Unified Butterfly Recorder

Design Document

Team 12
Client: Reiman Gardens
Advisor: Dr. Rover
Team Members/Roles
Grace Wigen-
Jeremy Marchesani-
Anthony Mazzie-
Zach Wingert-
Ryan McNally-
Timothy Ellis-

Team Email: sdmay23-12@iastate.edu
Team Website: https://sdmay23-12.sd.ece.iastate.edu/

Revised: 12/2/2022/ Version 2

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Summary of Requirements

List all requirements as bullet points in brief.

- UBR must allow users to perform a butterfly identification survey and send the completed survey data to a database for storage and retrieval.
- UBR must access GPS and other peripheral sensors on the device and include this in survey data.
- UBR must be able to export survey data to a third-party database, PollardBase, and also be able to export the data as a .csv file.
- UBR must be able to map out the route taken on the survey and identify points where they saw a butterfly
- UBR must have the ability to take a picture of a butterfly for future identification
- Able to add different sets of things (bees, trees, etc.)
- UBR must be able to store GPS and time stamp data without a connection to the internet.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

We have determined that some relevant courses are, SE309, SE319, and COMS363.

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

We have learned a lot more information about PWA's as this was something that we didn't have any exposure to in our classes. Another large piece of knowledge that we acquired was AWS. Only 1 member of our team had any knowledge of this and it is a critical part of our project, so we ended

up learning how powerful it actually was. One final piece of knowledge we learned about was React. In 319, we briefly went over React, but nothing to this extent.

Table of Contents

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

UBR: Unified Butterfly Recorder

PWA: Progressive Web App

AWS: Amazon Web Services

CI/CD: Continuous Integration and Deployment

1 Team

1.1 Team Members

Grace Wigen, Jeremy Marchesani, Anthony Mazzie, Zach Wingert, Ryan McNally, Timothy Ellis

1.2 Required Skill Sets for Your Project

AWS, React, PWA, Relational database

1.3 Skill Sets covered by the Team

AWS - Anthony Mazzie

Relational Database - Ryan McNally, Anthony Mazzie

React - Grace Wigen, Jeremy Marchesani, Zach Wingert

PWA - Timothy Ellis

1.4 Project Management Style Adopted by the team

Our team will be adopting a hybrid waterfall+agile approach for this project due to the fact that the requirements for this project are well laid out from the start.

1.5 Initial Project Management Roles

Team organization: Grace Wigen

Client Interaction: Tim

Individual Component Design: TBD

Testing: TBD

Research: TBD

2 Introduction

2.1 Problem Statement

Entomologists and researchers, specifically at Reiman Gardens, need a way to quickly and efficiently collect, store and transfer data gathered from their butterfly identification volunteers when they go for their survey walks. The problem with the current method of recording surveys on paper is that it

is slow and more difficult to manage, and the problem with the current UBR applications is that they are either not cross-platform or do not have longevity and break after an update. It is important to the staff at Reiman Gardens to be able to offer an app that can unify users under one platform while still providing the interesting features of the old app. This can be solved with cross-platform or web-based mobile applications.

2.2 Intended Users and Uses

The intended primary users of this application will be trained volunteers supporting the field of butterfly research by going on 'surveys', or walks where they take a predetermined route and count how many butterflies they find on the way, and what species they are. These users are characterized by their training, their personal interest in researching butterflies, and their willingness to learn. These users will be interested in an easy-to-learn tool that works quickly and reliably.

A subcategory of the primary users are distinguished by being users of the original Android application. They will want the new application to be easy to transition to.

Secondary users include people who want to use the surveying tool for their own purposes, and aren't necessarily affiliated with Reiman Gardens. They won't have any official training, and will hopefully use the customizable features of the app to go on their own surveys and record populations of animals or plants that they themselves are interested in. They will want the app to be customizable and easy to learn.

Researchers will care that the app exists because it will provide more data to them, as well as be able to transfer the data gathered directly from the app to their database. Butterfly identification volunteers care that this app exists because they are able to quickly and more easily collect data while on their walks.

2.3 Requirements & Constraints

Functional requirements:

1. UBR must allow users to perform a butterfly identification survey and send the completed survey data to a database for storage and retrieval.
2. UBR must access GPS and other peripheral sensors on the device and include this in survey data.
3. UBR must be able to export survey data to a third-party database, PollardBase, and also be able to export the data as a .csv file.
4. UBR must be able to map out the route taken on the survey and identify points where they saw a butterfly
5. UBR must have the ability to take a picture of a butterfly for future identification
6. Able to add different sets of things (bees, trees, etc.)
7. UBR must be able to store GPS and time stamp data without a connection to the internet.

Resource requirements:

1. UBR must be designed to minimize battery use on the user's mobile device.

2. Needs some sort of cellular signal (before submitting) and ideally has signaled the majority of the route.

Physical requirements:

1. UBR must be available from users' mobile devices, specifically Android and iOS phones.
2. Phones must have GPS and camera capabilities.

Aesthetic requirements:

1. UBR must be similar enough to the older Android UBR so that returning users can pick up the new system with little discomfort but also intuitive enough that new users can learn how to use it quickly
2. UBR must have a clean feel and be approved by our client

User experiential requirements:

1. Users must be able to tally the butterflies they see very quickly, ideally without even looking down if they spot a large group of the same butterfly.
2. Users must not be annoyed by using UBR.
3. Have a detailed GPS so users can easily see where they spotted butterflies
4. Must be faster than recording butterflies using the old paper method.

Economic/market requirements:

1. UBR must be free to use.
2. UBR must be built in a way that will minimize maintenance, as there is very little budget for support after development.
3. UBR must be designed to use a minimal amount of resources because there is very little budget.

Environmental requirements:

1. UBR must be available across the United States.
2. UBR must not somehow harm or disturb the butterflies it is built to tally

UI requirements:

1. UBR UI must be intuitive and easy to use.
2. UBR UI must be simple and consistent across the application.
3. UBR UI must be offline-first; that is, the application can operate a critical subset of its core functionality without the internet.

Other requirements:

1. UBR must operate at full capacity on both Android and iOS operating systems.
2. It must be built so that a future update, iOS, for example, will not break the application altogether.

2.4 Engineering Standards

1. Agile- will be used to make sure that we are on track with what the client needs.

2. ISO 15910- This standard will help when documenting our application
3. ISO 9126- This standard will help us develop/evaluate the quality of our application
4. ISO 15504- This standard will help us to access the process of our software
5. HTTPS - Web security standard that will ensure data is safely transferred and stored between the mobile phone and our server. The security of this data is important as we will be collecting personal data like GPS location which is sensitive data.
6. Hashing - We will store any password information as hashes so we avoid the risk of leaking user's passwords.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our team will be adopting a hybrid waterfall+agile approach for this project. This is one of those rare projects where the requirements, even down to the final UI, are very well laid out from the start as we have previous versions of this product to base our design off of. We will still want to keep in close contact with our clients and stakeholders, but their input will likely be more on the level of course-correction and not complete redesigns of certain parts of the project. Our client has a history of liking to test out the application's features as they are developed, which is another reason we feel that the combination of waterfall+agile will suit our project best.

3.2 Task Decomposition

- Working application that helps people track butterflies
    - UI
        - Based on existing UI from old application
            - Map
                - Display map of current survey location
                - Pins
                    - When butterfly observation is made
                        - Include color for species
                        - Stretch goal: widgets on map that expand pins
                    - After 15 seconds of not seeing butterflies
                - Draw lines connecting breadcrumbs in chronological order
            - Butterfly tracking menu
                - Lists of butterflies
                    - 'Favorites' or most recent list
                - Enable users to input a species list
                - Zones
                    - Manual
                    - Stretch goal: Automatic

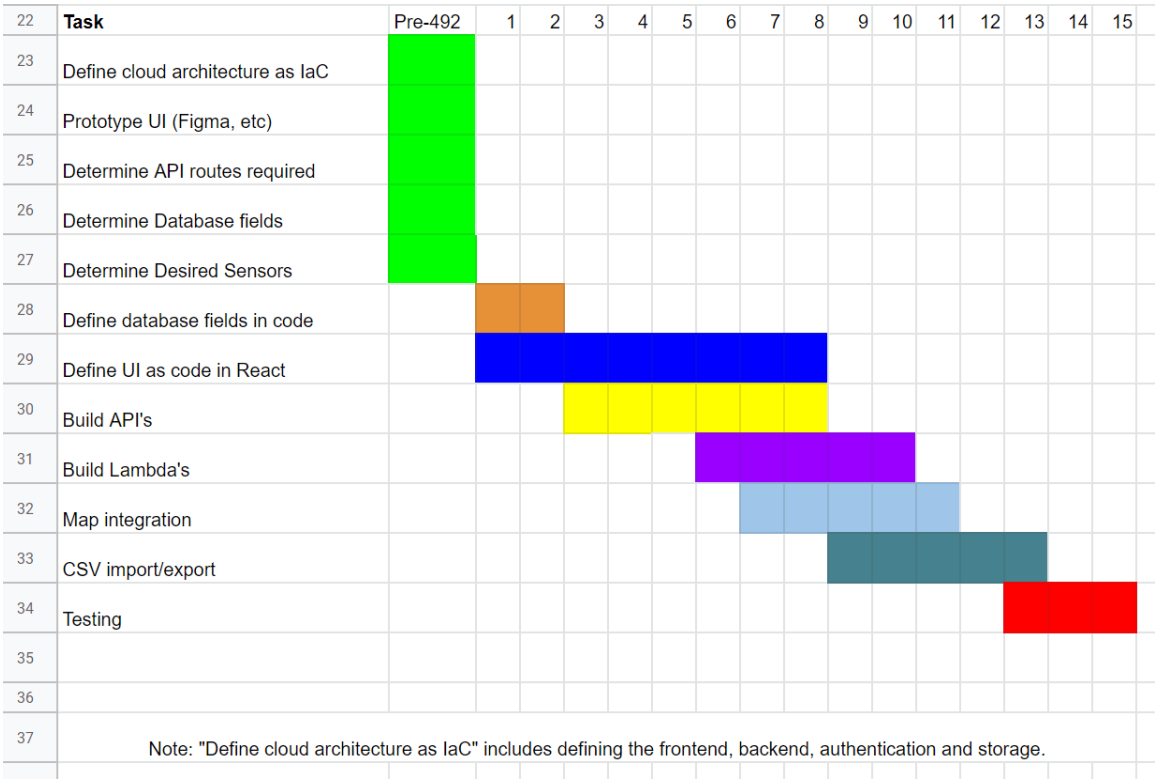
- Log screen
    - Edit data in case of user error
        - Take picture
        - Edit butterfly type
        - Edit number of butterflies seen
- Login
    - To see old surveys
- Past surveys
    - List of butterflies seen
    - Stretch goal: Map of route (path walked by surveyor)

- Backend
    - Maintain published butterfly lists
    - Exporting unsimplified surveys
- Cloud
    - Store unsimplified surveys and user data
    - Export data to PollardBase
        - Compile data into csv file
        - Users can export CSV file
        - Stretch Goal: connect with PollardBase API to send simplified surveys

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. Create basic UI for tracking butterflies
    a. Create tracking and counting system
    b. Use GPS calls for the map screen
    c. Prepare space for butterfly lists
2. Create backend
    a. Receive log exports
    b. Allow users to view old logs
    c. Provide published lists/groupings of butterflies for download
3. System for exporting simplified survey data to PollardBase

3.4 Project Timeline/Schedule



3.5 Risks And Risk Management/Mitigation

- Map (be able to have a functioning map when offline): 0.3
- Add sensor data (we may not have access to all the phones sensors from before so we will likely have to use API's that may change): 0.6
    - When using this API's we will have to add extra validations to allow the app to still function, even in the case the API changes. We will need to allow the app to essentially "skip" that call when it sees that it no longer works and keep proceeding while giving us a warning that it did not work.
- Cloud (No one on our team has any experience with the cloud except for one member so learning this will be a slight risk): .03
- PWA : 0.1
    - May not be able to implement all of our features in offline mode.

3.6 Personnel Effort Requirements

Task	**Description**	**Hours per week per student until task is completed**	**Total hours spent by all team members**
Define cloud architecture as IAC	Define the foundational architecture of the	5	50

	entire application as code		
Prototype UI	Formally define what new UI will look like	2	20
Determine API routes required	Discover and define all use cases for API calls	2	20
Determine database fields	Discover and define all fields that will be required to be stored by database	2	20
Determine sensors required	Discover and define which mobile sensors we will need access to	2	20
Define database in code	Write code to define database fields as IaC	5	60
Define UI in React	Main buildout of front end, using prototype as example	10	60
Build API's	Main API buildout, build all routes required for application	10	150
Build Lambda's	Main logic buildout, build all backend logic as Lambda functions	10	150
Map Integration	Incorporate map into front end	10	250
CSV import/export	Add ability for users to upload CSV of previously recorded CSV data	10	200
Testing	Test all app functionality, adjust as needed	10	150

3.7 Other Resource Requirements

This application does not need any physical resources as it is just an application that is used from a cellular device. The cellular devices will be the pre-existing property of the volunteers.

4 Design

4.1 Design Context

4.1.1 Broader Context

Our design problem is to make it easier for people to collect data on butterflies. In turn, this makes research on butterflies easier as, ideally, there will now be more usage of the app. The app will also now give more/new data fields to the researchers than the paper copies were before. We are designing for the person who goes out and sees/counts the butterflies. The communities that are affected by our design are both the data collectors (walkers) as well as the butterfly research community. The societal need our project has is that it can help keep track of how the butterfly population is doing and may be able to track trends and take action earlier if something is wrong with a large amount of butterflies.

Area	**Description**	**Examples**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

Public health, safety, and welfare: UBR will help to track the butterfly population by providing an easier way to collect data. With an easier way to collect data, people will be more likely to collect

data. With this data, we will be able to see how the butterfly population is doing and keep them healthy so that there is a healthy population of butterflies to keep our ecosystem in check.

Global, cultural, and social: UBR will allow for easier sharing of research data between institutes. UBR will automatically upload each survey's data to the PollardBase repository, a data entry and management system for butterfly monitoring programs, and will make sharing data between participating programs easier than it was before using pen and paper observations that would have to be uploaded by hand. This may lead to more collaboration between Universities and other research institutes.

Environmental: UBR will have little to no environmental impact. Cell phone signals have no known interaction with butterflies and pose no threat to them or their environment. If we host our PWA with AWS as a serverless application,, there will be a small amount of energy used to host the site which will have a small impact on carbon emissions from that data center, and it will be more energy-efficient than having a dedicated server running all of the time. UBR will eliminate the need for paper and pen surveys so less paper will be used for this type of work in the future.

Economic: There are no existing applications that fulfill this niche, so we will not be disrupting an existing market. There will also be no need to be competitive, as this application is a specialty tool that requires some training to use.

4.1.2 Prior Work/Solutions

No similar products exist in this market. We are only competing with the old paper/pen survey method. One could say that we are 'competing' with the original UBR

There are different versions of the UBR application. The original version was an native android application. There is also an IOS version that was once available on the App Store.

PWA	Android	IOS
+cross platform, so more potential users +automatic upload to PollardBase +easier to maintain as web standards change much slower than mobile application standards +super easy to access on any device -potentially more difficult to develop than native application	+better integration with device sensors +can store survey data locally on device -only available on Android devices -harder to access as you need to know how to download android APKs -must export data by hand via CSV file	+better integration with device sensors for temp and location -no longer available due to IOS version changes -cannot maintain

Mobile Web Application	Pen/Paper Survey
+allows for collection of more precise data like GPS location, time information, and weather	+very easy to use and learn +no maintainability concerns

data +allows for data to be sent directly to PollardBase +ability to take pictures of a butterfly if species is unknown, allowing for identification later on -requires a mobile device with ability to connect to internet -more overhead with developing app and maintaining it	+has reliably worked in the past -requires data to be processed by hand for both data collection and when it is sent to PollardBase -no metadata collected

4.1.3 Technical Complexity

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles –AND–
2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.
3. Our design choice to use the cloud, AWS, to build the application is not something most of us are familiar with. We will need to spend time learning how the cloud works versus a traditional API and database setup.

4.2 Design Exploration

4.2.1 Design Decisions

1. PWA vs Cross platform vs native vs hybrid development. This was an important decision for how we are going to meet the needs. We've decided to do PWA as it has an offline feature that we really liked, it will be better in the long term due to not having any iOS updates to worry about, and it is more easily developable than developing an app and having to do it for both iOS and Android.
2. Hosting on Cloud vs Iowa State web server. This decision was important because that is where a lot of our backend logic will be held. It also won't really cost anything extra since we will be not hitting any of the thresholds that require a payment.
3. When to ping a gps location. We decided that we will ping a gps location every x (number is still to be determined) seconds. But when a user clicks that they have seen a butterfly, we will need to grab the location of the user there. So when this happens, we will just reset our timer of when to re-ping the location of the user. We are doing this so that there are less api calls which should help to speed up the app, as well as drain less phone battery.

4.2.2 Ideation

1. PWA
   a. Pros
      i. Applications accessible offline
      ii. Accessing them on different platforms is not a problem
      iii. Much cheaper and faster to develop than a different native solution for each platform

        iv. They adapt to different screen sizes – great responsiveness
        v. Interaction and navigation resembles native apps so it is easy for the user to understand how to use a PWA
        vi. No installation process which can be appreciated by impatient users
    b. Cons
        i. Weaker performance on iOS and less support for Apple devices
        ii. Limitations when it comes to hardware and operating system features
        iii. They are not available on the app stores
        iv. They use much more battery power
2. Cross platform
    a. Pros
        i. Time and cost-effective thanks to developing both app versions at the same time
        ii. Reusability of code so we don't have to create a separate base for each platform
        iii. Easier to reach more target groups with an app for iOS and Android
        iv. Adding features doesn't require a lot of work because code is consistent – in native development usually codebases are much different
    b. Cons
        i. Dependency on framework when it comes to hardware, operating system and UI features
        ii. Some parts of code still need to be written separately because of the differences between platforms
3. IOS native
    a. Pros
        i. Fast performance
        ii. Can function in an offline environment
        iii. Support for device APIs
        iv. UI components custom to each platform which boosts user experience
        v. With access to needed hardware – easier to prevent bugs
        vi. Screen-ratios are not a problem because of layouts for each platform
        vii. No need to depend on open-source libraries
    b. Cons
        i. Android users cannot use the app
        ii. Would need to build a separate android native app to make available for more users
        iii. A lot of updates happen, so maintainability is an issue, not long lasting
4. Android native
    a. Pros
        i. Fast performance
        ii. Can function in an offline environment
        iii. Support for device APIs
        iv. UI components custom to each platform which boosts user experience
        v. With access to needed hardware – easier to prevent bugs
        vi. Screen-ratios are not a problem because of layouts for each platform
        vii. No need to depend on open-source libraries
    b. Cons

        i. Apple users cannot use the app
        ii. Would need to build a separate ios native app to make available for more users
5. Hybrid
    a. Pros
        i. Low cost development
        ii. Maintenance is easy
        iii. Using native APIs is available, so we can use device features like camera or GPS
        iv. Adding new features to hybrid mobile apps is simple with one codebase
    b. Cons
        i. Complex apps won't perform great with this solution, more features slow it down
        ii. Does not work offline
        iii. One codebase means that the hybrid mobile app works the same everywhere – might cause problems with using certain features provided by iOS or Android only

4.2.3 Decision-Making and Trade-Off

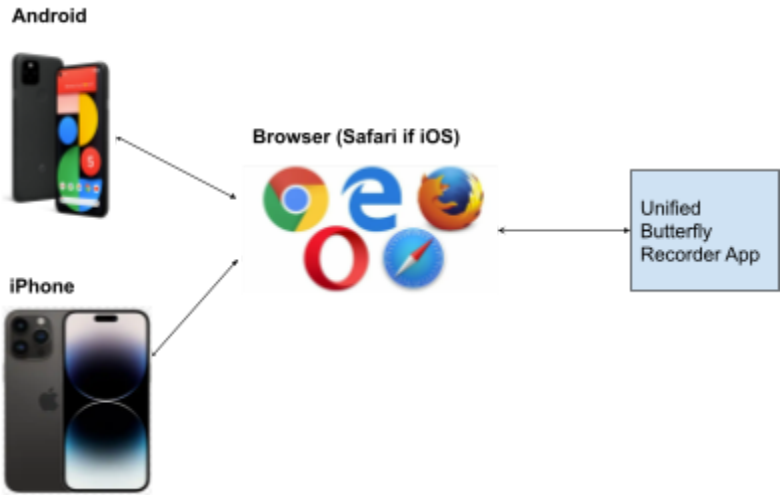
We decided to do a simple pro/con list for each of our options listed in section 4.2.2. If a dealbreaker was identified, we did not consider anything further about that particular route. "Deal Breaker" list below:

- PWA
    - No deal breakers identified. This helped us to reach a final decision.
- Cross-platform
    - Dealbreaker: Longevity
- iOS Native
    - Dealbreaker: App needs to be available for Android users as well as iOS users.
- Android Native
    - Dealbreaker: App needs to be available for Android users as well as iOS users.
- Hybrid Web App
    - Dealbreaker: Application not available offline.

4.3 Proposed Design

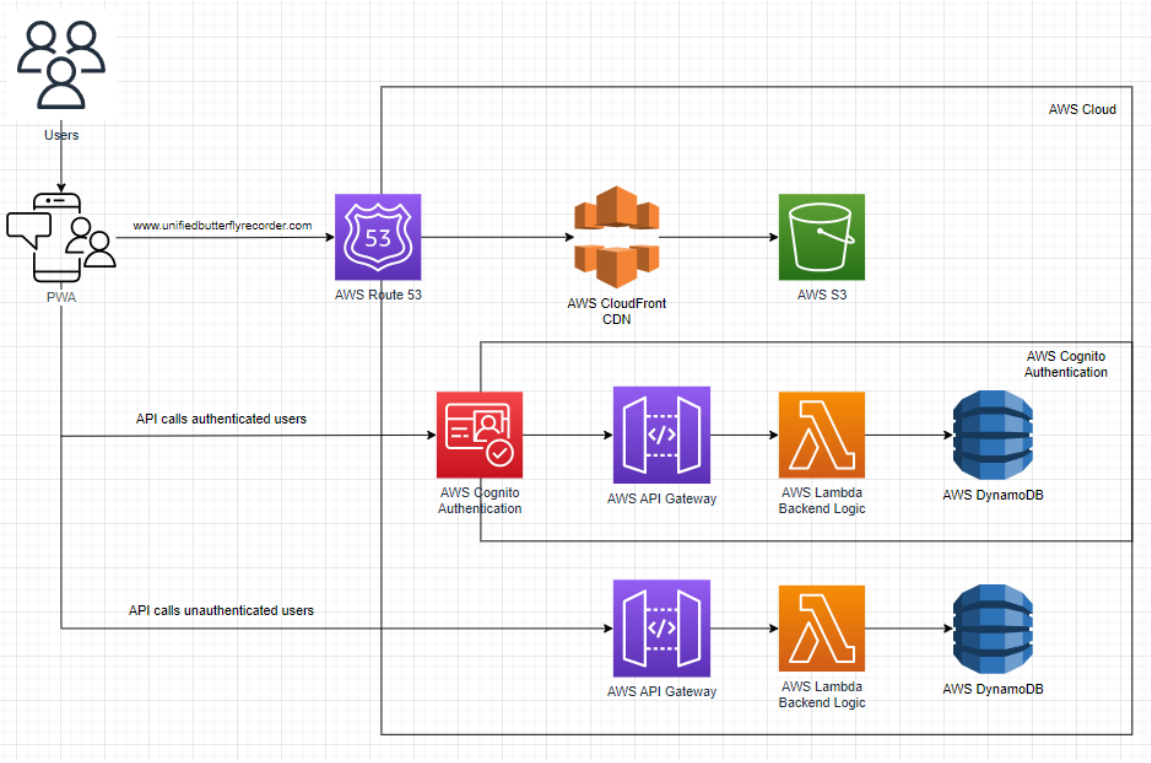
4.3.1 Overview

In order to access this application on a mobile device, users will need to use the browser (Safari if the user is on iOS) to access the application. Once the users enter the URL of the application, they are then able to start using the application. The application is broken down into two parts, a front end where the users are able to interact with the application, and a backend where the logic of the application is stored. This backend is also where the data from the application is stored.



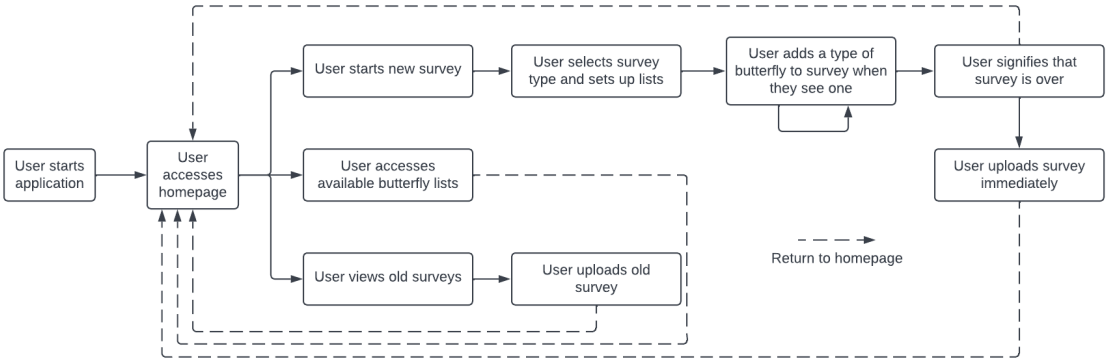
4.3.2 Detailed Design and Visual(s)

We plan to leverage a React with PWA frontend hosted in an AWS S3 bucket accessible with AWS Route 53 via the website url. This PWA will make API calls to the backend, and we may have branching access to the backend depending on whether or not unauthenticated access to survey storage on the cloud is permitted. We will leverage Lambda functions on the backend as well as DynamoDB, all hosted on AWS.

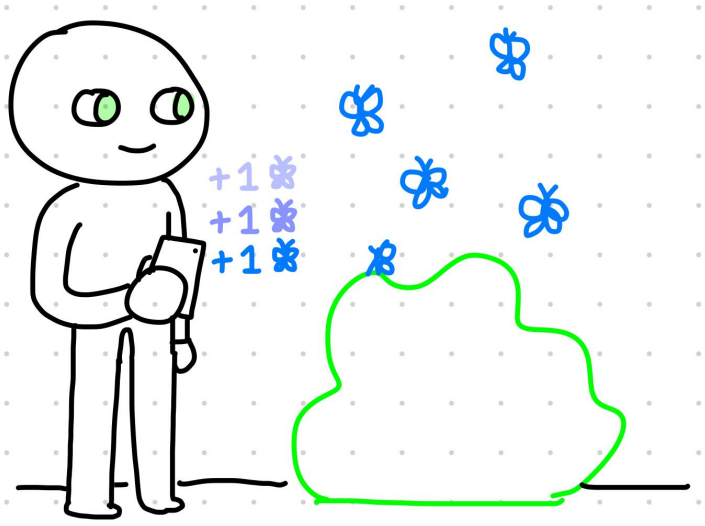


4.3.3 Functionality

This is designed to work as a surveying tool. The most frequent interaction users will have with the system is to add new butterflies to a 'survey.' Additionally, once a survey is complete, a user will have an opportunity to upload their survey immediately to PollardBase, or if they aren't connected to the internet or created a survey by accident, they can upload it later or not at all.



The client has specified that the process of adding new butterflies of the same type, say if you run into a large number of one type during a survey, should be so easy that one doesn't need to be looking at their phone to do it.



4.3.4 Areas of Concern and Development

Previous versions of the application that we will be basing our User Interface on have been tested by our end users and satisfy most of their requirements. Our changes to the existing design will allow for the application to be accessed on all mobile devices instead of just Android devices. We also plan to iterate on the current design by creating an API that will automatically upload the survey data to the PollardBase repository.

Our biggest concern is the offline capability of the application. We are not sure about the number of features we can use offline, as well as the ability and accuracy of those core features. Since this offline mode is the most essential aspect of the program, it is the toughest unknown for us. Also, being able to store the data safely until back online to transfer to the back end is a crucial part of the program, as institutions need to be able to have and use the data once the survey is completed (and back online).

Our immediate plan is to change from having an app that is a Native app to an app that is a Progressive Web Application (PWA). This will solve the problem for iOS users due to the fact that an iOS update won't shut the app down as it did in the past because this app is not specific to iOS. The PWA is a way to have the app function almost identically to the Native app that was previously built while avoiding the possibility of the app becoming entirely irrelevant due to an iOS update that is out of our control.

4.4 Technology Considerations

PWA: We are opting to implement the mobile application as a progressive web app (PWA). This allows us to make one app that is accessible through android and apple devices through the web browser. PWA has an advantage over typically mobile websites because of the offline functionality and consistent native-like user experience. The main drawbacks of PWAs are the increased power consumption and the inability to access as much device data as a native app. PWAs, because of the web development stack, are also going to be much more maintainable and durable throughout the app's life.

Cloud Hosting: Rather than hosting the web server and database on an Iowa State hosted device, we have opted to leverage AWS cloud services in our design. Using the cloud allows us to do all of the backend logic in one place. We've chosen this alternative because of the IaaS capabilities of the cloud and many of the solutions that come right out of the box.

React: It has a lot of documentation, so there are resources to look at if/when we get stuck, however, since React updates so fast, documentation can be scarce for newer features. It makes dynamic information easy to display and has the JavaScript library to utilize, but React is UI only, so it may be harder to integrate our backend with our front end. One weakness of using React is the limited experience our group has using React. So there will be a learning curve to using it.

4.5 Design Analysis

The main thing we have built and tested has been a sample PWA app using cloud services with offline functionality and basic pinning of locations on a google map interface. Anthony, who has the most cloud experience, took the lead in scoping out these basic features in his test app.

One design decision we are contemplating is the use of authentication, offline functionality, and storage of survey data. How sensitive is this data, and how important is it to users that they can access their previous surveys? This impacts whether or not we use the AWS Cognito Authentication Service, and whether or not we allow users to create surveys from offline mode.

If a user is logged in, the user would be able to store their surveys in the cloud and be able to view past surveys. There will be an option to create a survey while not logged in, but it would be subject to being deleted after the information has been submitted.

5 Testing

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

- React App frontend components
    - How:
        - Setup/Teardown tests using beforeEach() and afterEach() functions.
        - Mimic user events, data fetching and rendering using the act() function.
        - Condition checking using the expect() and toBe() helper functions.
    - Tools: Jest and/or the React Testing Library.

- AWS Cloud Development Kit (CDK) infrastructure-as-code
    - How: Two ways, fine-grained assertions or snapshot tests.
        - Fine-grained assertions example: "this resource has this property with this value" .. "this lambda function has a runtime selected, and the runtime equals NODEJS_16".
        - Snapshot test example: store a baseline CDK template, run tests using new code against this baseline, to determine the new code works exactly the same way as the original.
    - Tools: AWS CDK Assertions module and/or Jest.

Helpful unit testing resources

ReactJS: https://reactjs.org/docs/testing.html

AWS: https://docs.aws.amazon.com/cdk/v2/guide/testing.html

5.2 Interface Testing

Interfaces:

- AWS Route 53 connecting the browser running the Progressive Web App (client) and the www.<insertname>.com domain name.
    - We can test that this DNS query is resolving to the correct data by checking the uptime of our website. There are tools like BetterUptime or StatusCake that we can leverage to get alerts if our website is no longer accessible/down from the web browser point of view.
- AWS API Gateway connecting the client and the backend lambda logic.
    - We can use API Gateway Console to test our Rest API calls. This link describes how we can check the status code of API responses, time taken for calls, and even the response body. This allows us to test our backend business logic at the gateway interface.

5.3 Integration Testing

Integration testing in this case can be achieved with react-testing-library again which has many methods for mimicking the actions of a user navigating a page, such as finding and clicking buttons or filling out text fields. It will actually be relatively critical to have good integration testing, as there are lots of circumstances in the app where a user needs to, say, fill out a form in certain ways, and checking that the forms are valid will require the cooperation of many components at once. It would be ideal to automate these tests to make sure that a form can handle all types and combinations of valid input, in situations where the input is derived from limited lists of items.

5.4 System Testing

System testing is used to ensure that the application is completed as a whole. We will use this to allow us to know that all of our functionality is working both, individually and together. This will help to see that these individual components are working correctly, as well as when we bring them together. We will use Jest to first individually test certain functionality. Once we have passed those tests, we will combine multiple functionalities that are more complex to ensure that these have also passed. This should be closely aligned with the requirements as we must test every functionality that we have so we know the product is fully completed.

5.5 Regression Testing

To ensure we do not break old functionality, we will be using GitLabs CI/CD tools. This will allow us to run all new versions of the code through a series of rigorous tests to ensure that the code will build, perform as intended, and deploy successfully. The main thing we need to make sure of is that the project will successfully compile and run for each new version that is pushed to the repository. The CI/CD pipeline tools automatically try to compile and build each new version, and alert us if

the build fails. The CI/CD pipeline tools also allow us to set up tests to make sure that the app functions as intended. Specifically, we will be testing to make sure that each new version of the application is able to build, receive user data from the AWS backend, send new user data to the AWS backend, record new surveys locally, and deploy to the web properly.

5.6 Acceptance Testing

We can demonstrate that our solution meets functional and non-functional requirements by allowing the user to use the application before it's 100% complete. If we create beta models, our client can test small subsets of the application and provide us with feedback. It will be important to let the client know what functionality we are having them test so that their expectation is on the same level as ours for what should be functioning correctly. Acting upon client feedback will be critical to ensure our team meets the needs of the end user.

5.7 Security Testing (if applicable)

We are still working with the client on major design decisions that may require the application to have authentication. If we choose to do authentication, we can use similar techniques that are being used for our React Frontend testing, i.e. Jest and the React Testing Library. We can test our backend authentication configuration in the AWS cloud by using fine-grained assertions and snapshot tests, provided by the AWS Assertions module.

5.8 Results

Test	Correct	Incorrect
React Front End	Easy to use Intuitive Uncomplicated Simple	Complex Difficult to navigate Cluttered Complicated Confusing
AWS CDK Tests	Resources provisioned to specification	Resources provisioned not to specification
Interface Test - Website Uptime	No alerts from the uptime monitor	Alert from monitor saying the site is down
Interface Test - Gateway	Response time <3 Seconds Request body data is as expected Success Status Code	Unusually long response time Request body is does not contain needed data Failure Status Code

6 Implementation

4. Create basic UI for tracking butterflies:
    a. Create tracking and counting system
       Weeks 3-8
    b. Use GPS calls for the map screen
       Weeks 7-11
    c. Prepare space for butterfly lists
       Weeks 1-2
5. Create backend
    a. Receive log exports
       Weeks 3-8
    b. Allow users to view old logs
       Weeks 3-8
    c. Provide published lists/groupings of butterflies for download
       Weeks 3-8
    d. Survey storage
       Weeks 1-2
6. System for exporting simplified survey data to PollardBase
   TBD based on client feedback of PollardBase integration

7 Professional Responsibility

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Area of Responsibility	Our Own Words
Work Competence	Get the work that needs to be done, when it needs to be done, and at the required quality. NSPE vs IEEE They both agree that engineers should not take on tasks that they are not capable/qualified for, but NSPE expands on this idea with a section about how engineers should not even add contributions to documents unless they are qualified to do so.
Financial Responsibility	Have the best interest of your client/supervisor in mind, not only understanding how they want their money spent but also communicating financial matters with the ownership and care you would have with your own money. NSPE vs IEEE NSPE includes quite a bit more on the details of this responsibility, although the IEEE does include details about accurate

	estimates and crediting others' property.
Communication Honesty	Tell the truth, even when it makes you look bad, and don't let false/poor communication lead to surprises later on. NSPE vs IEEE IEEE focuses on the well-being of the public, as well as early reporting of potential conflicts, but the NSPE includes information about the holistic reporting of all pertinent information.
Health, Safety, Well-Being	Act in such a way that does not harm others, and maintain thoughtfulness about the effects of your actions. NSPE vs IEEE IEEE includes sustainability in this category, as well as communication honesty in case your actions do negatively impact the public. NSPE focuses on the privacy of clients and more legal issues.
Property Ownership	Don't take others' ideas or assets as your own when you don't have rights to them. NSPE vs. IEEE Same as the Financial category
Sustainability	Act in such a way as to have a non-negative impact on the longevity and health of your organization, industry, and environment. NSPE vs. IEEE No NSPE Canon
Social Responsibility	Provide value to others, working to benefit others rather than simplifying working for the extrinsic motivators of your work assignment. NSPE vs. IEEE NSPE does not have specific rules of practice for this field but rather a professional obligation to serve the public interest. IEEE does include some information about informing the public as part of your social responsibility

7.2 Project Specific Professional Responsibility Areas

Area of Responsibility	Importance In the Context of our Project
Work Competence	Medium - Typically I would say high, but after learning about the focus on only picking up work that you are qualified in, I feel like that impact is relatively small considering that we are all students and will all need to surpass some learning curve to get these technologies up and running.
Financial Responsibility	Low - It seems that the potential costs of our project are already low and that our client has not given us any financial requirements.
Communication Honesty	High - It's all too easy not to provide the needed product by simply brushing issues under the rug. I have often seen this form of procrastination in school projects, and it is very dangerous when your supervisor is not on the same page about the status of your work.
Health, Safety, Well-Being	Medium - The UBR product needs to have a user-centered design, meeting the needs of the researchers involved in butterfly tracking. We can greatly improve their quality of reporting if we can automate certain tedious tasks.
Property Ownership	Low - We have been given ideas to build off of from other senior design teams, and we are encouraged to do so. I also think the main technology we will be borrowing is necessary boilerplate code in our cloud infrastructure.

Sustainability	Low - Although we can have an environmental impact on how software is produced, our project is relatively trivial compared to more physical products.
Social Responsibility	High - We have a responsibility to Reiman Gardens and the research network to provide the service that they are asking for. This customer satisfaction is critical to the project's success.

7.3 Most Applicable Professional Responsibility Area

Communication and honesty:

For our project, the most important area of professional responsibility is communication and honesty. In the context of this project, communication and honesty means that we keep our client, advisor, and fellow team members up to date on the current status of the project. We have done this by setting up a weekly recurring meeting with all stakeholders to provide updates on the status of the project, as well as ask questions to the client and advisor so that we can make the best decisions moving forward. We also meet as a team in class, and a few times a week outside of class as needed to discuss the project. This has benefited us when we needed to make large design decisions about the project and had the ability to quickly meet with the client the determine his users' needs and what design decisions would suit them best.

8 Closing Material

8.1 Discussion

The largest requirement that we hoped to deliver on but now believe we probably won't be able to is the offline geographical pinpointing. A native application could get a device's GPS data without an internet connection, but the PWA we've decided to go with requires an internet connection to successfully complete a GPS request. We have discussed this at length with our client and missing out on some geographical data some of the time will not kill the project.

We already have some ideas in mind for how to work around this problem smoothly. Some GPS data is better than none, so getting the exact GPS position of every butterfly the user sees isn't a dealbreaker.

Otherwise, we are successfully meeting all of the requirements set forth by the client.

8.2 Conclusion

Completed Work Summary

- Working prototype
    - PWA
        - GPS calls
        - Map Integration
        - Time
        - Authentication

        - Repository
    - Cloud Infrastructure
        - Database Implemented for Authentication
        - Domain Route 53
        - S3 Bucket
        - Defining some fields and routes
- Schedule
    - Milestones
    - Gantt Chart
    - Testing Plan
- Project Design
    - Requirements
    - User Needs
    - Risk

Contstrains

One issue that we've faced this project has been navigating the PWA specific design. We've needed to learn what device options we have through the web browser, and this have been a difficult task because we are not doing native development. We also have a dependency in our project on the development of the PollardBase API in order to upload data to the research database.

To overcome some of these challenges going forward, we want to be in communication with the contact given to us at PollardBase. There is a chance we will be able to integrate with PollardBase during the scope of our project. We also want to test PWA features early in development so that we can verify our dependencies before investing time in relevant sprints

8.3 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

8.4 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.

8.4.1 Team Contract

Team Members:

1) Zach Wingert

2) Timothy Ellis

3) Ryan McNally

4) Anthony Mazzie

5) Jeremy Marchesani

6) Grace Wigen

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

2:30-3 PM Mondays with Dr.Rover via Zoom

2:00-2:30 PM every other Wednesday with Nathan, in person, central campus

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Discord

3. Decision-making policy (e.g., consensus, majority vote):

Majority vote

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Meeting minutes are stored in a shared Google Drive folder.

Switch who takes notes each meeting.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

Team members should be present (virtually and mentally) and on time for meetings. If you cannot be there, let the team know via Discord.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Aim to get things done before they are due. Communicate with others if any help is needed.

3. Expected level of communication with other team members:

Communicate often. If you need help, ask. If you have extra capacity to get things done, let the team know.

4. Expected level of commitment to team decisions and tasks:

There's no reason why we cannot get an A in this class. Do A-worthy work.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction,

individual component design, testing, etc.):

Team organization: Grace Wigen

Client Interaction: Tim

Individual Component Design: TBD

Testing: TBD

Research: TBD

2. Strategies for supporting and guiding the work of all team members:

Communicate often, ask questions, and be open about where you are in your work especially if you have roadblocks.

3. Strategies for recognizing the contributions of all team members:

GitLab to track project progress

Collaboration and Inclusion

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

team.

Anthony Mazzie - AWS cloud development, serverless/event-driven infrastructure-as-code cloud development, web development (ReactJS)

Zach Wingert: Backend/PLSQL, C, Java, Oracle Apex

Timothy Ellis: Native Android development, organized, architecture mindset when programming

Jeremy Marchesani: Android development, Flutter/Dart Development, ReactJS, Java

Grace Wigen: basic industry full-stack web development experience, native Android development in education environments, art asset creation, Java, Javascript, C#.

Ryan McNally: SQL ETL/Backend work, Android development, java, basic scripting.

2. Strategies for encouraging and support contributions and ideas from all team members:

Transparent team environment. Team members are encouraged to participate, ask questions, and communicate often. All ideas are welcome.

3. 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?)

Communicate concerns with the individual, escalate if needed to professor/advisor

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Get an A and be well prepared for the next semester. Don't leave all the work to the last minute.

2. Strategies for planning and assigning individual and team work:

Discuss these at our weekly meetings as well as on Discord. Use GitLab Issues

3. Strategies for keeping on task:

Being organized, sharing deadlines, communicating status via GitLab, communicating via Discord

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Group will discuss the infractions internally during weekly meetings.

2. What will your team do if the infractions continue?

Elevate discussion to team advisor/professor.

****

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) Ryan McNally_____ DATE 9/12/2022

2) Grace Wigen_____ DATE: 9/12/2022

3) Zach Wingert_____ DATE 9/12/2022

4) Anthony Mazzie_____ DATE 9/12/2022

5) Jeremy Marchesani_____ DATE 9/12/2022

6) Timothy Ellis_____ DATE 9/12/2022