FLUTR

Design Document

sddec24-11

Reiman Gardens

Dr. Diane Rover

Amanda Friis - Full Stack Developer, Documentation

Tayler Barnhart - Team Leader, Cloud Integration

Nathan Geater - Full Stack Developer (Back End)

Alex Brown - Front End Development Leader

Muralikrishna Patibandla - Integration & Back End Development Lead

sddec24-11@iastate.edu

sddec24-11.sd.ece.iastate.edu

DOCUMENTATION - FLUTR
2024

CLIENT: MR. NATHAN BROCKMAN
ADVISOR: DR. DIANE ROVER

Group Members

Amanda Friis - Full Stack Developer, Documentation
Tayler Barnhart - Team Leader, Cloud Integration
Nathan Geater - Full Stack Developer (Back End)
Alex Brown - Front End Development Leader
Muralikrishna Patibandla - Integration & Back End Development Lead

Executive Summary

Development Standards & Practices Used

The Flutr system will be almost entirely a software-based project. The software systems we plan to use for the project include the React code base for frontend development, MongoDB and Springboot for the backend database and control, and DigitalOcean for cloud hosting of the entire system. Additionally, we will be using Git to host our codebase, which allows us to break up tasks into smaller groups and create branches to allow different team members to work on their respective tasks. We will be implementing an agile-style sprint system during the latter half of the project timeline, as this integrates well with the feature-creation system we will be using on Git. As such, each developer will be documenting feature creation through each sprint, which will allow us to move the project forward at an even pace. All code will be reviewed by another developer before pushed to production.

For the Kiosk system, we will be doing basic documentation of the purchasing and implementation. The majority of Kiosk work will be testing, as interior temperature is the biggest problem with the system. Bi-weekly testing will be done on the kiosk to ensure proper performance.

The Engineering Standards we are considering include IEEE Std 1063-1987, ISO/IEC 27001, WCAG, and IEEE 830, all of which are documented in Section 2 below.

Summary of Requirements

For this project, these are the requirements we will be following.

Architecture

- Database
    - Cost-effective
    - Scalable for multiple institutions
    - Editable
- Front-end
    - Consistent coloring/ UI design
    - Big, readable buttons
    - User-friendly (ie. simple steps, readable, etc.)
    - Alphabetical ordering of both scientific and common names

Features

- Users
    - Super user: Creates other primary admin users
        - Can pull all institution data
    - Head admin: One per institution, Creates other admin users
    - Admin: Logging releases / shipments
    - General public: No login
    - Kiosk mode: No login, No exterior links
- Release
    - Same options as Flutr currently has, but more navigable and user-friendly
    - Tracking which shipment a butterfly comes from before release
- Shipment
    - Adding/Deleting new suppliers
    - Editing shipments
- Data tracking
    - Flight time based on daily release
    - Shipment export

    - Release export
    - Butterfly export
- Editability
    - Shipment and release database entries
    - Edit butterflies (ie. common name, flight time)
- Kiosk
    - Environmentally sound internal and external components
    - User-friendly interface
    - Make it unable to leave the kiosk software

Optional Requirements

- Translation
- Using voice to enter butterflies
- Auto-generate paper for incubation

Applicable Courses from Iowa State University Curriculum

The most applicable course taken by all team members is ComS 309: Software Development Practices. In this class, we were each tasked with creating an Android mobile application, using Android Studio, Git, and Springboot. Our team will be mimicking similar practices, as we will be splitting into a frontend and backend team, while implementing similar Git style, using separate branches for developing new features. Additionally, the backend software will be very similar to the class, both using Springboot to manage the backend database.

Other classes that some team members have taken that apply to this project include ComS 319: Construction of User Interfaces and SE 329: Software Project Management. ComS 319 helped teach general Javascript coding and the creation of UI systems for software projects. This also includes object-oriented programming applications to UIs and event-driven programming. SE 329 focused on general software project management, including using different tools to manage projects and analyzing different softwares.

New Skills/Knowledge acquired that was not taught in courses

New skills that the team will need to acquire include Cloud systems, React UI coding, and CSV importing and exporting into a backend database.

For Cloud systems, some members have general Cloud knowledge, but hosting full websites and backend databases on a single Cloud server is new to all members.

Although all members have general frontend experience through classes, React is a new language to all team members, and as such, the frontend team will be focusing on learning React, as well as how to integrate React code with the backend database.

Finally, as previous shipment data will need to be imported into the new system, formatting the backend in a way to allow for this import will be a new step for all members. This includes the importing process, which will require the team to find the tools necessary to do this. Shipment data will also need to be exported to a CSV, which will likely use the same tools as importing to do.

These are all new skills that the team will need to acquire in order to successfully complete this project.

Introduction

Problem Statement

Butterfly gardens around the globe complete the same process as each other every day. Gardens will release butterflies into their habitats at least once per day, they then have to record the species of butterfly they released and which shipment they came from. Despite this being such a common process across all of the institutions in the world, they don't all record things in the same way. Enter Nathan Brockman and the people of Reiman Gardens. Years ago, a Senior Design Group made an application called reimanbutterfly.com. This app supplied the entomologists of Reiman Gardens with a tool that allows them to track their shipments, mark butterflies as released, and generate a report for the USDA at the end of the year.

Nathan at Reiman Gardens has decided that their current solution is far outdated and needs to be completely redesigned. He also wants the platform to be generalized so that any institution that he grants access to can use the software and the process can be standardized across the world.

An additional benefit of this software is that with the butterfly data already collected, it can easily be formatted into a guest facing page that displays fun information about the location's butterfly habitat. We can display things like a butterfly of the day, special notes from the entomologists at that location, and statistics about which butterflies are present in the habitat.

In addition to the app, we will also be overhauling Reiman Gardens Kiosk that will display our completed project. This will be a Reiman specific part of the project, but can pose some interesting problems. The case we have to build in is incredibly limited in terms of space, will be kept in an environment of 80℉ and high humidity.

Intended Users

For our project, we have 3 main users - Global Admin, Local Admin, and General Users.

Global Admin (GA)

Who is this?

In our system, we plan to have only 1 Global Admin - our client Nathan Brockman. The GA is someone who knows the system well and oversees the use of Flutr around the world.

What are their needs?

The Global Admin needs a way to oversee the flight houses and the butterflies in them. The difference between the GA and the Local Admin is that the GA also gains the ability to service other flight houses (each run by an LA). The GA can create new flight houses under the Flutr system for LAs to use for their flight houses.

What do they gain?

From this, the global admin gains the ability to easily control the work of other flight houses and LAs. This includes the ability to add new butterflies and their photos, as well as manage the databases of each house. This helps solve the problem of a lack of admin system in the previous system, where the creation of other flight houses was tedious and the control from the GA was very limited.

Local Admin (LA)

Who is this?

A local admin is either the leader or a worker of any butterfly garden that the Global Admin has registered into the system.

What are their needs?

Local Admins need to be able to enter their shipments, “pull” butterflies from those shipments and release them into their garden, update the common name and estimated lifespan of a butterfly species, update the look and information of their customer facing page and add more LAs to their own garden’s page.

What do they gain?

The LAs gain a sense of uniformity between all butterfly houses that use Flutr. They also gain good record keeping tools and a way to organize their information in a fun and interactive way for their guests.

General Users (GU)

Who is this?

General users are guests at the butterfly garden. They don't require any sort of login, this is just the default page, expected to be accessed from either the kiosk or on the guests personal mobile devices.

What are their needs?

GUs want to learn about the butterflies that can be found at their location, get fun facts about the location and be provided with an overall good user experience.

What do they gain?

GUs gain knowledge about the butterflies surrounding them. They also receive fun facts and some specific information from LAs in the form of the "Note".

Requirements, Constraints, and Standards

Requirements & Constraints

Employee UI

- Editing of current and past butterfly tracking data
- Creation of new flight houses with customization options
    - Butterfly common name
    - Flight house logo/branding
    - Employee creation for every house, with respective user access levels (constraint)
- Butterfly photo editing (Only for super user)
- Exporting of butterfly tracking data

Visitor UI

- Display information specific to each Butterfly Garden
- Good landing page that shows all gardens using the platform
- Kiosk Specific version so users can't escape site through external links like social media links (constraint)

Kiosk

- Sufficient cooling method to keep the components cool in the high-temperature, humid conditions of the flight house
- Proper sealant system to keep bugs out of the interior of the kiosk and prevent damage to cables and other components (constraint)
- Efficient, resilient touch interface for visitor interaction
- Fast and lag-free user experience with kiosk software

Engineering Standards

IEEE Std 1063-1987 (Standard for Software User Documentation)

- Ensuring that the software documentation is comprehensive, clear, and useful to all user groups. This standard covers the applicability, purpose, document usage, conventions, issue reporting instructions, tutorials, and glossary, which are all important to allow for easy adoption and maintenance of the project.

ISO/IEC 27001 (Information Security Management)

- Protecting sensitive data related to the operations of different flight houses, as well as user and employee data collected through the system. This standard outlines the best practices for establishing, implementing, and maintaining information security.

WCAG (Web Content Accessibility Guidelines)

- Ensuring the website and kiosk UI are accessible to all visitors, including those with disabilities. Compliance with this standard makes the website's content more accessible to a wide range of people with disabilities.

IEEE 830 (Recommended Practice for Software Requirements Specifications)

- Guides the development of clear software requirements specifications, in order to ensure the final product will meet the needs and requirements of the client and all user groups.

Project Plan

Project Management/Tracking Procedures

As features and work begin to be implemented, we will be adopting an agile sprint strategy, where the team will run 2 week sprints in order to assign work and implement features. The main idea behind this is that it will allow our team to work in smaller groups on different things, all while staying on the same track with one another. With the Flutr project, the project has a wide array of different features and problems we need to solve (which can be tied to the requirements needed). Allowing members to take smaller pieces and work on them over a couple of weeks helps with time management, as school, clubs, sports, and other things can make it difficult for the group to meet together often.

For tracking progress, we will use a couple of different tools. The first being the weekly reports that the team fills out - this makes for an easy tracking method, as we can see a short description of the project each member is working on and the hours that each member has put in each week. Additionally, we will also be using Gitlab to track individual features with milestones. This allows us to track more detailed information about the problems a member might be facing. Additionally, we will communicate over Discord multiple times a week for general check-ins and questions. Finally, the team leader (Tayler) will periodically check in with each member to make sure all problems are solved and that the team is moving forward together.

Task Decomposition

The tasks in this project mainly pertain to the web app, though a small bit deals with the kiosk. As such, they will be separated into different sections.

Web Application

- Finalize frontend and backend tools
- Implement database and app onto cloud hosting service
    - Design database to hold long periods of tracked data, as well as butterfly facts, details, and images
- Create communication interface between frontend and backend
- Create front page framing & general website design
- Build the butterfly tracking system & connect with backend database
- Allow previous database access & editing
- Export previous data to external source (PDF, Spreadsheet)
- Implement security admin system with multiple levels of admin & permissions
- Design app for multiple devices, including computer, tablet, and phone touchscreen
- Create automated butterfly counter with death counting
- Use data for intuitive maps, charts, and facts about the current butterflies in the database
- Allow for generation of new houses
    - Incorporate a simple tutorial system for other houses
    - Allow for house customization from the head admin
- Create documentation of our process and final design for bug testing and future updating

Kiosk

- Research correct computer parts needed for the kiosk
- Design & implement “parental controls” software to prevent the kiosk from accessing the internet freely
    - Implement QR code system to allow users to find the social media on mobile devices without leaving the Flutr mainpage.

- Test hardware parts in hot and humid environment (80℉ and 80% humidity)
- Install computer parts into kiosk
- Test software and internet prevention in kiosk

Project Proposed Milestones, Metrics, & Evaluation Criteria

- A New Computer System Will Be Built for the Reiman Gardens Kiosk (End of Spring Semester)
    - Cooling System Will be sufficient to get through the summer months.
- Login System Created On The Backend with all User classes accounted for
- Basis for landing screens created on frontend. Functional with simulated postman server.
- Login connected on frontend.
- Location Creation and Head Admin Creation process implemented full stack.
- Entry of shipment information at individual locations created.
    - Allows for editing of submitted forms. IMPORTANT.
- Locations can perform “Release” action on their database of received butterflies.
- Head Admins can customize their location’s database specifics such as Common Names, Estimated lifespans
- Head Admins can customize the look of their customer facing website (logo, color scheme, social links, contact information, ‘The Note”)
- Frontend displays the customer facing site for unauthenticated users.
- Statistics and Butterfly of the day created for customer facing website.
- Quality of life improvements (Switching Common and Scientific Names on the fly, picture toggling.)
- Beta Test by November ‘24
- Final Touches/ Bug fixes found in Beta by Mid December ‘24

Project Timeline/Schedule

Target 1: Kiosk

Planning

- Decide on what hardware we want to use

Design

- Create a plan for how to fit everything into the kiosk

Development

- Tear apart old kiosk to make room, find what we need to replace
- Replace all old tech inside kiosk with new tech

Testing

- Set up in Reiman Garden Pavilion
- Fix any concerns that arise during testing period

Deployment

- Permanently install

Phase	Start Date	End Date
Planning	March 1, 2024	March 16, 2024
Design	March 17, 2024	March 20, 2024
Development	March 21, 2024	April 19, 2024
Testing	April 20, 2024	May 4, 2024
Deployment	May 5, 2024	May 10, 2024

Target 2: Website Creation

Planning

- Requirements gathering
- Research on platforms
- Study previous projects for their failures and successes
- Develop general time frame

Design

- Choosing platforms for frontend and backend
- Wireframes for potential web pages
- Finalize requirements
- Plan software architecture

Development

- Create database
- Develop webpages
- Implement features

Testing

- Website testing
- Database testing
- Communication testing

Deployment

- Set up in Reiman Gardens for initial user-testing period

Phase	Start Date	End Date
Planning	February 1, 2024	March 10, 2024
Design	March 11, 2024	March 31, 2024
Development	March 18, 2024	May 15, 2024
Testing	April 15, 2024	September 1, 2024
Initial Deployment	May 10, 2024	May 15, 2024

Target 3: Website Customization

Planning

- Decide on database framework to use (Mongo DB)

Design

- Prioritize intuitive and functional design and aesthetics for visitor and staff experience

- Add fun facts for users to see

Development

- Build login system for different flight houses with their own permissions

Testing

- Website testing
- Database testing
- Communication testing

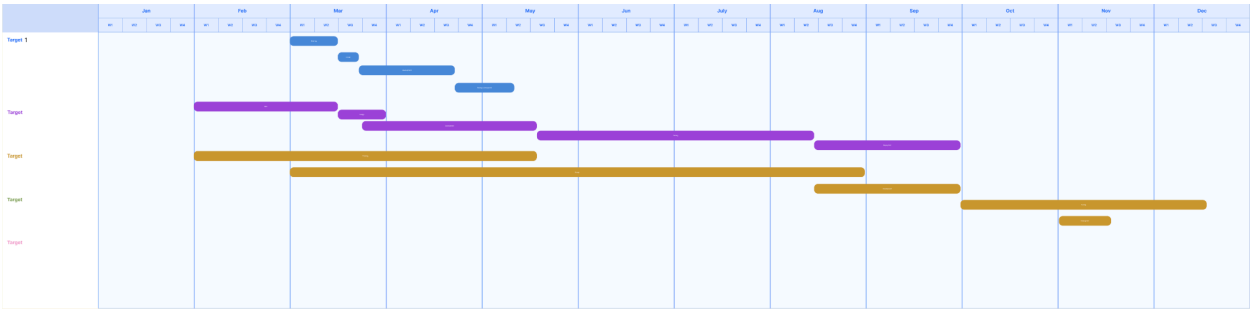
Deployment

- Deploy to other pavilions

Phase	Start Date	End Date
Planning	February 1, 2024	May 10, 2024
Design	March 1, 2024	September 1, 2024
Development	August 15, 2024	September 30, 2024
Testing	October 1, 2024	December 10, 2024
Deployment	November 1, 2024	November 15, 2024

Gantt Chart

Beginning February and ending in December.



Risks & Risk Management/Mitigation

Web Application

- Implement Database and App onto Cloud Hosting Service

    - Risk: Overspending or underestimating required resources.
    - Probability: 0.5
    - Mitigation: Use cloud services with extensive documentation, accessibility features, and scalable payment models. Conduct in-depth analysis to compare services.
- Design Database for Long Periods of Data Storage
    - Risk: Inefficient data retrieval and storage costs.
    - Probability: 0.2
- Create Communication Interface between Frontend and Backend
    - Risk: Incompatibility issues leading to poor performance.
    - Probability: 0.1
- General Website Design and Front Page Framing
    - Risk: Design not being responsive or user-friendly on all devices.
    - Probability: 0.2
- Build Butterfly Tracking System
    - Risk: Tracking system inaccuracies or inefficiencies.
    - Probability: 0.6
    - Mitigation: Prototype early and test with real data. Act on and incorporate feedback from the client, staff, and volunteers based on testing.
- Implement Security Admin System
    - Risk: Inadequate security leading to unauthorized access.
    - Probability: 0.5
    - Mitigation: Use a tried and tested authentication framework. Regular security maintenance and updates.
- Design App for Multiple Devices
    - Risk: Poor user experience on certain devices.
    - Probability: 0.1
- Create Documentation for Maintenance, Updates, and Customization
    - Risk: Documentation being unclear, insufficient, or complex.
    - Probability: 0.5

- Mitigation: Follow best practices in technical writing. Regular updates and peer reviews of documentation. Conduct user testing for feedback on features and simplify the UI.

Kiosk

- Research and Install Computer Parts
    - Risk: Parts not functioning as expected in the environment.
    - Probability: 0.7
    - Mitigation: Select components rated for extreme conditions. Test and compare different options in various possible conditions.
- Design and Implement Administrator Control Software
    - Risk: Software being bypassed or failing to function.
    - Probability: 0.2

Personnel Effort Requirements

Task	Hours	Explanation	Source
Authenticate admin level users at individual location	30	Need to ensure that the super user can create other admins and admins can create other general users at their given location. Need to ensure security at each authentication level.	These estimates are gathered from our own personal experiences from classes and internships.
Take in information about shipment of butterfly from the user and save (Species, count, supplier)	40	Webframe design and development of webpage: 10 hours Database design and development: 10 hours	
Take in information about which butterflies are being release into habitat and which shipment they came from	40	Webframe design and development of webpage: 10 hours Database design and development: 10 hours	
Edit general	30	Ensure user can edit a webpage	

information about the location and the site		based on their individual location	
Update shipment database	20	Edit/add/delete information from the shipment database	
Update list of which butterflies are in habitat, and start their respective timers	30	Develop life timer based on release and estimated life span Edit/add/delete information from the release database	
Serve current information to the general public	10	Home-page website that users can view to see butterflies currently flying	

Table 3.1: Personal Effort Requirements

TOTAL HOURS: 200 hours

HOURS PER PERSON: 42 hours

Given a time period of February - December, and excluding summer (about 33 weeks)

HOURS PER WEEK PER PERSON: 1.2 hours

Other Resource Requirements

This project requires the use of a hosting service, which DigitalOcean is currently the chosen option, to deploy the site on. It also requires a domain, which will be “flutr.org” and is currently owned by Reiman Gardens. While both of these are closely related to financial resources, they can sometimes be forgotten as a resource needed.

Design

Broader Context

We are designing a niche product for a niche user group. There are two user groups for our project, Entomologists at Butterfly Gardens across the world, and the general public that visit those gardens. This project is a tool for learning, a tool for workplace data management, and a tool to unify the field of Entomology.

Area	***Description***	***Examples***
Public health, safety, and welfare	The main people affected by this are the entomologists at the various gardens using the product. Their wellbeing is benefitted by making their work activities easier to accomplish.	Welfare of workers in butterfly gardens improved by better practices and easier work.
Global, cultural, and social	This project is all about unifying the processes that entomologists across the world complete, or at least the record keeping portion of it. Having one software system that everyone uses, should be very effective in bringing all of these people together throughout the world.	Global effects related to unification of processes across the world of entomology.
Environmental	Our project is almost entirely digital and should have little to no environmental impact. It will require some power to run, but it is a relatively lightweight software system.	Lightweight programs will have a negligible impact on environmental factors.
Economic	Our client Mr. Brockman will be paying for all hosting of this application in perpetuity through Reiman Gardens. Other gardens will be given access to this software for free as an incentive to get them to use it.	Products will be available for free to all Butterfly Gardens that request access.

Table 4.1: Project Considerations

Prior Work/Solutions

Our team is redesigning and improving the previous work of two senior design projects.

reimanbutterfly.com

The first attempted solution is *reimanbutterfly.com*. One advantage of this software is that there are multiple ways for guests to interact with the website. There is a home page with a welcome message, butterfly and fact of the day, and a personal note from our client. There is also a statistics page that provides interesting visuals. Finally, there is a butterfly search feature that allows a user to input certain features they see on a butterfly. From there, they can narrow down which butterfly they are looking at while in the pavilion or search any butterfly in general. Along with the butterfly gallery, these features curate a personal connection and interactivity for guests that enhances their experience and learning.

Another advantage of this solution is that it provides a relatively easy way to track daily release and shipment data. This application allows Reiman Gardens to create and input various data from shipments. This includes what butterflies they receive, if they have emerged, if they are damaged or sick, and more. It also provides a way for Reiman Gardens to keep track of which butterflies are currently released in the pavilion.

While this application provides a variety of impressive features, it has its limitations. The most major of which is that this software was not developed to be distributed outside of Reiman Gardens. Our client is asking for a solution that can help pavilions world-wide.

flutr.org

The second iteration has most of the advantages of *reimanbutterfly.com*, but was developed with the intent of distribution. However, there were disadvantages that kept it from fully reaching that goal.

One of these disadvantages is that the UI/UX is clunky and not user-friendly. For example, the user cannot sort the butterfly via the common name instead of the scientific name. This can be a barrier to entry for new users. Another disadvantage is that the backend was set up in a way that makes it difficult to pull data that isn't directly related to shipments. This makes it harder to create the same visual graphs that is on *reimanbutterfly.com.* The most

major disadvantage is that flutr is currently rigid. As in, there is no option to add new suppliers of shipments, and there is very limited editability of certain data points. Due to these list of issues, our client has decided to have our team rework flutr, rather than release it in an unfinished state.

Kiosk

The second part of our project is a redesign of the kiosk within Reiman Gardens. We are incorporating aspects of the previous work by using the same metal case. This is an advantage in terms of lessening our overall workload, but also limits our design options. There are also two major disadvantages of the previous design.

Firstly, the previous design was not sealed properly. This allowed bugs and other pests to damage the internals of the kiosk. Secondly, there was no proper cooling in place to account for the extreme heat and humidity of the pavilion environment.

However, the software running on the kiosk met all the requirements our client wanted. The only disadvantage to this software is that the guest was able to find a way out of the software and onto the internet.

Technical Complexity

Our application is a sophisticated intertwining of technologies such as React.js, Spring Boot, MongoDB, and hosting solutions. We are creating this application from the ground up and in a way, we have to work as though we are a startup company. We also have the task of rebuilding the kiosk that Reiman Gardens have. This is a complex task because there are a lot of factors against us, mainly environmental. The harsh 80 degree temperatures and high humidity of the butterfly enclosure are difficult on computer components and on the ability to cool them effectively. Combine this with the small profile of the case and it is quite the feat.

Design Decisions

Software

User Hierarchy

One key decision we made for our software is the formatting of our user hierarchy. We have decided to have a super user, head admin, admin, general public, and kiosk. One reason this decision is important to our project is that our client requested to have sole access to certain data from the pavilions; however, he also does not want to have to micromanage each pavilion. By having a separation between super user, head admin, and regular admin, we can achieve this by allowing the super user access to data, and the head admin to create regular admin accounts. This way, our client only has to create a head admin for a pavilion, and they can handle the rest internally.

Furthermore, our hierarchy also allows a separation between the general public and kiosk mode. The general public can view butterflies currently in flight, view interesting statistics, find information about those butterflies, and follow a link to the pavilion's social media. The kiosk mode has all of these features, excluding the links to social media. This is to ensure that a guest cannot accidentally or intentionally leave the Flutr app on the kiosk. We decided to use a QR code on the kiosk, rather than a link..

Customization

Another key design decision for our software is a high level of customization for each pavilion. One area with a high level of customization is a pavilion's home page. The customization options include editing the homepage to have a 'welcome note' with any information they wish to present to their guests, choosing coloring, uploading images, and enabling statistics and butterfly of the day. This allows a pavilion to truly feel as though the software was personally designed for them, which is important to having a positive user experience.

Another area of customization that is useful is being able to edit a butterfly's common name and their probable longevity. Nathan informed us that while scientific names are universal, common names are not. Therefore, we decided that a pavilion can edit the

'common name' to differ from what Reiman Garden uses, but cannot edit scientific names. Furthermore, having a user be able to edit their own probable longevity (lifespan) of a released butterfly can provide important data for Nathan and the pavilion. If a butterfly has a shorter probable lifespan than the average, that pavilion can study what needs to be changed to prolong the lifespan.

Kiosk

The most major key design decisions we will need to make for our kiosk hardware is to ensure it is environmentally sound. The environmental factors we need to consider are high humidity, heat, and bugs. We must also replace the damaged touchscreen. To make this decision, we have made two plans that we will present to Nathan.

The first plan is to have the kiosk software running inside the lobby, and run an ethernet cable to the kiosk. This plan avoids having to factor in the environmental elements, but could have issues with powering the kiosk and Nathan might prefer to not have a cable coming from the lobby to the pavilion.

The second plan is to completely gut the current kiosk and replace damaged components. This requires more planning around the environmental factors, but will allow us to improve the components and implement a better cooling system.

Ideation

For our project, many of our design decisions are based on the work and results of the groups that worked on this project in past years. Our client underlined his preference for a solution that included tried and tested features of the outdated versions of this project, and so our ideation process revolved around finding parts of the previous solutions that were missing or nonfunctional.

We considered the following during our ideation process:

- Through discussions with our client, Nathan, and our observations of the various procedures at Reiman Gardens, we gained insights into the necessities and wants for the project.

- By analyzing the limitations and issues faced by the previous versions, we identified technologies that we had expertise in which we could carry over. We also identified frameworks that offered responsiveness, user-friendliness, scalability, and versatility.
- We also prioritized enhancing the user experience to make the application more accessible, intuitive, engaging, and efficient for its various user groups.

For our design decision relating to the technologies we picked for our project, we considered the following:

- Drawing inspiration from the cloud hosting implementation in the previous iterations. We had extensive meetings with our client and after thorough research, we decided to continue using DigitalOcean for our cloud hosting platform.
- The most recent version of Flutr was built with Go and MongoDB, and we identified that MongoDB was the most efficient, scalable, and supported database architecture option. However, since our team had no experience with Go, we decided to build our project's backend using SpringBoot, which all of us had some degree of expertise in. We considered various alternatives as well, such as C#, Django, and Node.js.
- When considering frontend-backend integration, we needed to decide on an efficient frontend framework, and after detailed discussions and research, we decided on using React, which is very modern and well-supported, in addition to being versatile.

Decision-Making & Trade-Off

For our design decision-making regarding the backend framework we picked for our project, we considered the following:

- Go: Was considered for the existing iteration of Flutr, along with its performance advantages. However, we decided against it due to the team's unfamiliarity with the language.

- SpringBoot: Was considered for the team's experience developing applications in or alongside SpringBoot in previous courses, along with its comprehensive feature set, community support, and versatility.
- C# (with .NET): Was considered due to its powerful features and professional capabilities. However, the team was unfamiliar with this language, and we found it not as suitable for our project.
- Django (Python): Was considered for its rapid development and its suitability for data-driven applications, but we considered its performance abilities not on par with the potential of SpringBoot.
- Node.js: Was considered for full-stack potential and efficient I/O handling. However, we had concerns about security and complexity.

Our decision to proceed with SpringBoot as our backend framework was driven by the team's prior experience with the language. We also chose it to avail its extensive community support and versatility. The vast compatibility options, productivity benefits, balanced performance, and development efficiency were key factors in our selection of SpringBoot as the most suitable technology for our project.

Proposed Design Overview

Our design allows for butterfly garden workers to access a web application and keep track of the data related to their butterfly shipments. From these shipments, workers can also keep track of the butterflies released into flight from these shipments. This data is collected for required report generation at the end of the year. Admins can customize things about their specific butterfly site, like the estimated lifespan of specific species as they see fit. With this data already collected, it is easy to make an experience for guests of the butterfly garden that they can access through a kiosk or their own personal devices.

Detailed Design & Visuals

The Flutr application will use multiple different tools to implement the functionality, including Springboot, React, Github, and DigitalOcean. Shown below is a detailed diagram of the tool implementation, followed by a listing of how each component will interact.

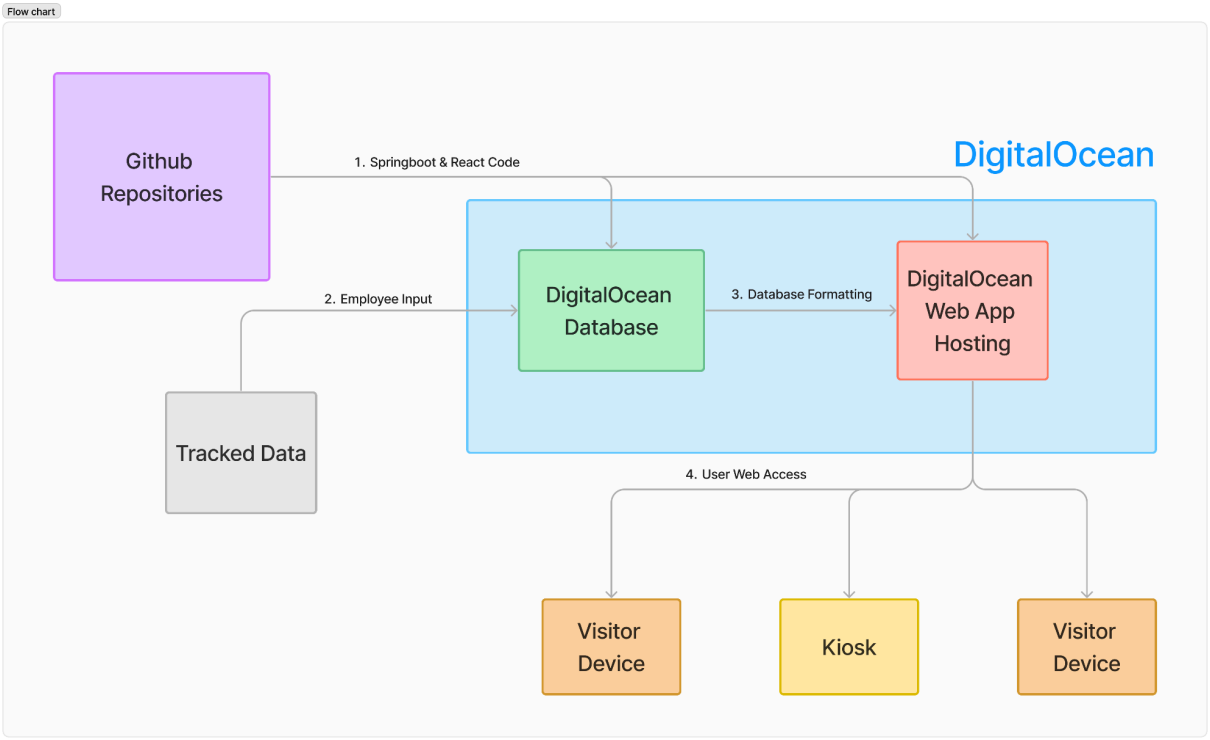


Figure 2.2: Hosting Diagram

Figure 2.2 shows a basic interaction diagram of all tools used to host the website, with each arrow being detailed further below.

1. All code will be managed through Github, where the primary code bases will be Springboot for controlling the backend database, and React for creating the UI in the hosted web application.
2. Using the hosted web app, employees will be able to add to and edit the database holding previous butterfly data. This will be done using a private section of the web app, guarded by a login screen.
    a. NOTE: Although not shown directly, the employee tracked data will come from a portal in the web app, not through a separate database or web app,

nor by hand. All employee access to the database will be done through the web application.

3. The database will relay data to the web app, which in turn will format the data into user-friendly charts for visitors to learn with.
4. The visitors will access the web app on their personal devices or on the Reiman Gardens kiosk located inside the flight house.
    a. NOTE: The kiosk feature will not be an option for other flight houses who use this system.

For each component, a description of their usage is given below.

Github

Github will be used for hosting our codebase. The primary repository will hold both the frontend and backend implementation, each in their own respective folder. Team members will create branches to implement features, which will be merged into the main branch and thus updated into the hosted web app and database.

Springboot

Springboot will be used to design and implement the MongoDB database hosted in DigitalOcean. This code will be hosted on Github.

React

React is another code language that will be used for building the frontend web app for both visitors and employees to interact with. This is a commonly used frontend code language, which will also be held on Github and implemented into the web app. Both React and Springboot will communicate together to send data between the web app and the database.

DigitalOcean

DigitalOcean is the primary cloud hosting platform which we will use to hold both the database and the frontend app. By hosting both on the same cloud platform, DigitalOcean provides easy connections between both processes.

DigitalOcean also provides automatic implementation of Github code - when code is updated on the main branch of Github, DigitalOcean will automatically deploy both the database and the web application to reflect these changes. This removes the need for a CI/CD pipeline.

Functionality

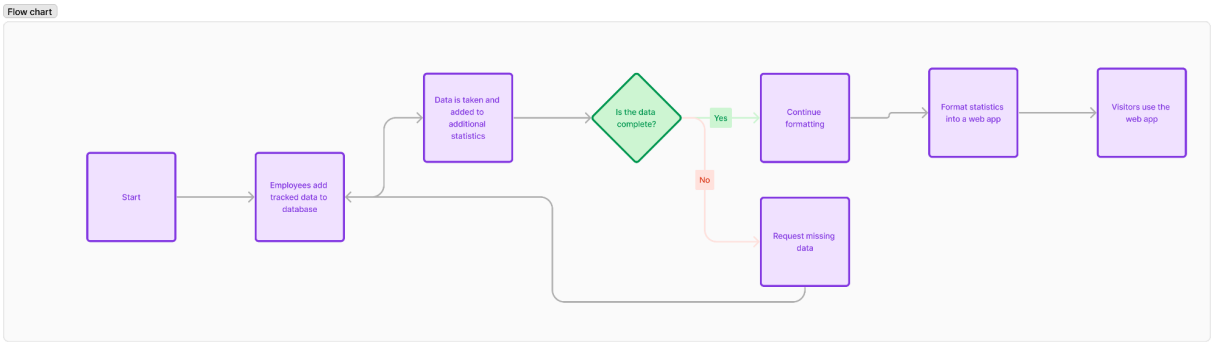
As this project has 2 main users, the functionality for each user is different.

Employees

For the employees, the main functionality will be tracking the shipments and butterflies released into the flight house. To do so, they will be marking each butterfly they are released into the system, using an easy tracking UI. This will then be transported to the database where all past data will be held.

Visitors

For visitors, they will be using an interactive web app in order to learn more about the flight house and the butterflies inside. For this, we will take the employee tracked data, as well as additional facts in order to create an easy-to-use web app. Visitors will both be able to use their phones as well as the in person kiosk at Reiman Gardens.



The flowchart shown above details the data flow between the database and the user devices.

Areas of Concern & Development

Based on our multiple requirement gathering meetings with Nathan, we are confident that our current design meets his needs as a super user. We designed our product so that it improves features that Nathan likes on the current app, and adds features he feels are missing. Another group that is similar is other pavilions. We also feel that our design meets their needs because our client is an industry expert. He has guided our design to not only be useful to Reiman Gardens, but to butterfly pavilions as a whole.

Our final group of users is the general public. Our current design meets their needs by creating an interactive element to the pavilion. With Flutr, a guest can find more information about a butterfly they are currently seeing within the pavilion. This will improve guest experience and engagement.

One primary concern for delivering our final project is ensuring that the software can be easily integrated into other pavilions. We must ensure that international users can utilize all the key features of our software without us present. Our immediate plans for a solution is to create an in-depth user manual with an index, glossary, screen captures, feature highlights, and more. We also plan to include contact information so other pavilions can ask questions, suggest improvements, or report bugs. A question we have for our TAs is what is the best way to test this software to ensure as smooth of a rollout as possible? It would be beneficial to find and fix as many bugs as we can while it is still local.

Another primary concern is environment-proofing the kiosk hardware. We need to ensure that our core components do not overheat in a high humidity environment. On top of having proper cooling, we also need to prevent bugs from getting into the case. A question we have for our TAs is what recommendations they have to improve cooling and ways we can prevent bugs from the case?

Technology Considerations

Our tech stack is made up of a DigitalOcean hosting solution, React.js Frontend, Springboot for the backend API, and MongoDB for the backend database. Our client wants us to “make it cool” and be on the cutting edge, but not so cutting edge that the technology is unproven. We chose popular frameworks that are on the leading edge of the space. React is a great improvement upon the basic usage of Javascript. It has been around for a while now and is Open Source, while still backed by Meta. It is very powerful and can be expected to be supported for a long time. DigitalOcean as a hosting solution was chosen because our client already uses them to host other applications, and some members of the team are already familiar with its workings. Springboot was chosen for our Backend API because we all have some exposure to it from COM S 309. Some of us have more experience than others, but it is built on Java, and we are all familiar with that. Finally, we chose MongoDB as our backend database because some members of the team had experience with it and due to its scalability.

As part of this project also involves rebuilding the Kiosk in the Reiman Gardens Butterfly Wing, we have to make some hardware choices as well. We are building a middle to low range computer that is able to display a webpage and interpret touchscreen input. This is not the primary concern of the group when it comes to our hardware considerations. The major focus of this computer build is small footprint and thermals. This computer will be located in the Butterfly Garden and needs to be able to survive with constant uptime in a humid environment that is kept at a consistent temperature of 80℉. The case for this computer is also quite small and not very accommodating to large coolers. A couple of solutions discussed have been using a Raspberry Pi, Watercooling a micro-atx computer build, and potentially even running cable outside of the butterfly enclosure (either fiber optic or CAT6 Ethernet) and running the computer outside of the warm environment. It is undecided at the moment.

Design Analysis

So far, we have laid groundwork for the backend's framework and began writing a frontend application in React. Things have essentially been going to plan and we will move forward with said plan. The goal for the coming weeks is to develop both in parallel and come to a point where we can integrate them.

Testing

Unit Testing

The units we will be testing include SpringBoot backend services, MongoDB database models, and our React frontend components.

- We plan to test our SpringBoot application services using JUnit and Mockito for backend Java tests, isolating and testing each class and external dependencies.
- Our MongoDB models can be tested using MongoDB Compass to ensure data validation and verify that database rules are being enforced.
- To test our React components, we can use the React Testing Library to isolate modules from the DOM and verify their functionality and behavior based on different states. We could also use Mocha and Chai for more detailed testing.

Interface Testing

The interfaces in our design include API endpoints, database interactions, and the user interface.

- Interactions between the frontend and the backend are done through RESTful APIs. We will use Postman for testing and validating request handling, responses, and error management.
- Interaction between the backend and the databases is done through the Hibernate framework and will be tested using Postman and JUnit.
- Cypress can be used for end-to-end testing of the application to simulate user actions and verify interactions between UI components and the backend.

Integration Testing

The most critical integration paths in our design are:

- User Session and Access Management: Ensuring that user sessions are correctly initiated, maintained, and terminated across different operations and components. Ensuring user authorization flows are correctly implemented across the system.
- Data Integrity: Ensuring seamless data flow between the frontend application, backend server, and the database collections.

Testing these integration paths can be done using the built-in Spring Integration Tests to simulate the running application and verify integration points, and Cypress to automate browser-based integration tests and cover scenarios like login, data entry, and navigation.

System Testing

We intend to carry out comprehensive testing to ensure the combined operation of the system as a whole, involving stress testing, load testing, and scalability tests. To perform these tests, we intend to use GitHub Actions to run all tests whenever changes are committed and catch errors early.

Regression Testing

Regression testing shouldn't be all that important with the extensive planning we are doing, but we will still complete some. We will continuously test our product as we add new features, especially when they interact with old features.

Acceptance Testing

We will continuously demo to our client Nathan Brockman in order to make sure we are completing requirements as he sees fit. We also are shooting to finish the application in the early days of November to have a bit of a beta test before we are completely done with the project and have graduated.

Security Testing

The data stored in this project is not all that sensitive, but it would be a good idea to simulate a few attacks that could be run on our software.

Results

These tests together should do a good job of comprehensively solidifying our application as a modern and secure application that can successfully fulfill its requirements without the fear of error. All the different fields of testing should make our application as strong as it can be.

Implementation

The majority of our implementation plan was stated throughout the design section of this document, mainly focusing on the details of each overarching design decision. For implementation, we will be adopting an Agile sprint style of management, as our project can easily be broken down into “features” that individual team members will be able to pick up in each sprint and implement into Flutr. Shown below (Table 6.1) is the planned management/implementation style.

Cloud Integration	**Agile Sprint - Features**	**Testing**
Integrating the frontend and backend system into DigitalOcean, while creating a communication bridge between each end	Creating features to add onto the frontend and backend sides. Implementing each page on Flutr	Load testing among various devices to ensure Flutr can handle a large number of users, among other tests.
Mid Aug - Mid Sep	Mid Sep - Mid Nov	Mid Nov - Mid Dec

Table 6.1: Implementation Time Table

However, this will not always be the style. For the first few weeks, we will focus on implementing the website fully into the Cloud hosting software, as well as integrating the frontend and backend together in a seamless way. This will be focused more on a “milestone” system and less of a “sprint” system, as almost all team members will need to work together to implement this stage.

The agile sprint stage will follow this, and will last the majority of the semester, as feature creation will be the vast majority of the next semester. Each sprint will be 2 weeks long, in which members will present their feature at the end of each sprint. Weekly meetings will also occur for members to communicate about current status and to ask questions to other members.

Towards the end of the semester, we will switch away from the sprint style for testing, in which members will do various testing of the Flutr website in order to fix various bugs. As the plan for the website is to be deployed on November 25th, this will give us a few extra weeks to respond to large usage and change the system as needed.

Throughout the whole project, we will actively communicate with the client in order to keep the website focused to their own liking. This should help prevent a large change being needed towards the end of the project, as the team won't stray away from the client's needs.

Professional Responsibility

Areas of Responsibility

Professional Responsibility	IEEE Code of Ethics	Addressing Responsibility	Difference from NSPE version
Work Competence	I.6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations	Emphasizes the importance of being competent in work and not taking on tasks beyond expertise.	The NSPE Code of Ethics also emphasizes competence but includes a broader scope of responsibilities, including the need to "perform services only in areas of their competence." (I.2)
Financial Responsibility	1.4 ... to reject bribery in all its forms I.5 ..., to be honest and realistic in stating claims or estimates based on available data,...	Promotes honesty and realism in financial matters, ensuring truthful claims and estimates.	The NSPE Code includes a similar principle but also specifically addresses financial matters related to bidding and competition, requiring engineers to "be truthful and objective in reports, statements, and testimony." (II.3.a)
Communication Honesty	I.5 to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, ..., and to credit properly the contributions of others	Encourages open and honest communication in technical work, including accepting criticism and giving proper credit.	The NSPE Code emphasizes similar principles of honesty in communication but also highlights the importance of expressing opinions opinions “founded upon knowledge of

			the facts and competence in the subject matter." (II.3.b)
Health, Safety, and Wellbeing	I.1 to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	Prioritizes public safety, health, and welfare, emphasizing sustainable development practices.	In terms of health, safety, and wellbeing, both codes share a similar ethical stance, with the NSPE code providing slightly more detail regarding environmental concerns.
Property Ownership	I.3 to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist	Stresses avoiding conflicts of interest and disclosing them when present in property-related matters.	The NSPE Code covers similar ground but also includes specific provisions regarding proprietary information, confidentiality, and conflicts of interest. (III.5)
Sustainability	I.1 to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might	Emphasizes sustainable development in professional duties.	The main difference in the context of sustainability between the IEEE and NSPE codes lies in the specificity of language.

	endanger the public or the environment		
Social Responsibility	I.1 to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	Underscores engineers' responsibility towards societal welfare and sustainable development.	The NSPE Code similarly stresses societal welfare but includes additional provisions about "being guided in all their relations by the highest standards of honesty and integrity" (III.1), which sets a broader ethical standard for engineers' conduct in social contexts.

Table 7.1: Professional Responsibility

Project Specific Professional Responsibility Areas

Work Competence - This applies to our project. We have each been clear and honest about what our abilities are and what we can bring to the project in terms of our experiences and our expertise. We have assigned roles and responsibilities in the team to make the best use of our skills. We are performing well in this area.

Financial Responsibility - This certainly applies to our project. There are two areas of the project that are particularly relevant. Firstly, the database and web hosting must be done responsibly so that only the hosting that is needed is paid for, and no unnecessary charges are made due to mistakes such as leaving unnecessary servers running, causing costs to increase dramatically over time. Secondly, the physical kiosk has a specific cost, and having a proper budget is important. Currently, we are performing well in this area.

Communication Honesty - This applies to many aspects of our project. Communication between team members and communication with our client and advisor is important to maintain steady progress and goodwill between members. We are currently communicating well.

Health, Safety, and Wellbeing - This area currently seems to be less critical in our project. The database accessed by the entomologists and website accessed by both the staff and public, do not pose any risks to health and safety. The project does not change how any person interacts with the butterflies physically.

Property Ownership - This area is relevant to our project. Our client wishes to be able to access the data entered into the system by all organizations that will use it. This is for the purpose of aggregation and comparison of data and statistical analysis. There should be no other sensitive information, however. We are performing well in this area.

Sustainability - This is not significant in our project. The only operating costs of the project is the energy it takes to host a small website and a database, and the costs of the parts for the kiosk at Reiman Gardens specifically.

Most Applicable Professional Responsibility Area

Social Responsibility - This is very relevant to the project and is perhaps the most significant. Our project directly deals with the scientists at Reiman Gardens and other butterfly pavilions, as well as the public who will be interacting with our project through the public website. Because of the importance of this area in our project, we are actively working with our client and plan to interact with other users to make the right choices in this area. Currently, we are performing well in this area.

Closing Material

Discussion

Through Flutr, we intend to standardize and improve the butterfly management system for our client, Reiman Gardens, as well as for partner flight houses across the world. The goals we expect to achieve are to create a robust, scalable, user-friendly, and efficient platform to track, log, and manage butterfly releases, shipments, and employee data. Flutr will also implement a visitor view of the flight house, providing educational content through a visually appealing UI. Our project aims to meet the requirements of our client through the completion of this highly customizable and efficient application, along with the installation of kiosk hardware for Reiman Gardens.

Conclusion

Over the course of this semester, our team has built test applications for the backend and frontend, as well as built a foundational database architecture to be integrated into the system. We have performed extensive research into components for the kiosk that considers the unique conditions of the flight house to ensure reliable, environmentally conscious, and long-lasting operation.

We have also performed research into cloud hosting, database architecture, integration development, and the ethical responsibilities and standards we must adhere to as engineers. By compiling all our research and findings into comprehensive documentation, we aim to secure a high degree of client satisfaction and work to meet requirements based on in-depth deliberation and constant communication.

References

IEEE, “IEEE Code of Ethics,” *ieee.org*, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html

National Society Of Professional Engineers, “NSPE Code of Ethics for Engineers,” *National Society of Professional Engineers*, Jul. 2019. https://www.nspe.org/resources/ethics/code-ethics

Team

Team Members & Management Roles

Amanda Friis - Full Stack Developer, Documentation

Tayler Barnhart - Team Leader, Cloud Integration

Nathan Geater - Full Stack Developer (Back End)

Alex Brown - Front End Development Leader

Muralikrishna Patibandla - Integration & Back End Development Lead

Required Skill Sets For Your Project

Amanda Friis - React, Springboot, MongoDB, backend design

Tayler Barnhart - Cloud implementation, frontend design, Kiosk hardware creation

Nathan Geater - HTML, CSS, JS, backend design

Alex Brown - HTML, CSS, JS, React, Full Stack Development & Interaction

Muralikrishna Patibandla - SpringBoot, MongoDB, Postman, React Integration

Skill Sets Covered By The Team

Amanda Friis - UI/UX development, documentation, backend development

Tayler Barnhart - Cloud design, frontend UI creation, general computer hardware design

Nathan Geater - Backend development, testing, documentation

Alex Brown - UI Development, Software System Architecture, General System Architecture

Muralikrishna Patibandla - Backend Development, Database Architecture, Testing

Project Management Style Adopted By The Team

For our development throughout the project, we will be using an agile system, through 2 week sprints. The team will split into different groups to tackle features during each sprint, with each member attempting to create features as the sprints progress. Additionally,

members will log their hours spent each week on the project, and bi-weekly check-ins will be used to keep the project on track.

Team Contract

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
   Thursdays, 3:30pm Parks Library/TBD
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
   Face-to-face, Discord
3. Decision-making policy (e.g., consensus, majority vote):
   Consensus
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
   Discord overview

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
   Let the team know in advance if there is a schedule conflict
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
   Split evenly, share problems, ask questions
3. Expected level of communication with other team members:
   High
4. Expected level of commitment to team decisions and tasks:
   Do the work assigned to you, ask for help when needed, help others if unassigned work

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
   We will decide as we figure out team members' strengths.
2. Strategies for supporting and guiding the work of all team members:
   Communicating about our own and each other's work, keep ticket board updated on git
3. Strategies for recognizing the contributions of all team members:
   Track git ticket board, track hours

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
   Tayler - frontend development, cloud platforms, hardware design
   Alex Brown - Front End, Hardware, Would like to be a little backend involved as well.
   Amanda Friis- Frontend and backend development, CAD, CNC
   Nathan Geater - Frontend and backend development
   Murali - Backend development and frontend integration

2. Strategies for encouraging and supporting contributions and ideas from all team members:
   Open communication between members, allowing for differing ideas and being open to others opinions.
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

Talking at meetings and bringing it up to other members to talk about together.

Being open to communicating about problems and keeping each other accountable for their work is key in our success.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
   By the end of the semester we would like to be finished with the majority of planning and hopefully already working on the project.
2. Strategies for planning and assigning individual and team work:
   Creating a sprint plan on Gitlab(?) that we can all assign work for each person in order to track work.
3. Strategies for keeping on task:
   Setting specific goals for each meeting/sprint that each of us follow.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
   Members will be held responsible knowing that team members can report to the course instructors if members do not follow the rules we have set forth
2. What will your team do if the infractions continue?
   Be honest about a team member's performance when it comes to peer reviews.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _Tayler Barnhart_____ DATE __01/30/2024_____

2) _Amanda Friis_____ DATE _____01/30/2024

3) _Nathan Geater_____ DATE __01/30/2024_____

4) _Muralikrishna Patibandla_____ DATE _01/30/2024_____

5) _Alex Brown_____ DATE _01/30/2024_____