# Digital Pet Task App

# Team

Sarah Jorissen, Laura Lopez, Jonathon Hoffman

# Abstract

Each day of our lives, we are faced with filling up our day with tasks that we don’t actually want to accomplish. We constantly have to either force ourselves to do them or have to get something out of it in order to get the job done. Due to natural human nature, we need an incentive to feel accomplished after completing our tasks. In order to assist in this constant issue, our Digital Pet Task App provides a solution. The pet helps to incentivize the user to stay productive and on task by tying the care of the pet to a to-do list. If tasks are finished in a timely manner, the pet will be fed & cared for, and continue to grow. However, neglecting your tasks also means neglecting your pet, which will eventually lead to its death, and starting all over again.

# Description

The Digital Pet Desk App is a web-based application that provides an interactive calendar and task list to organize activities, job duties, chores, etc. In addition to this, The Digital Pet Desk App gives users an interactive pet that they take care of as they complete things on their task list. The more tasks users complete on time, the better taken care of their pet will be. The opposite effect will occur if they fail to complete items on their task list on time.

Tasks can be added and checked off via the task list, or scheduled in advance through the calendar. The calendar will allow users to schedule tasks on a particular date at a particular time. Larger tasks that will take time to complete can be broken down into smaller pieces that can be completed in shorter intervals that help care for the pet.

Alongside the checklist, the pet itself will be displayed, showing its current status. Statuses can include how hungry it is, or its mood if it’s been neglected for too long. If the user continues to neglect the pet, it will eventually show a little gravestone to indicate the pet has died, and an option to start over with a new pet is shown. The pet will start off as a baby and slowly grow over time, similar to a Tamagotchi. If a pet dies and a new one is generated, the user’s task list and calendar will remain as it was at the point of the previous pet’s death.

The goal for the semester is to complete the front-end of the web app that persists in local storage. When the user adds an item to the to-do list, the app will serialize the items on the list and make a call to localStorage.set. On loading the app, it will load the existing state from local storage. That way, the tasks will persist between sessions on the same computer. If this goal is reached with time to spare until it must be presented, the next step will be to implement a web server that will allow the app to be run from any computer.

To achieve this, the app itself will be built using Create React App, and Remix can be used to implement the web server.

# Features

| **End-of-semester** | **If there is time** | **Cannot be completed** |
| --- | --- | --- |
| View Calendar | Overlap application onto other windows in use (browser extension?) | Users are able to customize small details of their pets |
| Add task to calendar | Notification when task is coming up via calendar | Users can share their tasks lists with other users |
| Remove task from calendar | Email monitoring  (notifies user of new emails) | Users can have their pets interact with other users pets for increased pet lifespan |
| Set tasks at a certain time interval on calendar | Allows the user to choose between different pet options | Users can share their pets happiness scale via their preferred social media |
| View task list | Allows the user to choose light/dark mode for UI | Turning the web application into a mobile app |
| Add to task list | Allows the user to manipulate UI positioning |  |
| Remove a task from the task list (Ex: a task no longer needs to be done) | Create a log in page for user authentication |  |
| Check off an item from the task list | A score-type system to record pet lifespans |  |
| User pet is fed upon completing a task | Increasing pet lifespan by playing with it |  |
| User pet deteriorates at the expiration of uncompleted tasks |  |  |
| User pet dies after a certain point of negligence / failure to complete tasks |  |  |
| User pet announces to user when it is content or unsatisfied |  |  |
| Save task list/calendar status upon death of a pet |  |  |
| Regeneration of a pet if one dies |  |  |

# 

# Technology

* **Chrome web browser:** The primary browser this app will be developed and tested for.
* The team will be developing on **Windows and Ubuntu**, mainly focused on Windows.
* **WebStorm IDE (by JetBrains):** A quality JavaScript IDE that works on multiple operating systems and helps simplify and streamline many necessary tasks. Free to use for college students, so all team members are able to access it.
* **HTML, CSS, JavaScript:** Required languages for building a web application.
* **React/Create React App:** Create React App helps to streamline the creation of a React project and comes with a variety of 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.
* **Remix:** Full-stack React framework which we will mainly be using to run the web app off a server.
* **FullCalendar:** Customizable calendar for React. No need to reinvent the wheel.
* **Linode:** For implementing a web server for our project, once the front-end is complete.
* **Discord:** For quick communication with team members.
* **Outlook:** For more long-form communication, sending necessary files, etc.
* **PivotalTracker:** For keeping everyone on task and seeing what still needs to be done.
* **GitHub:** version control.

# Team members’ background

## Sarah Jorissen

I’ve worked on a couple of web app projects personally, so I have a good amount of experience with the languages involved. I’ve only recently started working with React and Remix with one of my latest projects, but I am confident enough that I’ll be able to keep learning and improving with both as we work on this project. I’ve used GitHub for quite a while now, and I feel fairly confident using them.

(As a bonus, I’m also married to a web developer who works with React, so we’ll always have someone on-hand to pester for help if needed.)

## Laura Lopez

I am unfamiliar with the JetBrain IDE that will be used for development. I have also never programmed in HTML or CSS. I am somewhat familiar with JavaScript through previous experience of converting TypeScript into JavaScript. Due to this, I am not familiar with the React library or Remix and FullCalendar tools. I am somewhat familiar with Linode but not extremely comfortable with it. I am very familiar with Git and GitHub technologies that will be used to share work amongst the team. My primary responsibility will be for any and all activities regarding the task list system.

## Jonathon Hoffman

I am unfamiliar with the JetBrain IDE that has been chosen to develop the project, but I do have programming background in HTML, CSS, some Javascript, Java, and Python due to my student worker job and prior coursework. I am not familiar with the React, Remix, or FullCalender tools and libraries. I do not have any experience with any server-related development, but I am very familiar with Git and GitHub due to prior coursework. I will be responsible for any and all activities regarding the calendar system that will be implemented into the project.

# Dependencies, Limitations, and Risks

## Dependencies

| **Dependency** | **What is the risk?** | **Possible solution** |
| --- | --- | --- |
| Server will need to be active during development, if time permits web deployment. | Web application will be unusable if server is down. | Have a backup server service predetermined in case of this emergency. |
| Team members unfamiliar with some of the technologies which build off of each other. | Team members may be unable to learn how to use these technologies. | Team members that are unfamiliar with these technologies can seek help from team mates or Dr. Nicholson. |
| Task list must be implemented prior to the implementation of the calendar and pet. | If task list is not implemented properly, calendar and pet will become unstable. | Run several tests throughout development with all team members to ensure that the project can move forward. |
| Calendar must be implemented in order for the pet to know what time the task must be completed by. | Pet will be nonfunctional if calendar is not developed properly. | Run several tests throughout development with all team members to ensure that the project can move forward. |



## 

## Limitations

| **Limitations** |
| --- |
| Team member Laura Lopez and Jonathon Hoffman do not have experience working with servers. Laura Lopez and Jonathon Hoffman will seek help from Sarah Jorrisen for help in these areas if needed. |
| Team members Laura Lopez and Jonathon Hoffman have no experience using the JetBrains incremented development environment. They will seek help from team member Sarah Jorissen and internet resources if needed. |
| Application will be limited to Windows users and the Google Chrome browser. Could be extended to other OS’s or browsers if time permits. |
| Team member Laura Lopez does not have experience with HTML or CSS. Team member has limited experience with JavaScript. Team member will be guided by team mates and internet resources to close learning gap. |



## 

## Other Risks

| **Other risks** |
| --- |
| The possibility of another pandemic, could potentially put the project at risk of not getting 100% finished by the designated deadline date. |
|  |
|  |
|  |



# Timeline

| **Week** | **Task** |
| --- | --- |
| Week 1  8/29 - 9/5 | 1. Set up tools, make sure everyone is on the same page 2. Sketch out basic UI for app, decide how to tie features together 3. Decide on design for pet & its different statuses |
| Week 2  9/5 - 9/12 | 1. Begin task list development. 2. Decide on basic UI design for all parts to follow (colors, etc.) 3. Create basic art for pet to use for testing features 4. Implementation of adding and removing items from task list. 5. Implement the FullCalendar API |
| Week 3  9/12 - 9/19 | 1. Create UI of task list 2. Add tasks to calendar 3. Set different “conditions” that determine how the pet will look/respond & test these |
| Week 4  9/19 - 9/26 | 1. Create UI of task list 2. View task list implementation. Checking items on and off task list. 3. Remove tasks for calendar 4. Create UI for viewing individual tasks on calendar 5. Implement pet aging & changing over time intervals, depending on status |
| Week 5  9/26 - 10/3 | 1. Setting tasks at certain time intervals for calendar 2. Create UI for editing individual tasks on calendar 3. Make announcements to user when they are coming close to a due date for their task |
| Week 6  10/3 - 10/10 | 1. Adjustable calendar UI 2. Allow regeneration of pet upon the death of one 3. Create a settings window for application |
| Week 7  10/10 - 10/17 | 1. Prepare for midterms 2. Mid-semester meeting/Progress management (Longer than Scrum) |
| Week 8  10/17 - 10/24 | 1. Confirm calendar and task list interaction is successful 2. Confirm task list and calendar are unaffected by pet death |
| Week 9  10/24 - 10/31 | 1. Confirm pet is programmed to follow task list and date/time provided by the calendar |
| Week 10  10/31 - 11/7 | 1. Implement notification system for upcoming tasks/dates in calendar 2. Possibly implement pet to overlay other windows other than desktop |
| Week 11  11/7 - 11/14 | 1. Create poster for project (due the 14th) |
| Week 12  11/14 - 11/21 | 1. Complete any unfinished items |
| Week 13  11/21 - 11/28 | 1. Thanksgiving Break 2. Complete any unfinished items |
| Week 14  11/28 - 12/5 | 1. Complete any unfinished items / Prepare for finals |
| Deadline  12/6, 1:30 - 3:30 PM | 1. Submit final report 2. Live demonstration and presentation 3. Submit all code to D2L and GitHub |

