DOCUMENTATION - FLUTR
DESIGN

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 Developer

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.

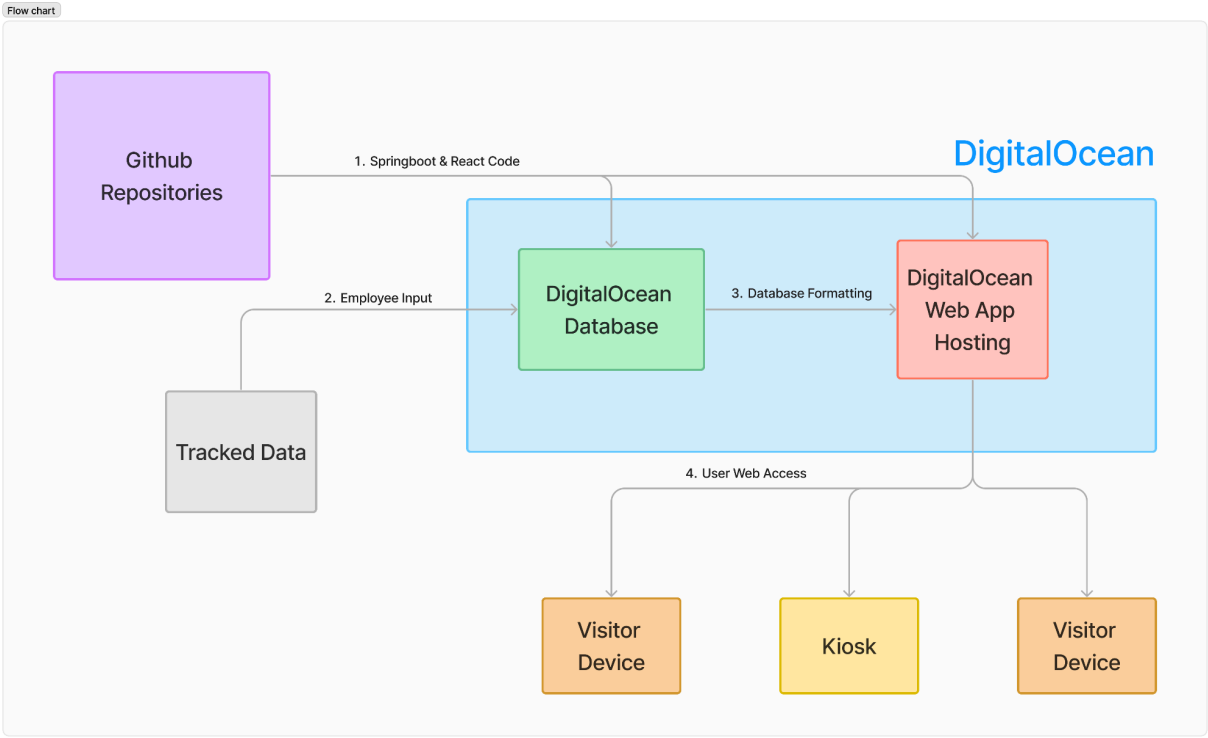


Figure 2.2: Hosting Diagram

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

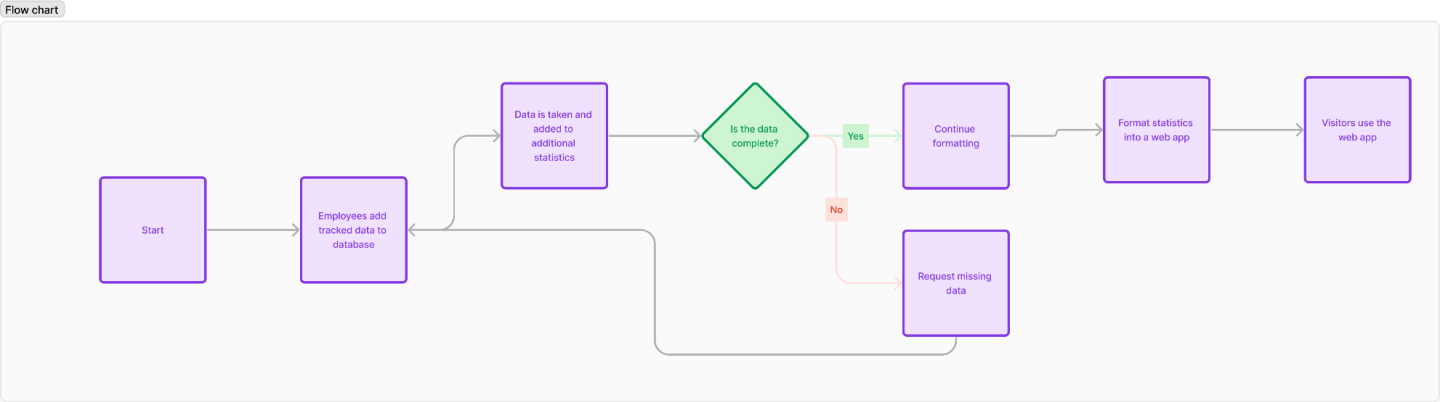
As this project has 2 main users, the functionality for each user is different.

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 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.