Unified Butterfly Tracker

COM S 402 Senior Design Document

Team 4
Marble Island Butterfly Project
Nathan Brockman, Mario Pantaleo, Jake Garnett, William Schulte,
Spencer Kane
Team Members/Roles
cssdgroup4f23@iowastate.onmicrosoft.com
Team Website

Revised: Date/Version

EXECUTIVE SUMMARY

Development Standards & Practices Used

We used Scrum with two week sprints for this project. We organized our tasks using the Gitlab issue board. The web application is being hosted on a server for long term use and maintenance.

Summary of Requirements

- Data Tracking for Butterflies
    - Ability to enter data from mobile or computer in an engaging and efficient way
    - Ability to save specific data based on what type of form was filled out
    - Ability to review saved data and update it
    - Ability to see data saved from others in the same group
    - Graphics of recent data added
- Different groups for different Facilities
    - Ability to create groups to share data
    - Ability to request access to a group
    - Separation of databases so that group information is safe from users outside of group
    - Ability to create facility admin and global admin

Applicable Courses from Iowa State University Curriculum

- COM S 127
- COM S 228
- COM S 309
- COM S 311
- COM S 319
- COM S 362
- COM S 363

New Skills/Knowledge acquired that was not taught in courses

- Knowledge of how to use Django
- More in depth knowledge of Python
- More in depth knowledge of Figma
- How to use PyCharms
- Knowledge of how to communicate with clients
- More in depth knowledge of how to communicate with team members
- More experience working with a team

Table of Contents

1 Introduction <NOT TO EXCEED TWO PAGES> 3

1.1 Acknowledgement 3
1.2 Problem and Project Statement 3
1.3 Operational Environment 3
1.4 Requirements 4
1.5 Design Assumptions and Limitations 4

2 Architectural Diagram << ONE TO TWO PAGES MAX> 5

3 Components <FOUR PAGES MAX> 6

3.1 Component-1 (builds into ONE executable/process) 6
3.1.1 Module 1 6
3.1.2 Module 2 <REPEAT FOR AS MANY MODULES THE COMPONENT HAS> 6
3.2 Component -2 <Repeat for as many Components that THE SYSTEM HAS> 6

4 Data Decomposition <THREE PAGES MAX> 7

5 Detailed Design 8

6 Design Rationale 9

6.1 Design Issues 9
6.2 <Issue 1> 9
6.2.1 Description 9
6.2.2 Factors affecting Issue 9
6.2.3 9
6.2.4 Alternatives and their pros and cons 9
6.2.5 9
6.2.6 Resolution of Issue 9
6.2.7 9
6.3 <Issue 1> 9
6.3.1 Description 9
6.3.2 Factors affecting Issue 10
6.3.3 10
6.3.4 Alternatives and their pros and cons 10
6.3.5 10
6.3.6 Resolution of Issue 10
6.3.7 10

1 Introduction <NOT TO EXCEED TWO PAGES>

1.1 Acknowledgement

We thank the Reiman Gardens and the Marble Island Butterfly project for their continued support during the design process, and for providing us with example data to aid in the design process. We also thank the Oregon Zoo and the Woodland Park Zoo for their husbandry data which aided us in our database design.

1.2 Problem and Project Statement

General Problem Statement: The data entry process for many butterfly husbandry facilities is slow, monotonous, and inefficient. Many facilities track butterfly data on pen and paper which can lead to messy data and organization. They all record different data on facility specific data forms. It is difficult for facilities to collaborate due to the lack of standardized data collection, records, and availability.

General Solution Approach: The proposed solution is to create a web app that will record butterfly husbandry data. By making it a web app, users will be able to enter data from a lab or in the field. The data entry process will be efficient, fast, and standardized. The web app will also attempt to take advantage of sensors already used by the facility to further streamline data collection. The web app will gamify data entry to make it less boring. It will also encourage collaboration by allowing facilities to optionally share data with each other. If a facility chooses to share data, other facilities will be able to see the shared data, graphs, and statistics.

1.3 Operational Environment

Our project's end state will merely be a web-server, with no physical components aside from the main server that users will connect to with their browser. Ideally the physical server itself will reside within a rack that will be maintained by an IT staff. There are no components that any user needs to bring into their facility to make our software work, and thus it should be perfectly fine running off in a server farm somewhere.

1.4 Requirements

Summarize Functional and Non-functional requirements for your project (from your Requirements document). KEEP IT BRIEF)

Functional requirements:

- Data Entry and Storage: Data entry being simple and streamlined will be essential to the web application. The data will need to be stored efficiently and securely so it can be retrieved quickly.
- Data Sharing with other facilities: Sharing data will allow facilities to easily access and view data that is shared with them. This will allow facilities to collaborate and work together.
- Facility creation and management: Facility creation and management will be needed for new facilities joining the service and choosing what data to share.
- User creation and permissions: User creation will allow new users to join the service and permissions will determine what level of control the account will have.

Non-functional requirements:

- Gamify data entry: Gamifying data entry will make the data entry process less monotonous.
- Automate data entry with sensors: Automating
- Direct Messaging to other users

1.5 Design Assumptions and Limitations

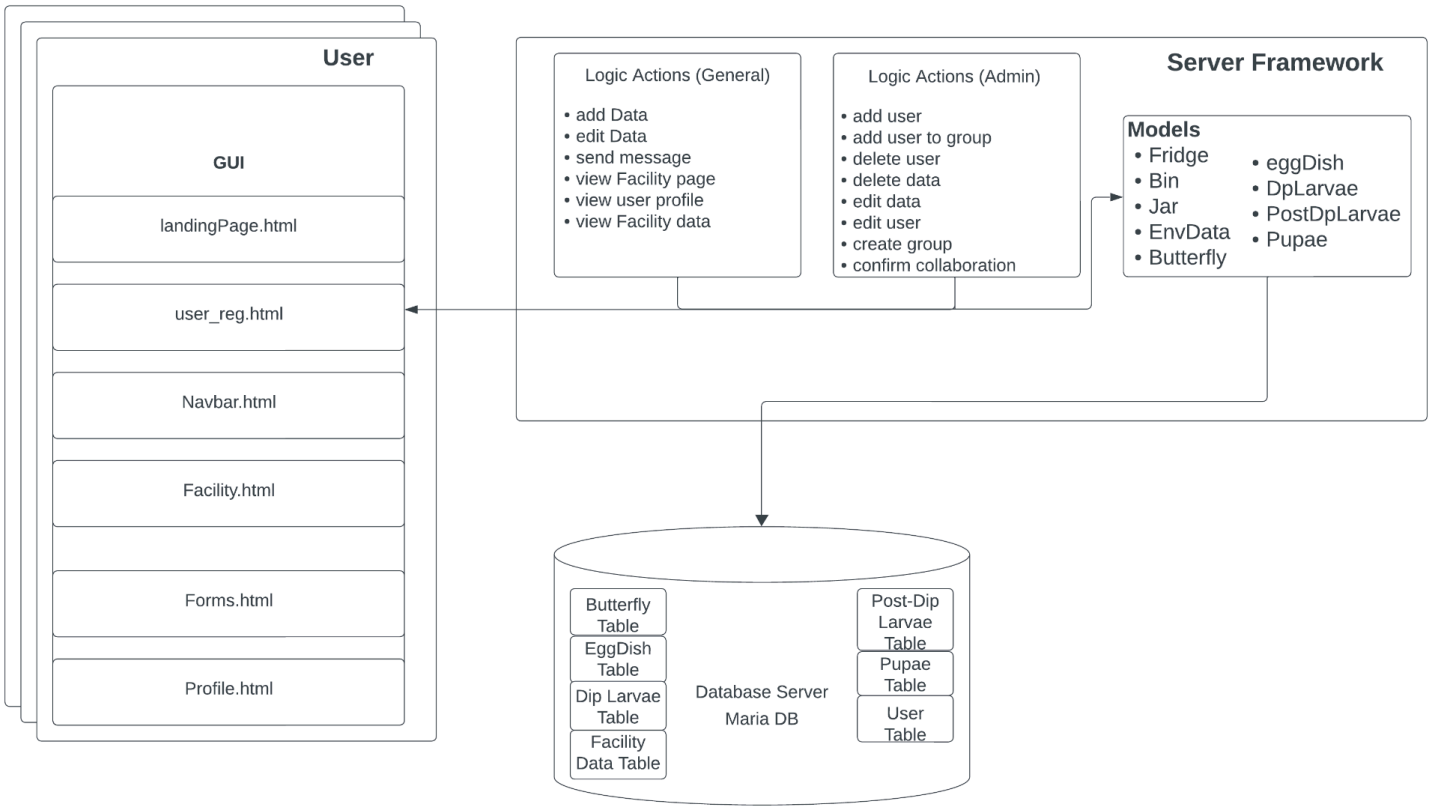
Assumptions:

- The web application will need to support multiple facilities, each with one to three users entering data concurrently.
- Background is red and white to match the Iowa State color scheme.
- Django framework will need the following models: Fridge, Bin, Jar, EnvData, Butterfly, eggDish, DpLarvae, PostDpLarvae, and Pupae.
- Database will need the following tables: Butterfly, EggDish, DipLarvae, Facility, Post-Dip Larvae, Pupae, and User.

Limitations:

- Unable to access phone hardware for automated data collection. Facilities have their own sensors that can accomplish something similar.
- Facility sensor data collection will be limited by sensors available.
- The number of facilities and concurrent users will be limited by server hardware.

2 Architectural Diagram



3 Components

3.1 Component-1: Data Entry (builds into ONE executable/process)

3.1.1 Module 1: HTML

- Description: Responsible for defining the structure and layout of the data entry forms. Includes HTML markup creation for input fields, buttons, etc.
- States: Will have different states based on the form
- Dependencies: it depends on Forms and Views for generation and rending of the content
- Parts:
    - HTML Markup: Defines the structure of the data entry forms.
    - Static Elements: Non-changing elements like labels, headers, and footers.

3.1.2 Module 2: Forms

- Description: Handles the generation of the forms based on predefined templates or configurations. It interacts with the HTML module to inject dynamic content into the forms.
- States: Dynamic states based on the form being displayed or edited
- Dependencies: Depends on the HTML for rendering the forms and the Models module for retrieving form configurations.
- Parts:
    - Form Templates: Predefined structures for different types of data entry forms.
    - Dynamic Form Generation: Logic for populating forms with dynamic content.
    - Event Handlers: Functions that respond to user interactions within the forms.

3.1.3 Module 3: View

- Description: This module is responsible for rendering dynamic content on the user interface. It interacts with the forms module to display dynamically generated forms and updates the UI based on user input.
- States: UI states based on the content being displayed or user interactions.
- Dependencies: Depends on the forms module for obtaining dynamic content and the models module for fetching data to display
- Parts:
    - Dynamic Content Rendering: Logic for displaying dynamically generated content.
    - User Interface Updates: Functions for updating the UI based on user actions.

3.1.4 Module 4: Models

- Description: A model contains essential fields corresponding to the database for each of the data form entries.
- States: States related to data validation and processing.
- Dependencies: Depends on the DB module for database interaction and may depend on external libraries or plugins for specific functionalities.
- Parts:
    - Data Models: Representations of the data structure used in the application.
    - Data Validation Logic: Rules and checks to ensure data integrity.
    - Database Interaction: Functions for storing and retrieving data from the database.

3.1.5 Module 5: DB

- Description: The DB module is responsible for managing the connection to the database and executing database operations.
- States: Connection and transaction states
- Dependencies: Depends on the models module for data to be stored or retrieved.
- Parts:
    - Database Connection Management: Logic for establishing and maintaining a connection to the database.
    - Database Operations: Functions for executing SQL queries or commands.

3.2 Component -2: Facility

3.2.1 Module 1: View Facility

- Description: This module enables users to view information about a facility, including any graphs, data, or relevant content.
- States: UI States vary based on which facility you are able/allowed to view, it also depends on what specific content they wish to view such as viewing the data, graphs, etc.
- Dependencies: This module depends on the models or database interaction module to fetch facility information from the database.
- Parts:
    - Facility Information Display: A section or view where information about the facility is shown.
    - Data Display Components: Elements to present data and graphs associated with the facility.
    - User Interaction Controls: Buttons or options for users to navigate or interact with the facility's information.

3.2.2 Module 2: Create Facility

- Description: This module is responsible for allowing users to create a new facility within the system.
- States: States include the initial state (before creation) and the state during the facility creation process.
- Dependencies: It depends on the Models module for data validation and the Database Interaction module to store the newly created facility data.
- Parts:
    - Facility Creation Form: A user interface for inputting facility details, such as name, location, and description.
    - Admin Authorization: Logic to verify that the user initiating the creation is an administrator.
    - Data Validation: Ensures that the data entered during facility creation is valid and follows specific rules.

3.2.3 Module 3: Invite User to Facility

- Description: This module enables users to be invited to join an existing facility within the system by local admins of the facility.
- States: States include the initial state (before joining) and the state during the invitation process.
- Dependencies: It depends on the database for storing user invitations and user authentication to verify that the local admins have the authority to send out invites.
- Parts:
    - Invitation Management UI: A UI for initiating and managing invites.
    - User Authentication: Verification of the admin's authorization to send invites.

3.2.4 Module 4: Edit Facility

- Description: This module allows administrators to edit the details of a facility, such as its name, description, or any other relevant information.
- States: States may include the initial state (start of the editing process), user input, and confirmation of changes.
- Dependencies: It depends on the database for updating facility information and user authentication to ensure that only authorized admins of a facility can make changes.
- Parts:
    - Facility Editing Form: A user interface for making changes to facility details.
    - User Authentication: Verification of the admin's authorization to edit facility information.

3.3 Component -3: User Authentication

3.3.1 Module 1: Login

- Description:
- States:
- Dependencies:
- Parts:

3.3.2 Module 2: Register

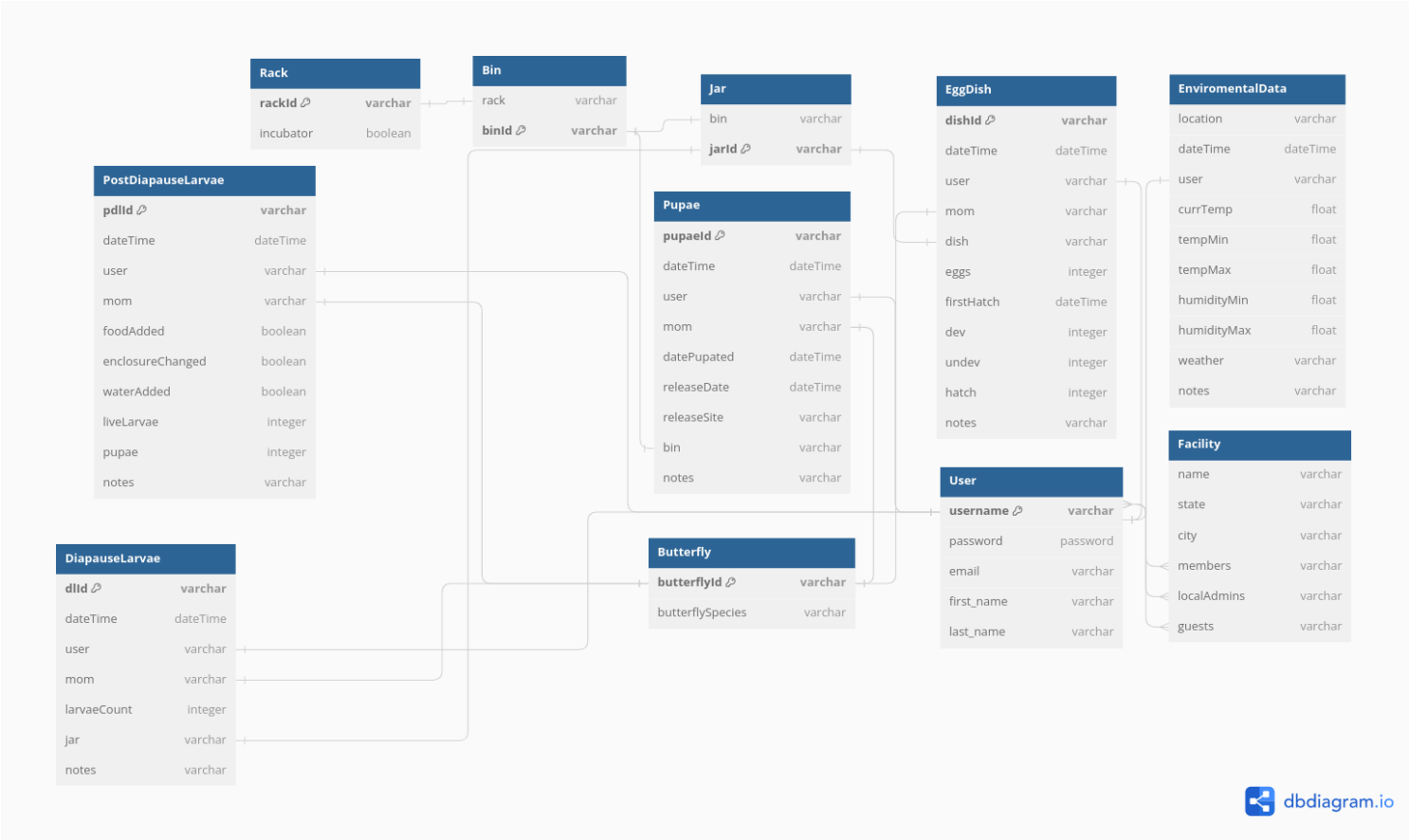
- Description:
- States:
- Dependencies:
- Parts:

3.3.3 Module 3:

- Description:
- States:
- Dependencies:
- Parts:

4 Data Decomposition

<Here give information on what persistent data your project stores and their relationships. Basically, describes files, tables, entity relationship etc. This is just entity-relationship diagram if you have a relational database.>



Most of the data being collected in each table pertains to the type of information that we saw being collected. The main relationships that we want to emphasize here would be that each entry has a user, for tracing who took the data. We also want to emphasize the relation to the mother butterfly as well, as every single stage in the process is also tracking the parent at every single stage of development.

5 Detailed Design

NOT REQUIRED <Java Docs to be used instead>

Give a link to your documentation folder here.

6 Design Rationale

<Follow instructions given separately; Note that I expect this to be a **major part** of the document>

6.1 Design Issues

- Limited access to the sensors on the phones for data collection
- The app must be cross platform with no major differences between any devices

6.1.1 Lack of phone sensor input

6.1.1.1 Description

Our project mentor, Nathan, really wanted to be able to let the users take environmental data straight from temperature and pressure sensors on the smartphones of the individuals recording the data. This is helpful for tracking any possible change that could be affecting these butterflies and how they grow so that we can better understand their species and help to conserve them.

6.1.1.2 Factors affecting Issue

The sensors would need to be on any mobile phone and accessible to the user to submit. As it would be, not all phones have these features, making it unreasonable to try to autofill information that is not attainable.

6.1.1.3 Alternatives and their pros and cons

One top alternative discussed has been to purchase a type of sensor that they already use, then access it via its software api to autofill the information based on what area they select and what information can be taken from the sensor in that area. A pro of this is that we could reliably automatically gather some information from the site without it being required to enter. A con of this would be that any facility that wants to use this would have to buy a sensor, likely one of the same brand so that its api calls would work.

6.1.1.4 Resolution of Issue

6.1.2 Gamification

6.1.3 Description

Nathan has requested that we make the data entry engaging such that we can convince possible users that it is worth their time. To this end, he suggested a gamification of the data entry so that it would appeal to users. However, he also wants it to be as efficient as possible.

6.1.4 Factors affecting Issue

Data entry is data entry, and our ideas about making it “fun” or more engaging will end up slowing down the rate of entry and thus lowering the efficiency.

6.1.5 Alternatives and their pros and cons

The alternative is to not try to gamify it in which it will be quite boring to enter the data. The only pro to this is that we can instead hasten data entry by using a previous form to autofill data or some similar template form.

6.1.6 Resolution of Issue

FEEL FREE TO ADD APPENDICES AS NEEDED. UPDATE TOC BEFORE SUBMITTING