Final Report

Small Pupae Big Data

Team: sdmay20-01b

Team Members:

Manthan
Matthew Markose
Mark O'Meara

Executive Summary

Reiman Gardens receives shipments containing butterfly pupae. Reiman gardens needs a system where they can store all this information and analyze that data. They also need a way to make this work for multiple gardens. The following components were needed to make the app: An Authentication System, Butterfly Gallery, Plant Gallery, Custom Homepage, Multiple Users, Butterfly Search, Shipment Search, Shipments Modification, Butterfly Modification, Compatibility with large amounts of data.

The current system supports only one garden. Some components needed to be shut down on it to make it work. Some features like multiple users and shipment search did not already exist. The butterfly search on the current system was also broken.

Our app was to solve all those problems and add all extra features as required. We set out to accomplish this by using google's firebase which is a cloud based hosting service. In the final app multiple users were implemented using OKTA.

Engineering Standards and Design Practices

- Hardware is minimal in purpose, mainly a client facing web device and temperature/environment probe to create a standard of easy access and analysis
- Software practices planned are for easy adaptability for multiple potential clients, thorough data analysis presented in a simple way for clients, and cohesive design.
- Standards: all were considered; mainly ethics for pupae growth and competition

Summary of Requirements

- Ability to enter data into a UI
- Update and remove data entered into the system
- Provide automatic error catching for cleaning data
- Perform data analysis on all entered data
- Create necessary end of year reports
- Update in nearly real time
- Provide metrics to other suppliers on how their pupae are doing against other suppliers as a whole
- Visualize data sets per species of pupa
- Ability to export data for use in JMP or R
- Extra: Documentation on how the system works
- Extra: Make it mobile browser friendly

Applicable Courses from Iowa State University Curriculum

- CS: basics (101, 227, 228), algorithms & development (309, 311, 319, 329, 339)
- Statistics: intermediate (301), R (480)

New Skills/Knowledge acquired that was not taught in courses

- New framework potential so far: VueJS, Python integration
- Learning more about entomology and pupae growth
- Applying data analysis and software design for a real client-facing problem

Table of Contents

List of figures/tables/symbols/definitions

1. Introduction

1.1 Acknowledgement

Anita Westphal (of Reiman Gardens) - We would like to thank her for meeting with us to walk us through the client-facing process we are trying to improve and add insight to.

Nathan Brockman (of Reiman Gardens) - We would like to thank our official client, Nathan Brockman, curator at Reiman Gardens, for working with us and giving us the opportunity to work on this project.

We would also like to thank our project advisor, class professor, guest speakers, and classmates for being influential to our skills and opportunities in this senior design project.

1.2 Problem and Project Statement

Each week thousands of pupae are shipped from butterfly farms and ranches from around the world to butterfly flight houses worldwide, like the Christina Reiman Butterfly Wing found at Iowa State University's Reiman Gardens. This industry, in most cases, is considered a very green industry as the indigenous people use their native plants to produce a commodity then can sell on the world market, at the same time enhancing and protecting their native environment. Since its opening in November of 2002, records of the emergence success of each pupa has been collected and sent annually to the United States Department of Agriculture Animal and Plant Health Inspection Service Plant Protection Quarantine unit. To date Reiman Gardens has received over 300,000 pupae from its suppliers.

Not all pupae that are shipped to flight houses emerge properly or at all. Each facility worldwide provides conditions they believe to be best for the pupal stage of butterflies and moths. The conditions provided are usually based on generalizations, in most cases all species receiving the same treatment expecting a similar success outcome. It has been shown that some facilities do better with some species than others but the root cause for those success or failure on a particular species has not been explored fully. In 2014 Reiman Gardens Entomology Staff started reaching out to other butterfly flight houses around the world to request their emergence data to help develop best practice standards to improve emergence success across facilities.

The Entomology staff at Reiman Gardens has begun analyzing the data that has been received over the years but the current method of; receiving emails with Excel files, copying all the data into an Access tables, creating Access reports and then manipulating that data so it can be analyzed with "JMP" is very time consuming and cumbersome, affecting the long term success of this project.

The team proposed an online platform solution where facilities can enter and upload their emergence data. The team will create a database to manage all that data points that go along with emergence data and develop an interface to extract requested information. The data can then be used to provide some predetermined statistical analysis into the interface, but also give the capability to download in an easy format to utilize with "JMP" and/or "R" for analysis for more advanced calculations. Coupled with statistical analysis, storage and viewing of historical data will be available through the interface to check track trends and provide insight into different techniques.

1.3 Operational Environment

The operational environment will be of no concern as it will be an online platform that will be used on facility computers or faculty phones.

1.4 Requirements

Table 1.1: Requirements

ID	Requirement	Description
R1	Data entry interface for the data, with a focus on ease of use	Currently, most gardens record data in Excel. The final product must have a simple, easy to use interface to encourage gardens to switch systems. The product must have better features than Excel or paper and pencil, to ensure a beneficial transition to the system.
R2	Ability to update and remove data	Currently, the garden has no way to update data after it has been entered.
R3	Provide automatic error catching for cleaning data	Automatic error catching and cleaning of data would reduce time spent in the data collection phase of the research.
R4	Perform data analysis on all entered data	Data analytics metrics: ● End of day: what are the best methods (temperature, light, how close pupae are hung together, humidity, etc.) to give the highest rate of proper emergence (on a per species basis) ● Raising the whole industry standards for emergence ● Reiman is willing and wants to do experiments (and perhaps other facilities are as well), but currently the data is more observational ● Although health and life after emergence matters, for the purpose of this project the team will only focus on variables up to proper or improper emergence
R5	System to create end of year reports	All gardens must submit a variety of statistics each year regarding emergence. The system must report these statistics.
R6	Update in nearly real time	The system must update in nearly real time so that facilities can get a continual status update. This may encourage facilities to continue using the product.
R7	Provide metrics to other suppliers on how pupae are doing against other suppliers as a whole	The system must have a way to show the supplier how the pupa are doing and how they are doing against the average. ● Should not encourage competition between suppliers or gardens ● Should display potential ranges of success (For example, display maximum emergence rate knowing how many have released properly, how many haven't survived, and how many there are total) ● Must accept entry upload from gardens (To some extent, data should be taken how the gardens have it readily available)
R8	Visualize data sets per species of pupa	Data visualization should be done on a per species basis. The product must have visually appealing aspects, including graphs directed at a non-technical audience.
R9	Ability to export data for use in JMP or R	Data needs to be able to be exported in a way that can be plugged into JMP or R for more specific data analysis.

R10	Extra: Documentation on how the system works	Although little documentation has been done in past projects, the product should have all relevant documentation included.
R11	Extra: Make it mobile browser friendly	Add in features to make the website mobile friendly so that the workers can do everything from their phone instead of having to write it on paper and transfer it.

1.5 Intended Users and Uses

The end users will be other entomologists whom will use the application to enter data and perform basic data analysis. The entomologists entering the data will also be able to pull up historic data and compare results over time. The intended use of the data will be to determine the best environment and variables for proper emergence of the pupae to occur.

1.6 Assumptions and Limitations

Assumptions

- Data will not be coming in from more than a few users at a time
- Data sets will be large
- The primary users will be entomologists
- Data will be imported through Excel document
- Data will be labeled
- Sample template has been sent to users, therefore explanatory variables should be similar across gardens

Limitations

- The service available to use will have to be free and never accrue a cost.
- Storing large amounts of data will cost money. Product can only be accessed through Iowa State server. It must be deployed on Reiman garden application.
- Users must use template from Reiman gardens.

1.7 Expected End Product and Deliverables

Table 1.2: Deliverables

ID	Date	Deliverable
D1	November 30th	A UI for entering pupae data and generates reports
D2	February 15th	Data visualization and analysis of species
D3	March 30th	Historical data tracking and data supplier view
D4	April 30th	User manual and mobile friendly addins

Table 1.3: Deliverable Descriptions

ID	Expectation of Deliverable
D1	The UI delivered will be able to take data entered and store it in a database. The data entered will be accessible to the UI to generate reports as needed. The system must be easier to use than pen and paper while maintaining a visual appeal that will make the user want to use it. Data entered must be able to be removed and edited for correction purposes and update in real time. Data will also be cleaned before being stored so as to minimize any problems that will arise from improper data entry.
D2	System will be able to properly manipulate data for analysis and display the results in the UI. If the entomologist needs to perform more complicated calculations, then the data will be able to be exported. The data must be able to be separated by species and visualized on a per-species basis.
D3	Historical tracking will allow for the gardens to see how they are doing year-to-date. This will show if the changes made are contributing in a healthy or harmful way to the growth of the pupae. The data entered will be able to be seen by Reiman gardens and by the data supplier. This must enforce correct data entry and not create an issue with competition.
D4	All collected documentation will be used to determine a user's guide for the entomologists. The system will also be made to be mobile friendly, but not a mobile application. The mobile aspect must make it so the user can enter data easily without having to keep track with pen and paper.

2. Specifications and Analysis

2.1 Proposed Design

Progress

- Met with client to discuss deliverables
  - Met with client and experts in the field to learn who and how the team can help
- Divided portions of project amongst teammates
- Discussed technologies to use for different portions of project
  - Began creating and playing with pseudocode and simple unrelated projects to get familiar with frameworks

Functional

- Users must be able to import their data
  - Possibly Excel, Access, or other table data
- Provide data analysis
  - Visually as well as important statistics to track growth
- The data should be able to be modified
  - Admin/different level access to maintain data integrity across roles and client locations

Non Functional

- Use MySQL database or other well maintainable database or fork
- Interface should interact with JMP or R

2.2 Design Analysis

Up to this point, the team has primarily focused on defining requirements, clarifying user needs, and making a plan of action. The team has met separately with both entomologists, Nathan and Anita, to learn what they are looking for in an improved system. This worked well, since it allowed the team to independently see what each entomologist wanted and needed. In the future, however, the team will try to meet with both entomologists at the same time, so that each member of the team and the clients are on the same page.

Additionally, the team has spent time researching the best platforms and languages to make the project work the best. This includes looking at existing code and analyses, as well as researching external tools and gaining new ideas. This is working well currently, but will evolve as the project evolves.

The team's solution has a few key strengths. First, the solution extends the program that the client currently uses. This saves time and resources on the development side, and also eases transition for the client to the new system. Additionally, the system is intended for use across many facilities, which is beneficial for comparison and collaboration between facilities. The system will also improve the current minimal data analysis capabilities. With the new system, users may view standard analyses or download data to perform individualized analyses, allowing for maximum flexibility and ease of use.

One foreseeable weakness of the proposed system is the difficulty of adaptation for some research facilities. Current facilities have a wide variety of methods to collect their data. The methods range anywhere from Excel spreadsheets, to online entry, to paper and pencil. The team must design the system such that the learning curve is low, to allow as many facilities as possible to adapt to the new system. Although the solution will be as user-friendly as possible, the system may not be as simple as existing systems.

2.3 Development Process

The team will be working primarily in an agile environment and loosely following the design process below.

Figure 2.1: Development Process

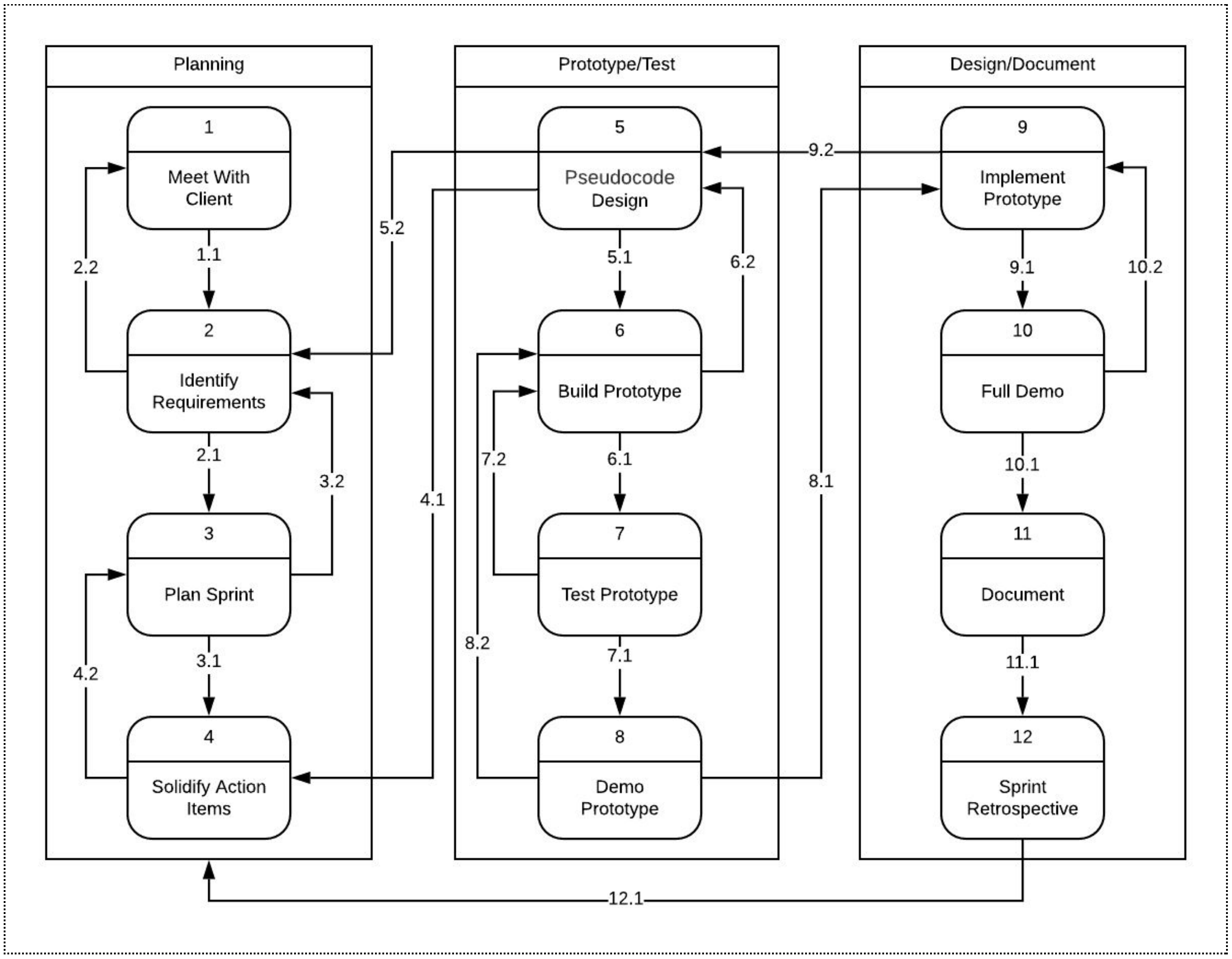


Table 2.1: Development Process Description

Step	**Plan**	**Description**
1	Meet With Client	Team will meet with the client to discuss planning documentation and determine any change in requirements. 1.1: If successful, identify the requirements for the project.
2	Identify Requirements	From client interaction we will determine requirements that need to be met for the next sprint. 2.1 If successful, begin planning sprint. 2.2 Else, meet with client again to clear up any confusion.
3	Plan Sprint	Work through requirements and determine what needs to be completed in the current sprint. 3.1 If successful, solidify action items. 3.2 Else, determine what is blocking the current work to be done and identify more requirements as needed.
4	Solidify Action Items	Determine the order in which sprint items need to be completed in so as not to block one another from being successful. 4.1 If successful, begin implementation 4.2 Else, review sprint items and determine what was missed if anything.
5	Pseudocode Design	As a team, pseudocode project so everyone has a clear picture of the design for this sprint. This will mitigate problems concerning overall design and ensuring pieces work together. 5.1 If successful, begin building a prototype. 5.2 Else, double check that no requirements were missed.
6	Build Prototype	Build a loosely coupled design that uses mock data to populate the necessary fields of the UI and design a backend that constructs objects the front-end service will consume. 6.1 If successful, test the prototype 6.2 Else, return to the pseudocode design phase and check that no pieces were missed in design or that a team member did not misunderstand the overall picture.
7	Test Prototype	Test the features against the actions items determined in step 4 to ensure that design meets requirements set by the client. Regression test as needed to ensure nothing new has been broke the existing code base. 7.1 If successful, setup meeting with client and demo prototype. 7.2 Else, go back to step 6 or further and determine what was missed in the design and fix/implement.

8	Demo Prototype	Set up meeting with client and discuss requirements met in the current sprint. Client will give approval or will determine that requirements were not met. 8.1 If successful, fully implement the prototype 8.2 Else, go back to step 6 or further and determine where build went wrong.
9	Implement Prototype	Fully connect the front-end and back-end and test functionality together. Make minor adjustments as needed and ensure all pieces are ready for production (client-side) testing. 9.1 If successful, setup meeting and do a complete demo. 9.2 Else, go back to step 5 to address any defects in implementation to ensure no pieces in the design were missed.
10	Full Demo	Run through a complete demo of product with client and do some testing with them. Get final approval and release for client use if applicable. 10.1 If successful, document process. 10.2 Else, fix implementation until client gives sign off.
11	Document	Document the process for how to use the system and complete any necessary reports. 11.1 Once completed, move onto sprint retrospective.
12	Sprint Retrospective	Meet as a team and discuss what went well and what did not go well. Discuss process improvements to effectively use our time better and other items in question or of concern. 12.1 Once completed, take new information into the next planning phase and begin iteration 2 or new project.

2.4 Design Plan

The application must be on an online platform so that other garden facilities can upload and view their pupae emergence data.

- Users should have a way to mass import data.
- Users should be able to mass import data from a variety of formats that other gardens facilities are using.
- Data entry should be easy for the user.
- Users should be able to modify their data post-entry to fix errors.
- A stretch goal is to support data entry from mobile devices.

The application must use a database to store, manage, and maintain the pupae emergence data.

- Data measuring should be cut off after the butterfly emerges.
- The database should have some built-in cleaning and error catching.

The application interface must have some built-in pre-determined statistical analysis of the data.

- Visualizations would be nice for users.
- Users should be able to visualize data on a per species basis.

The application must have an interface to extract relevant data from the database in an easy to use format for additional analysis using JMP or R.

- It would be helpful for users to have some documentation on how to use the interface.

The application interface must allow other garden facilities to compare their data against the group, but they should not be able to see other garden facilities individual data.

- Users should be able to upload, view, and compare data in real-time.

Based on the design requirements, the team plans to divide up the application into the following modules:

Figure 2.2: Module Diagram

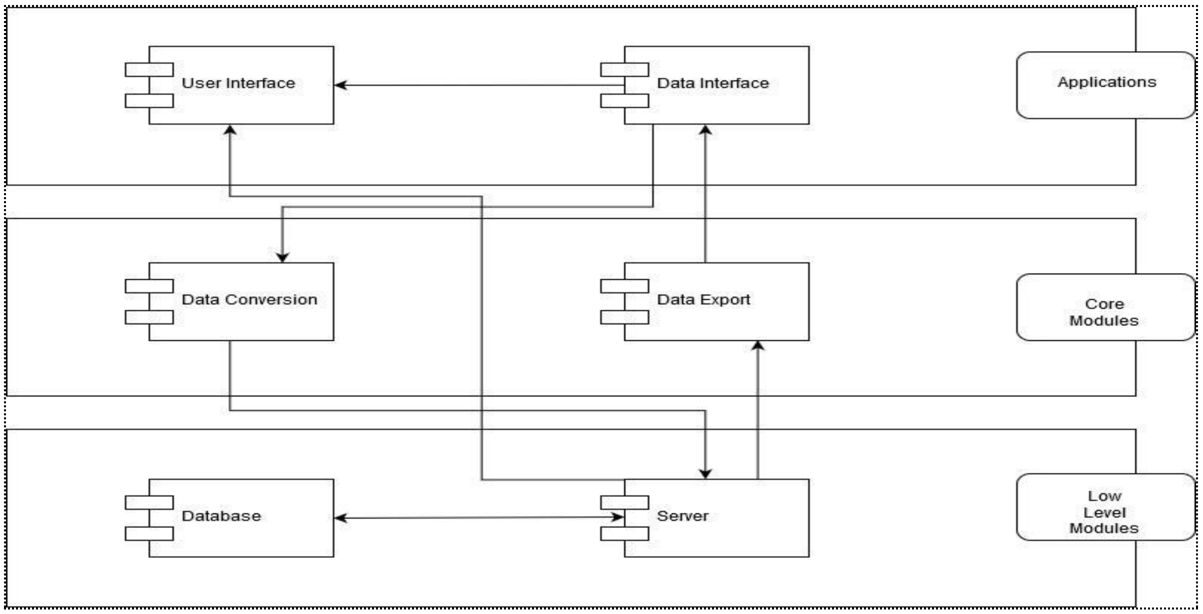


Table 2.2: Module Description

ID	**Module**	**Module Description**
M1	User Interface	Receives and displays data.
M2	Data Conversion Module	Converts data entered by the user into the proper format for the database.
M3	Database	Stores data, sends requested data to other modules.
M4	Server Module	Hosts the platform and acts as an interface between the database and the front end.
M5	Data Analysis Module	Uses JMP or R to do predetermined analytics of the data.
M6	Data Export Module	Exports data for analysis in JMP or R, with a potential capability to export to other storage formats like Excel.

3. Statement of Work

3.1 Previous Work And Literature

The team is creating an online platform where butterfly facilities around the world can submit files containing butterfly data. Data must be obtained from the file in order to return some statistical analysis on how a facility could improve their process to maximize the turnout of butterflies.

The team is developing an entirely new project for Remain Gardens. There is a product that already exists, which is the one they are currently using. That produce is limited to the collection of data from Reiman Gardens, and cannot accept data from other facilities. In the new product, other gardens will be able to send their files for storage and analysis. The team will be starting this project from scratch.

There are butterfly monitoring networks. These networks collect data regarding the populations of butterflies and their species. The data collected for this project is related to the environments that produce butterflies. This product is the only known product to collect data from multiple facilities around the world.

3.2 Technology Considerations

Table 3.1: Frontend Technologies

Technology	Strengths	Weaknesses
HTML	● Easier for another group to be able to pick after project is done ● Small learning curve	● Does not provide out of the box styling the way a framework does ● No state management
Angular	● Older and more established technology	● Has large learning curve compared to other frontend frameworks
React	● Provides reactive updating of data entry ● Widely used and modern ● Has state management system	● State management system, Redux, can be difficult to understand ● Not as easy for another group to pick up work later ● Has moderate learning curve
Vue	● Provides reactive updating of data entry ● Modern and raising technology ● Has state management system ● Has a CSS framework that allows for easy styling ● Simple to use and learn making it easier for another group to pick up work	● Not as easy for another group to pick up work later ● Newer and not as established

Table 3.2: Backend Technologies

Technology	Strength	Weakness
Node	● Could be used for frontend as well ● Highly scalable	● APIs are unstable ● Doesn't support multithreading ● Limited experience amongst team members
Python	● Could match language used for data analysis ● Easy to learn	● Limited experience amongst team members
MongoDB	● Easy to learn and use	● Other databases may work better with frontend

Table 3.3: Data Analytics Technologies

Technology	Strength	Weakness
R	● Data analytics lead is very familiar with it ● Open source ● Easy to create visualizations ● Lots of online resources	● Very specific to data analytics ● Only known by 1 team member ● Client hasn't used
JMP	● Very user friendly ● Client has previously utilized and has existing outputs	● Difficult to extend ● Not easy to automate ● Potentially problematic to integrate into website ● Not open source (requires license)
Python	● Up-and-coming data analytics language ● Easy to integrate with website ● Could match back-end language ● Lots of online resources	● Client hasn't used ● Team is less familiar with visualizing data in it
Firebase	● Modern backend as a service product, has matured since beta over the past few years ● Best cost/data ratio compared to most other popular options ● Supports server side node functions ● Built in analytics	● May run into roadblocks with data analysis ● Possibly have an issue with migration if a different back end is used in future

3.3 Task Decomposition

Using the following requirements determined for the project the design will be broken down into several tasks.

- Ability to enter data into a UI
- Update and remove data entered into the system
- Provides automatic error catching for cleaning data
- Provide perform data analysis on all entered data
- Create necessary end of year reports
- Update in real time
- Provide metrics to other suppliers on how their pupae are doing against other suppliers as a whole.
- Visualize data sets per species of pupa.
- Ability to export data for use in JMP or R.
- Extra: Documentation on how the system works
- Extra: Make it mobile browser friendly

Table 3.4: Task Decomposition

ID	Description	Dependent ID(s)
T0	Establish requirements	
T1	Setup server	
T2	Setup database	T0, T1
T3	Create UI for data entry (manual entry and CSV file upload)	T0
T4	Ability to enter, update, and remove data in database	T2, T3
T5	Catch data entry errors through the UI	T4
T6	Catch data entry errors server-side (bell curve on data?)	T4
T7	Basic data analysis	T4
T8	Export data for use in JMP and R	T4
T9	Display data analysis per species on the UI	T7
T10	Display data analysis for general information on the UI	T7
T11	Setup authorization system for outside Reiman Gardens	T1, T2
T12	Generate end of year reports	T7
T13	Historical data analysis	T7
T14	Display historical data to the UI	T12
T15	Display data to outside suppliers (historical, species, etc…)	T8, T9, T10, T13, T14

T16	How to use manual	T15
T17	Make it mobile friendly for data entry	T15

3.4 Possible Risks And Risk Management

Table 3.5: Risk Management

Error Type	Possible Error	Error Mitigation Plan
Cost	**N/A**	**N/A**
Materials	**N/A**	**N/A**
Equipment	**N/A**	**N/A**
Knowledge Area	● Depending on the technologies chosen, team experience will vary. ● Collective data analysis knowledge is limited to one person.	● Ask team advisor and professor for resources they are aware of that may help. ● Work closely with advisor as she has extensive data analytics knowledge.
Accuracy Issues	● Project is heavily dependent on data accuracy. ● Team members have no background in entomology, creating potential difficulties in catching inaccurate data.	● Obtain test data from client and actual values to run data against for accuracy. ● Test early and test often.
Data Breach/Loss	● Team has limited experience in data security.	● Research options and find resources. ● Create a backup system.

3.5 Project Proposed Milestones and Evaluation Criteria

Milestone 1: All services are communicating with one another (i.e. frontend, backend, server, and database).
Milestone 2: Data entry can be seen in the database and the data can be retrieved and analysis can be performed on data.
Milestone 3: General data and data per species is available and visualized on the UI.
Milestone 4: Project is fully deployed for use by Reiman Gardens.
Milestone 5: Outside user authentication is achieved and data can be separated by supplier.
Milestone 6: Historical, general data, and basic statistics can be seen by the UI for outside suppliers.
Milestone 7: Product is being used by outside suppliers and manual has been completed.
Milestone 8: Mobile friendly features have been enabled.

3.6 Project Tracking Procedures

The team will track progress through a kanban board and assign tickets for tasks to be completed. The kanban board will be supplemented with a design flow diagram that will track progress. Together, these tools will assist the team in ensuring completion of the project by May 2020. To ensure adequate progress is made and milestones are met, the team will meet once a week to update each other on progress.

3.7 Expected Results and Validation

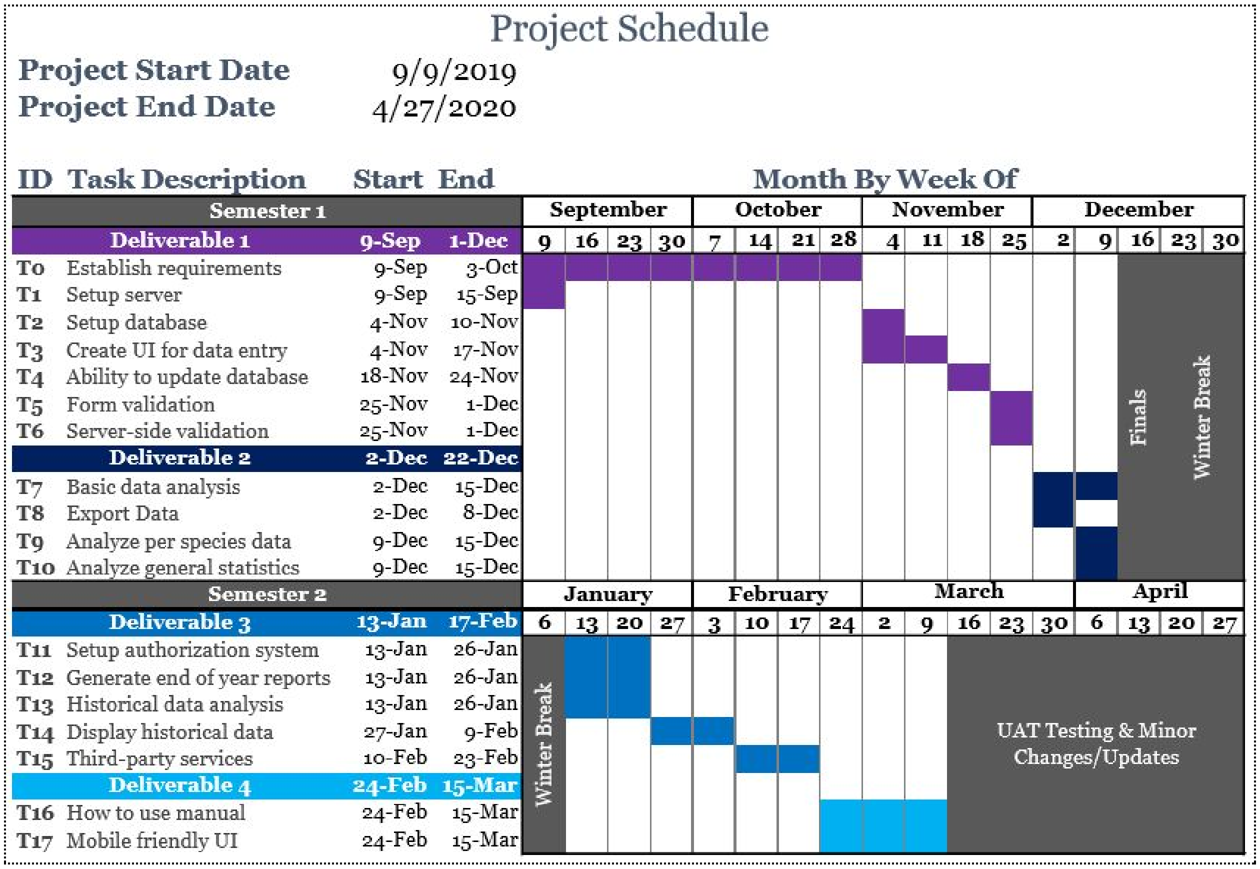
The desired outcome will be a UI that takes in various data points of pupa from a user. The data will then be used to perform an analysis on the various points of data that caused the pupa to successfully or unsuccessfully emerge. The system will be used to share this data with other facilities and contribute to a more successful emergence environment per species of pupa. Data will be available for export for deeper analysis with other systems if needed. The system will be able to validate and clean data to ensure that data is more accurate and mitigate the error of bad data points that can result from pen and paper. The system will be able to generate end of year reports and other required documents needed by facilities.

To validate that the solution works, the team will perform extensive user acceptance testing with the client before the solution is released.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Figure 4.1: Project Timeline



Project Schedule

Project Start Date	9/9/2019
Project End Date	4/27/2020

ID	Task Description	Start	End
	Semester 1		
	Deliverable 1	**9-Sep**	**1-Dec**
T0	Establish requirements	9-Sep	3-Oct
T1	Setup server	9-Sep	15-Sep
T2	Setup database	4-Nov	10-Nov
T3	Create UI for data entry	4-Nov	17-Nov
T4	Ability to update database	18-Nov	24-Nov
T5	Form validation	25-Nov	1-Dec
T6	Server-side validation	25-Nov	1-Dec
	Deliverable 2	**2-Dec**	**22-Dec**
T7	Basic data analysis	2-Dec	15-Dec
T8	Export Data	2-Dec	8-Dec
T9	Analyze per species data	9-Dec	15-Dec
T10	Analyze general statistics	9-Dec	15-Dec
	Semester 2		
	Deliverable 3	**13-Jan**	**17-Feb**
T11	Setup authorization system	13-Jan	26-Jan
T12	Generate end of year reports	13-Jan	26-Jan
T13	Historical data analysis	13-Jan	26-Jan
T14	Display historical data	27-Jan	9-Feb
T15	Third-party services	10-Feb	23-Feb
	Deliverable 4	**24-Feb**	**15-Mar**
T16	How to use manual	24-Feb	15-Mar
T17	Mobile friendly UI	24-Feb	15-Mar

***See table 3.4 for related task ID descriptions in more detail**

The team has divided the project into four deliverables and set the goal of having the first two deliverables done by winter break. The team thinks this goal is very achievable, and sets the project on a good pace to have the entire project finished by May 2020. The team is also aiming to get some of deliverable 3 done before winter break, because this deliverable is anticipated to be the most time-consuming deliverable.

The team is prepared to adjust the schedule as needed if deliverables take longer or shorter than expected. If any of the first three deliverables take longer than expected, we can push back and scale back deliverable 4, since the core functionality of the application will be finished in deliverable 3. As the project is currently progressing, the team may complete deliverable 3 earlier than expected. In this case, the team plans to expand non-functional requirements as part of deliverable 4.

4.2 Feasibility Assessment

This project has very clear functional requirements, but it also has many opportunities to implement additional functionalities. At minimum, the final product needs to have a website where a user can log in, enter pupae data, and obtain reports and statistics regarding the data they entered. This will require communication between front and back end, as well as communication with a reliable database.

The team views these functional requirements as very feasible, and plans to have the most important functionality implemented by mid-March. This timeline will give the team the remainder of the spring semester to implement other useful functionalities. This timeline also ensures the delivery of the most important functionality by building in catch-up time if needed.

The client has given the team flexibility to choose priority of many of the non-functional requirements based on available time. With this flexibility, the team believes the implementation of at least some features is feasible.

Foreseen Challenges:

Data Collection: As of now, different gardens collect a wide variety of data in many different formats. This can pose a potential challenge, since the data analysis portion of the project relies on consistent explanatory variables. Additionally, this may cause difficulty from a database perspective.

Security and Privacy: Each garden should expect their data to remain secure in the system. There should be varying levels of permissions to allow researchers (client) to view certain data that other users cannot.

Mobile Friendly: The client would like the website to work well on tablets and phones, which may pose a technical challenge to implement.

Minimal Maintenance: The client would like to use this product long after the team concludes working on it. Therefore, the team must build the product with durability and sustainability in mind. The client may not have the resources to update the code if something breaks or changes after the team leaves.

4.3 Personnel Effort Requirements

Table 4.1: Personal Effort

Task ID	Estimate of Effort
T0	**Establish requirements** - This is the most essential aspect of the project and it took 8 weeks to complete after several meetings with client and advisor.
T1	**Setup server using node js** - Node js has good documentation on setting up servers, which allowed the team to establish a local server for testing purposes within two hours.
T2	**Setup database using google firebase** - Since the team has limited experience with Firebase, the initial learning curve and set up should take approximately 1 week.
T3	**Create UI for data entry (manual entry and CSV file upload)** - This part of the project involves writing critical front-end code. This task will be ongoing for most for most of the next semester, but the initial setup

	should take approximately 2 weeks.
T4	**Ability to enter, update, and remove data in database** - This portion of the project involves making a basic CRUD web application, which should take approximately 1 week.
T5	**Catch data entry errors through the UI** - Every field must be validated in the front-end. This functionality should take about 1-3 weeks of concentrated effort.
T6	**Catch data entry errors server-side** - The data will also be validated using server-side checks by node and the db. This will be completed in conjunction with the front-end validation, which should take about 1-3 weeks.
T7	**Basic data analysis** - In order to provide the necessary statistics and visualizations, the data will be analyzed with a variety of models. The data analysis must communicate with the database. This can be completed alongside front and back-end development and should take 2-5 weeks.
T8	**Export data for use in JMP and R** - Once the communication between the data analysis portion and the database is developed, functionality to export data will be implemented. Data must be able to be exported in file formats accessible with JMP or R. This portion of the project should take approximately 1 week.
T9	**Display data analysis per species on the UI** - This task combines front-end and data analysis portions of the project. After data analysis has been completed, this task should take 2-3 weeks.
T10	**Display data analysis for general information on the UI** - Pages will be generated using data fields that were generated for individual species. To implement should take less than 2 weeks.
T11	**Setup authorization system for outside Reiman Gardens** - This task requires the communication between front-end, back-end, and database. Using existing authorization frameworks, this aspect of the project should take 1 week.
T12	**Generate end of year reports** - This task is an extension of generating reports and will be handled alongside it. It will take 1-2 weeks to ensure data is properly set and displayed.
T13	**Historical data analysis** - After preliminary analysis is completed, historical analysis should simply extend existing analysis. Historical data analysis will require the parsing of more data and should take approximately 2 weeks.
T14	**Display historical data to the UI** - This task utilizes the data from the previous task. To appropriately display mass amounts of data in a visually pleasing way could take up to 2 weeks.
T15	**Display data to outside suppliers (historical, species, etc…)** - In conjunction with the authorization system, the team will implement access tokens to allow data to be displayed based on authorization credentials. This task should take approximately 1 week.
T16	**How to use manual** - The manual will be a document regarding design and upkeep of the product. Additionally, it should provide instructions for use of the product. It should be created within a 3 week span while coordinating with the client.
T17	**Make it mobile friendly for data entry** - Throughout the development process, the team will be using

	materialize css framework which will allow easy extensibility to a mobile friendly version. Materialize css allows simultaneous development for all screen sizes. Therefore, this task will take no additional development time, but will be implemented alongside other tasks.

4.4 Other Resource Requirements

For physical resources, the team needs working computers to develop the product on. Software requirements resources would include a code repository and an IDE to develop in. Knowledge resources will mostly consist of internet sources, as well as occasional assistance and direction from the client and faculty advisor.

4.5 Financial Requirements

Our team has no financial requirements to meet for this project.

5. Testing and Implementation

5.1 Interface Specifications

The interface for the software product will be based on the same guidelines as of the existing software interface of the working system at Reiman Gardens. It will have input operations for Shipment Date, Arrival Date, Supplier, Species, Common Name, Number Received, Emerged in Transit, Damaged in Transit, Diseased, Parasites, Poor Emergence, No Emergence and Released. There will be an additional input interface for images in the details page of each individual emergence.

5.2 Hardware and software

5.2.1 Hardware

A system capable of running and maintaining node.js server. The minimum requirements for which include the following:

- 64-bit architecture
- Kernel version 3.10 or higher (On linux)
- 4 or more CPU cores
- A minimum of 16 GB or RAM
- A minimum of 25 GB of free disk space
- Three open ports for inbound TCP traffic: 8800, 8080 and 8081

It can also be assumed that the system will have to process HTTP and HTTPS traffic.(npmjs.com)

Additionally, the team will be using google's firebase which will be a hardware system in the cloud.

5.2.2 Software Stack

The major viewable interface, i.e the front-end will be based on vue.js and it will run on a node.js back-end. The team will be doing unit tests to check for code-completion and conformity. Google's firebase will be utilized for the database. The team will have the standard tests for schema, collections and triggers. The team will be using complex queries to check data integrity and consistency.

5.2.3 Types of testing

For functional testing, the following testing will be completed:

1. Unit Testing
2. Interface Testing
3. Integration Testing

For Non-functional testing, the following testing will be completed:

1. Performance Testing
2. Load Testing
3. Stress Testing

5.3 Functional Testing

5.3.1 Unit Testing

Each component will have a set of functions for completeness, efficiency and conformity. There are various tasks in the software project and each one of them has a function or class associated with it. There will also be multiple mock data classes for each input fields. The unit test functions will go through each interface, class and function and will also have checks for UI/UX consistency. The tests will be written in the form of an API so that it can be ported to a different project.

5.3.2 Interface Testing

The interaction with the user is one of the most important aspects of any software application. As mentioned above, the interface will be modeled on the already existing system to ensure familiarity and reduce the steepness of the learning curve. The team will also be testing the software interface based on the 'Nielsen and Molich's 10 User Interface Design Guidelines'. In addition to software testing, manual tests with the user interface will be completed with the client to get their feedback on user experience.

5.3.3 Integration Testing

The system will include various APIs for authentication, database, UI features etc. In order to ensure proper functioning of each of these various components, the team will test their stability with incremental testing. This is a bottom-up approach to testing in which one component will be added, then it's reliability will be tested, then another component will be added, and so on. This cycle will continue until all the APIs and components needed for the system to work correctly have been added and tested.

5.4 Non-Functional Testing

The team will be doing testing on factors like usability, reliability and performance. As the size of the database increases, it might face some issues with speed and reliability. The team will be testing those potential circumstances with mock data and extensive user usage reports. These will also include stress test and peak hour load tests.

5.5 Process

The team will start off by a bottom-up approach to integration testing. This will involve adding a component then checking it's usability and reliability and repeating it until all components have been added. For the testing process, team members will collaboratively write unit tests on each function, class, and interface. The purpose of unit tests is to ensure proper functioning of the code, i.e. code does what it is intended to do. When these tests successfully return true the team will move on to interface testing. This part of the process includes tests with the hands-on users of the product. Users will be asked to go through the software interface and report the hands-on usability of it. Additionally, the team will be going through each interfaceable component and match it with 'Neilsen and Molich's 10 Interface Design Guidelines'. After testing is completed on the functioning aspects of the application, the focus will move on to the long-term usability features that involve non-functional testing. For stress and performance testing, the team will inject the database with large amounts of mock data, comparable to what is expected to see in the final product. Along with performance testing the team will create a situation where many users are trying to access the same database to test for load testing on the large google firebase repository.

5.6 Results

The results are expected to very standard as the complexity of the project does not exceed the threshold which team members are used to working on. Many frameworks are available that do unit testing on node.js based projects. These frameworks may report the runtime, success percentage, and various other statistics, depending on how the team chooses to design them. The team expects to see failures in the database integration of the project and in the stress and performance tests. For the purposes of research the team has only ventured into things which are known concerns, however, there may be unexpected issues. The biggest challenge currently is meeting the requirements of the desired product and getting it to sufficiently work with google firebase.

6. Closing Material

6.1 Conclusion

Thus far the team has determined the technology stack to be used and created a fully connected initial project. The stack will include Vue, Vuetify, Node, Firebase, and Python. Vue will be the frontend framework used to structure the project and architect the solution. Vuetify will be the material design system to ensure all aspects of the website are fluid and have the same consistent look. Firebase will house all of the data for the project and accessed via a Node backend and analyzed with scripts written in Python.

According to the requirements outlined in section 1.4 the following plan of action has been set to achieve the goals of these requirements.

ID	Requirement	Plan of Action
R1	Data entry interface for the data, with a focus on ease of use	The initial project has basic layout using the aforementioned technology stack. The team will continue to push forward with implementing the elements of the form that are required by the client.
R2	Ability to update and remove data	The ability to add and remove data from the database has been implemented using VueFire. This was the optimal solution as it made connectivity to the database seamless while integrating Google FireStore with Vue.js.
R3	Provide automatic error catching for cleaning data	An approach to error checking that has been proposed will be using Yup validation. It has easy integration with form elements and type checking built into it.
R4	Perform data analysis on all entered data	The team will focus on a minimal viable analysis first, ensuring the client can have a basic model for all data entered into the system. Following the basic model, the team will construct more complex models to assist in the client's research.
R5	System to create end of year reports	Reporting requirements still need to be specified. The current approach will be to automate the report generation based off the data that is collected and then saved into any format that is specified by the reporting requirements. This approach will be the best for the client as it removes any need for manual data collection.
R6	Update in nearly real time	Vue.js is being used as the frontend framework to provide this functionality. It is a reactive framework that will update the UI in real time.
R7	Provide metrics to other suppliers on how pupae are doing against other suppliers as a whole	Steps have been taken into looking at a proper login validation. Some approaches discussed have been using basic Firebase and Vue validation and using Okta to provide the full service authentication. Further discussions with the client have to occur concerning if cost will be a concern for what level of security they are expecting.
R8	Visualize data sets per species of pupa	The data will be analyzed with Python and served back to the UI. The data will then be displayed in a graphical format to represent the data in a way that is easy to use and

		understandable. Meetings with the client will be necessary to determine the best way to present the information that makes it both easy to use and understand.
R9	Ability to export data for use in JMP or R	The data collected will have the ability to be exported as a CSV file for use with R or JMP. Both JMP and R can easily import CSV files, and this will give the client optimal flexibility for future research and modeling.
R10	Extra: Documentation on how the system works	This requirement will be integrated at the end of the design process after the basic structure is complete and finalized. The team has discussed ideas behind implementation and the most viable approach would be to create a downloadable document outlining the use. Another more time intensive approach would be to integrate the documentation into the website directly. Client feedback and team resources concerning time will be a factor into the progression of this decision.
R11	Extra: Make it mobile browser friendly	Vue.js has device settings that make it easy to adjust for each device size that the website could be viewed on. The site does not need to have a seperate mobile component, but it will be scaled to be viewed on a mobile device for data form entry.

6.2 References

[1] Evan You, “The Progressive JavaScript Framework,” Evan You, 2014 - 2019. [Online]. Available: https://vuejs.org/

[2] Google, “Firebase helps mobile and web app teams succeed,” Google, 2014 - 2019. [Online]. Available: https://firebase.google.com/

[3] Node.js Foundation, “Node JS,” Joyent, Inc, 2019. [Online]. Available: https://nodejs.org/

[4] npm, Inc., “Build Amazing Things,” npm, Inc., 2019. [Online]. Available: https://www.npmjs.com/

[5] Vuetify, LLC, “Vue Material Design Component Framework,” Vuetify, LLC, 2016 - 2019. [Online]. Available: https://vuetifyjs.com/