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

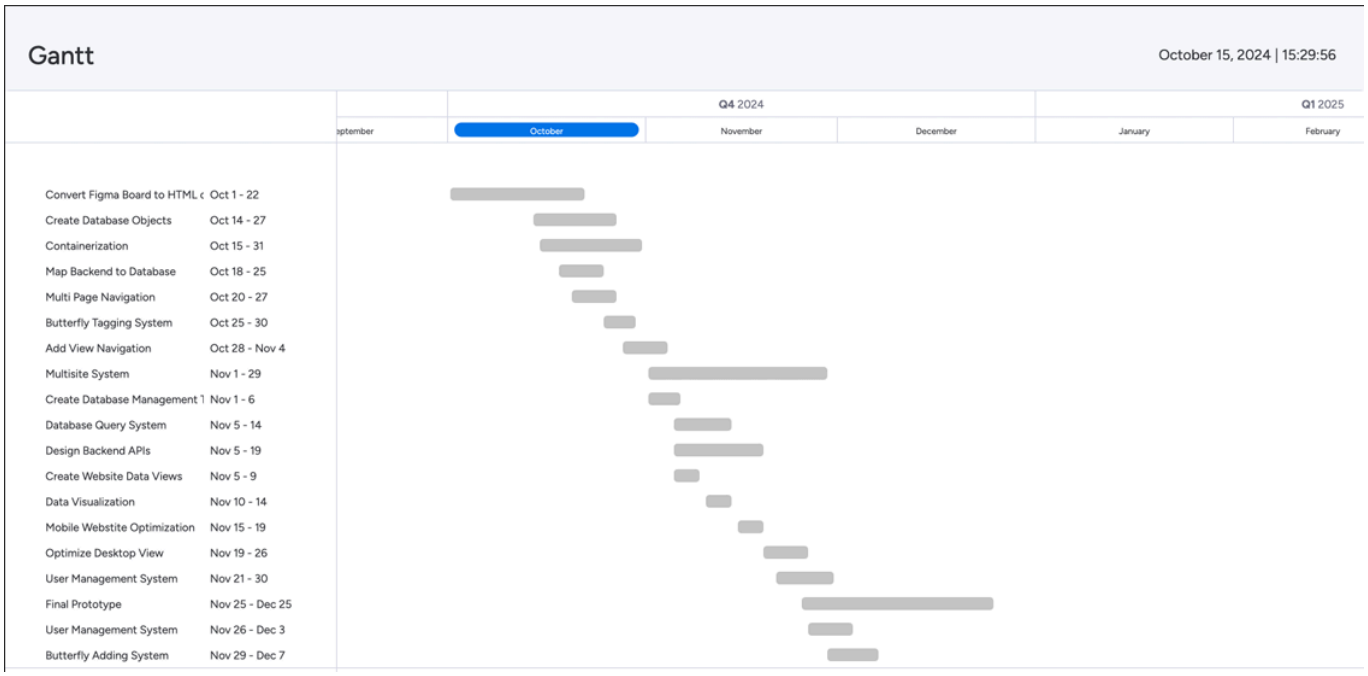
- UI Design Completion
    - Metric: Percentage of originally requested designs successfully converted to functional HTML.
    - Milestone: Complete the conversion of 100% of requested views into functional HTML components.
- Functionality Development
    - Metric: Percentage of interactive elements 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



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 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 yet explored.
	Butterfly data filtering and sorting	15	Ensuring that the filters and sorting methods are

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.
			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.
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.

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.
	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 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

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.
			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 feedback from the employees of Reiman
	Test website with guests and general public	5	Gathering feedback from the public utilizing surveys

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.