Global Butterfly Longevity Tracker

Design Document

sdmay25-03

Client- Nathan Brockman

Advisor- Maruf Ahamed

Team Members/Roles- Alex Herting(Frontend Developer), Andrew Ahrenkiel(Full Stack Developer), Carter Awbrey(Backend Lead), Charles Dougherty(Frontend Developer), Jaret Van Zee (Backend Developer)

Team Email- sdmay25-03@iastate.edu

Team Website- https://sdmay25-03.sd.ece.iastate.edu/

Date: 12-04-2024

Version 1.0

Executive Summary

The Reiman Gardens and other facilities with butterfly enclosures need to be able to quickly and reliably store and access butterfly tagging data electronically. It is important that this data is readily available to facility managers because it can help them optimize their enclosures to maximize the butterfly's life spans. We are creating a web application that allows guests and enclosure docents to be able to enter butterfly sightings while they are at an enclosure then the facility admins will be able to generate meaningful reports(life-span, number of sightings, time since last sighting, etc.) based on all of the data collected from the sightings. Our application will be able to adapt to various butterfly exhibits and will allow each exhibit to use their own unique tagging system.

For our frontend, we are using HTML./CSS/JS and will not be using a framework in order to maximize performance; for the backend, we will be using Spring framework, our database, we will be using MongoDB, and our server will be hosted on AWS. We currently have a minimal prototype that allows guests to enter butterfly sightings. This simple prototype will allow us to get feedback from our client so we can improve upon our current iteration of the software. Our web application is readily available to use on devices of all sizes, from phones to desktop computers. We have ensured that our pages can adapt and still be presentable under all screen sizes and in multiple browsers. Our next steps will be to implement admin users and allow for multiple facilities within the same web application, each with its own data. We are also looking into security features in order to keep our users and admins safe from any cyber threats.

Learning Summary

Development Standards & Practices Used

- ISO 639 - Language Code
- ISO/IEC 27001 - Information Security Management Systems
- ISO/IEC 25059 - Systems and Software Quality Requirements and Evaluation
- ISO/IEC 9797 - Message Authentication Codes (MACs)
- ISO/IEC 19772 - Authenticated encryption

Summary of Requirements

- Functional Requirements
    - User Authentication
    - Support User Group Hierarchy
    - Create Reports from Queried Data
    - Quick Response and Querying Times
    - Portability
- Resource Requirements
    - MongoDB Database
    - AWS Hosting Services
    - Cost Efficient Upkeep
- Aesthetic Requirements
    - Color Scheme
    - Typography
    - Images
- User Experiential Requirements
    - Ease of Navigation
    - Accessibility
- Database Requirements
    - Data Integrity
    - Scalability
    - Performance
    - Design

Applicable Courses from Iowa State University Curriculum

- SE 309
- SE 319

- Com S 363
- SE 185
- SE 186X
- CprE 230
- CprE 231
- SE 317
- SE 421
- COM S 252
- COM S 352
- Engl 314
- SpCm 212
- Com S 227
- Com S 228

New Skills/Knowledge acquired that was not taught in courses

- Hosting a web application on AWS
- Exporting a Figma board to HTML/CSS
- Expanded knowledge of Java Spring
- Json web tokens
- Expanded knowledge of MongoDB

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1. Introduction

1.1. Problem Statement

The Reiman Gardens and other facilities with butterfly enclosures need to be able to quickly and reliably store and access butterfly tagging data electronically. Butterflies within enclosures are currently being tagged; each facility needs a standardized way to enter and store butterfly sightings relative to their facility's tagging system. Facilities need a web application that allows guests and volunteers to quickly enter a butterfly tag sighting, as well as gives facility administrators a fast and reliable way to view database information. Reporting data is essential for facilities to make educated decisions on enclosure environments and derive conclusions based on certain butterfly species' lifespans.

A web application is in development to address the needs of any facility with a tagged butterfly enclosure. The web application needs to be easy to use for any user and easily accessible from any device to provide the highest level of sighting entries from any type of user. The web application will then be used to report on all sightings for administration purposes dynamically.

1.2. Intended Users

There are three groups of intended users for our application, each of which has different roles and privileges within the application. Each of those users is listed below.

Butterfly Enclosure Guests: A guest is a person who is visiting the butterfly enclosure for educational or entertainment purposes. A guest visitor could be a person of any age and background. Guests need an easy to use web application that they can enter a butterfly sighting in while visiting a butterfly enclosure. Since guests will be using a mobile device to enter butterfly sightings, they need a web application that can be used on any type of device. Guests also need a rewarding or valuable experience so they have the intent and reason to use the web app.

Guest users should be able to derive a sense of help or entertainment from using the web app while visiting the enclosure. We need to market the web application to guests so they can understand how they are helping the enclosure facility on a high level.

Docent / Volunteers: A docent or volunteer is a facility member who regularly spends time within the butterfly enclosure. A docent can also be a person of any age and background. Since a docent will use the web app much more frequently, they need a fast and easy way to continuously enter sightings. Docents will also benefit from an individual login so their sightings can be entered with more reputability than a guest visitor. Since docents are more reputable and garner more priority, docent accounts may also need to be configured with higher levels of privilege to database information based on administrative needs.

Docent users should derive a sense of help for the facilities enclosure more than just a guest visitor. Docents will be able to see the impact of the entered sightings and have a higher priority and sense of belonging within the web app.

Administrators: Administrators will be the person or people in charge of the butterfly enclosure at their given facility. An administrator will be a person with a higher level of knowledge of butterfly species as well as their given enclosure and tagging system. Facility administrators will need to

configure the web app for their facility and its tagging system as well as configure user accounts for docent and guest users. Administrators also will need full access to the database of butterfly sightings and need to be able to report on the data dynamically to benefit the facility.

Administrators will be the primary users of gathered and stored information as they can make more educated decisions for the future of their butterfly enclosures. Administrators can also be providers of data for other facilities and researchers to become more educated on environment variables and year-round habitat conditions for different butterfly species.

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

Functional Requirements

1. **User Authentication**
    a. The website must allow administrators to create accounts, log in, and log out securely
        i. Implement password recovery and reset functionality
        ii. Users should be able to register using an email address and password
2. **Support User Group Hierarchy**
    a. The website should handle different user types according to their permissions
        i. **Guest user**- needs to be able to submit the butterflies that they see during their visit within the web application because it updates butterfly information
        ii. **Docent**- needs to be able to easily insert butterfly sightings while maintaining a high level of credibility through secure credentials
        iii. **Exhibit Admin** - Needs to be able to view data from their site because they want to be able to use the data from their site for research
        iv. **Super Admin** - Needs to be able to view all data because they want to be able to use the data for research

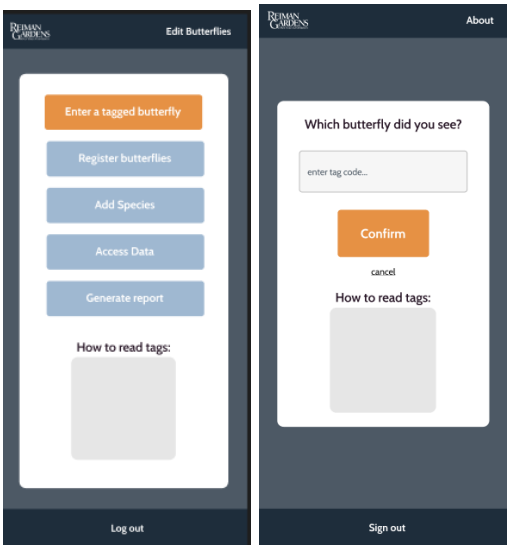

Figure 2.1 and 2.2 **Images display contrast between admin view (left) vs guest user view (right)**

3. **Create Reports from Queried Data**
    a. The website must be able to properly calculate statistics based on the data, and the admins request
        i. Must be able to calculate the average lifespan of specific butterfly types and all butterflies from one exhibit
        ii. Must be able to calculate the time since the last sighting of an individual butterfly
        iii. Must be able to calculate the number of sightings made on a single butterfly
        iv. Must be able to calculate the total number of butterflies from a specific species that are currently in the exhibit.
4. **Quick Response and Querying times**
    a. The website must be able to respond quickly to user requests, as well as provide data in a timely manner to admins.
        i. Page load time must be under 2 seconds **(constraint)**
        ii. Interactive elements will respond within 100 milliseconds from click time **(constraint)**
        iii. Fetching data should be performed in under 2 seconds **(constraint)**
        iv. Generating reports must be performed in under 3 seconds **(constraint)**
5. **Portability**
    a. The website must be able to adapt and perform on all devices and web browsers **(constraint)**

Resource Requirements

1. **MongoDB Database**
    a. The system will use MongoDB as its database in order to efficiently query, store, and retrieve data
2. **AWS Hosting Services**
    a. The system will use AWS for hosting the website to ensure reliability and security for users
3. **Cost Efficient Upkeep**
    a. The website should run for no more than $500 per month while maintaining a high level of performance **(constraint)**

Aesthetic Requirements

1. **Color Scheme**
    a. The website will use a consistent color palette throughout the website **(constraint)**
    b. The website will have sufficient contrast between text and background for readability **(constraint)**

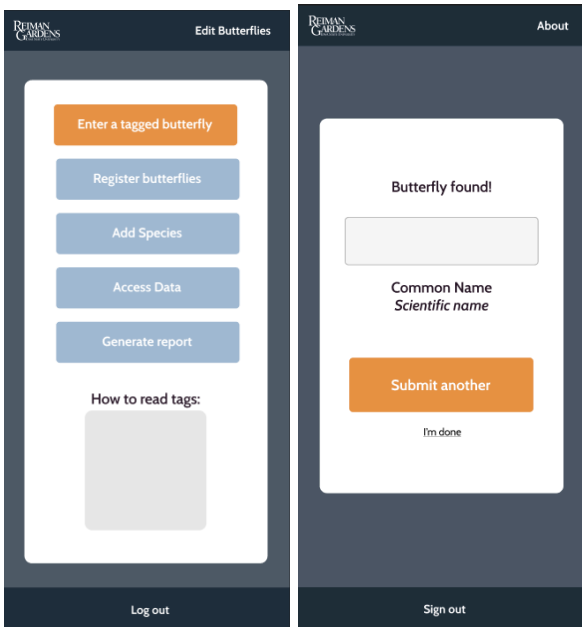

Figure 2.3 and 2.4 **Images from our website design that display color theme characteristics**

2. **Typography**
    a. The website will use web-safe fonts that are easy to read **(constraint)**
    b. The website must have adequate line spacing and margins for readability **(constraint)**
3. **Images**
    a. All images used in the website will be high-resolution images and graphics that are related to the content **(constraint)**

User Experiential Requirements

1. **Ease of Navigation**
    a. The website's layout should be intuitive and have a clear, logical structure, making it easy to navigate **(constraint)**
    b. The website will include navigation elements to assist the user **(constraint)**
2. **Accessibility**
    a. The website will provide alt text for images **(constraint)**
    b. The website will be available to use on all mobile devices **(constraint)**
    c. The website can be easily accessed by anyone with internet access **(constraint)**

Database Requirements

3. **Data Integrity**
3.1. Implement validation rules to ensure data accuracy and reliability
3.2. Ensure data remains consistent throughout with the use of constraints **(constraint)**
4. **Scalability**
4.1. The database must be able to upkeep performance as the amount of data grows **(constraint)**
4.2. The database will be able to scale and handle different exhibits being added to the system
5. **Performance**
5.1. The database must be able to complete queries in under 500 milliseconds **(constraint)**
5.2. The database must be able to complete GET, POST, PUT, and DELETE operations in under 500 milliseconds **(constraint)**

6. Design

6.1. The database must be designed in a way that will connect all collections efficiently in order to maximize performance

2.2 Engineering Standards

Engineering standards are crucial in everyday life because they ensure consistency, safety, and quality in countless different technology areas and products. These guidelines allow products from many different manufacturers to work together without needing to put in extra effort for each individual product. This promotes more innovation as it offers a clear framework for companies to aim for. This also reduces duplication of effort and ensures that technologies are both safe and efficient. Standards that are aimed towards safety are also of utmost importance. When engineering equipment is used in ways that could either aid or hurt a human, it is important to ensure safety in all systems. Consistency also allows for safety, ensuring random anomalies stay out of everyday situations and systems.

Here are three standards we have identified through research. We include how we define the standards as well as why they are important and applicable to our project.

- ISO 639 - Language Code - This standard refers to using language codes rather than the name of the language for identifiers. Language codes with two or three letters have many benefits, including being more identifiable to native speakers based on culture and some languages with similar names.

- This code was chosen because our team had not thought about the opportunity of non-english speakers utilizing the software and web app. Although the standard does not directly apply to an application offering multiple languages, it is something for our team to think about, consider, and incorporate to better the guest user experience.

- ISO/IEC 27001 - Information Security Management Systems - This standard defines that any company or application must put systems in place to mitigate risk related to the security of data owned or handled. This is important because it ensures that the management system is continuously being managed as new threats become apparent as time goes on.
  This code was chosen as we want to focus on security in our project especially for the user data and facility data that will be held. Since this project will be advertised to other facilities for their own use, it is very important to give them a safe and secure application for all users. they will be using this product as their own, and, therefore will need a secure application.
- ISO/IEC 25059 - Systems and Software Quality Requirements and Evaluation - This standard defines how software products and systems are crucial to stakeholders and users. Most notably, it mentions how AI is being used to replace human decision-making, and be based on incomplete data, which leads to a lesser quality of product. Defining the quality of a deliverable product ensures that it is of high standard and will not be dramatically vulnerable in practice.
  This code was chosen as it gives a general definition of quality for a software product. It also defines why high-quality software is important to a client or a user. Notably, the use of AI could lead to a lesser quality product or could be more vulnerable when used in practice.

> Ensuring we follow quality guidelines and do not replace human decision-making within the product's design is highly important.

Each team member chose a set of standards while researching.

All the standards chosen are different, as there are standards for many different topics and categories. Notably, the ISO/IEC 9797 gives security standards based on Multi-factor Authentication, specifically Message Authentication Codes, and using block ciphers and dedicated hash-functions. This is an extremely important and in-depth set of standards specifically focused on a single version of multifactor authentication. This standard was not chosen for the above as we feel our web app will not need a multi-factor authentication system implemented; however, this standard is something each of us is exposed to almost daily. Another chosen standard was ISO/IEC 19772, which is about authenticated encryption and information security. This is similar to the information security management system standard we listed above but is much more specific. Particularly, this standard emphasizes the quality of encryption and the need for encryption when handling user data and sending it from one place to another. This could very well be important to our project for encrypting user authentication and facility authentication to limit its vulnerabilities against attackers who may be analyzing network traffic.

Potential design modifications to incorporate the standards mentioned above:

First, we already have an emphasis on the importance of security for the application. The most notable change is to recognize that the application will be advertised to and used by other facilities. In these cases, other facilities will be treating the application as their own property, meaning that any vulnerability could mean a vulnerability to the facility itself. This means there needs to be a strong emphasis on the vulnerability scanning of the application to mitigate risk in the future. We may also need to look into more longevity for security in the case that the application is being used a few years down the road. This goes along with understanding the standard for authentication encryption and ensuring that we properly handle users' personal information. Language barriers are also something that we had not previously thought of for project requirements, which could vastly change the design of the application. Sitting down with the client and understanding the value a bi-lingual application may bring is an important step towards creating a better deliverable. In the case we do decide to add additional languages to the application, we can follow the above standard for language codes. Lastly, we need to shift our development process to incorporate quality standards and understand the risk of using AI during development. As long as all team members understand the risk to the project quality that comes with using AI during development, we can still keep human decision-making and high-quality for the deliverables.

3 Project Plan

3.1 Project Management/Tracking Procedures

We will be using Agile as our project management style for this project. The primary reasons that we decided on an agile approach were because of the flexibility, incremental delivery, continuous improvement, and collaboration aspects. Having independent tasks within our backlog is super important in software development. We do not want to have dependent tasks that could be delayed because of unexpected issues arising during development. We value the incremental delivery aspect of agile because we plan on producing prototypes for our client before providing him with the full version of the product. This ties into the continuous improvement aspect because as we produce prototypes, we will improve our product based on client feedback. The collaboration aspect of agile is excellent for software development. We have toned back the daily standup meetings to once a week to check in with one another and discuss the work we have done and plan to do.

We plan to track project progress using Git and GitLab. We have a GitLab repository set up with an issue board and milestones that are dated with deadlines. Inside GitLab we have also set up a scheme for our branch setup to keep our code organized, allowing for easier project progression due to simplicity. Git is an amazing tool for software development because of its version control, which is the main reason we decided to use it. It allows us all to collaborate on our repository simultaneously and promotes good coding habits.

3.2 Task Decomposition

Frontend

- Core HTML Development
    - Converting Figma boards to functional HTML
    - Multi-Page navigation
    - Facility management pages
        - Adaptable page color theming
    - User log-in pages
    - Mobile device optimization
- Backend Interactions
    - User sign-in implementation and authentication
    - Butterfly tagging support
    - Graphical data views
    - Butterfly data filtering and sorting
    - Unique butterfly tagging for each facility

Backend

- Database collection and Layout
    - Create a database management scheme
    - Create database objects
    - Map backend to database
    - Containerization
    - Design backend API
    - User sign-ins and permissions database
        - Secure authentication process
        - Quick and easy sign-in for repeat users

    - Data Querying
- Database butterfly storage
    - Database butterfly reports
    - Permanent Server hosting solution
    - Multi-facility tagging support

User Testing

- Test the website with those who will be interacting with the website
    - Test website with facility operators
    - Test website with guests and general public

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Frontend

- UI Design Completion
    - Metric: Percentage of originally requested designs successfully converted to functional HTML.
    - Milestone: Complete the conversion of 100% of requested views into functional HTML components.
- Functionality Development
    - Metric: Percentage of interactive elements from the design that function as intended.
    - Milestone: Ensure all interactive elements are fully functional and have corresponding tests written.
- Responsiveness
    - Metric: Whether a view is able to adapt to different screen sizes and aspect ratios while adequately displaying content.
    - Milestone: Ensure all implemented views adapt effectively to various screen sizes, including Desktop, Phones, and Tablets.
- Accessibility
    - Metric: Number of Web Content Accessibility Guidelines (WCAG) criteria met, as defined by W3C.
    - Milestone: Achieve an 'AA' level of accessibility by implementing the required WCAG guidelines.
- Compatibility
    - Metric: Compatibility tests pass rate across target browsers.
    - Milestone: Ensure our designs achieve a 100% pass rate in compatibility tests across major browsers, including Chrome, Firefox, and Safari.
- Client Acceptance
    - Metric: Level of client satisfaction with the design as measured through feedback.
    - Milestone: Achieve full client satisfaction with all designs, with no further changes requested after review.

Backend

- API Development and Integration
    - Metric: Percentage of API endpoints developed, tested, and documented.
    - Milestone: Implement 100% of API endpoints outlined in the project requirements, with thorough integration tests for each endpoint.
- Automated Testing Coverage

    - Metric: Percentage of backend code and branches covered by unit and integration tests.
    - Milestone: Achieve 100% code coverage and 100% branch coverage across systems to ensure comprehensive testing and reduce the likelihood of bugs in critical areas of the backend.
- Database Performance and Optimization
    - Metric: Average database query response time for data visualization features.
    - Milestone: Achieve a response time that is at least 50% faster than the previous design while maintaining equivalent end-user functionality.
- Tagging Adaptability
    - Metric: Capability to integrate specific butterfly tagging systems into the database.
    - Milestone: Successfully incorporate all widely used butterfly tagging systems adopted by major institutions.
- Security Compliance
    - Metric: Number of security vulnerabilities identified and remediated (tracked via penetration testing or security audits).
    - Milestone: Resolve 100% of critical vulnerabilities, aiming to avoid common security risks outlined by OWASP guidelines. Ensure that, to the best of our abilities, protections are in place against vulnerabilities such as broken access control, injection, cryptographic failures, and others.
- Error Handling and Uptime
    - Metric: Number of errors that impact user experience or server uptime.
    - Milestone: Implement logging and monitoring systems to ensure that the number of critical errors impacting user experience or server uptime remains at a minimum.
- Data Safety and Recovery
    - Metric: Risk of data loss in the event of a system failure.
    - Milestone: Implement robust data backup procedures to ensure that no critical data is lost during a system failure. Validate backups and confirm the ability to recover data in the event of a failure.
- User Data Management and Compliance
    - Metric**:** Compliance with data protection standards as outlined in the GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act).
    - Milestone: Achieve compliance with data privacy and protection standards, ensuring that all user data is encrypted and anonymized where applicable.

3.4 Project Timeline/Schedule

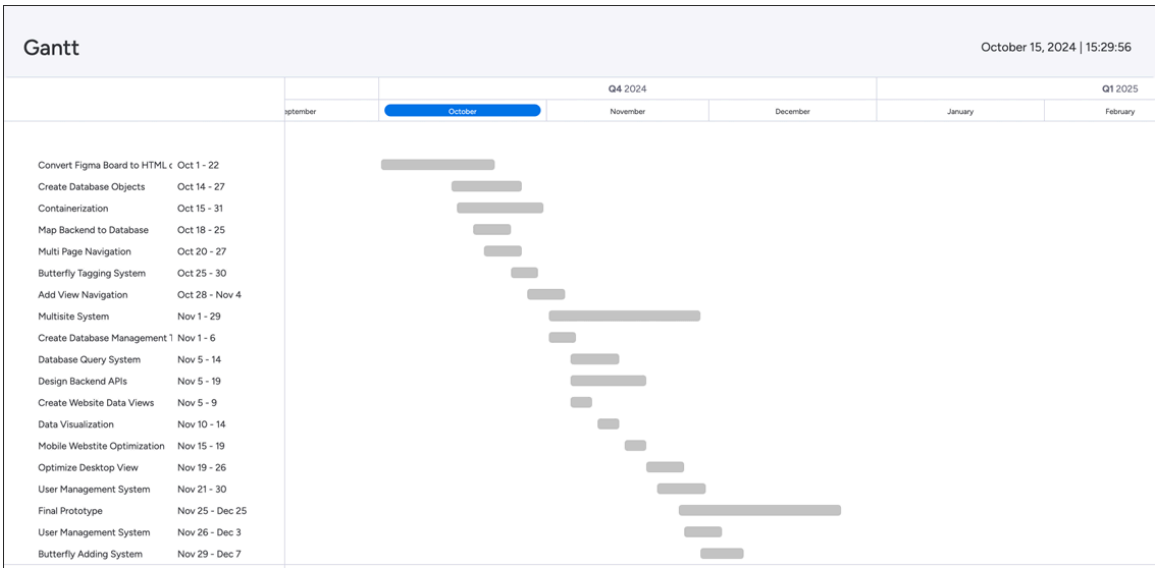

Figure 3.1

Gantt Chart Tasks:

- Convert Figma Board to HTML
  - Export figma board to code using PxCode
  - Correct responsiveness and design of the exported screens
- Create a Database Management Scheme
  - Brainstorm ideas for database collections and layout
  - Create a structure that will work best for our data types
- Create Database Objects
  - Create database collections
  - Define object parameters
- Containerization
  - Create a container for the backend
  - Create a container for frontend
- Map backend to database
  - Structure backend
  - Define object parameters within the backend code
  - Create and test requests to the database
- Multi-Page Navigation
  - Connect HTML screens via button clicks
  - Allow for user interaction in the UI
- Butterfly Tagging System
  - Allow user input of butterfly tags
  - Create requests to the backend for entering sightings
- Database Query System
  - Create efficient code for querying data
- Design Backend API
  - Plan and define endpoints
  - Implement endpoint calls
- Create Website Data Views
  - Implement admin interaction to create data reports

    - Implement calling to backend to gather correct data
    - Display the queried data
- First prototype
    - Deliver a baseline product that our client can test
    - Receive feedback from client
    - Change functionality based on client interactions
    - Repeat this process and iteratively improve our product
- Adding other butterfly systems
    - Implement ability for product to scale to other facilities
    - Allow for multiple admin users
    - Create butterfly tagging systems unique to each exhibit

Deliverables:

Week 4: Initial Concept and Design Review

Week 8: Present responsive screens to client

Week 15: Initial prototype with minimal functionality

Week 20: Second prototype with improvements based on client feedback

Week 24: Third prototype with improvements based on client feedback

Week 26: Finalized product

3.5 Risks and Risk Management/Mitigation

Tasks and Their Associated Risks

- Convert Figma Board to HTML
    - **Risk 1:** Views may not easily convert from Figma to HTML.
        - *Probability:* 100%
        - Mitigation Strategy: Begin conversion early in the project to allow sufficient time and resources for completion.
    - **Risk 2:** Views may not be fully responsive or accessible as planned.
        - *Probability:* 100%
        - *Mitigation Strategy:* Focus on achieving WCAG AA accessibility standards and conduct regular checks to ensure designs are adaptable to target screen sizes.
- Create Database Objects
    - **Risk:** Initial database objects may lack sufficient fields or functionality to meet project needs.
        - *Probability:* 45%
        - *Mitigation Strategy:* build out a thorough list of all data points that need to be stored and get each of those data points with their constraints to be officially signed off by our client.
- Containerization
    - **Risk:** Containerizing the app may be more complex than expected.
        - *Probability:* 50%
        - *Mitigation Strategy:* Conduct early research to verify the compatibility of components and ensure they integrate smoothly into a container.
- Database Query System / Map backend to database
    - **Risk:** Database performance may not meet client requirements.
        - *Probability:* 80%

        - *Mitigation Strategy:* Optimize the database structure and create indexes for high-demand queries. Track other potential optimizations during database creation to maintain performance.
- Multi-Page Navigation
    - **Risk:** Pages may not be easily navigable.
        - *Probability:* 60%
        - *Mitigation Strategy:* Test the navigation design with potential users to verify ease of use and page hierarchy effectiveness.
- Butterfly Tagging System
    - **Risk:** System may not support all common butterfly tagging methods used across sites.
        - *Probability:* 70%
        - *Mitigation Strategy:* Consult the client and potential site owners about their tagging methods and ensure the system can incorporate all identified methods.
- Design Backend API
    - **Risk 1:** Backend APIs may lack adequate security.
        - *Probability:* 50%
        - *Mitigation Strategy:* Stay aware of common security risks and implement safeguards, including minimizing stored user data and maintaining regular backups for data integrity.
    - **Risk 2:** Initial API list may not cover all required software functions.
        - *Probability:* 100%
        - *Mitigation Strategy:* Design APIs with extensibility in mind, allowing the team to add new functionality as needed.
- Create Website Data Views
    - **Risk:** System may struggle to display complex data views efficiently, potentially affecting performance and data accuracy.
        - *Probability:* 70%
        - *Mitigation Strategy:* Conduct performance testing for data-heavy views, consider pre-aggregating data to reduce load, and implement pagination for large datasets. Gather user feedback early to improve clarity and usability.
- First prototype
    - **Risk:** Initial prototype may not align with client expectations, resulting in delays due to rework.
        - *Probability:* 80%
        - *Mitigation Strategy:* Hold frequent, iterative feedback sessions with the client and conduct regular checkpoints to integrate feedback progressively, reducing the need for major adjustments.

3.6 Personnel Effort Requirements

Task	Subtask	Projected Effort(Person-hours)	Explanation
Core HTML Development	Converting Figma boards to functional HTML	20	Figma board to HTML conversions are clunky and require a lot of rework to function properly.
	Multi-Page navigation	10	Properly linking the web pages requires meticulous iterations and testing to ensure all links work.
	Facility management pages	20	The pages to update a facilities web page will be complex and require many different features to function.
	User log-in pages	5	The sign-in pages themselves shouldn't be too complicated on the HTML side.
	Mobile device optimization	10	Ensuring that mobile devices have the same experience can take a lot of optimization and rework.
Frontend-to-Backend Interactions	User sign-in implementation and authentication	15	Ensuring the sign-in is secure and cannot be exploited is a delicate process requiring time.
	Butterfly tagging support	15	Implementing tag posting and spotting will require complex interaction with the backend.
	Graphical data views	20	Creating useful and streamlined displays will require advanced methods we have not

Task	Subtask	Projected Effort(Person-hours)	Explanation
Core HTML Development	Converting Figma boards to functional HTML	20	Figma board to HTML conversions are clunky and require a lot of rework to function properly.
	Multi-Page navigation	10	Properly linking the web pages requires meticulous iterations and testing to ensure all links work.
	Facility management pages	20	The pages to update a facilities web page will be complex and require many different features to function.
	User log-in pages	5	The sign-in pages themselves shouldn't be too complicated on the HTML side.
	Mobile device optimization	10	Ensuring that mobile devices have the same experience can take a lot of optimization and rework.
			yet explored.
	Butterfly data filtering and sorting	15	Ensuring that the filters and sorting methods are efficient and effective will require deep analysis of the data and structure of the database.
	Unique butterfly tagging for each facility	15	Allowing facilities to utilize various tagging methods requires great consideration of the possible methods and how to allow for them.

Task	Subtask	Projected Effort(Person-hours)	Explanation
Core HTML Development	Converting Figma boards to functional HTML	20	Figma board to HTML conversions are clunky and require a lot of rework to function properly.
	Multi-Page navigation	10	Properly linking the web pages requires meticulous iterations and testing to ensure all links work.
	Facility management pages	20	The pages to update a facilities web page will be complex and require many different features to function.
	User log-in pages	5	The sign-in pages themselves shouldn't be too complicated on the HTML side.
	Mobile device optimization	10	Ensuring that mobile devices have the same experience can take a lot of optimization and rework.
Database collection and Layout	Create a Database Management Scheme	15	Creating a scheme that can effectively manage all of the data requires a lot of research and a deep understanding to meet the needs properly.
	Create database objects	15	The objects are essential to the structure of the database and may need to be reworked if not done correctly the first time.
	Containerization	15	Complex and requires proper

Task	Subtask	Projected Effort(Person-hours)	Explanation
Core HTML Development	Converting Figma boards to functional HTML	20	Figma board to HTML conversions are clunky and require a lot of rework to function properly.
	Multi-Page navigation	10	Properly linking the web pages requires meticulous iterations and testing to ensure all links work.
	Facility management pages	20	The pages to update a facilities web page will be complex and require many different features to function.
	User log-in pages	5	The sign-in pages themselves shouldn't be too complicated on the HTML side.
	Mobile device optimization	10	Ensuring that mobile devices have the same experience can take a lot of optimization and rework.
			implementation to improve performance.
	Map backend to database	20	Involves setting up complex database connections to the backend, which can cause efficiency issues if not properly mapped.
	Design backend API	20	Very important interfaces for the front end to interact with that must allow for scalability.
User Testing	Test website with facility operators	5	Distributing the website and gathering

Task	Subtask	Projected Effort(Person-hours)	Explanation
Core HTML Development	Converting Figma boards to functional HTML	20	Figma board to HTML conversions are clunky and require a lot of rework to function properly.
	Multi-Page navigation	10	Properly linking the web pages requires meticulous iterations and testing to ensure all links work.
	Facility management pages	20	The pages to update a facilities web page will be complex and require many different features to function.
	User log-in pages	5	The sign-in pages themselves shouldn't be too complicated on the HTML side.
	Mobile device optimization	10	Ensuring that mobile devices have the same experience can take a lot of optimization and rework.
			feedback from the employees of Reiman
	Test website with guests and general public	5	Gathering feedback from the public utilizing surveys

Figure 3.2

3.7 Other Resource Requirements

The main resource requirement for our project would be hosting the service for our client on AWS. This hosting service will be used to configure and control the status of the web application and make it available and easy to use for the client. Other external resources being used are MongoDB and Java Spring for the backend and database implementation. Java Spring is a free service that we are utilizing to secure CRUD requests between the front end and the database. Next would be the implementation of the database using MongoDB, which is a non-relational collection-based database. Here, we will store collections for each facility and track their specific tagged butterflies. Lastly, we are implementing Docker so our client can control all services from a single place. This includes the cloud hosting through AWS and the backend service implementation through a VM. Docker is a service offered by AWS, so integration will be quick and easy.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	Our project will directly impact our shareholders as they will be implementing it into their facilities everyday use. The project can improve the welfare directly by improving already existing processes and making everyday tasks more efficient.	Reducing time taken to enter new butterflies Increase accuracy of butterfly info Increase knowledge of butterfly needs
Global, cultural, and social	Our project aims to adapt to butterfly curators across multiple facilities and meet their needs. Multiple features of the website have been created specifically to meet the needs of those who are experienced in the field and provide them useful information.	Inaccurate data or information about butterflies could result in harmful changes to the butterflies treatment routines
Environmental	Our project can affect the population of butterflies by changing the way that different facilities care for them by providing detailed information.	Decrease number of butterflies in an atrium Increase the amount of food given to butterflies Increase lifespan of butterflies
Economic	Our project can reduce cost to our client by increasing efficiency in already existing processes and improving care methods for the butterflies.	Improved tagging efficiency leads to less wasted time for employees Better care mtheods can reduce costs from wasted steps

Figure 4.1

4.1.2 Prior Work/Solutions

Monarch Watch App - The closest existing application we could find to what we are currently developing. This app is used to track specifically monarch butterflies as they travel across the world. This is a paid service that allows people around the world to collaborate together to track the path of monarch butterflies as they migrate. For more about the app see [1].

Pros:

- Easy to use
- Provides accurate migration information
- Allows for multiple users to work together

Cons:

- Requires payment
- Does not provide detailed data
- Only used by one organization to track one species
- Built for long-distance use

Solar-powered radio tags - Solar-powered radio tags and RFID tag combinations are close to being applied to butterflies. Currently, a small group is testing this technology with butterflies in the wild. This would allow for the automation of detection and tracking of butterflies without manual entry from visible tags. For information about these tags, see [2].

Pros:

- Precise tracking ability
- Automated tracking
- Low weight of 0.06 grams

Cons:

- Not currently available on the market
- Expensive
- Still untested and could have issues

Small RFID Tags - Currently some of the smallest RFID tags could be small enough to use on butterflies. Unfortunately, these tags have very small range and high costs. “Smaller tags have a shorter read range since they cannot capture as much energy from a reader antenna.” For more information about the size and range of small RFID tags, see [3]

Pros:

- Automated tracking
- Small enough to be used on most butterfly species

Cons:

- Expensive
- Low range

Previous Project - A previous project to make a similar system was created a few years ago by a different senior design group from computer science. This project has functioned as a baseline but has many underlying issues that plague the site and need to be addressed. The client has requested an entirely new website be created to replace the old system.

Pros:

- Free
- Sticker method of tracking

Cons:

- Hard to navigate
- Long load times
- Missing polish
- Missing needed features

4.1.3 Technical Complexity

This type of application does not currently exist in the industry; facilities commonly use CSV files to store information that is entered manually. All reporting data is challenging to report on, and no cross-facility data is being shared or utilized. Our application addresses this need for any facility to have a standardized way of entering data and reporting on it with the additional advantage of comparing data across facilities.

The system has three components similar to any other software web application. Complexity grows when implementing a custom frontend for any number of facilities that utilize the application. This needs to be adaptable and provide growth for when we are no longer developing. Creating a way that each facility can have its own UI along with tagging logic increases the complexity of the project. Having all the data stored in one location provides an easier way to report on data from multiple facilities but calls for strong logic and reporting options for users due to tagging system implementations. Taking raw sighting data and displaying it dynamically to administrators and researchers to a point where they can derive conclusions is the end goal of the project.

This requires extensive frontend development experience to be able to create a quick and easy-to-use application that is also scalable for any facility. Providing adaptive new endpoints through quality dynamic code poses challenges on the backend, which will manage any number of facilities that use the application. Database expertise is required not only to design and implement a multi-facility database but also to query and display information in valuable ways. This project requires experience from an extensive list of software technologies and systems, whereas usually, a developer focuses specifically on a single expertise or technology.

4.2 Design Exploration

4.2.1 Design Decisions

Butterfly Longevity Project – Figma This board contains all UI designs and the connection between each screen. This is important because we need to ensure that the screens have all the necessary

functionality for the client's needs. We also need to ensure that they are in locations that would make sense to our clients and the users. This is arguably the most important design aspect of our project as it directly affects user experience in every way. You can not recover from a bad UI design. We have presented this figma board to our client and have gotten approval.

MongoDB Atlas: Cloud Document Database | MongoDB We have decided to use MongoDB as our database which is a very important decision to our web applications performance. This decision was important because it will directly impact how long it takes to load and query data. For our data types and the amount of data our database will store, we will benefit most from using MongoDB. MongoDB is very flexible and great at handling a dataset that will continue to grow without losing performance which is a main concern of ours.

Cloud Computing Services - Amazon Web Services (AWS) We decided to use AWS to host our web application which is very important to our applications availability. AWS offers high availability (99.999%) making it extremely reliable for our client and for the other users. AWS also allows for you to upgrade your plan at any time in order to account for a higher amount of traffic. This will allow our product to scale and have no performance issues. It is also very cost-effective in comparison to other options and is a pay-as-you-go price model, meaning we can cancel the service at any time.

4.2.2 Ideation

To decide on what database to use, we first looked at all of our data that would need to be stored within the database. Next we estimated which data sets would have exponential growth and which would remain similar in size as time goes on. We also listed performance and scalability as two of our most important features in the database. We chose those two traits specifically because our client's previous product had struggled with that and we were looking to improve on the previous design. Our 5 considerations and why we chose to use/chose to not use them are as follows:

MongoDB (Using):

Pros

- Flexible data schema
  - Allows for easy changes in data collections
- Allows for scalability of data size without a tradeoff in performance
- High-performance speeds
  - Fast querying speeds
  - Fast retrieval speeds
- Very little to no cost

Cons

- Does not support traditional SQL joins
  - Can lead to limitation in querying
- Potential redundant data
  - In turn increasing storage costs

MySQL (Not using):

Pros

- High performance speeds
    - Fast querying speeds
    - Fast retrieval speeds
- Built-in security features
- Supports vertical scalability

Cons

- Does not support horizontal scalability
    - i.e. adding traits to a data type
- Lack of schema flexibility
- Limited JSON support

Oracle (Not using):

Pros

- High performance speeds
    - Fast querying speeds
    - Fast retrieval speeds
- Multi-platform support
    - Available on Windows, Linux, Unix and more
- High scalability without performance tradeoffs

Cons

- High costs
- Resource intensive
    - Significant demand of CPU
- Built for enterprise applications

IBM DB2 (Not using):

Pros

- High performance speeds
    - Fast querying speeds
    - Fast retrieval speeds
- Supports XML and JSON formatted data types
- High scalability without performance tradeoffs

Cons

- High costs
- Complex setup and features in order to make the most of the application
- Low flexibility of data schemas

MariaDB (Not using):

Pros

- Open source and free to use
- Compatible with MySQL
- Flexible storage engines

Cons

- Lacks advanced features that can improve performance
- Low performance with large data sets
- Lacks professional support and there are not a lot of resources out there on the product

4.2.3 Decision-Making and Trade-Off

We created a weighted decision matrix to make a final decision on which database we would utilize for the project. We prioritized the performance and scalability of the platform first, as the previous project struggled with performance issues and could not handle a large amount of data. Flexibility and cost are also important factors because of the vast range of data that needs to be stored and the low budget needed to keep the project running.

Weighted Decision Matrix (Scores range 1-10)

Database	Performance (25%)	Scalability (25%)	Flexibility (20%)	Cost (20%)	Querying Support (10%)	Total Weighted Score
MongoDB	9	9	9	10	7	9
MySQL	8	6	5	20	8	7.3
Oracle	9	9	6	3	8	7.1
IBM DB2	9	9	4	4	8	6.9
MariaDB	7	5	6	20	7	6.9w

Figure 4.2

Based on the scores of the table, we decided MongoDB was the best fit for our project after receiving a score of 9. Some of the following factors affected our decision and how we scored the databases.

Scalability: Horizontal scalability in MongoDB is a large advantage when handling large amounts of data that will only grow over time. We expect the database could contain many years of butterfly data that could cause issues on other platforms.

Flexibility: MongoDB allows for a very flexible schema structure that suits our needs, allowing the database to adapt as the project evolves without requiring complete restructuring.

Cost: Since MongoDB is open-source, it offers a great cost advantage over most other options, such as Oracle and IBM DB2, which require licensing and resource costs.

Performance: MongoDB's fast querying and retrieval speeds address previous performance issues the previous group has experienced, making it an even stronger candidate for our use.

4.3 Proposed Design

4.3.1 Overview

We will follow a Model-View-Controller design pattern for our website. This means that there are three main components to the website. The first component is the view, which is the part of the website that a user will see and interact with. This part of the website must have a well-designed user interface and will need to take in user input safely. The second component is the model, which relates to all the data for the users, butterflies, and facilities that must be stored. We need to establish standards of what exact data will be stored, this will prevent any clashing of data or inconsistencies in the database. The final section controller, this part of the website, is responsible for moving any data from the database of information to the actual display on the website. The controller will need to input data and any data that the database provides in a secure and efficient manner.

4.3.2 Detailed Design and Visual(s)

Front End

The front-end implementation of the project is not using a framework such as React or Angular. Instead, it is written in pure HTML, CSS, and JS to avoid performance issues and high device compatibility. All UX design follows the Butterfly Longevity Project Figma board which can be found at the following link: Butterfly Longevity Project Figma.

PxCode is a tool used to generate HTML code from Figma boards. We utilized this to generate HTML code for each view that is presented on the Figma board. This leaves us with 14 generated views, so 14 generated HTML, CSS, and JS code files. A few views from the Figma board cannot be easily containerized and will be later implemented manually into the system. These views would include the database spreadsheet view and the admin home page setup view.

The code then needs to be optimized to resize correctly to mobile, tablet, and desktop views of the web app. For each view, this will be done quite differently since the containerization of objects in the view is quite different. The main point of reference is through CSS files and styling guides, ensuring that top-level containers rely on viewport height and width rather than a pixel amount of some other variable identifier. Furthermore, low-level containers will need custom styling from developers to ensure proper function for all screen sizes.

For a more in-depth example of how the app will work, watch the video demo at this link: https://youtube.com/shorts/4BnmfrQZxho

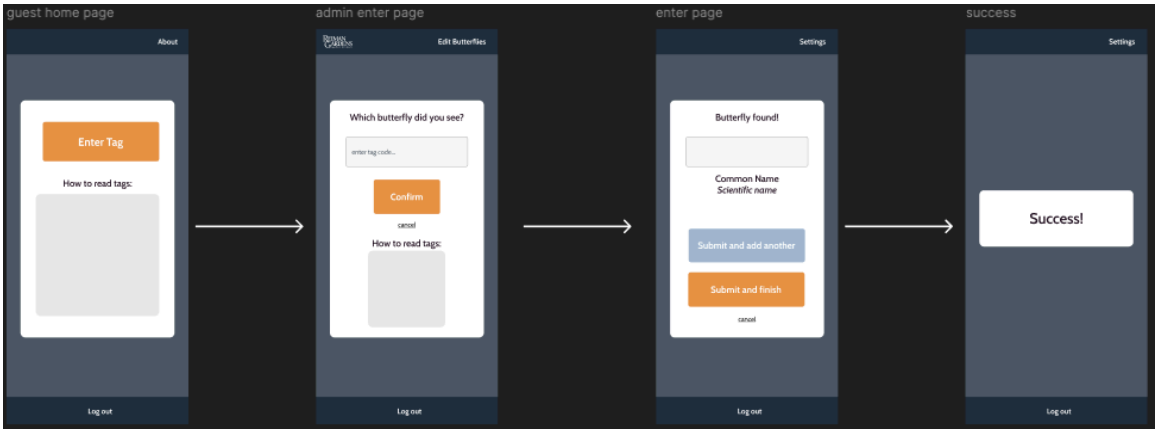

Figure 4.3 (Example of guest user experience)

Another notable Frontend performance feature to be implemented is storing reporting information on the client side. As of right now a problem with the reporting is when searching the table or adding filters to generated reports, a new server request is made with every change. instead we can have all this filtering be done on the client side to avoid server lag and improve the overall performance of the system.

Implementation of report views for mobile devices are yet to be created. If the client desires to keep the table report for mobile devices then reporting views will be optimized for mobile and tablet devices. Ensuring these are easy to understand and use is up to developer implementation and will be constantly updated and corrected for improvement.

Backend

As mentioned above, Java Spring is the utility service that is being used for the backend of the application. There are two main purposes of the Backend, communicate data from the backend to be displayed to the user on the Frontend, and take user data from the Frontend and store it in the database. This is done by following a Get, Post, Put, Delete HTTP connection API, which is readily available through the Spring service.

Controller classes within the backend will be created based on views. Currently, All the Login pages will have an associated controller, the Generating Report page will have an associated controller, and the Tagging pages will have an associated controller. The image below shows an example of a Get and Post mapping implemented for testing within the controller.

```
15      @GetMapping("/entities")  Andrew Ahrenkiel
16      public List<Butterfly> getAllEntities() { return butterflyRepository.findAll( type: "test"); }
19
20      @PostMapping("/add")  Andrew Ahrenkiel
21      public Butterfly addEntity(@RequestBody Butterfly butterfly){
22          butterflyRepository.save(butterfly);
23          return butterfly;
24      }
25  }
```

Figure 4.4 (Example test code)

Each of the following table entries will have its own implementation of a request mapping. All requests will use JSON for communication as that is the form the data will be stored in. Request code can be written in multiple ways and will be up to the developer's interpretation for implementation.

Get	Post	Put	Delete
Get Facility Theme	Login Request	UpdateFacility Theme	Delete Invalid Butterfly Sightings
Get Facility Assets (Logo, Tag Method, etc.)	Post Butterfly Sighting	Update Facility Assets	Delete User Accounts
Get Butterfly information for specific facility	Create Facilities and Facility Admin Accounts	Update Admin Account password or information	Delete Facility and Facility Information
Get all butterfly information with specified filters	Add a new Butterfly Species	Update a butterfly sightings information	Delete/Remove Butterfly Species

Figure 4.5

Requests will use JSON request bodies when needed rather than parameter variables to keep api url simple and easy to understand. Response bodies will also be given in JSON so data will not have to be altered after retrieving it from the database. Requests should be reused whenever possible to avoid duplicate requests in different controller classes.

Database

MongoDB is the chosen service for our database, with the reasoning shown in the above sections. I will describe the high-level database collection schema and how it will hold information for our users. It's notable to mention that MongoDB is a non-relational database meaning it does not utilize tables to hold information. Rather, MongoDB stores data in collections; within collections are objects that store data in a JSON format.

To maximize performance for each facility, we have chosen to create a collection for each facility that will be utilizing the web app. The following model resembles how the database collections will be organized.

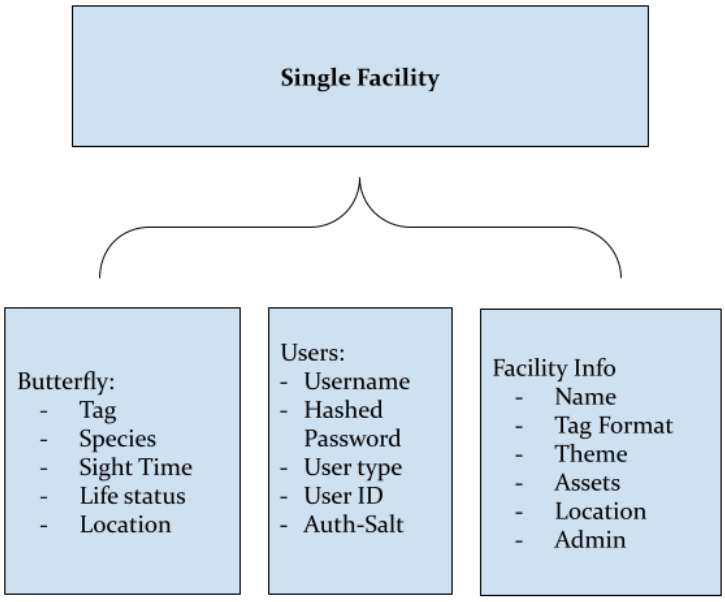

Figure 4.6 (Example of Database collection hierarchy)

Each facility will have its hierarchy of collections with the above fields being included in each collection object. Separating the facility butterflies from each other makes it much easier to give reporting information based on that individual facility. This means only their butterfly data will be parsed for filtering and returned to the user.

In the special case, a super admin will request butterfly reporting information from multiple facilities; this is the only time a join condition will be implemented. Since different facilities may have the same tagging method, it's essential to include the location of each butterfly directly within its database object entry. This means even when multiple facility butterfly collections are joined together and presented, Butterfly entries can still be distinguished by location.

Figure 4.7

For a more high-level and easy-to-understand model of the design, please refer to the following document: Detailed Design and Visuals

4.3.3 Functionality

Our design centers on two primary use cases: logging butterfly sightings into the database and outputting and analyzing this sighting data. The app enables users to easily input sightings, tagging location, time, and other details. These inputs are stored in a database for streamlined analysis and retrieval.

Additionally, the app includes a robust user management system, allowing for varying access levels based on user roles. This ensures that different users, from researchers to citizen scientists, have tailored access to the database and app features. For example, while a researcher may access detailed data and analytical tools, a casual user may only input sightings.

The design is also adaptable across institutions, enabling each to manage its own data access policies and customize the user experience as needed. This flexibility ensures the app serves diverse institutional needs, facilitating collaborative and secure data collection across multiple organizations.

4.3.4 Areas of Concern and Development

Our current design is on track to meet all project requirements and user needs. So far, any challenges we've encountered have been manageable with our chosen tools and design approach, which offer the flexibility and scalability needed for our application. The extensibility of our tools has allowed us to adapt smoothly, and we see no need for major changes at this stage.

At present, we have no significant concerns about the development process and, thus, no specific issues to address. However, as we move forward, we'll continue monitoring for any potential areas needing refinement and remain open to feedback from clients, TAs, and faculty advisers to ensure our design remains aligned with expectations.

4.4 Technology Considerations

MongoDB: This database is ideal for flexible, semi-structured data like sightings. While it lacks some transactional integrity compared to SQL databases, its scalability and JSON-friendly structure make it a strong choice. Although a SQL database like PostgreSQL could improve data consistency, MongoDB's flexibility aligns better with our needs.

Spring Boot: Chosen for its robustness and strong support for RESTful APIs, Spring Boot simplifies backend development and integrates seamlessly with MongoDB. Although complex, its built-in features reduce boilerplate, saving time. Alternatives like Node.js could offer a lighter stack, but Spring Boot's stability and Java-based environment are ideal for our goals.

HTML/CSS/JavaScript: These core web technologies provide broad compatibility and control over the UI, allowing us to build a responsive and accessible interface. While frameworks like React or Vue.js could streamline development, using plain HTML/CSS/JavaScript keeps our front end lightweight and manageable.

4.5 Design Analysis

So far, we've successfully translated our frontend design concepts from Figma into HTML, which is now fully viewable and loadable. However, we have yet to implement the backend functionality and connect the pages, which will primarily involve JavaScript. In the future, we will need to link the frontend to the backend APIs, which are being developed concurrently. On the backend, we've made progress on implementing the APIs and connecting the database to support our needs. We have completed basic functionality for butterfly tagging and sightings but still need to implement user authentication, user management, and additional API features to complement the tagging and sightings system. Looking ahead, we plan to focus on finalizing these integrations and ensure the full system works seamlessly. While the overall design is feasible, we are addressing key functionalities in parallel to avoid any delays in the user experience or data management.

5 Testing

5.1 Unit Testing

Frontend units being tested:

- UI Components
  - Manually opening each page in three separate browsers (Chrome, Edge, and Safari) to ensure compatibility
  - Manually checking the responsiveness of each page to different screen sizes by adjusting the screen size in the browser to ensure compatibility with all device types
- Calls to the Backend
  - Create mock calls to Postman to ensure that all endpoints are responding correctly and our system properly handles errors
- Page Navigation
  - Manually click through the navigational elements, ensuring that we can reach each page from one another.
- Individual Functions
  - Set up unit tests to isolate each individual function
  - Set up unit tests to test different functions being called in succession to ensure functionality does not change

Backend/Database units being tested using JUnit tests:

- Tagging System
  - Tag-System Relationships
    - Tags correctly belong to their associated tag systems.
    - Tag systems can identify the tags they encompass.
  - Tag Definitions
    - A specific tag is properly checked against a tag definition to confirm inclusion.
    - All tags generated by a specific tag are correctly validated as part of a given definition.
  - Equality and Hashing
    - The equals and hashCode implementations for tags, tag components, and tagging systems are tested to:
      - Guarantee consistent behavior in data structures like HashSet and HashMap.
      - Ensure that equality works as intended across various use cases.
- Database Actions
  - Document Requirements
    - Each database document has specific logic requirements, and unit tests verify that these requirements are consistently met.
  - Duplicate Prevention
    - Duplicate entries cannot be added to the database, following defined constraints.
    - Business logic for uniqueness is enforced for all objects.
  - Insertion and Deletion Integrity
    - When a document is inserted, all linked documents are updated or created as necessary.
    - When a document is deleted, related documents are either safely removed or updated to reflect the change, preventing orphaned records.

    - Permission Validation
        - Each database operation (e.g., insertion, deletion, update) is checked against the permissions of the request sender.
        - Unauthorized requests are rejected, ensuring secure and compliant access to sensitive data.

5.2 Interface Testing

Frontend to Backend:

- Testing API endpoints
    - Validate that the endpoints are correct and using the right request method (GET, POST, PUT, or DELETE)
    - Ensure correct error handling for invalid inputs and edge-cases

Backend to Database:

- Query Validation
    - Validate query data flow, results, and result formatting
    - Confirm response time is appropriate and query is optimized for performance without sacrificing correctness

5.3 Integration Testing

Backend Integration:

- Integration tests involve a running MongoDB instance (or an embedded database) to verify actual database operations.
- Once the database and backend are connected, manual checks are performed to validate key workflows like user creation, tag assignment, and data retrieval for sightings.

Backend-Frontend Integration:

- Once the frontend and backend are connected, we manually test the data-driven functionalities to ensure they work as intended. This includes workflows such as:
    - User authentication and authorization.
    - Tagging and spotting processes.
    - Displaying data for various institutions and users.
- Since the API is already verified during backend testing, the focus is on ensuring the integration works as expected in real-world scenarios.
- This approach ensures that any issues arising from frontend-backend interactions are identified and resolved efficiently without duplicating API-level verifications.

Screen Functionality Integration for Frontend:

- Navigational Integration
    - Ensure that the new page can be properly navigated to and from based on existing screens
- Data Fetching Integration:
    - Ensure the page properly receives and displays all data obtained from requests to the backend

5.4 System Testing

For frontend, we will have automated unit tests that will navigate through both common and uncommon workflow patterns. We will reach a coverage level of at least 90% with this set of unit tests. We will utilize the interface testing strategies in our system tests by setting up mock calls through Postman in our unit testing scripts. This will allow for testing data to not interfere with production data.

Frontend testing will include unit testing and end-to-end testing to ensure a viable user experience. End-to-end tests are arguably the most important tests to conduct; however, unit testing helps create a foundation for each module to ensure proper usage. End-to-end testing will be the final form of testing implemented within the test suite for the front end, ensuring all GUI elements are functioning properly.

Our system-level testing strategy ensures the backend operates reliably by combining automated and manual approaches. Unit tests verify individual components, ensuring their functionality in isolation. Interface and integration tests use a hosted MongoDB instance to validate operations like inserting, retrieving, and deleting data while confirming relationships and constraints are upheld.

For deployment, we perform manual testing to ensure services are accessible externally and workflows like user registration and tag assignment function as expected. This layered approach ensures all components and their interactions meet project requirements while focusing on practical, real-world functionality.

5.5 Regression Testing

We will have a set of automated test cases that we will run each time with new coding updates. We have a CI/CD pipeline setup within GitLab to help us achieve this. We are using git and gitlab which will easily allow us to revert back to older versions of the project at any time. Due to our agile workflow, regression testing is made quite easy. Valuable continuous integration testing will be able to run previous tests on new versions of code before it is merged into the main branch and creates conflict in deployment. Each branch will be tested when committed to, ensuring no previous test cases are broken in the process of development.

5.6 Acceptance Testing

We will ensure performance metrics are met, this includes page load time and request response time. We will ensure that exported files of data reports are of a high quality and manageable download size. Along with these quantitative acceptance tests, we will also have regular qualitative testing in the form of meetings with the client/product owner to ensure project scope and goals are met. These qualitative tests will require strong communication with the client to meet the needs of his expectations.

5.7 Security Testing

Our project does value security to an extent. Since our web app will be marketed to other facilities for usage, information privacy is a strong concern. Sensitive information for each facility involved

could be held in the database or on the site's frontend. This being said, we are taking security testing principles into account when developing the web app.

Since we are using AWS and Docker to host both the database and frontend web app, we will have a strong built-in security infrastructure in place. Most of our work will be done to ensure backend requests cannot be manipulated in any way. This means no SQL or query injection into fields for an attacker to retrieve database information pertaining to facility or user account data. Along with this we will ensure that each request is properly formatted and cannot have request parameters nor request bodies manipulated by an outside user to try and break the site. This will be implemented with both frontend and backend logic to ensure user input is treated properly by the system.

Lastly, vulnerability testing will be one of the later forms of security testing we do. As stated before, AWS should have a strong infrastructure in terms of machine security, meaning we will have little work to do when configuring host machine security; however, vulnerability testing, port scanning, and more will be administered to ensure safety. This will also be adapted with AWS machine configuration to ensure ports are closed, secure versions of operating systems are chosen, and no severe connection vulnerabilities are present in which an attacker could gain access and enumerate through.

Lastly, since we are using AWS we will have longevity with machine security as AWS security systems will be updated even after we have completed development on the project.

5.8 Results

Our testing results thus far indicate strong compliance with requirements and user needs. All unit tests have passed successfully, validating the individual components of our system. Additionally, manual testing of workflows and components confirms that the design performs as expected. While these tests do not guarantee an error-free implementation, they significantly reduce the likelihood of bugs in the software.

Qualitatively, client feedback has been positive, particularly regarding the UI design. The service's speed meets user expectations, and further performance testing is deemed unnecessary at this stage due to the already satisfactory user experience. Moving forward, our focus will remain on maintaining this level of reliability as new features are integrated.

6 Implementation

Frontend

The goal for the end of the semester was to have a functioning prototype of the guest user experience on a live server. This includes each guest page having full functionality on any devices, navigation between pages being properly implemented, and requests to the backend sending and receiving information to display on the page.

Views

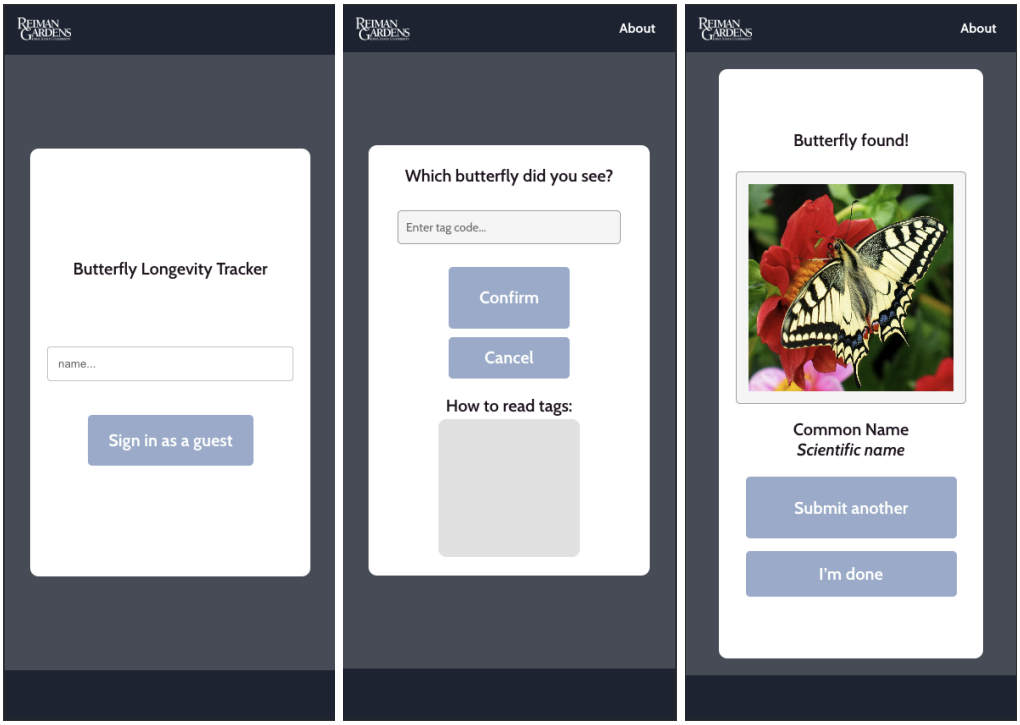

Figure 6.1

Each view is configured to handle all mobile, tablet, and desktop devices. Navigation of pages is implemented in javascript through button listeners. Error handling is implemented based on required input fields for each page, not letting the user progress until the proper information is included. Upon making a tag request, a database call is made to create a new sighting, which returns the information of the tagged butterfly if found and returns an error message if the tag does not exist in the system, in which the user will not be redirected.

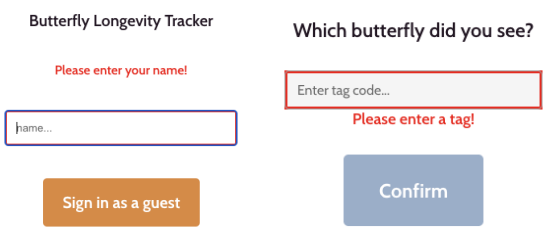

Figure 6.2

Backend

The goal for the end of the semester was to have a functioning database and backend that implements requests to retrieve, modify, and store reporting information. This includes database table design, request style, and properties, and Spring setup and implementation.

The backend uses the Java Spring framework to handle requests between the frontend and the database. Requests are split into four controller classes that handle retrieving and providing information to the database. These controllers are currently Butterfly, Domain, Species, and User which have a combined 19 requests. The createSighting request is the prominent one that is currently used in the guest experience. Formatting follows a JSON structure with the same format being implemented throughout; the body of the createSighting request is shown below.

Figure 6.3

```
{
    "username": "{{userName}}",
    "domainId": "{{domainID}}",
    "tag": {
        "background": {
            "shape": ".",
            "color": -65536
        },
        "foreground": [
            {
                "shape": "X",
                "color": -16777216
            },
            {
                "shape": "Y",
                "color": -16777216
            },
            {
                "shape": "Z",
                "color": -16777216
            }
        ]
    }
```

The request body sends the username of the currently logged in user along with the facility ID that is currently being used. Tag information is then passed to the backend (so any tag type can be used) which allows for the backend to find the proper tagged butterfly entry. The foreground of the tag is sent in an embedded list, which allows for any type of tag; here, we are specifying the alphanumeric entries on the tag that the user has entered.

This request then proceeds to fetch the tagged butterfly and construct a new sighting entry for the given facility within its facility collection table.

The request will respond with a similar JSON body which is used to populate the following screen for the frontend. This includes the butterfly's common name, species name, and a picture provided by the user of the given butterfly. This can be seen in the example image from the frontend section above. Username, facility information, and sighting time are entered into the sighting database, which can later be reported on.

Database structure consists of a multitude of MongoDB collections, Butterfly, Domain, Species, and User, each of which is represented by a specific object. Here's a rundown of the information contained in each object:

- Domain
  - Unique ID
  - Name of the Facility
  - Tagging System Used by the Domain
- User
  - Unique ID
  - Username
  - Password (hashed for security purposes)
  - Domain (as a reference)
  - Permissions
- Species
  - Unique ID
  - Scientific Name
  - Common Name
  - Description
  - Image
- Butterfly
  - Unique ID
  - Species (as a reference)
  - Domain (as a reference)
  - Tag Identifier
  - List of Sightings

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	Definition	ACM Code of Ethics	Team Description
Work Competence	Work is completed to the highest quality within our teams ability	2.1 Strive to achieve high quality in both the processes and products of professional work	Regular code testing and coding reviews prior to merging changes to our main branches
Financial Responsibility	Keep costs to a minimum while maintaining quality	1.2 Avoid harm	Researched several hosting services and picked the one that is most cost-effective
Communication Honesty	Give updates on work you have done, plan to do, and are currently doing	1.3 Be honest and trustworthy	Sharing with each other what we are working on
Health, Safety, Well-Being	Ensure the security of the facilities and users	1.2 Avoid harm	We are looking into implementing security for our website
Property Ownership	Respect the ownership of resources and ideas available to us	1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts	We ensure that all of the content we use is either cited or free for use
Sustainability	Ensure environmentally friendly software and use sustainable tools	1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.	We chose technologies that are well-established and will be serviced for many years to come
Social Responsibility	Upkeep user engagement and involvement amongst guests at the facilities	2.7 Foster public awareness and understanding of computing, related technologies, and their consequences.	We designed our UI in a way to keeps our users engaged

Figure 7.1

What we are currently doing:

Currently, we are keeping regular contact via Discord messaging to update each other on the work that we are doing. We are using Git and GitLab for version control as well as quality control. We utilize code reviews and code tests prior to merging any code changes to our main branches. This ensures that we are keeping a high quality of code and will reduce the risk of finding bugs down the line. We can see our progress through our GitLab issue board, which has all of the things we plan to do, are currently working on, and have finished working on. We have implemented existing software features in order to improve our design and increase sustainability.

Area we are doing well in:

Work Competence: We have ensured that all code changes are reviewed and tested by at least one other member of the team before merging a branch. This ensures quality work and keeps us all accountable for fixing any oversights in our own code caught by another team member.

Area we need to improve in:

Communication Honesty: We currently do not always keep each other updated on what we are working on and there can be disconnects between what we expect someone to be doing vs what they are actually doing. This can be harmful to our progress as we don't want to assume something is being done when it is actually not being worked on. Going forward, we will start to implement more frequent communication of our work via online calls and discord messaging.

7.2 Four Principles

	Beneficence (Promoting Good)	Nonmaleficence (Avoiding Harm)	Respect for Autonomy	Justice
Public health, safety, and welfare	Our project design greatly increases the performance and ability to report administrators, which affects their day-to-day workflow.	Our design does not harm the general well-being of users in any way.	Our design allows each facility to configure their UI in their own way which can benefit their personal brand and image.	Our design will have the same functionality for each facility, not creating an imbalance of features from location to location.
Global, cultural, and social	Our design promotes a more accessible and well-performing workplace within any facility that decides to utilize our application.	Our design does not harm the values of any specific cultural groups or people,	Since our design isn't required to be used by guests, there is no cultural autonomy impact on our design.	Our project design will benefit any facility enclosure workspace for administrators, volunteers, and guests in a multitude of ways.

Environmental	The design does aim to provide data to administrators and researchers to benefit the longevity of butterflies, which I would say is a good thing to promote.	Our design does depend on tagged butterfly enclosures which could be seen as a harm to the natural environment for butterflies.	Tagging butterflies could have an environmental autonomy impact, but our project builds on the basis that the butterflies are already tagged. However, this is likely still a negative impact on butterfly autonomy.	Our project design will provide butterfly enclosure administrators with relative information on butterfly life spans based on species. This can be used to derive the environmental impacts of enclosures on specific species' lifespans.
Economic	Our web application will provide Butterfly Exhibits with an inexpensive way to maintain their data while keeping a high-quality user experience for their guests.	We chose an inexpensive hosting service in order to mitigate the amount of money our client needs to spend.	Exhibits don't have to pay for a new tagging system, as our web application will be able to handle any system they choose to use.	All exhibits will have access to the web application at no cost.

Figure 7.2

Environmental Beneficence is very important to our project. Since our design relies on guest visitors to utilize the application, we need to prove to them that they are helping the facility create a better environmental life for the butterflies based on species. Our design will already ensure the availability of data to administrators; however, we still need guests to participate in sighting entries to provide more quality data.

Environmental Autonomy is weak in our project due to the reliance on tagged butterflies. Tags on butterflies disrupt their natural life; however, we believe this is made up for the positive impact the design of the project can have. Our project also did not propose tagging of butterflies. Butterflies in facilities were already tagged for information research purposes in which we are providing a valuable tool for facilities to utilize.

7.3 Virtues

List and define at least three virtues that are important to your team. Describe what you will do or have done as a team to support these virtues among all team members.

Each team member should also answer the following:

- Identify one virtue you have demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - How have you demonstrated it?
- Identify one virtue that is important to you that you have not demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - What might you do to demonstrate that virtue?

Andrew

Accountability has been a virtue I have demonstrated thus far in our senior design project. This is important to my character and taking responsibility for my actions, along with taking responsibility for my assigned work for the team. Persevering and communicating with my team to ensure I meet deadlines and quality work is keen when it comes to accountability. I have shown this virtue by being clear and honest with my team in my work progress and communicating my own responsibilities for the project.

Determination is a virtue that I would like to work on moving forward with the project. Although I think I am determined to meet deadlines and project goals, I haven't acted on this virtue up to this point. being more determined in the future could help the team collaboratively push the project further than its intended goal and boost team performance. To demonstrate this virtue, I could take more of a leadership role and accept more accountability for the project in order to increase determination in finishing the project for not only me but my team as well.

Alex

Collaboration is a virtue that I have shown while working on this project through participating in code reviews and sharing my work and knowledge with other group members. I ensure that all of the code I add is tested and reviewed by another team member to align with our team's standards. I make sure that everyone is aware of what I am working on and make sure to update team members on the work we need to complete.

Resilience is a virtue that I could improve upon going forward with the project. I feel that sometimes I will run into a difficult problem and it will deter me from completing work that I need to get done. I sometimes will put the work on hold for too long and it could delay my completion time when I could have completed it earlier.

Charlie

Reliability is a virtue that I have shown consistently throughout this project. It is important to always follow through with your word given to your teammates, especially when working with deadlines as it can cause incomplete products if not properly addressed. I have shown this throughout the semester by ensuring anything I commit to is complete by the needed deadlines.

Creativity is a virtue that can lead to great solutions for a problem. Unfortunately, I feel that sometimes I have not taken full advantage of this virtue and resort to the safe option of doing what I know already. This virtue is important to innovating and creating new better solutions to existing problems. To demonstrate this virtue, I could start to put more effort into thinking of new possible methods to accomplish issues that I may have previously handled in a different way.

Carter

Tenacity has been key for me during this project. It's important to follow through on commitments, especially with deadlines, to avoid leaving tasks unfinished. I've made sure to meet every deadline and complete all my responsibilities on time throughout the semester.

Punctuality is crucial for keeping everything on track, but I recognize I've struggled with being on time for meetings and check-ins. Being punctual helps maintain momentum and respect for everyone's time. To improve this, I plan to better manage my schedule and set earlier reminders to ensure I'm always on time.

Jaret

Adaptability is a crucial skill and component that a software engineer must have. Many times a project may need to be adjusted or an approach will need to be changed since there are flaws existing in the original plan. The ability to willingly adjust your plan and make the necessary changes is an important part of the job of a software engineer

Perseverance is a virtue that is important for a software engineer to possess. Software development is not always a pretty process and can bring many headaches to the user. This virtue has been demonstrated throughout the semester since there have been several small roadblocks and challenges that the team has faced and have needed to overcome.

8 Closing Material

8.1 Conclusion

The main goal of the project was to work with our client to understand the needs of the project and to plan our approach for development. Different website screens have been converted from Figma boards that were received from our client which demonstrated a desired appearance of the interface. A prototype has been created that represents a basic form of the back-end environment, which contains models for each of the major objects that need to be stored.

During this first part of our project, we set out several goals for us as a team to achieve. Our first goal was to learn how we could work together as a team. During our time together, we have been able to learn about each other's strengths and weaknesses in our work and how to cover each other. Our second goal was to build a good working relationship with our client and supervisor. This goal we have partially accomplished, but due to the nature of our project, we have not been able to be in

contact with our client or supervisor that often since we have had no major updates or deliverables. Our third goal was to continuously present high-quality work that meets the deadlines. We have continuously presented exemplary work to our professors, clients, and supervisors. Our fourth goal was to be in regular communication with our other team members. This goal has been met since regular weekly meetings have been held and additional communication has been made. The final goal was to create a prototype of our project that we could present to our professors, clients, and supervisors. This final goal has not technically been met yet, however, it is expected to be completed before the end of the semester.

Reflecting on our goals and the progress that has been made on them, while many of the goals were achieved, there were a few that our team fell short of. The biggest goal that the team fell short of was staying in regular communication and building a working relationship with our client and supervisor. Our team struggled to stay in consistent communication with our supervisor, which hurt our relationship. It was determined that we need to take a step back and put more emphasis on our communication. We began to meet with our supervisor more regularly and have been working on repairing our relationship. Our team members were all in regular contact with each other which allowed for each member to thrive in their own regard.

8.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE). See link:
https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf

[1] K, Baum, “Monarch Listing Announcement Expected Next Week,” monarchwatch.org. Nov. 26, 2024. [Online]. Available:https://monarchwatch.org/blog/. [Accessed Dec. 4, 2024]

[2] K, Ball, S, Burcher. D, Puma, “The Sky’s the limit for Monarchs Wearing Solar-Powered Radio Tags,” beecityusa.org. Dec 7, 2023. [Online]. Available:https://beecityusa.org/the-skys-the-limit-for-monarchs-wearing-solar-powered-radio-tags/. [Accessed Dec. 4, 2024]

[3] M, Roberti, “What is the World’s Smallest RFID Tag?,” rfidjournal.com. [Online]. Available:https://www.rfidjournal.com/ask-the-experts/what-is-the-worlds-smallest-rfid-tag/. {Accessed Dec. 4, 2024]

[1] K, Baum, “Monarch Listing Announcement Expected Next Week,” monarchwatch.org. Nov. 26, 2024. [Online]. Available:https://monarchwatch.org/blog/. [Accessed Dec. 4, 2024]

[2] K, Ball, S, Burcher. D, Puma, “The Sky’s the limit for Monarchs Wearing Solar-Powered Radio Tags,” beecityusa.org. Dec 7, 2023. [Online].

Available:https://beecityusa.org/the-skys-the-limit-for-monarchs-wearing-solar-powered-radio-tags/. [Accessed Dec. 4, 2024]

[3] M, Roberti, “What is the World’s Smallest RFID Tag?,” rfidjournal.com. [Online]. Available:https://www.rfidjournal.com/ask-the-experts/what-is-the-worlds-smallest-rfid-tag/. {Accessed Dec. 4, 2024]

8.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.

9 Team

Complete each section as completely and concisely as possible. We strongly recommend using tables or bulleted lists when applicable.

9.1 Team Members

- Alex Herting
- Andrew Ahrenkiel
- Carter Awbrey
- Charles Dougherty
- Jaret Van Zee

9.2 Required Skill Sets for Your Project

- Agile
- AWS Server Hosting
- CI/CD
- Cyber Security
- Database Management
- Data Security
- Database Design
- Gitlab
- Git
- Git Command Line Interface
- HTML/CSS/JS
- Java
- Java Spring
- JUnit Testing
- MongoDB
- Project Management
- SCRUM
- Social Engineering
- Software Architecture
- Software Testing

9.3 Skill Sets covered by the Team

Jaret Van Zee

- Agile
- CI/CD
- Cyber Security
- Data Security
- Database Management
- GitLab
- Git
- Git Command Line Interface
- HTML/CSS/JS
- Java
- Java Spring
- JUnit Testing
- MongoDB
- Project Management
- SCRUM
- Social Engineering
- Software Architecture
- Software Testing

Andrew Ahrenkiel

- Agile
- CI/CD
- Cyber Security
- Database Management
- Data Security
- Database Design
- GitLab, Git, Git CLI
- HTML/CSS/JS
- Java
- Java Spring
- JUnit Testing
- MongoDB
- Project Management
- SCRUM
- Social Engineering
- Software Architecture
- Software Testing

Charles Dougherty

- Agile
- Database Management
- Database Design
- Gitlab
- Git
- Git Command Line Interface
- HTML/CSS/JS
- Java
- Java Spring
- JUnit Testing
- MongoDB
- Project Management
- Software Architecture
- Software Testing

Alex Herting

- Agile
- Database Management
- Database Design
- Gitlab
- Git
- Git Command Line Interface
- HTML/CSS/JS
- Java
- JUnit Testing
- MongoDB
- Project Management
- SCRUM
- Social Engineering
- Software Architecture
- Software Testing

Carter Awbrey

- Agile
- AWS Server Hosting
- CI/CD
- Cyber Security
- Database Management
- Data Security
- Database Design
- Gitlab
- Git
- Git Command Line Interface
- HTML/CSS/JS
- Java
- Java Spring
- JUnit Testing
- MongoDB
- Project Management
- SCRUM
- Social Engineering
- Software Architecture
- Software Testing

9.4 Project Management Style Adopted by the team

Agile

- GitLab issue board used to assign and track team progress
- Weekly group “stand up”
- Project tickets/issues split amongst group members
- Issues identified and added to the backlog
- Biweekly client check-ins acting as product owner

9.5 Initial Project Management Roles

1) Alex Herting	Frontend Developer
2) Andrew Ahrenkiel	Full Stack Developer
3) Carter Awbrey	Backend Lead
4) Charles Dougherty	Frontend Developer
5) Jaret Van Zee	Backend Developer

9.6 Team Contract

Team Members:

1) Alex Herting	2) Andrew Ahrenkiel
3) Carter Awbrey	4) Charles Dougherty
5) Jaret Van Zee	

Team Procedures

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

(1) Regular team meetings outside of class will be done virtually over Discord. The regular team meeting will occur on Friday at 6 pm.

(2) The preferred method of communication for updates, questions, reminders, or general information will be done through Discord chat or Discord voice call. If communication is needed for any inner conflicts or major issues, it will be done in person or via Discord video/voice call.

(3) Decisions will be made by a majority vote since there is an odd number of team members. Team members should be given a fair amount of time to discuss their choice and why they want that choice. Team members in the minority vote should comply with the decision made. Team members in the majority should see if they can compromise by including any qualities of the minority's decision.

(4) Carter Awbrey will be the official team auditor for team meetings. The backup team auditor will be Jaret Van Zee. The team auditor will be expected to take notes from each meeting. The notes should well document the current progress of each team member's work, any roadblocks, and planned work. All important information should also be documented in the auditor's notes, such as new meeting times, deadlines, and important workflow changes. The auditor should make the notes available no more than 1 day after the meeting to the rest of the team.

Participation Expectations

1. **Expected individual attendance, punctuality, and participation at all team meetings:**
2. **Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:**
3. **Expected level of communication with other team members:**
4. **Expected level of commitment to team decisions and tasks:**

(1) All team members are expected to show up on time to any scheduled team meeting unless they communicate prior to the meeting. All team members are expected to participate whenever they can in a meeting. All team members should be actively listening whenever another team member is speaking. Conflict and discussion are okay in a team meeting, but members must respect each other's ideas–even if they do not agree with the idea being discussed.

(2) Team members will need to put their full effort into whatever work they are presenting to the rest of the team. All hard deadlines should be met ahead of time to review each other's work. Any teamwork will be divided equally among all team members as fairly as possible. A soft deadline will be determined, likely 2-3 days before the hard deadline, that all team members should have their segment of work completed by.

(3) Team members are expected to regularly update their other team members. If any team member runs into a roadblock, they should notify their team members in no more than 24 hours. If a team member runs into any difficulties or roadblocks, a discussion or team meeting should be done in order to solve the issue.

(4) A team member is expected to work on their work or tasks for an average of three hours per week. Three hours per week is not a hard limit; this number could be higher or lower on a weekly basis. These three hours will not include any regularly scheduled team meetings.

Leadership

1. **Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):**
2. **Strategies for supporting and guiding the work of all team members:**
3. **Strategies for recognizing the contributions of all team members:**

(1) These are the leadership roles that each team member fulfills:

Jaret Van Zee - Database Manager & Timeline Organizer

Carter Awbrey - Project Manager & Visionary

Alex Herting - Frontend Manager

Charles Dougherty - UX/UI Design Director

Andrew Ahrenkiel - Team Organization & Technical Design

(2) These are the strategies each team member will follow to support and guide the work of other team members.

Jaret Van Zee - I will ensure that all team members are getting assigned the work that is appropriate for them. I will also make sure that work is getting split fairly between the team members and that work is getting done on time.

Carter Awbrey - As the project manager, I will oversee the communication between members to ensure that deliverables are met and that our product meets the needs of our client. Additionally, I will work to address and mediate issues that may arise between team members. This doesn't limit my work to solely leadership based contributions, but does include them.

Alex Herting - As the frontend manager, I will be responsible for creating tickets related to the front end and assigning them to members of the team. Checking in to see how far along we are on tasks and re-assigning tickets accordingly.

Charles Dougherty - It is essential that all parts of the team work collaboratively to create one whole product rather than pieces strung together. I will ensure that each person feels included and understands the goal of what is currently being worked on and make sure that what has already been completed follows the requirements. I will also contribute to handling conflicts in code or design choices to create a better final product, even with differing ideas.

Andrew Ahrenkiel - As the Team Organization leader, I will ensure weekly meetings have applicable purposes and are productive. I will be responsible for the Gitlab issue board and tracking member progress. As the Technical Design Lead, I will ensure all code within the development branches is functional and appropriate; I will review the majority of code merges to ensure code quality.

(3) These are the strategies each team member will follow to recognize each other's work.

Jaret Van Zee - Keep to set soft and hard deadlines for work that needs to be completed. Will do regular check-ups on team members to ensure they are doing okay on their own work.

Carter Awbrey - Follow set team deadlines and communicate regularly and clearly with other team members.

Alex Herting - Setting appropriate deadlines and checking in weekly on the front-end tasks to make sure that we are meeting those deadlines.

Charles Dougherty - Follow deadlines, checking in on each area of the project and not just where I am currently working, checking GitLab commits and pushes, and discussing future improvements/changes.

Andrew Ahrenkiel - Gitlab issues, weekly check-ins, code commits and pushes, feature branches, etc.

Collaboration and Inclusion

Describe the skills, expertise, and unique perspectives each team member brings to the team.

Strategies for encouraging and supporting contributions and ideas from all team members:

1. **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?)**

(1) These are a list of skills, experience, expertise, and unique perspective that each team member brings to the team:

Jaret Van Zee - He is currently working as an IT intern. Jaret also has skills in back-end development and database management. Jaret can use his knowledge from his cyber security minor to bring new perspectives to the team and the overall security of the software being developed.

Carter Awbrey - Skills include server-side and cloud integration having worked with cloud products many times in the past. Additionally I have experience writing UI in HTML/CSS/JS or using React frameworks. I have professional experience writing backend/server software in .NET and Java as well as integrating that with frontend products.

Alex Herting - Skills include full-stack development, having experience with React framework and SQL for databases. I currently work as a software developer, which has given me experience with the software development process and managing large workloads. Have experience in Javascript in web development.

Charles Dougherty - Experienced in working in teams to develop a strong final product. During my internships, I have worked in SQL and Mongo databases and also developed many frontend applications. Finding new solutions to a problem is one of my strong suits as I enjoy finding new technologies and discovering what can be done within the limitations.

Andrew Ahrenkiel - Skills include full-stack development, particular experience with both React and Angular, along with Mongo, SQL, and Oracle. I can bring a unique viewpoint of the SDLC from my internship experience with Wells Fargo, having worked as a DevOps engineer and a Fullstack Software Engineer. I also recently worked on a database migration project as part of my internship, which may provide me unique expertise in database design

Goal-Setting, Planning, and Execution

1. **Team goals for this semester:**
2. **Strategies for planning and assigning individual and team work:**

3. **Strategies for keeping on task:**

(1) These are the official goals that this team will strive to achieve this semester:

I. Learn how to work with each other as a team.
II. Build a good working relationship with our client(s) and supervisor
III. Continuously present high quality work as a team and individually to our professors, client(s), and supervisor. Ensure that all work completed meets its expected deadline.
IV. Regularly communicate with each other about how progress on our work.
V. Create a detailed and exact prototype of product we plan to build for our client in the following semester.

(2) Whenever new work is assigned to the team, a timeline should be created that the team will follow to make sure that the work is completed by the deadline. Each team member will be assigned work that best meets their expertise or interest. All work will be divided evenly to the best of our ability.

(3) During each team meeting, each team member will discuss what work they have completed, what work they plan to complete in the next few days, and if the member has run into any issues. The other team members will compare their current work progress to the expected timeline of progress to ensure that the team member is getting their allotted work complete.

Consequences for Not Adhering to Team Contract

1. **How will you handle infractions of any of the obligations of this team contract?**
2. **What will your team do if the infractions continue?**

(1) If any team member knowingly violates the team contract, the following procedures will be met.

First Infraction - A soft warning, the team member that committed the infraction will asked by another team member to not commit that infraction again

Second Infraction - A strong warning, one or two team members will directly say how the team member's infractions have affected the team and what they've done

Third Infraction - Official Team Meeting, an official team meeting will be made in order to discuss how this can stop in the future. A resolution must be reached and a remediation plan must be created in order for the team meeting to be successful

(2) If a team member is consistently committing infractions against the team after the warnings and team meetings. All team members except the team member committing the infractions will meet with the SE 491 professors to discuss a best possible solution for the rest of the solution. This may involve 491 professors directly talking with the team member committing the infractions, a specialized plan may be created for the team member, or possibly removal of the team member from the team.

****

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) Andrew Ahrenkiel DATE: 09- 19 - 2024

2) Alex Herting DATE: 09 - 19 - 2024

3) Charles Dougherty DATE: 09- 19 - 2024

4) Carter Awbrey DATE: 09- 19 - 2024

5) Jaret Van Zee DATE: 09- 19- 2024