

Unified Butterfly Recorder

DESIGN DOCUMENT

Team 12

Client: Rieman Gardens, Nathan Brockman

Advisor: Dr. Diane Rover

Team Members/Role:

Timothy Ellis - *Frontend*

Ryan McNally - *Backend*

Anthony Mazzie - *Backend*

Lucas Onwu-Chekwa - *Frontend*

Zach Wingert - *Frontend*

Jeremy Marchesani - *Frontend*

Grace Wigen - *Frontend*

Team email: sdmay23-12@iastate.edu

Team Website: <https://sdmay23-12.sd.ece.iastate.edu>

Table of Contents

Table of Contents	2
1 Revised Project Design	4
1.1 Changes from CPRE 491.....	4
1.2 Functional requirements:.....	4
1.3 Non-Functional Requirements.....	4
1.3 Standards.....	5
1.4 Engineering Constraints.....	5
1.5 Security.....	6
1.5.1 Physical Security.....	6
1.5.2 Cybersecurity.....	6
2 Implementation Details	7
2.1 Architecture Overview:.....	7
2.2 React App Implementation.....	8
3 Testing	9
3.1 Testing Process.....	9
3.1.1 Unit Testing.....	9
3.1.2 Interface Testing.....	9
3.1.2 Integration Testing.....	10
3.1.3 System Testing.....	10
3.1.3 Regression Testing.....	10
3.1.4 Acceptance Testing.....	11
3.2 User Testing.....	11
3.3 Testing Results.....	11
4 Context	13
4.1 Related Products.....	13
4.2 Related Literature.....	14

4.2.1 Disign Statment.....	14
4.2.2 Areas of Concern:.....	14
Appendix I - Operation Manual.....	16
1 Introduction.....	16
2 Login/Signup.....	16
3 Preview Old Surveys.....	17
4 Start a new survey.....	17
5 Pause, Cancel, or Finish a survey.....	18
6 Favourite a butterfly.....	18
7 Review Current survey details.....	19
8 Edit Current survey general details.....	19
9 Edit Current survey sighting details.....	20
10 Set Default values for the app.....	20

1 Revised Project Design

1.1 Changes from CPRE 491

Throughout the semester, we had to make some design changes to better match our customer needs and the scope of the project that we could support. The major design changes are listed below.

1. No integration with PollardBase. We would have liked to connect to the 3rd party butterfly information repository; however, their development of the API that would have allowed us to do this fell behind schedule and is still not ready for use.
2. No longer an 'offline first' application. Due to restrictions related to PWA's access to hardware features, we can not get the GPS coordinates of the user when they are offline. This means that for full functionality, the user must have an internet connection.
3. No ability for users to take photos of butterflies in our app and associate them with a sighting. This change was made to make the scope of the project smaller and more accomplishable in the given time frame.

1.2 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 .csv file format.
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 be able to store GPS and time stamp data.

1.3 Non-Functional Requirements

1. UBR must be available from users' mobile devices, specifically Android and iOS phones.
2. Devices must have GPS capabilities and the ability to connect to the internet.

3. 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
4. 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.
5. UBR must be built in a way that will minimize maintenance, as there is very little budget for support after development.
6. UBR must be designed to use a minimal amount of resources because there is very little budget.
7. UBR must operate at full capacity on both Android and iOS operating systems.
8. It must be built so that a future update, iOS, for example, will not break the application altogether.

1.3 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 - A 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 users' passwords.

1.4 Engineering Constraints

1. Data persistency: The information in the app persist upon every app refresh. When a user refreshes the app, they should not lose the current survey data already collected.
2. Compatibility with different browsers and platforms: The app is designed and developed to work seamlessly across different web browsers, operating systems, and devices, including desktops, laptops, tablets, and smartphones.
3. Responsiveness and scalability: The app is designed to adapt to different screen sizes and resolutions without sacrificing performance or user experience.
4. Offline functionality: The app is designed to function offline or in areas with low network connectivity by caching and storing data locally.

5. Performance: The app is designed and optimized for fast loading speeds and low latency to provide a smooth and seamless user experience.
6. Security: The app is designed and developed with security best practices in mind, including secure connections, data encryption, and protection against common web security vulnerabilities.
7. User experience: The app is designed to provide an intuitive and engaging user experience that meets the needs and expectations of the target audience.
8. Integration with existing systems: The app is designed to integrate with existing systems and platforms, including third-party APIs, databases, and content management systems.
9. Cost-effectiveness: The app is designed and developed to be cost-effective, considering the budget, resources, and time constraints of the project.
10. Maintenance and support: The app is designed and developed to be easy to maintain and support, including regular updates and bug fixes.

1.5 Security

1.5.1 Physical Security

The application is somewhat limited in physical security, as anyone who has the link can access the application. But in order to control this, we provided the client with admin privileges which allows them to keep track of user activities and block or delete a user if found to be a security risk.

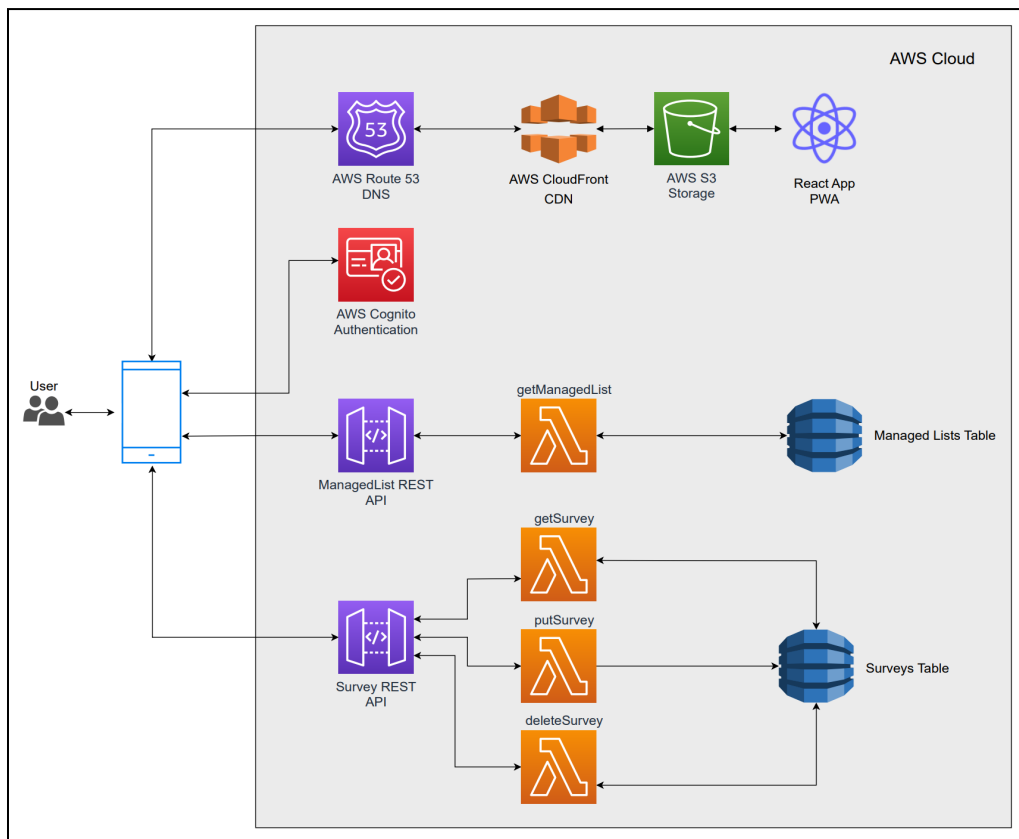
1.5.2 Cybersecurity

- Authenticating users: All users require a valid email and password to sign up for the app. This email needs to be verified before they can move forward, there restricting the number of random users in the application and ensuring every user is valid.
- Input filter: Each input collected in the app through forms is validated by the aws back to prevent injections (or Cross Site Scripting).
- Storage: User information, credentials, and data are stored securely on the cloud. Each user can only access their own data to ensure data integrity and confidentiality.

2 Implementation Details

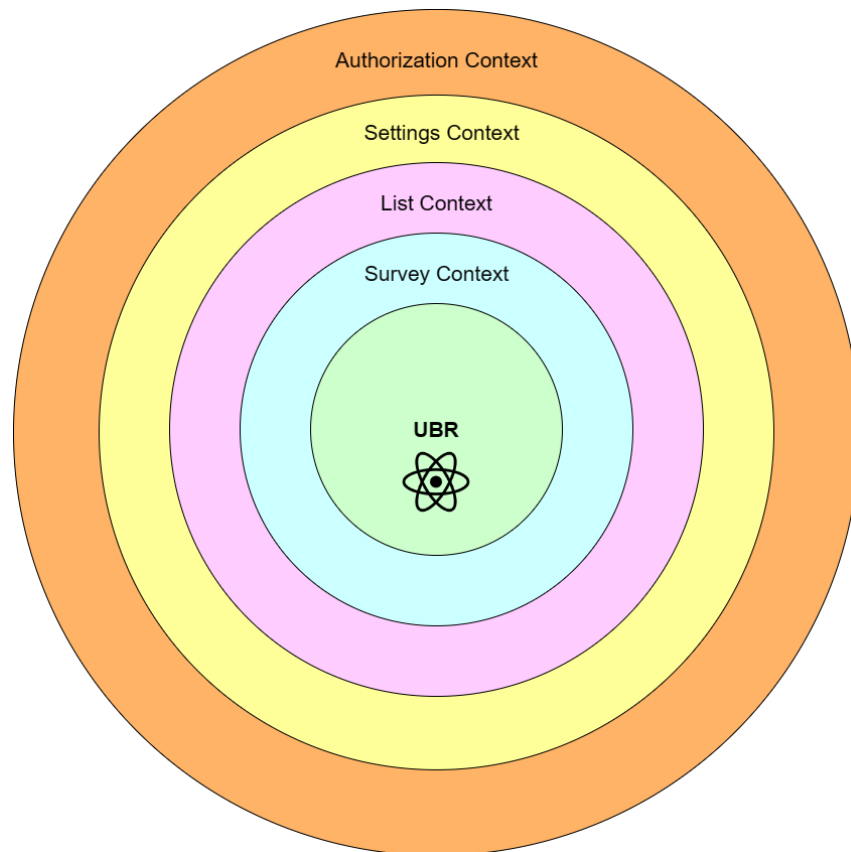
2.1 Architecture Overview:

For the URB application, we have implemented a Progressive Web Application (PWA). A PWA is a web-based application that provides an experience similar to a native mobile application but allows for some major advantages compared to native apps, like being cross-platform and more maintainable. UBR uses a React app frontend and AWS cloud services for hosting, DNS, storage, and authentication. Specifically, our React app is stored in an Amazon S3 bucket; it is then served up to the AWS CloudFront Content Delivery Network and then passes to AWS Route 53 DNS for hosting. User authentication is handled by Amazon Cognito services. We use an Amazon DynamoDB to store all of our user's data, with one table for users' survey data and one table for managed lists. These tables are accessible to us through Amazon's API gateway using REST APIs, where we can get, set, and delete surveys and lists using Lambda functions.



2.2 React App Implementation

The URB applications front end handles internal data movement in two ways, React contexts and local storage using the browser's cache via IndexedDB. The diagram below shows the hierarchy of the contexts used within the application to handle data movement. This hierarchy allows data to be passed to child components and is how the majority of the internal data is passed around. In some cases where persistent data is important, we use the IndexedDB to store semi-permanent data like user preferences and settings in the browser's cache. In the case where data must be fully persisted, we use the Amazon DynamoDB as previously described in the architecture overview. The DynamoDB stores all of the survey data and list data and is fully backed up in the cloud as soon as a user completes their survey.



3 Testing

3.1 Testing Process

3.1.1 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.

3.1.2 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.

3.1.2 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.

3.1.3 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.

3.1.3 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.

3.1.4 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.

3.2 User Testing

User testing has been the main focus of our testing efforts. Our client and his assistants have been user testing the application throughout the development process and have given feedback that has shaped the app and the way we developed it. Our team would meet with our client at least once a week to review changes in the app as well as getting feedback and discussing requirements and functionality. This testing process helped reveal bugs early and has greatly improved the quality of our application.

3.3 Testing 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

4 Context

4.1 Related Products

There is no 'industry standard' for the type of application we have developed. The only other similar products that we have knowledge of are the previous iterations of this application from previous Senior Design projects. Other than that, the industry standard for this was to complete the surveys using a pen and paper form, and then to manually upload data from that. A list of pros and cons comparing our implementation of the URB app to previous version is listed below

PWA	Android	IOS
<ul style="list-style-type: none"> +cross platform, so more potential users +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 a native application 	<ul style="list-style-type: none"> +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 	<ul style="list-style-type: none"> +better integration with device sensors for temp and location -no longer available due to IOS version changes -cannot maintain

Pros and Cons of mobile application VS traditional pen and paper survey:

Mobile Web Application	Pen/Paper Survey
<ul style="list-style-type: none"> +allows for the collection of more precise data like GPS location, time information, and weather data +allows for data to be sent directly to PollardBase +ability to take pictures of a butterfly if species is 	<ul style="list-style-type: none"> +very easy to use and learn +no maintainability concerns +has reliably worked in the past -requires data to be processed by hand for both

unknown, allowing for identification later on -requires a mobile device with ability to connect to internet -more overhead with developing app and maintaining it	data collection and when it is sent to PollardBase -no metadata was collected
---	--

4.2 Related Literature

4.2.1 Disign Statment

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 number of butterflies.

4.2.2 Areas of Concern:

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.

Appendix I - Operation Manual

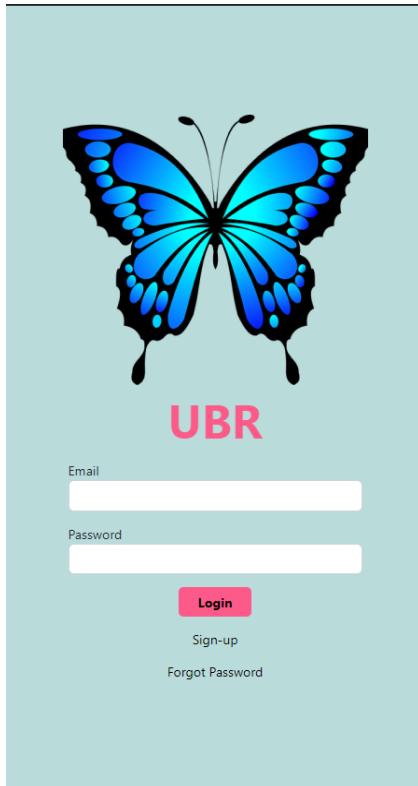
1 Introduction

Our system is Progressive Web App (PWA). It is a type of web application that uses modern web technologies to provide users with a native-app-like experience on the web. PWAs can be accessed through a web browser like any other website, but they have additional features and functionality that make them feel like native apps. Besides accessing the site online, users would be able to download a version of the application to the mobile device, which is what sets it apart from other normal web applications.

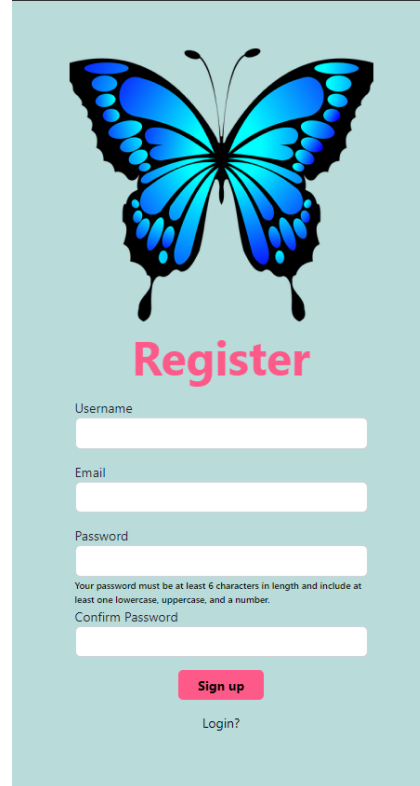
Our system is called Unified Butterfly Recorder, its sole purpose is to allow a user to run butterfly surveys. A survey in our use case is a quick or long trip (mostly by foot) where a user clicks on a butterfly from a list when spotted on their walk.

2 Login/Signup

When a user accesses the link <https://www.unifiedbutterflyrecorder.com/> they are met with a login page. A user can either login with their email and password if they already have an account or signup for one if they don't have an account



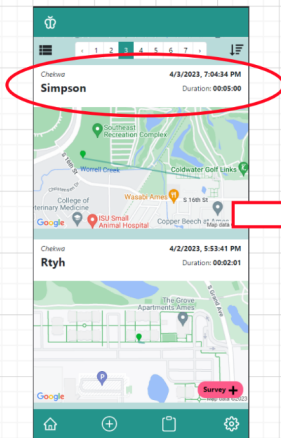
The login page features a large blue butterfly illustration at the top center. Below it, the text "UBR" is displayed in pink. The page contains two input fields: "Email" and "Password". A pink "Login" button is positioned below the password field. At the bottom, there are links for "Sign-up" and "Forgot Password".



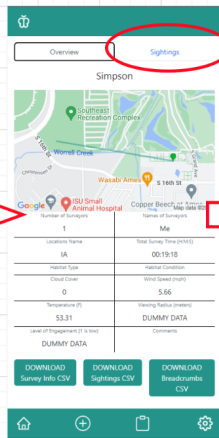
The register page features a large blue butterfly illustration at the top center. Below it, the text "Register" is displayed in pink. The page contains three input fields: "Username", "Email", and "Password". A pink "Sign up" button is positioned below the password field. Below the password field, there is a note: "Your password must be at least 6 characters in length and include at least one lowercase, uppercase, and a number." Below this note is a "Confirm Password" input field. At the bottom, there is a "Login?" link.

3 Preview Old Surveys

On successful login, users are presented with their previous surveys if they have any or an empty page if none



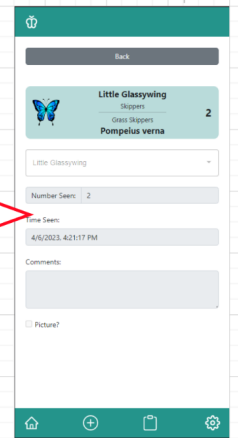
Home Page
Click on the header of the previous survey you want to preview



Overview Page
It brings you to the general overview page where you can see general data and also download survey information



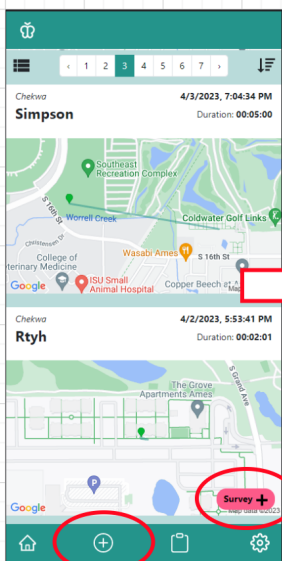
Sightings Page
Here you can see all the sightings made during that survey. Click any one to see more info



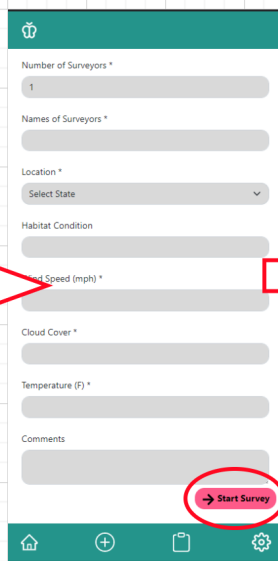
Sightings Details Page
More info about the sighting is available here

4 Start a new survey

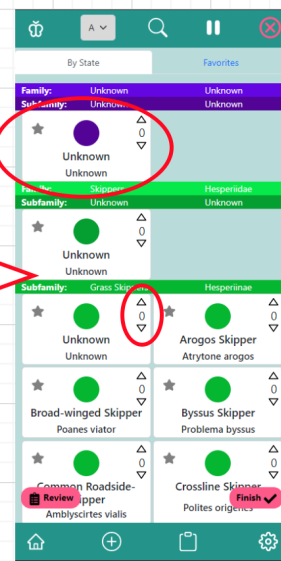
A user can start a new survey in two ways,



Home Page
Either of those buttons can be pressed to start a new survey

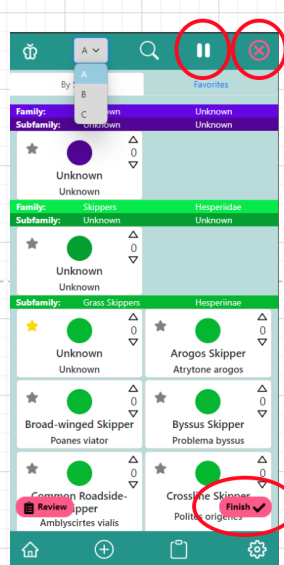


Survey Form Page
Fill out the survey details



Active Survey Page
Add butterflies when spotted by clicking on the cards or arrow button to increment or decrement count

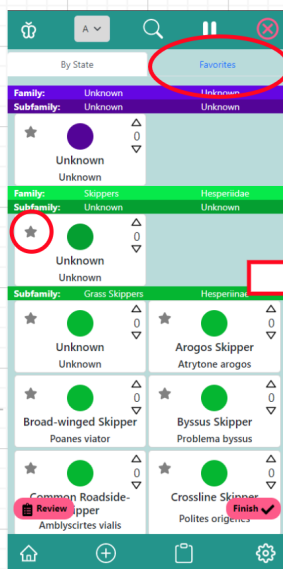
5 Pause, Cancel, or Finish a survey



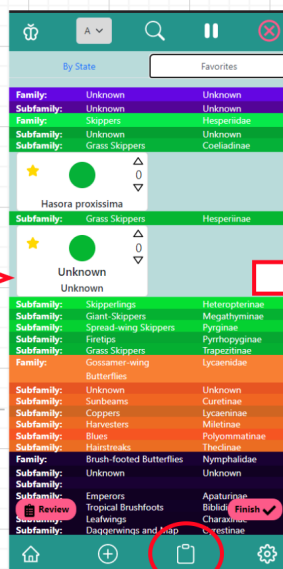
During an active survey you can either pause the survey, cancel the survey or finish the survey (submits it to the database)

6 Favourite a butterfly

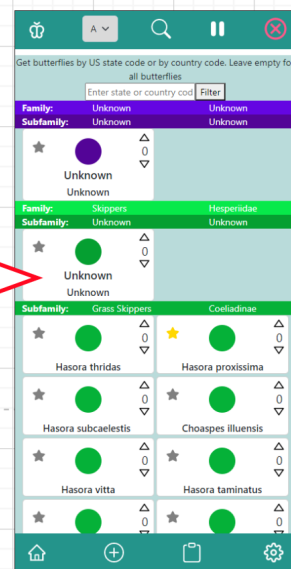
A user can favourite a butterfly in two ways



Active Survey Page
During an active survey click on the star icon of a butterfly to favorite it. Click on the favorites tab to see all favorites

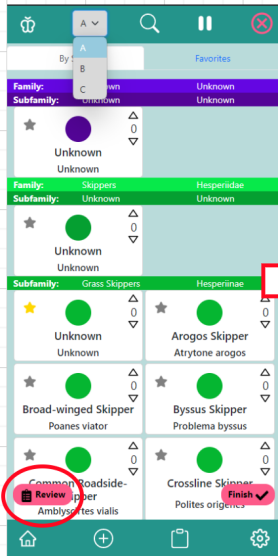


Favorites
All your favorites can be easily accessed during an active survey. You can also favorite from the all list page

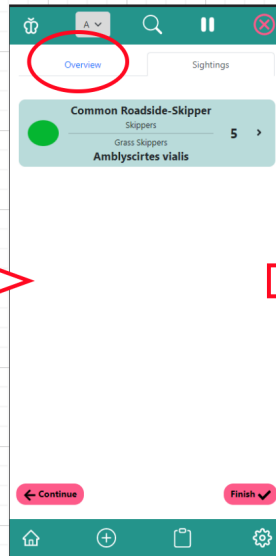


All Butterfly list Page
A list of all butterflies in our database

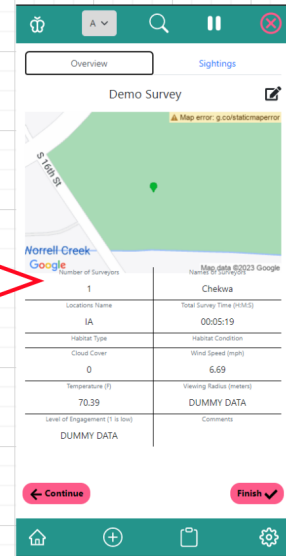
7 Review Current survey details



Active Survey Page
During an active survey click the Review button to preview current survey data so far

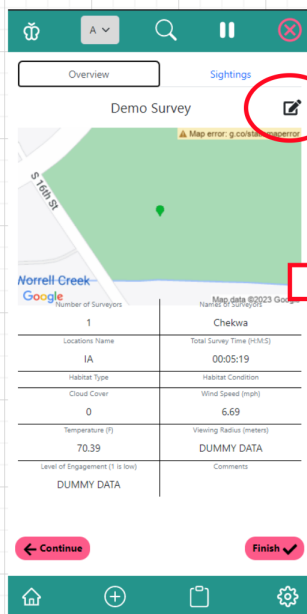


Sightings Page
Here you can see all the sightings made during the survey. Click the overview tab to see general info

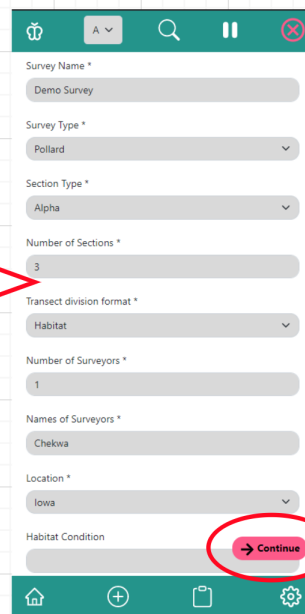


Overview Page
It brings you to the general overview page where you can see general data

8 Edit Current survey general details

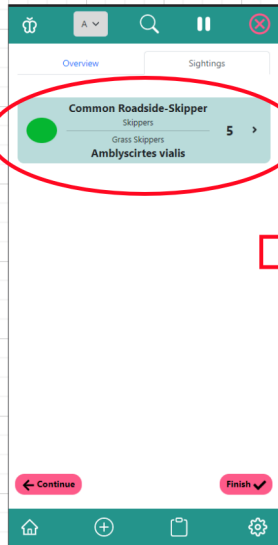


Overview Page
Click on the edit icon at the top right to update the current survey details

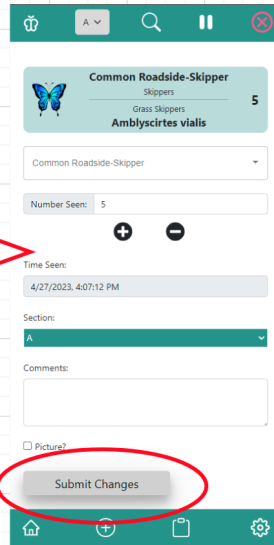


Survey Form Page
It brings you to the form page. You can update some values here and click Continue when done.

9 Edit Current survey sighting details

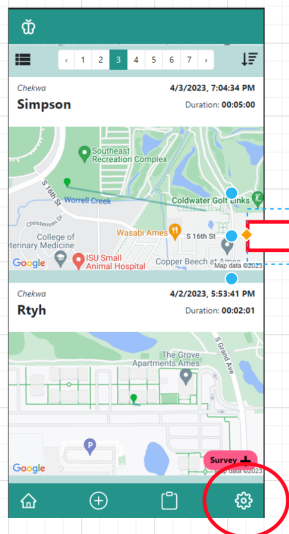


Sighting Page
During a survey, click a sighting to update sighting details

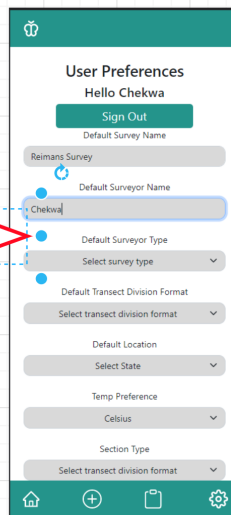


Sighting Details Page
Here you can update details about the sighting.
Click Submit Changes when done.

10 Set Default values for the app



You can set default values by clicking the settings icon anytime



Settings Page
Set default survey values, by updating any form input, and it saves automatically.
You can also sign out of the app on this page