROkNitePJ",
            "name": "Wash Dishes",
            "description": "Wash and put away the dishes in the sink.",
            "penalized": false,
            "done": false
          },
          "-NIZWNqveiZ6grgFNcBm": {
            "id": "-NIZWNqveiZ6grgFNcBm",
            "name": "Mop Floors",
            "description": "Mop the kitchen and bathroom floors.",
            "penalized": false,
            "done": false
          }
        }
      }
    }
  }
}
```

The team (Sarah) had decided on using Realtime Database over Firestore early on, due to the misunderstanding that it would be able to provide what the app would need, and that it may be simpler to implement than Firebase. Unfortunately, this mistake became very obvious much later down the line when we began to implement date- and time-sensitive features, and realized that being forced to pass these to and from the database as strings created major problems in the latter half of the project.

The structure of the database was laid out so that when a user is created, Firebase would generate a unique user ID for them, which we could then use to store their pets and tasks in the database. This allows each user to be able to access their respective data, but prevented from accessing others'. Their tasks would also be organized by date, to make it easier for all tasks of a specific date or date range to be pulled, for features such as the daily task list and calendar. As pets and tasks are added, updated, and deleted, the changes would be reflected in the database. This helped keep the data consistent across all components of the app, and any other devices that access it as well.

Sarah - I had no previous experience with Firebase before this project, and what little experience I had with databases at all were based around SQL, so implementing all the different ways for our app to communicate with our database was probably the biggest challenge of this project for me, but also where I learned the most. Many mistakes were made, but going forward, I know better ways to try to do this in the future.

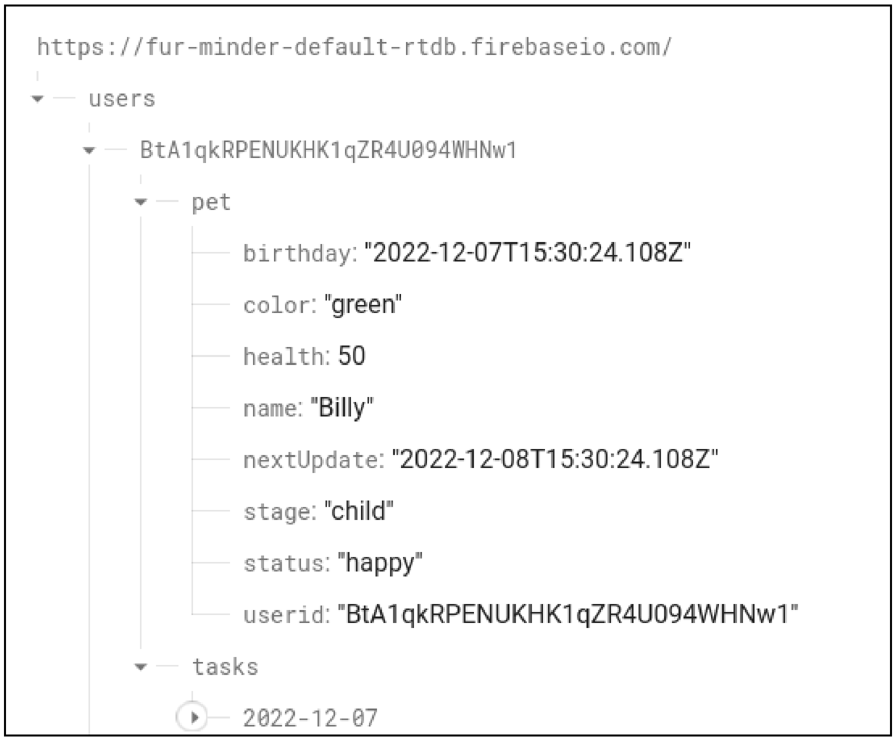

```
https://fur-minder-default-rtdb.firebaseio.com/
users
  BtA1qkRPENUKHK1qZR4U094WHNw1
    pet
      birthday: "2022-12-07T15:30:24.108Z"
      color: "green"
      health: 50
      name: "Billy"
      nextUpdate: "2022-12-08T15:30:24.108Z"
      stage: "child"
      status: "happy"
      userid: "BtA1qkRPENUKHK1qZR4U094WHNw1"
    tasks
      2022-12-07
```

Figure 2: Data in Realtime Database console

Communication

The team decided on using Outlook and Discord as a way of maintaining stable communication throughout the life-cycle of the project. The main form of communication was a discord server where we maintained several different channels for different uses such as “general”, “tech-support”, “off-topic”, “notes-resources”, and “documents” channels. The usage of these different channels enabled the team to maintain a strong and organized chain of communication that helped maintain the team throughout the weeks of the project. One of the other key reasons we decided to mainly use Discord, was because the service provides a screen-share feature and voice chat feature.

Jonathon - I think our decision for using Discord was brilliant because of the several features it provides and because each of us was fairly used to Discord. For example, we can video call, voice chat, and even share screens. I also feel like it made organizing our group a lot easier with the use of different channels throughout our group's discord server.

Laura - Discord basically housed almost all of our conversations that were text-based or over-the-phone/screen share. It was the most convenient form of communication when we weren’t together. It was also a great way to easily share document links so that we could all work on them during different phases of the project.

Sarah - My own distance from campus made it hard to meet in-person without prior planning, so being able to also work together remotely was a huge boon. Not only could we call and share screens, but someone who wasn't able to be at their computer actively working could also chime into conversations as needed.

Version-Control

The team didn't have sway on the decision for this technology as it was a requirement by the Professor for the project. The team was required to use GitHub as a way to maintain version control throughout the life cycle of the project. What's interesting about this requirement is that our IDE, WebStorm, has a built-in feature that connects to GitHub automatically which allowed the team to easily maintain a version-controlled application and not have to worry about the craziness that is known as GitHub.

Jonathon - I have only used GitHub once prior to this class and when I did it was the bare minimum, so I had absolutely no idea what I was doing in terms of version control with GitHub. Thankfully, my other team members were quite knowledgeable and explained everything I needed to know within the first week of development! I feel like this made development significantly easier compared to if we didn't use Git or version control at all.

Laura - I have used GitHub on several occasions for both personal assignments and group projects. I felt pretty comfortable using it. The biggest difference for me was using the desktop version of GitHub, as I had only used the command line in the IDE for previous projects. GitHub Desktop was simpler than using the command line, but I actually found Webstorm's Git system easier to use than GitHub Desktop.

Sarah - Like the others, I'd only really used GitHub on my own in the past, and so learning to do it as a group was an interesting challenge. Being unfamiliar with Windows and GitHub's desktop app for it, I was a little worried about potential problems using two different methods might create, but thankfully WebStorm once again has several useful features for Git that helped with that.

Design:

This design section of the paper will be a description that will cover the entire functionality of the application. This will be broken into sections to make it easier to follow and understand.

Create Account Window

Upon loading the webpage, furryminder.com, the user will be prompted with a large “Create Account” window. Within this window, the user is prompted with text fields to provide a valid email address, their password, and confirm the previous password. There will be a “Submit” button at the bottom of the window that, when pressed, will take the user to the Create Pet window. Within the window, the team provided a way to make the password visible, in case the user wants to check their input. What if the user already has an account? The team at Furry-Minder has that covered with a “Login” button at the bottom of the window next to “Already have an account?”, which when pressed will take the user to the “Login” window. If you notice, there is no way to get to the core application without having an account! This was done on purpose because it was the easiest way to manage who has access to the application, and what the user sees. Because much of the main display of the page relies on accessing the user’s tasks and pet information, it made more sense to prompt the user to create an account and pet (or to log in) if they access the page without an account.

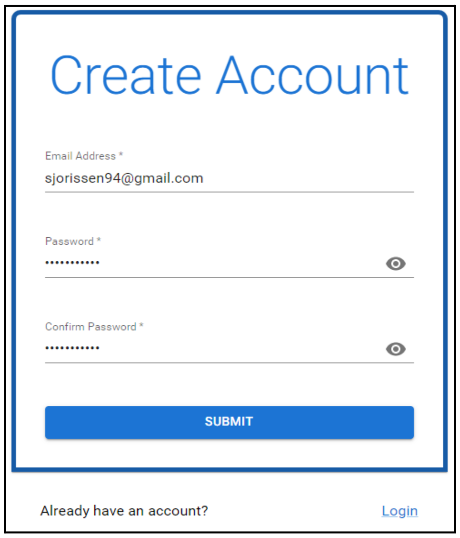


Figure 3: Create Account Window

Pet Creation Window

This window will only be seen when a user creates their account or if their pet passes away from horrible mistreatment. This window will prompt the user for their Pet’s name with a text field for the user’s input. At the bottom of the window, there will be a “SUBMIT” button that, when clicked, will randomly generate their pet’s appearance while also pushing the new pet’s information such as Name, Color, Age, and Species to the database. After clicking the “SUBMIT” button the user will be transferred to their very own user-specific homepage!

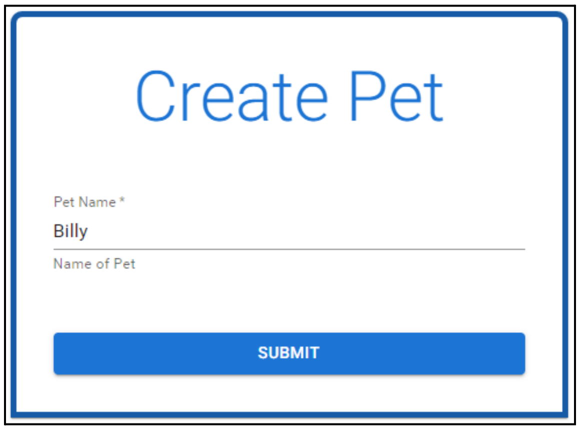


Figure 4: Pet Creation Window

Login Window

As stated in the “Sign-up Window” section there will be a “Login” button located at the bottom of the “Create Account” window. After clicking this button the user will be directed to the "Login" window where they will be prompted for the Email Address and Password that they used to create an account. Similarly to the design of the "Create Account" window, there will also be a "Create Account" button at the very bottom of the "Login" window that when clicked will direct the user to the "Create Account" window. Just like the "Create Account" window the team of Furry-Minder developed a way for the user to take their password out of a protected view, to check if their input was correct. If the user provides incorrect information for their email address or password, they will be given an error message, and have to resubmit with the proper information to log in and access their homepage.

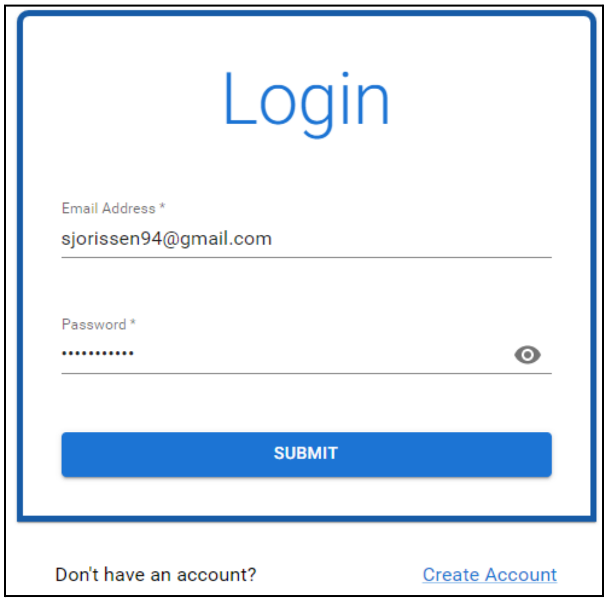


Figure 5: Login Window

The Homepage

The user's homepage is the central hub for the Furry-Minder application and is specific to each user. In the minimalistic design, the team of Furry-Minder created the user's very own task list labeled "My Task List" on the right of the screen. Within the user task list window, they will have a "Calendar" button to open up their user-specific calendar, a "+" button to add tasks to their user-specific task list, and a section to view their tasks for the current date (the white space below the header). To the far right of the screen, we will see the user's pet window! In this window we will have the pet's name centered at the top, the pet itself moving around in the window, followed by its health bar, and finally a "Clothing Hanger" button for opening the Customize Pet menu.

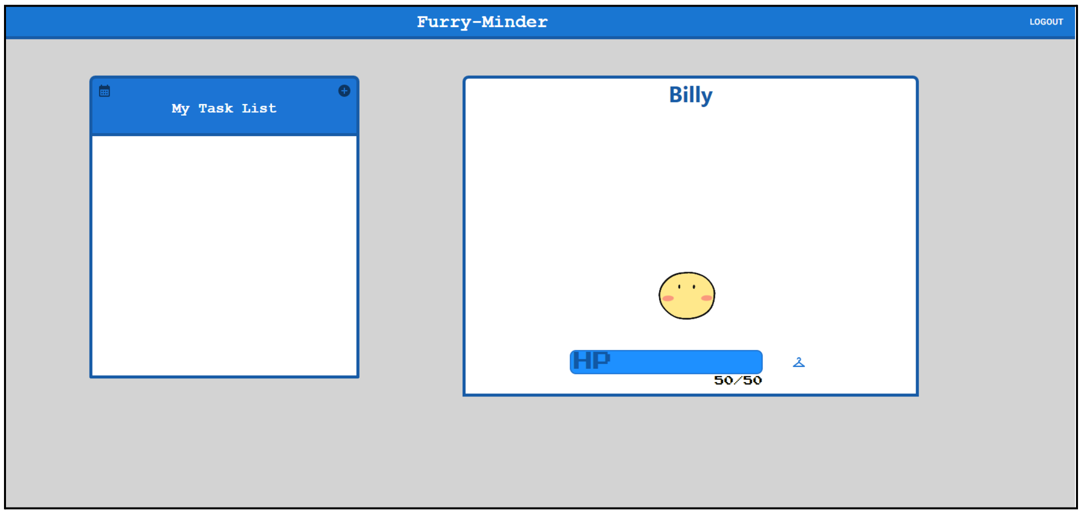
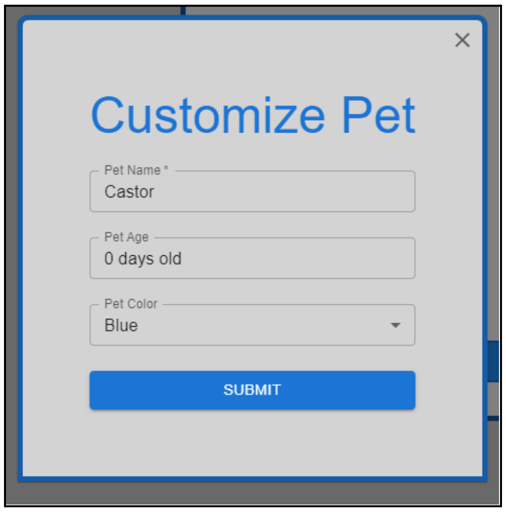


Figure 6: User Homepage

Pet Customization Menu

The Pet Customization modal will open when the “Clothing Hanger” button on the user’s homepage is clicked. Within this modal, the user will see two text fields with their pet’s name, the pet’s age, and a drop-down box for the pet’s color. All fields can be changed except for the pet’s age, which will change over the lifetime of the pet. The team of Furry-Minder developed several species of pets and even more color variations. With the release of version 1.0 of Furry-Minder, the team allowed the user to choose a color for their pet if they didn’t like the color they got when the pet was randomly generated. This can be done by clicking the dropdown box labeled “Pet Color” and choosing from the selection: red, yellow, blue, green, or “void” – a “special” color meant to look a bit different and more interesting than the standard ones. Once the user makes all the changes they wish to make, the user will have to click the “SUBMIT” button to finalize and submit those changes to the user’s profile on the database.


Customize Pet

Pet Name * Castor

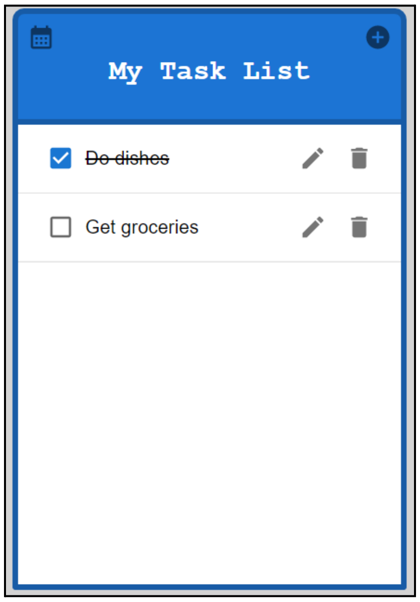
Pet Age 0 days old

Pet Color Blue

Figure 7: Customize Pet Modal

Task List Window

Back on the user’s homepage, the Task List Window can be seen on the left side of the screen labeled as “My Task List”. The Task List will be composed of a “Calendar” icon that will open the user’s calendar modal, and a “+” icon that will open the add task modal and will have space to view tasks for the current date. When the user has tasks for the current date they will be displayed in the whitespace under the blue header tab with the title “My Task List”. Each Task will have an empty checkbox to the left of the title of the task, and to the far right will have a “Pencil” button for editing the task and a “Trash Can” icon for deleting the task. If the user hovers over the task, the task’s description is displayed as a tooltip. When clicking the “Pencil” icon, the Edit task modal will open. When clicking the “Trash Can” icon, the task will just simply be deleted from the user’s task list and


My Task List

✓ ~~Do dishes~~

☐ Get groceries

from the user's profile on the database. A very important aspect of the user's task list is the ability to mark tasks as complete. That's because in order to keep the user's pet healthy they must complete tasks, and in order for the database to know which tasks are complete, the user would have to check the check box of the task. When the user checks the check box, the title of the task will be struck through to visually show the task is done and when the check box is unchecked the title will go back to its normal state.

Figure 8: Task List Window

Add Task Modal

In order to reach this modal, the user will either have to click the "Pencil" icon that is located next to the task on the user's task list or by clicking the task on the user's calendar. After taking either approach, the "Add Task modal" will display and will prompt the user for a few things. In order for the Task List component to truly function as a real-life task list, the team of Furry-Minder wanted the user to provide a Task's Name, the Task's Description, and the Task's Due Date. If the user intends on adding a task to their Task List, they are required to provide the Task's Name and its Due Date, where the description is optional. To make selecting dates easier for the end user, the team of Furry-Minder implemented an API called the Date-Picker which will allow the user to click a "Calendar" icon that will open a calendar where they choose a date from there. Once the user wants to finalize their task submission, they will have to click the "SUBMIT" button that will push the new task to their profile on the Furry-Minders database. Also, let's say, for example, that the user opens the modal but changes their mind and doesn't want to create a new task, how do they exit the modal? The team of Furry-Minder developed two ways of exiting the modal, one would include an "X" icon in the top right-hand corner, and the other method is by clicking outside of the modal.

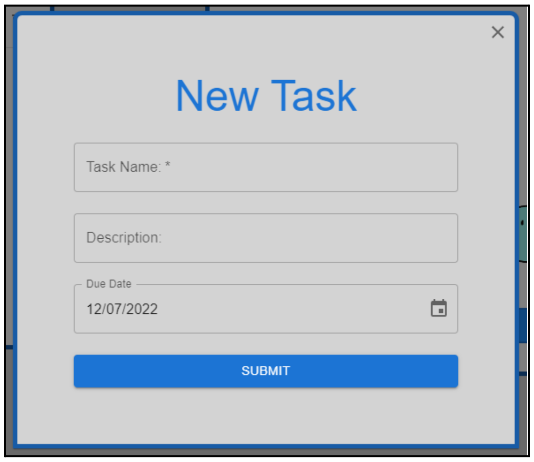
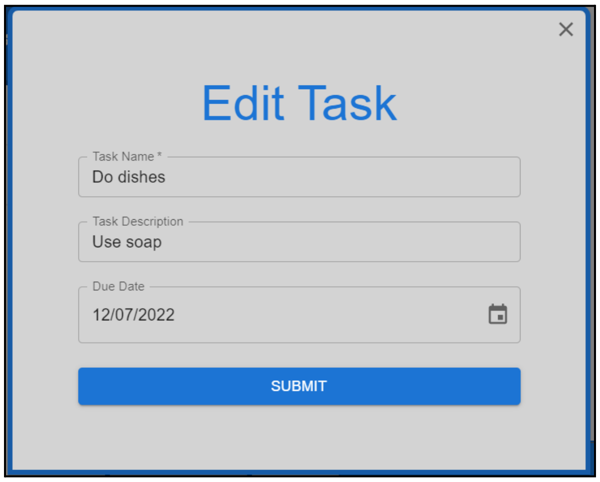


Figure 9: Add Task Modal

Edit Task Modal

In order for the user to view the "Edit Task" modal, they must either click the "Pencil" icon on the task within the user's Task List on their homepage or by clicking the task when viewing their Calendar. This modal is very similar to the "Add Task" modal, where the user will see three fields. These three fields include the Task's Name, the Task Description, and the Task's Due Date. The difference between the "Add Task" modal and the "Edit Task" modal, is that the "Edit Task" modal pulls the task's information from the user's profile on the Furry-Minder's database and will allow the user to change what text fields they wish to change. The same stipulations still apply to the task in this modal, there must be a Task Name and a Due Date, otherwise, the user will be prompted to fill the text fields with proper information. Once the user wants to finalize their task edits, they will have to click the "SUBMIT" button that will push the new task information to their profile on the Furry-Minders database. Also, let's say, for example, that the user opens the modal but changes their mind and doesn't want to edit the task information, how do they exit the modal? The team of Furry-Minder developed two ways of exiting the modal, one would include an "X" icon in the top right-hand corner, and the other method is by clicking outside of the modal.



Edit Task

Task Name *
Do dishes

Task Description
Use soap

Due Date
12/07/2022

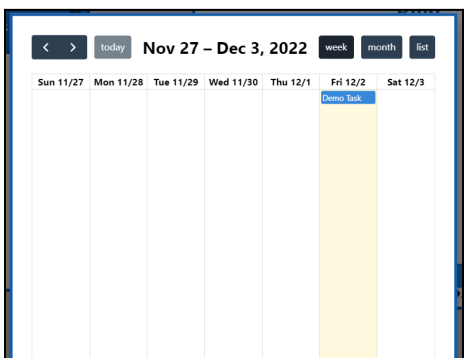
SUBMIT

Figure 9: Edit Task Modal

Calendar Modal

Back on the user's task list within their homepage, there will be a "Calendar" icon that, when clicked, will initially generate the user's calendar in the weekly view. For any tasks that were created, they will be visible on the days in the weekly view. There will be 'previous' and 'next' buttons that are displayed as "<" and ">" icons that allow the user to traverse the dates of the calendar and a 'today' button that will take the user back to the current date. There are three options that change the view of the calendar in the top right corner, a 'week', 'month', and 'list' view. When the user clicks the 'week' button, the calendar will change to the week view of the calendar, if the user is not already on the current dates week or month. When the user clicks the 'month' button, the calendar will change to the 'month' view of the calendar. When the user clicks the 'list' button, the user will see a list of all tasks that the user has within their profile in the Furry-Minder database. If there is a task that the user wishes to change, all the user has to do is click the task and the "Edit Task modal" will display. Finally, when the user wants to exit the calendar, they can click outside of the modal and the calendar will close.

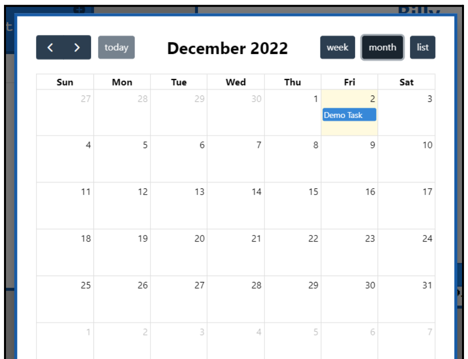
Figure 10: Calendar Week View



Nov 27 – Dec 3, 2022

Sun 11/27	Mon 11/28	Tue 11/29	Wed 11/30	Thu 12/1	Fri 12/2	Sat 12/3
					Demo Task	

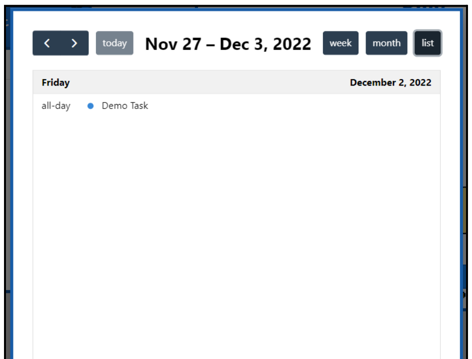
Figure 11: Calendar Month View



December 2022

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2 Demo Task	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Figure 12: Calendar List View



Nov 27 – Dec 3, 2022

Friday December 2, 2022

all-day ● Demo Task

How to deploy or build the application:

For this guide, we will assume you have a basic understanding of GitHub and npm. From the project's repository, clone the project. Navigate to the project's directory and run `npm install` to install the needed packages. The project as written is tied to our own Firebase account, which the app can still run and connect to (for now). However, you can only interact with the app as a user, and won't have access to authentication or the database. To set your own, you can follow the full tutorial Google provides on Firebase's website. Here, we will go over the steps needed for this project briefly.

Once you've created a project and you're at the project console, you'll be greeted by a banner that says "Get started by adding Firebase to your app." Since we definitely want to add Firebase to our app, go ahead and click on the "Web" button (the third one). Give your app a name and click "Register." It'll bring up the next step, prompting you to run `npm install firebase` and giving you a code snippet. Copy everything in `firebaseConfig` and in your project directory, navigate to `pet-task-app/src/api/firebase-config.js/`, and replace the old `firebaseConfig` with yours.

Once done, you'll be directed back to the console. Under "Build" on the sidebar, you'll see the list of different features that Firebase offers. For this app, we'll only need to use Authentication, Realtime Database, and Hosting. We'll go over each of these individually:

Authentication

Choose this and click "Get Started." Furry-Minder only requires users to sign up with an email and password at the moment, so under the "Sign-in method" tab, click on the button that says "Email/Password." Toggle the first switch that says "Enable," then click "Save." And that's it! That's all you have to do to set up Firebase authentication.

Realtime Database

Choose "Realtime Database" from "Build" in the sidebar now. Select "Create Database," and a modal will pop up asking you to select your database location and click "Next." On the next page, leave it in locked mode and click "Enable."

Now, go ahead and navigate to the "Rules" tab. The rules for the database can be found under `/pet-task-app/database.rules.json/`. Copy those and paste them over the default rules in the console.

If you navigate back over to "Data," you'll see that the database is empty. We don't actually need to add anything to this. When the program makes a call to a specific path in the database, it will create that path if it doesn't already exist. The project files do contain an `emulatorData.json` file which can also be imported here if you'd like to quickly add some data to start with – but make sure that you replace the UID at the top with a *real* user ID listed under Authentication. To use the file, click on the dot menu icon in the far right of the database's header and select "Import JSON." Navigate to the file, select it, and click "Import," and you should see a new line appear in the database that says "users." click on the arrow next to it to expand and view your current users' IDs, and click on those to view their specific information. If you want to manually add any data through the console, be sure that it follows the same format as the file uses, or the app will not be able to reach it properly.

Hosting

The database was the most complicated part of this whole setup. Now, we can move on to setting up Hosting. Select it from the "Build" menu and click "Get Started" like the others. It'll give you another npm install command to run and give you the option to show the steps for adding the Firebase JS SDK to your app. Since we already did this, you can leave that unchecked and move on to the next step.

Here you'll be given another two commands to run. Use `firebase login` to log into your Google account in your project directory. Once done, run `firebase init` next, and you'll be given several options to choose from:

1. Select Realtime Database and the first Hosting option. Scroll through the list with the arrow keys and use the spacebar to select them. Then hit enter when you're done.
2. And after setting up, Firebase will ask if you want to overwrite `database.rules.json` with the rules in the console. Type "n" or just hit enter (no is the default) to continue.
3. Leave "public" as the public directory.
4. Furry-Minder is a single-page app, so enter "y" when prompted.
5. Choose "no" for automatic builds and deploys with GitHub.
6. Choose "no" when asked to overwrite `index.html.`

Finally, the Firebase setup is complete!

The last step will tell you to run firebase deploy, but don't do that just yet. Back in your project directory, run `npm start build` first. Then, you are finally ready to run `firebase deploy --only hosting`.

Known bugs:

Edit Task Window - Although attempted to fix several times, the edit task window has experienced a bug in which when a task is edited from the task list, the date gets kicked back to the day before. This bug occurs when any of the fields in the edit task modal are edited. However, this does not occur when editing a task from the calendar component. One of the main issues we encountered was being forced to store our dates as strings with a Realtime Database.

Future work:

Toggle between dates in the task list

This feature would allow the user to be able to select the date they wanted their tasks displayed on the task list by using a left and right arrow button. If they do not have tasks scheduled on a certain date, then they would not have any tasks displayed. If they select a date that has one or more, they should be displayed on the task list for that due date.

Ability to set tasks to be repeated

This feature would allow the user to be able to set a task to be repeated daily, weekly, or custom. This would prevent the user from having to input a task several times if it is a task that they do often.

Add more species and color options for the pet

This feature would allow the user to have a larger variety of pet options to choose from when they allow the pet to grow into an adult. This would also allow the user to have more colors to choose from when they initially are assigned a pet. Initially, there had been six different colors (“polka dot” being the sixth option, and second “special” color), and six different adult options. Users would start with a child who looked the same for all possible species but could grow into one of two “teen” options. Each of these “teen” pets could then grow into one of three possible adult options, with the other three only able to grow up from the other one.

After a pet is fully grown, and the user has kept them up for a certain period of time, users will have the option to unlock new colors and species of pet that they can then change to whenever they’d like in the customization menu.

Add the ability to customize the pet with accessories

This feature would allow the user to unlock accessories over their pet’s lifetime after completing a certain amount of tasks. These accessories would be able to be taken off the pet or be able to be changed if they had several accessories unlocked. As a child, a pet can only wear one accessory at a time but will be able to wear more as it grows up. Users will start out with no accessories but can unlock them over time by completing tasks on time and taking good care of their pets. Accessories will also be tied to users rather than pets, so should a pet die and the user creates a new one, they still retain all of the accessories they'd earned before.

Stricter password requirements

At this time, the password requirements for creating an account are not very strict and can be pretty much anything the user chooses. In the future, we would hope to have the password requirements have to be at least 8 characters long, including an upper and lower case letter, a number, and a symbol.

User notifications

Users will be able to set up email and push notifications to alert them when a task deadline is upcoming, or to warn them when their pet's health has gotten below 50%. Users can toggle these options on or off in their settings menu.