Unified Butterfly Recorder: iOS Final Report

Senior Design Team DEC14-16

Eric Larssen | CJ Mankin | Sean Shickell

Client: Reiman Gardens
Nathan Brockman - Anita Westphal

Advisor: Dr. Diane Rover

Table of Contents

- Project Overview
  - Background
  - Surveys, Sightings, and Protocols
  - Target Devices
  - Problems
  - Objective
  - Functional Requirements
  - Non-Functional Requirements
- System Design
  - Architecture
  - Functional Decomposition
  - Design Decisions
- Detailed Design
  - Database (Schema)
  - User Interface
  - Sightings
  - Remote Export Support
- Testing
  - Simulation and Logging
  - Developer Testing
  - Field Testing
- Project Management
  - Standards
  - Schedule and Timeline
  - Work Breakdown
  - Risks
  - Communication
- Summary
  - Acknowledgments
  - Conclusion
  - Maintainence
- Appendix A: User Manual
  - Getting the App
  - Basic Surveying

Project Overview

Background

Butterfly population levels are an effective way of indicating climate changes in an environment. The 2013 Unified Butterfly Recorder team created an Android application (app) that changed the way butterfly sightings are recorded around the world. Popularity of the app was exceptional, but Apple users, including several Apple-exclusive organizations, were unable to participate. Thus, the demand for a similar iOS app led to the creation of this team.

Surveys, Sightings, and Protocols

The butterfly research community consists of a myriad of organizations and individual scientists that all collect information that they may deem necessary or important. A **survey** is a collection of butterfly sightings gathered in one general location over a time span that is generally less than a few hours. A **sighting** is simply a sighting of one or more butterflies of a particular species and the data surrounding it. Organizations handle sightings generally the same way albeit collect different data, where they differ is how they conduct surveys. We call these different surveys **protocols**. Below are the protocols we have implemented in our application.

Pollard Walk

A surveyor repeats an identical route several times over the course of years, recording sightings visible within a specific range of the path, aiming for consistency of recording. This can allow more rigorous statistical analysis.

Distance Sampling

Record the distance of sightings from a specific line or point in order to estimate local distribution and abundance.

Presence-Absence

Simply record whether a particular species has been sighted at all in an area.

Meandering

Similar to field trips, this protocol involves individuals or groups walking an indeterminate path looking for as many individuals as they can find.

Mark-Recapture

Surveyors capture butterflies with nets. They then place a mark on their wings and release them. Later they return to the same location and capture more butterflies, making notes when they find already marked butterflies.

Target Devices

After discussions with our client and advisor we decided to support all devices that support iOS 7 or later. This turned out to be a preliminary choice, as Apple introduced iOS 8 during the second half of this project. iOS 8 introduced some new features in development and testing, and all devices that supported iOS 7 also support iOS 8. Because of this, we shifted our target to iOS 8.

We chose to support iOS 8 devices because they are modernized, and support Apple location services, either through assisted-GPS in cellular models or broader WiFi services in standard models. The presence of location services is a requirement whenever possible due to the enhanced functionality it provides our app. The background location determination provides weather information for the survey and gathers a GPS gridpoint for a sighting when it is made.

When considering physical devices, our application is less complicated in that most Apple devices have the same general features as far as the developer is concerned. The major exception to this rule is in screen size and resolution, as devices range from smaller iPod touches to a full size iPad and some models contain Retina displays.

Finally, we had to consider the presence of onboard sensors. The Android team worked to support the presence or absence of pressure, temperature, and ambient light sensors in their application to gather such information in the background when a sighting was made. Of these legacy choices, the Apple devices that we choose to support by choosing iOS 7 devices will only contain ambient light sensors. The client has accepted that pressure and temperature fields will have to be filled by manual entry or weather service information.

Problems

In general there is no national or global standard for the type of data collected, how it is recorded, or the methods of collection. The result is multiple, organization-specific databases, each collecting or excluding their own unique set of datapoints. The lack of such a standard creates data disparity, and thus complicates analysis and fosters error-prone conclusions.

More specific problems being addressed by our project are listed below in no particular order.

- Apple users are unable to utilize the Android app and are currently entering survey information manually using paper and pencil.
- The aggregation of data requires manual effort and is therefore prone to human error.
- Third party organizations need to convert the data from the Android and iOS app to their own systems, and currently have to do so manually.

Objective

Data fragmentation, collection methods, and data storage continue to impede progress toward a single, unified database of butterfly sighting information that can be used to aid conservation efforts around the world. The Android app took the first steps to achieving a solution to these problems. We aim to continue this effort by creating an iOS application to introduce and include a larger subset of the surveyor community to the Unified Butterfly Recorder framework.

In doing so, we will aim to uphold the standards set by the Android Team in usability, reliability, functionality in our application. We will continue to support the easy collection of survey data across multiple protocols and efficient collation and exportation operations to conform to the needs of the iOS community.

We will continue to work with the community to raise awareness for the Unified Butterfly Recorder framework and try to convince more organizations to adopt it’s data collection applications. We will move beyond mere emulation of the Android app and work to continue surpassing the quality and efficiency of current surveying methods by interacting with and responding to feedback from the community.

Functional Requirements

Summary

Allow the user to record any amount of information they wish to store from sightings in a survey data structure. Collect as much information from the device as possible, relating to location and conditions. Allow users to export their data to external store, email, or web server.

List of Inherited, Cross-Platform Requirements

The following is a list of functional requirements provided to the Android team we inherited with the project:

Automated Data Collection

- Breadcrumbs (waypoints on a map for route tracing)
- Date and Time
- Wind Speed and Direction
- Humidity
- Cloud Cover

Manual Data Collection

- Increment / decrement a sighting instance for each species
- Habitat description and conditions
- Mark and recapture data entry
- Categorized behavior notes
- Location name
- Surveyor names and count
- Comment section
- Manual weather entry
- Photo capture

iOS Specific Requirements

- Modify collection methods to support manual entry where automatic collection is impossible.
- Allow for location services failure in areas where they are unavailable
- Account for disparity in GPS gridpoint accuracy across devices

Web Export Requirements

Allow submission of the survey and its data to to a server which will put the data into a database.

Non-Functional Requirements

Inherited, Cross-platform Requirements

- The battery must drain no more than 50% during a 4 hour survey on recommended settings with an average phone
- Should require little or no training
- Data should not be lost upon exceptional conditions
- Allow users to record data in absence of a data or wifi connection
- Usable on a device without a subset of supported hardware features and sensors
- Minimize network traffic
- Submit data via wifi or cell network
- Failure handling on data submission
- Leave app in middle of survey, and continue survey later
- Minimize CPU usage

iOS Specific

- Stay consistent to the Android workflow design
- Maintain Apple Human Design standards
- Preserve consistent user experience across platforms
- Ensure similar database schemas and data collection between apps

System Design

Architecture

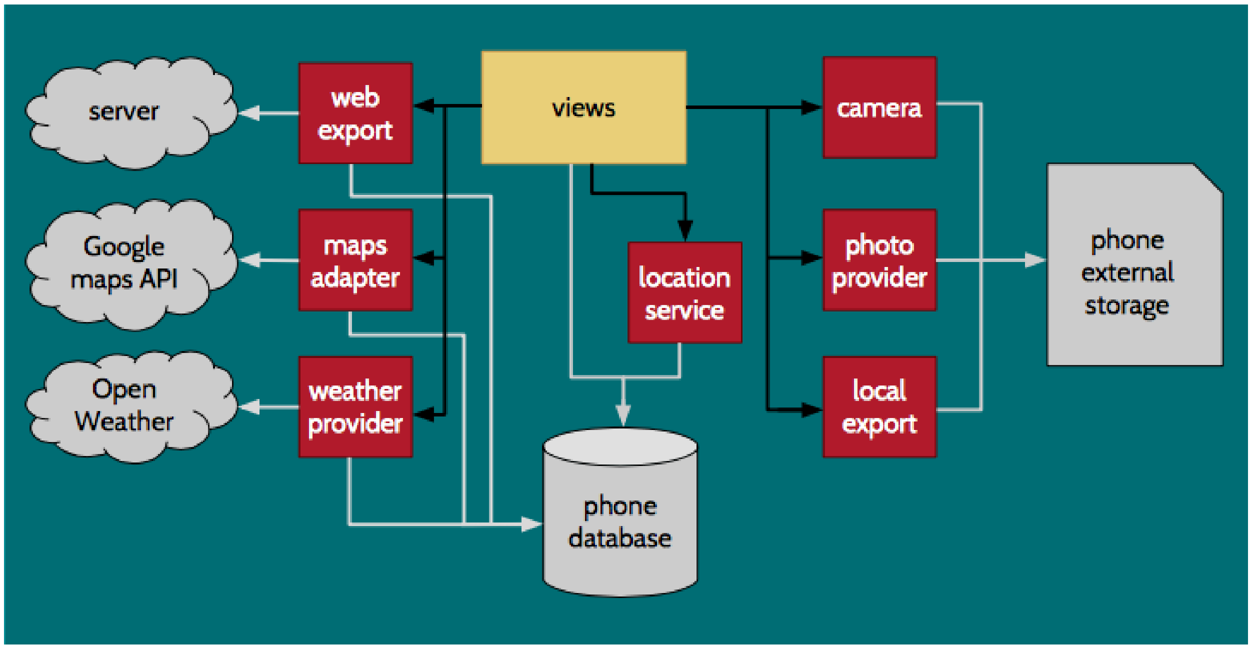


Figure A. Information summarized from poster of Senior Design Team 08, December 2013: http://butterflies.ece.iastate.edu/files/ubr-poster.pdf

Android

The architecture summarized in Figure A., displayed above, represents the system architecture for the Android app. This architecture is included in our report because it represents the primary constraint in our design. The following items summarize the features of the existing architecture.

- The user interface is achieved through subclasses of the Android Activity class (pictured in yellow)
- UI events trigger actions by intermediate services, adapters, or providers (pictured in red)
- Intermediaries write data directly to application or device storage (pictured in gray)

iOS

As the continuation team, our design process needed to adapt the Android architecture into a model that is compatible with the iOS framework. Figure B., displayed below, represents the architecture that arose from that design process. The following items summarize the differences from the original architecture

- The user interface is achieved through subclasses of iOS UIView class
- UI events trigger functions provided by relevant frameworks (e.g. CoreLocation) included in the build path
- iOS applications restrict writing to an Apple device's physical disk, so there is no phone external storage
- Location services require interaction with a separate framework than Apple Maps, as opposed to a single system within the Google Maps API
- iOS includes UI support that eliminates the need to Observer functions

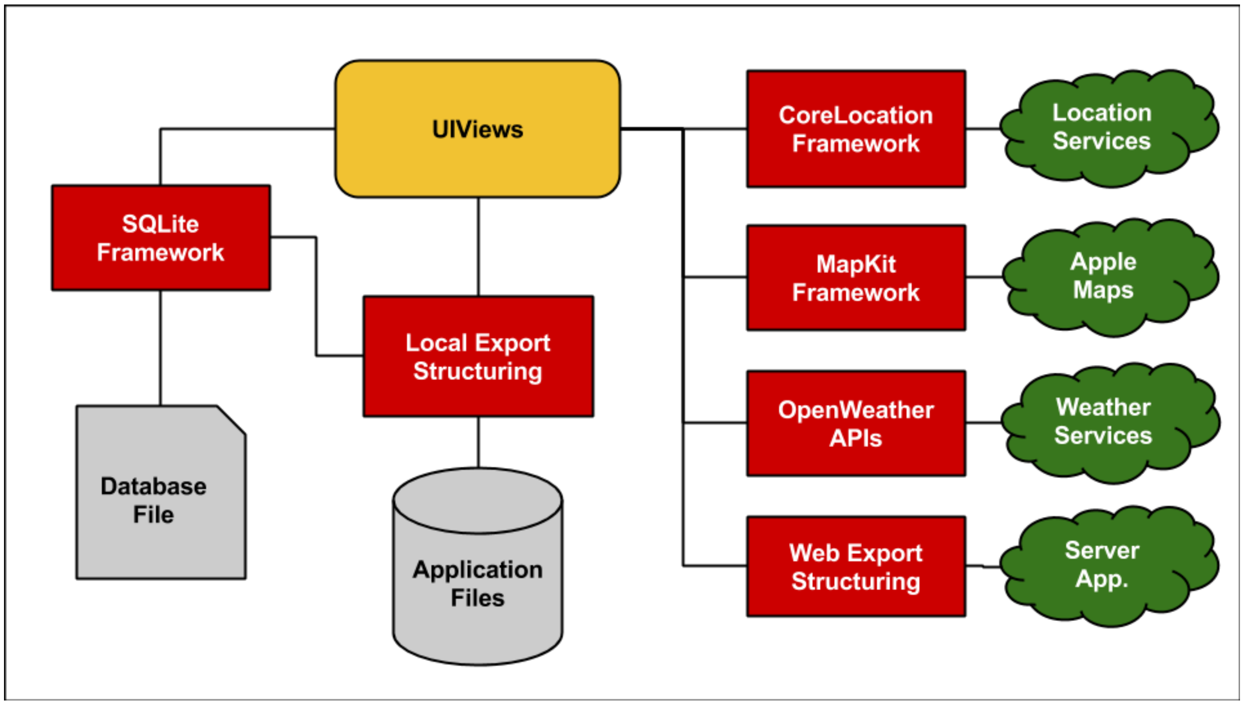


Figure B.

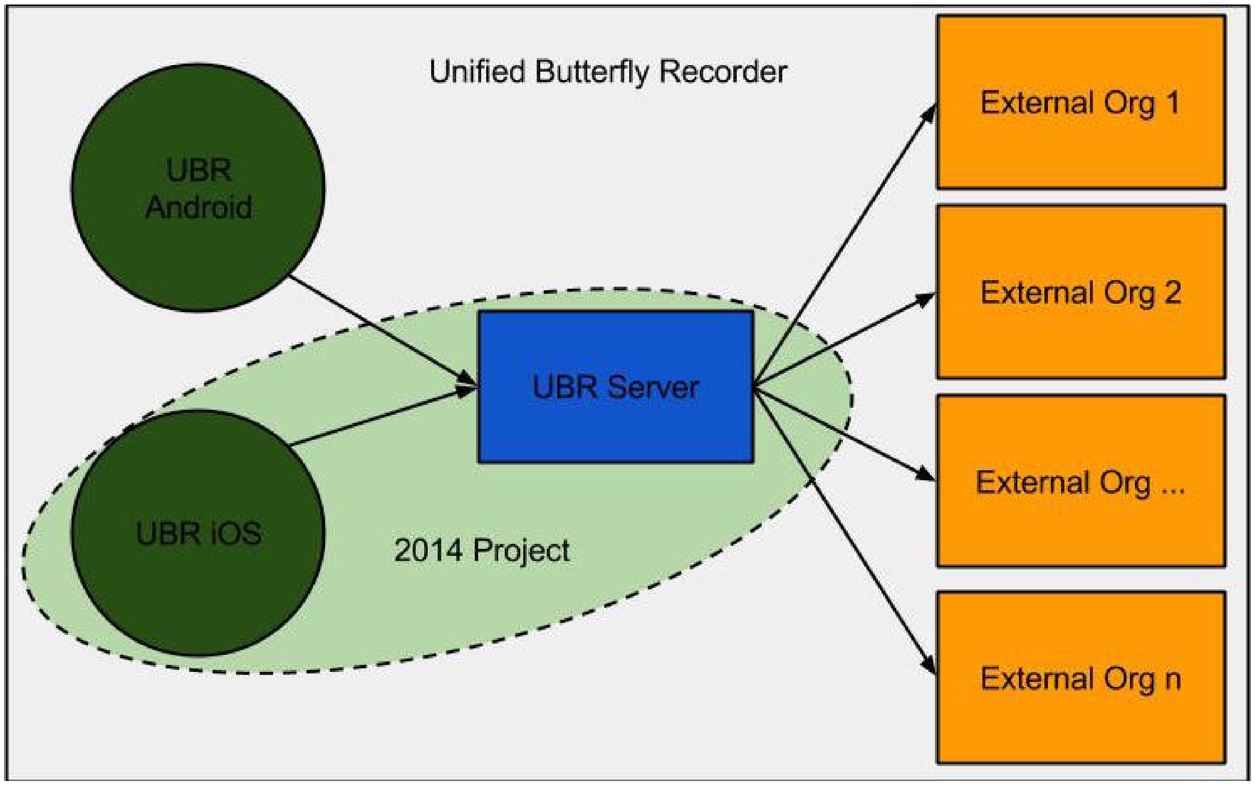


Figure C.

Server

Figure C., displayed above, serves two purposes. The first is to represent the full Unified Butterfly Recorder framework. The second is to illustrate the scope of this project. Originally, the scope of this project was limited to the development of the iOS application. When the Android Team was developing their application, they created a prototype server to receive data from the application via PHP as another export option. This idea never moved past a proof-of-concept phase in the Android project. After many discussions with our clients and advisor, we decided to take it a step further and create a server application that would serve as an interface for external organizations to receive data from and for both applications to send data to. We will explain why we chose this route later in the Detailed Design, Remote Server section.

Functional Decomposition

Views

In Figures A and B, these are represented by the yellow boxes. In Figure A, the interfaces are defined using XML, using Java Activity classes to define functionality of the interface. Each Activity class also defines how and when the Activity will transition to a different Activity using an object called an Intent. Within iOS, interface building is encapsulated in a Storyboard file. A storyboard is used to define each screen, or View Controller, place user interface elements like buttons and text fields, and define screenflow using transition objects called Segues. Each element is then linked back to an Objective-C subclass of the ViewController class, which acts as a delegate for callbacks made as a result of UI events monitored by each element.

Since a well-crafted application can appear similar and have similar functionality in both frameworks, it is easy to assume that programing between them is trivial. Contrary to intuition, an Android button behaves very differently from an iOS button. An Android Intent interacts differently from an iOS Segue. These differences manifest not only in the different programming languages, but in how each object interacts with the underlying application framework.

Interfaces and Services

In Figures A and B, these are represented by red boxes. Several items are also included here that are not shown, because they do not fit the description of an interface or service in the iOS framework as they did in the Android framework. This is because they are included in the CoreServices framework that governs all iOS applications.

Apple Maps API

The MapKit framework provided by Apple is used to display the user location, provided by Location Services. If there are any sightings associated with the survey, they are marked on the map as annotations and include information callouts that display choice information about that sighting when tapped. When the user starts a survey, a breadcrumbing trail is drawn that shows their walking path.

Camera

Apple provides a service that can call upon the camera to take a new picture or select an existing one from the phone’s Photo application. In our application, a sighting can include a photo and in some cases is necessary for validation when submitted to a third party organization’s database.

Local Export Structuring

A custom-written set of methods that structures the data housed in the dynamic SQLite file into a set of comma separated value (CSV) files. The user is able to specify which CSVs to create and then email the created files as attachments.

Location Services

The CoreLocation framework provided by Apple is the application's way of communicating with Apple's Location Services. Based on the network connection and device, the means of deriving a GPS gridpoint depend on the included device hardware, data connection, and WiFi connection.

Open Weather API

An external framework that provides an interface to supply the application with weather data for a given GPS gridpoint. The response is in the form of a JSON message and parsed into fields used by the Sightings and Survey tables. The data is requested and populated within the user interface automatically if Location Services are available, and require no interaction by the user.

SQLite

A third party framework (included as a usable framework by Apple) for emulating an SQL database within a special database file. The application uses this framework to interface with the database and perform CRUD operations on tables containing survey, sighting, breadcrumb, and miscellaneous information recorded and modified during a survey.

Web Export Structuring

A custom-written set of methods that structure the data stored in the SQLite database files as a JSON object that can be sent to and interpreted by our intermediate server. It is important that after we export the data, we retain the survey locally so that the user may edit the survey. If the survey is submitted again, it modifies the server with the modified fields.

Storage

Phone Storage

The device's internal storage is used to hold both photos taken for sightings and the database files used by the SQLite framework. The application contains two SQLite database files. The first contains all dynamically created information, such as surveys and sightings. The second houses all of the static data regarding possible butterflies to create a sighting for. The data in the second file is never modified by the application itself. This was a step made by the Android team to ensure that this data could never be corrupted during an application crash.

With iOS applications, each application contains a Documents folder, much like a desktop file system for a user. Applications are unable to access any file not contained in their own filesystem. This is contrary to Android devices, who have a publically accessible file system. iOS devices have no public filesystem at all. This constraint restricts certain functionalities available in the Android application, such as the idea of a custom butterfly search list.

<u>Server Storage</u>

A server and database were provided by the university for use with this phase of the project. This storage will act as an interface between the application and any external organizations that would make use of the data. The server also happens to host our team website. The database on the server has a similar schema to that of the application database. The server, however, is able to take full advantage of a MySQL database. More storage space on the server relative to mobile devices also allows for greater scalability to increased demand from external organizations.

Design Decisions

Location Services Support

The most important aspect of the Unified Butterfly Recorder applications is in how it simplifies the survey process through automatic data collection. The bulk of this automation relies on an accurate GPS gridpoint for the device's current location. We quickly discovered the major difference between Android and iOS support for user location determinations: Android devices contain a chip which iOS devices lack. The consequence of this difference is that an iOS WiFi device that is unable to connect to a WiFi network can not make any determination as to it's current position. An iOS cellular device that cannot connect to a WiFi network or get a reliable signal from it's cellular provider will encounter the same problem.

As one might imagine, many butterfly surveys take place out of the realm of a WiFi network, and possibly out of the realm of a cellular network. Many long conversations were conducted to address this problem. Several suggestions were considered, including the use of third party devices that interface with the device to feed it GPS data, as well as using the last known location in conjunction with the accelerometer and gyroscope on the device to perform rough calculations by hand. All such suggestions proved to be prohibitive. Third party devices rarely provide GPS to any application but the application it was designed for. Also, using device processing power to make on-the-fly GPS calculations would degrade application performance during heavy use, such as breadcrumbing. Such calculations are also difficult to code for background execution, i.e. when the app is not being displayed.

A breakthrough came when we collectively considered what type of person would be using an iOS WiFi device to perform a survey. This type of person most likely has an iOS cellular device as well that can create a WiFi hotspot. The user would then be able to link their WiFi device to this hotspot and get available location updates. This type of assumption makes the possibility of having no network access at all fairly low. This requirement has been accepted by the client. The application was already required to operate in the presence of no data connection, so we simply had to build standards of operation when a location could not be determined. This amounted to setting the location to null in data records and resting Location Services hardware after a period of failed location queries.

Server Side Processing

At the semester mark we were faced with the task to expand the scope of the project. Since we were aware that the previous design team had already made a proof of concept export to the database, we decided to expand on that and take a step in solving a larger problem. We wanted to solve the dilemma of how external organizations were going to incorporate our data into their systems without manually entering it into their systems and without any security vulnerabilities for their users. What we decided to do is create a Python Flask app on the server that would handle data coming in from both the Android and iOS apps. This data would then make the data into objects and distribute it to the other organizations. This however did come with more design choices, which compiled into more and more work.

Soon we began communicating with some external organizations who showed interest in our system, such as Butterflies And Moths Of North America (BAMONA), eButterfly, and others only to discover that the organizations did not have a means to take all of the data the applications were collecting. In fact, they were only interested in the data they would traditionally log, and would not associate any extra data with a particular sighting once it was in their system. This meant that some of the data the user was collecting, manually or automatically, was not going to be used and would ultimately be discarded. Our client had made it clear that they did not want to house any data or administer a database for any application data, because they did not want to be seen as a competing data collection organization. Doing so would discourage other organizations' surveyors from using the iOS and Android applications. From these problems and constraints, we decided that until someone would be willing to take all the data, we would store the information on an Iowa State server that would not be made public and not used for academia. This would ensure that such information would never be dissociated from any particular sighting, regardless of where it was exported and the schema of the destination database.

Once the structure and logistics of the system were determined, we needed to define how data would be submitted to the server and to a corresponding organization. Certain organizations need authentication to submit data to their server, and this provided yet another hurdle. At this point we could either take the username and password as user input from the device and then send it to the the organization, but decided that it would not be an acceptably secure method and requires too much effort on the part of the user. Our next viable option is less secure, but allows for export operations without requiring any device or personal identifier information from the user. Our idea is that we would let the organizations set a key and send it out to their user group. Upon submission of a survey, a user attempting to submit to a particular organization would need that organization's key. The server would then use this identifier to export data to the correct organization.

Unfortunately, we have been unable to get a clear idea of the whether this solution is acceptable. Although the data distribution to the organizations work is not complete, we feel that our communications with these organizations have not only improved our relation with them, but has made them more willing to work with us than ever before.

Detailed Design

Database (Schema)

Breadcrumb Table

- key - ID
- Columns
    - Survey ID → Links back to Surveys table

- Time - Time the breadcrumb was logged
- Latitude - GPS
- Longitude - GPS
- Accuracy - Accuracy of the location determination
- Speed - Speed at which the user was moving when the breadcrumb was logged

Butterfly Table

- key - ID
- Columns
    - Generic Name
    - Scientific Name
    - Common Family Name
    - Scientific Family Name
    - Common Subfamily Name
    - Scientific Subfamily Name

Sightings Table

- key - ID
- Columns
    - Survey ID → Links back to Surveys table
    - Generic Name
    - Scientific Name
    - Common Family Name
    - Scientific Family Name
    - Common Subfamily Name
    - Scientific Subfamily Name
    - Location
    - Number
    - Temperature
    - Wind Speed
    - Wind Direction
    - Cloud Cover
    - Time
    - Behavior
    - Gender
    - Condition
    - Comment
    - Wing Length
    - Illuminance
    - Relative Humidity
    - Pressure

- Transect
- Photo Name
- Ambient Temperature
- Mark Found
- Mark Added

Survey Table

- key - ID
- Columns
    - Type
    - Name
    - Start Time
    - End Time
    - Number of surveyors
    - Names of surveyors
    - Location Name
    - Comment
    - Habitat Type
    - Habitat Condition
    - Wind Speed
    - Cloud Cover
    - Temperature
    - Engagement
    - Viewing Radius
    - Uploaded
    - Online Temperature
    - Online Humidity
    - Online Time
    - Online Wind Speed
    - Online Wind Direction
    - Online Pressure
    - Online Sunrise Time
    - Online Sunset Time
    - Online City
    - Online Country
    - Online Max. Temperature
    - Online Min. Temperature
    - Online Cloud Cover
    - Online Rain
    - Online Snow

Transect Table

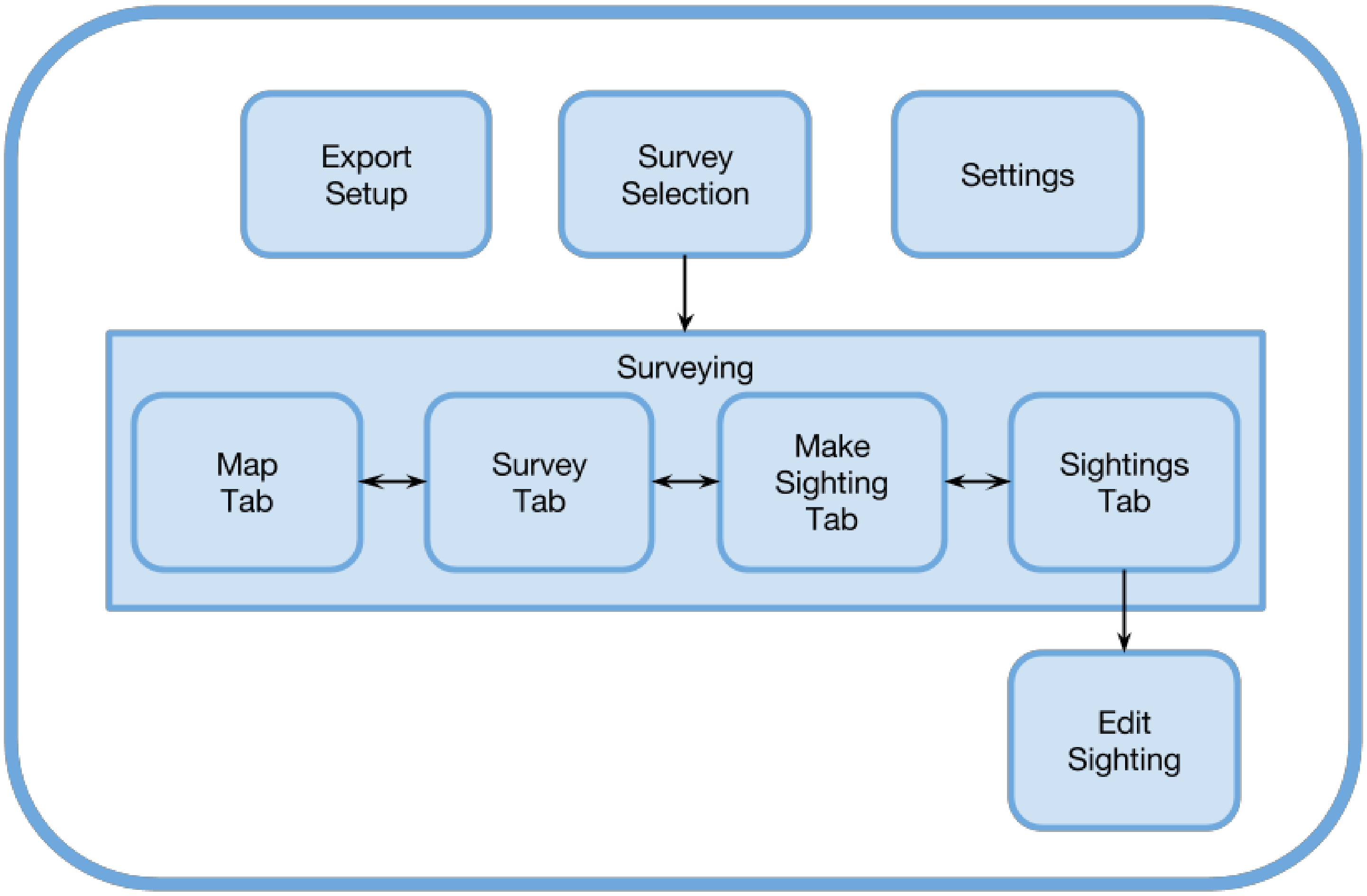
- key - ID
- Columns
    - Name
    - Description

User Interface

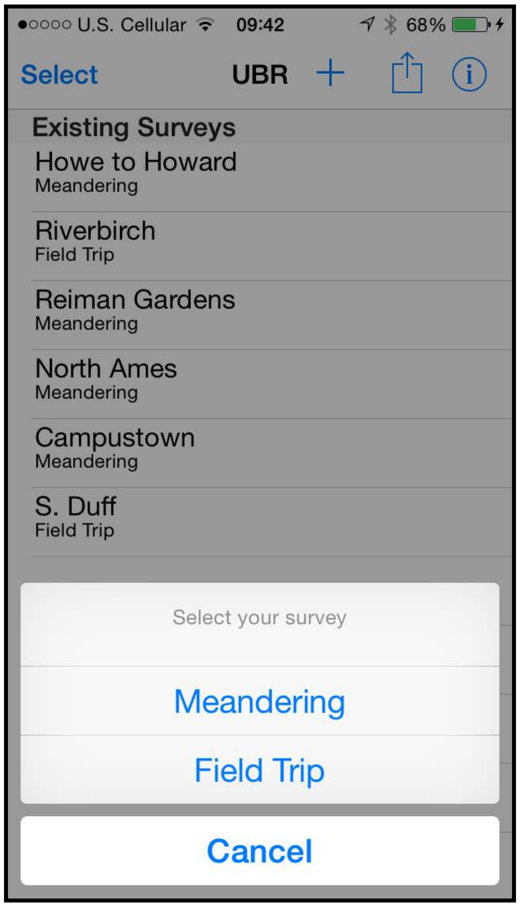
Target Audience

The mobile application targets a particular community, but there are many kinds of individuals that actually carry out a butterfly survey. These individuals can have widely varying levels of expertise, in both butterfly surveys and mobile application use. Following the Android application's model, we have strived to make this application easy to use and understand from its first use. This purpose has been further served by adhering to the Apple Human Design standards, which all seasoned Apple users are subconsciously aware of.

Screenflow Diagram



Page Descriptions



Existing Surveys

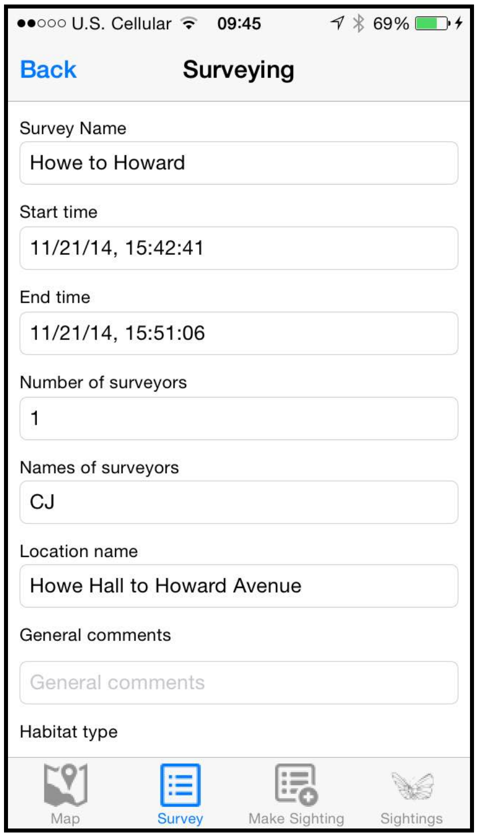
Howe to Howard	Meandering
Riverbirch	Field Trip
Reiman Gardens	Meandering
North Ames	Meandering
Campustown	Meandering
S. Duff	Field Trip

Survey Selection

The landing page when UBR is launched. From here, a user is able to select a survey or perform per-survey operations such as exporting and deleting. Users can also create a new survey of a particular type. Each type of survey has it's own method of capturing and using data.

Surveying

Once a user selects a survey, they are in a state known as "surveying." During this time, all operations, performed manually or automatically, are done with respect to the survey that was selected. Sightings and Survey informations can be modified at any time by navigating to their respective panels.



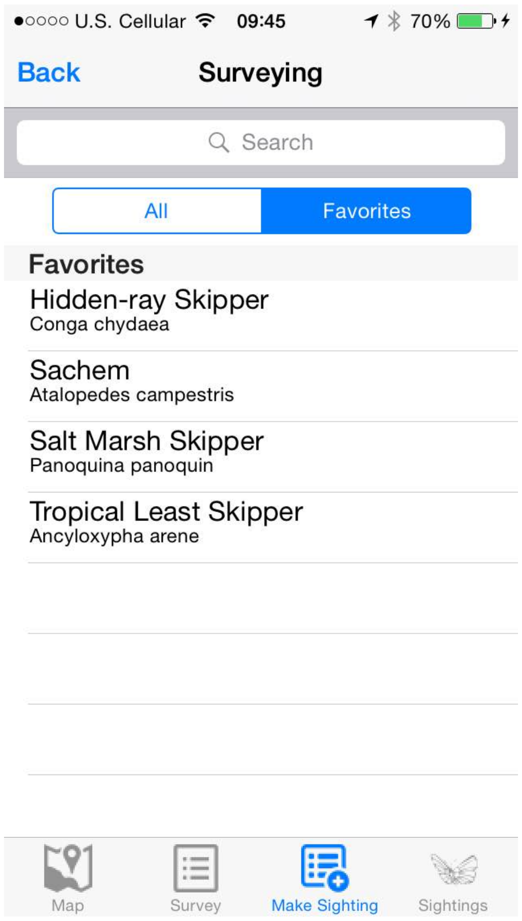
Survey Name: Howe to Howard
Start time: 11/21/14, 15:42:41
End time: 11/21/14, 15:51:06
Number of surveyors: 1
Names of surveyors: CJ
Location name: Howe Hall to Howard Avenue
General comments: General comments
Habitat type

Survey

The survey page may be different depending on which protocol is chosen. This tab allows the user to input some information unique to that particular survey. The same data will be collected automatically for all surveys. For data fields that are not present in a particular survey protocol, the database will register null values. When a user selects a survey, this page is automatically shown for all protocols except Incidental, as this information is typically the first to be edited. Certain fields can be set to auto-populate, such as surveyor name and location name.

Make Sightings

In this screen, the user is able to perform the most basic survey operation - to log a sighting. The list is impressive, but they can customize the list be selecting a region and subregion in the Settings page. They are also able to search by species name and family name. If they log a particular butterfly often, they can register that butterfly as a favorite. Tapping a particular row creates a sighting. They can tap anywhere in that row to increment the counter.



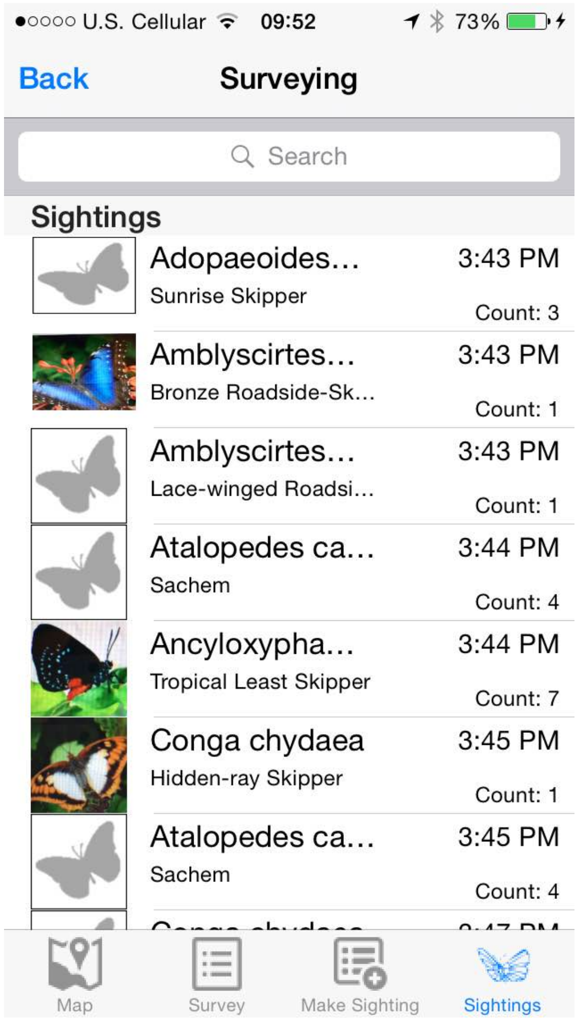
Favorites

Hidden-ray Skipper
Conga chydaea

Sachem
Atalopedes campestris

Salt Marsh Skipper
Panoquina panoquin

Tropical Least Skipper
Ancyloxypha arene



Sightings

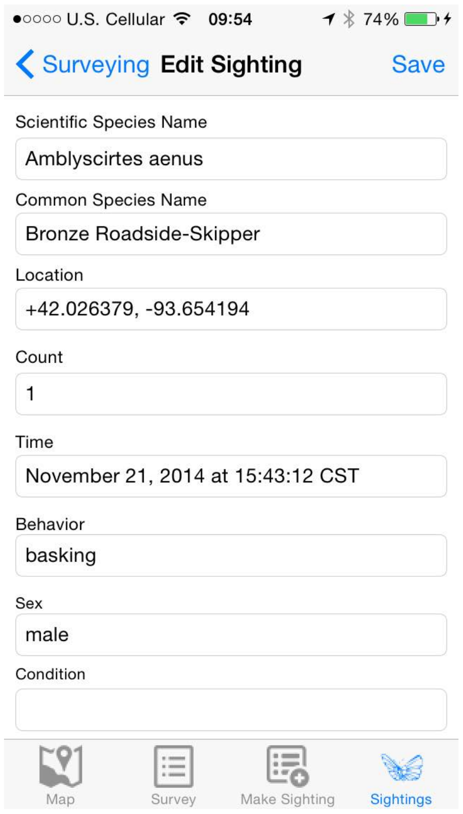
Adopaeoides…	Sunrise Skipper	3:43 PM	Count: 3
Amblyscirtes…	Bronze Roadside-Sk…	3:43 PM	Count: 1
Amblyscirtes…	Lace-winged Roadsi…	3:43 PM	Count: 1
Atalopedes ca…	Sachem	3:44 PM	Count: 4
Ancyloxypha…	Tropical Least Skipper	3:44 PM	Count: 7
Conga chydaea	Hidden-ray Skipper	3:45 PM	Count: 1
Atalopedes ca…	Sachem	3:45 PM	Count: 4

Sightings

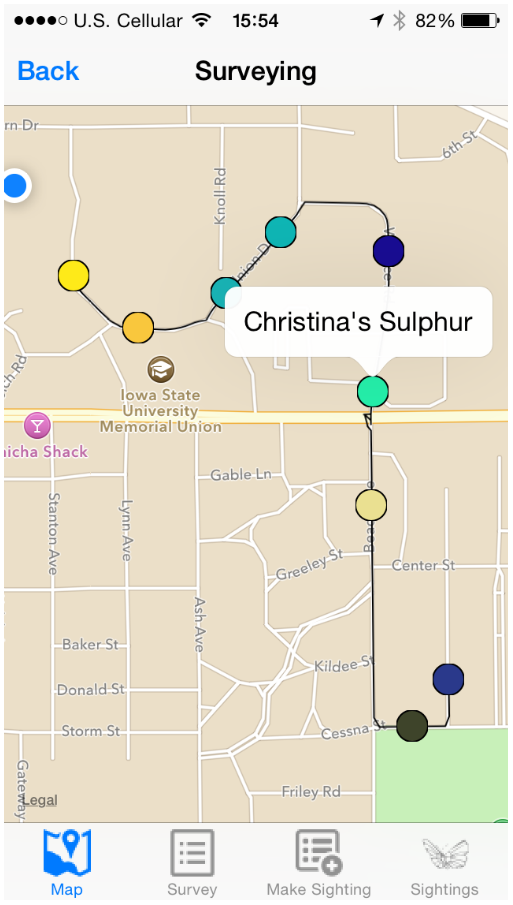
The sightings tab will show all of the sightings that a user has made for a particular survey. Here they can see a brief overview of the information for each sighting, include it's scientific name, generic name, count, and the time at which the sighting was logged. Long pressing a sighting will call up an alert that allows the user to delete the sighting that was pressed, incase a sighting was made mistakenly

Edit Sighting

If a user taps a sighting in the Sightings tab, the advanced details for that sighting is displayed. These details include manually entered information regarding the nature of the sighting, including gender, behavior, and other physical properties of the butterfly. Users can also take a picture and see some of the automatically populated information for the survey, if they choose. Changes can be saved at anytime by tapping "Save Sighting" or the back arrow to navigate back to the Sightings tab.



Scientific Species Name: Amblyscirtes aenus
Common Species Name: Bronze Roadside-Skipper
Location: +42.026379, -93.654194
Count: 1
Time: November 21, 2014 at 15:43:12 CST
Behavior: basking
Sex: male
Condition:



Map

The screen is a neat capability that is unique to a Unified Butterfly Recorder survey - the ability to plot the sightings made for the particular survey via annotations and trace out the path that the user walks via breadcrumbing. Each annotation will also display basic information about which sighting is being represented. The color is also unique to each species/family combination. Butterflies from the same family will have similar colors, but all colors are distinct.

Remote Export Support

Server Design

When we decided to create our server application we determined that it would be okay to use languages and frameworks we were more comfortable with since it will be our own project from scratch. Since, Eric Larssen had those most amount of experience in the particular field, he chose to create a Flask Python application which is a very lightweight and portable application. The application without data distribution is very simple as it takes the survey information from the mobile applications in the JSON format and enters the data into a SQLAlchemy model, which is an ORM for python. Once it is in a model, it becomes trivial to add to the MySQL database. Once the server handles distribution, it will have more functionality that will be able to be modified and sent.

Testing

Developing a mobile app requires testing the application logic and event handling as well as code integration with the user interface. App testing was performed in three general categories: validation testing within Xcode, interface and integration testing on developer devices, and field testing within the community. Automated testing was used wherever possible, but requirements still changed frequently enough that our main method of testing was scenario validation

Simulation and Logging

This phase of testing was focused on ensuring that code contained in each module responsible for application decision making was sound. This included accessing the onboard database, communicating with the various Apple frameworks to obtain and apply user location data, and dynamically generating data for use in populating fields in the user interface.

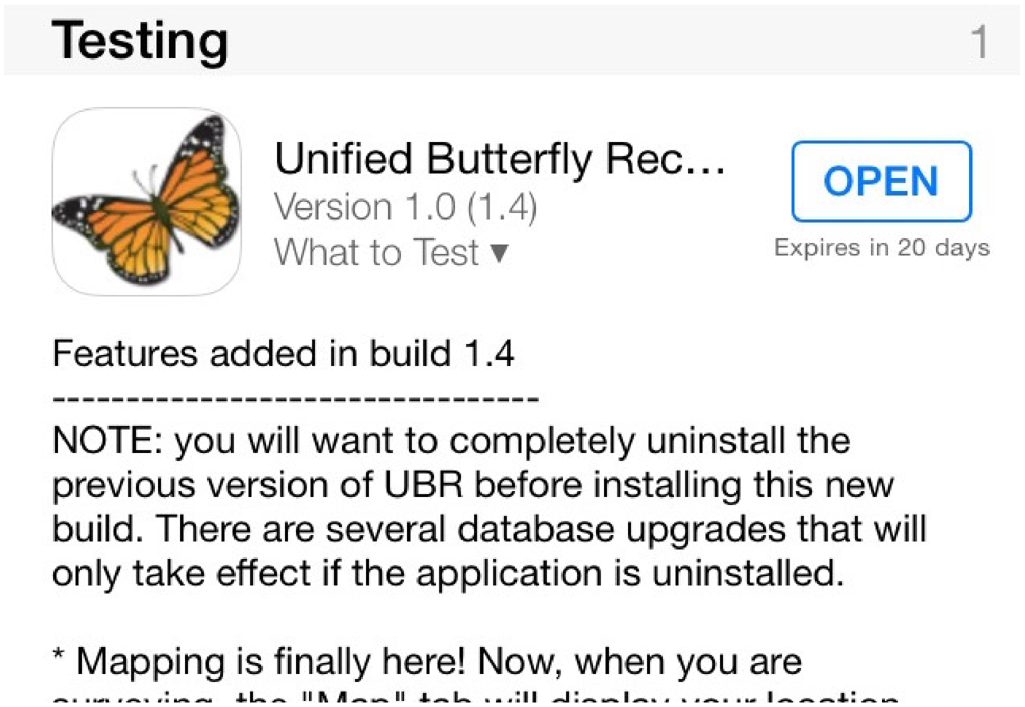
Testing utilized the use of the iOS simulator provided within the main Xcode bundle and logging tools used to communicate the result of unit tests. The simulator can also be used for basic user interface testing, but the simulator has some shortfalls that necessitate another phase of testing.

Developer Testing

During this phase, we as the developers would test the application on our personal Apple devices. The device would be connected to the computer running the application in Xcode, and we would be able to run use-case tests while monitoring the logging output utilized in the first phase of testing. The focus in this phase is user interface integration with the backend logic code.

Being able to test on an actual device provides several advantages over the provided simulator. The simulator is less robust in user location determination and has no support for certain features, like taking a new picture. Testing with a physical device before releasing to the community allows us to ensure that a certain subset of features is available for each new iteration.

Field Testing



Unified Butterfly Rec...
Version 1.0 (1.4)

Features added in build 1.4

NOTE: you will want to completely uninstall the previous version of UBR before installing this new build. There are several database upgrades that will only take effect if the application is uninstalled.

* Mapping is finally here! Now, when you are

Once the next iteration had been screened, we were ready to release it to the community for field testing. Testers included members of our client group at Reiman Gardens as well as several power users in different organizations. Implementing this system raises awareness for the application and keeps our development efforts in touch with the needs of the community as the app moves forward.

iOS 8 included developer support for a system called TestFlight. TestFlight is integrated within iTunes Connect, the Apple evaluation system used to eventually release an app to the app store. After initial setup, each new iteration, or bundle, is uploaded to iTunes Connect and immediately made available for download to registered TestFlight testers via the free TestFlight app.

Testers are able to read release notes for each new bundle and provide feedback to the developer email provided. New testers can be added at any time and invited to test the application.

Project Management

Standards

As described earlier, there are several ways to conduct butterfly surveys that are accepted by the industry. We have developer our app to support as many of these protocols as possible, and tailoring the user experience and relevant data fields to match the protocol that is being performed. Following these criteria ensure that we collect relevant data for each protocol.

Schedule and Timeline

The first few months of the our project was spent learning Objective-C and learning the Android version of the app to understand how it works. We also met with the client several times to understand the importance of the app and get an idea of their vision for the whole system. The database schema was implemented in the Spring, along with a few basic functionalities.

Unfortunately we were set back from our initial timeline of having our minimal viable product out during the summer. However, when we came back for the second semester we hit the ground running and released our MVP to a select set of power users that were able to use and test the functionality of the application. By the end of November we had 90% of the functionality complete of that of Android application, along with a deployed application to the server.

Work Breakdown

Team Contribution

As a group we decided it was very important to keep everyone informed, even if one of us had to miss a meeting, since our group consisted of only three members. This proved extremely valuable throughout the semester as everyone seemed to be on the same page. All group members were broadly informed about the development, but as with software development teams in industry, we each had a facet of the development that we were best at. CJ was in charge of device database interfacing, UI research, and location services integration. Sean was in charge of implementing the mapping and breadcrumbing services, while Eric focused mostly on the server application, server database, and interfacing with external organizations.

Team Roles

Person	Role
Eric Larssen	Team Leader, Scrum Master
CJ Mankin	Webmaster, Lead Developer
Sean Shickell	Communications, Documentation

Risks

Loss of a Team Member

In the situation of a loss of a team member, we would without a doubt become strained with the amount of work each remaining team member would take on. Since we have strived to keep every team member informed and involved, and we have great communication between members, we would not be forced to backtrack to understand the departing team member's work. Our progress would be significantly slowed, however, as an application's development is a heavy task for a group of two.

Lack of a Team Members Contributions

In all projects, in academia or the workplace, one group member performs some majority of the work, whether it is by a large or small margin. This person can often feel the weight of the team on their back. At the same time there can also be a person(s) that does little or nothing throughout the project and let the team down. Originally, we had not developed a plan of action for this scenario. Fortunately, our team dynamic changed from week to week to meet our changing requirements, so that certain members were able to contribute more than others on certain weeks. These contributors rotated often from week to week, so that no team members was doing the most or least of the work in the group

Loss of a Third Party Service

By careful design by both the Android Team and our team, we were able to stay loosely coupled from all third party libraries except of course the iOS platform and its standard libraries that our application runs on. The only external service that could possibly terminate support is the OpenWeather API, which serves as an automatic data collection tool and is not required for functional operation.

Communication

Email

Whether communicating with our advisor or clients, we seemed to always fall back to email. It may not get the credit it deserves in most projects, however it was used as our primary form of communication. Trying to organize an event with 5 people would otherwise seem impossible without the use of email.

Google Calendar

Although we did not use this to communicate through the traditional fashion, we started off the second semester by filling out our schedules on a common calendar so that when we needed to make a meeting, we could either have the person we wanted to make a meeting with make an appointment, or we could easily see the time slots available for the three of us to meet. This came in handy more than a couple of times.

Trello

Trello is a very open ended way of managing tasks across different subjects. In our case we used Trello as a Kanban board, in our Agile development. We keep a backlog, current iteration, open issues, and done board. Our backlog of issues are those that have yet to be started. Typically issues listed at the top are the next to be implemented in the next iteration. The current iteration issues are the those that are currently being addressed. The open issues are issues that need to be addressed by someone other than the person who initially was assigned to the ticket. Finally, the done column is where the finished tickets go.

We were able to use this board to communicate not only to our team but to our client the progress of certain tickets and if we had any questions we would tag them on an issue which sends them an email. We were not very traditional in the way our agile process worked because we could not simply work on the project everyday let alone meet everyday, so instead we met every week.

Summary

Acknowledgments

We would really like to thank everyone for their patience while working with us. There are many groups that were not as fortunate to have a client, or an advisor, who are as understanding as ours. Our team and project definitely had its rough patches, but we got through them with their help.

We would specifically like to thank Dr. Diane Rover for spending the time she has over the last year with us. Coming from a background of iOS development, she could appreciate the challenge of creating an iOS application with three developers who had little to no experience in mobile development. However, her honesty is what we are thankful for the most. When it came to reviewing our poster, documents, and presentations, her sometimes brutal honesty is what we needed to change the quality of our project from good to great. Finally, the guidance she provided was invaluable when it came to making the design decisions described previously.

Finally, we would like to give a special thanks to our clients, Nathan Brockman and Anita Westphal, who took time out of their busy schedules to meet with us every week. Their insight and forethought astounded us at times when we look back at some of the suggestions they had made early on that helped us in the long run. Meeting us every week meant that they got to see sometimes lots of progress made, and other times not so much. They certainly understood that some features come faster than others and were understanding when something was late. As experts in their field, we were constantly energized by their enthusiasm and their dedication to this project and our team, and it helped enforce the importance of this project.

Conclusion

From citizen scientists to experts in the butterfly field, we were tasked with creating an iOS application that would benefit all surveyors in creating a survey with the most accurate information with as little input as possible. Although many of the difficult design decisions we covered by our predecessor, we were faced with many decisions ourselves. The port from an Android application to an iOS application is not as easy as it may seem and although they may look visibly similar, it took much effort to make it so. Although the iOS devices are more similar to each other, they lack the capability of that of the Android and so compromises had to be made. These are but a couple challenges we faced in our two semesters, and have discussed the others in the document above.

After one semester of work, we decided to bite off more and try and solve a problem where organizations that use the app have to manually convert the data to their schema. In doing so, we have interacted with more than a few organizations who are willing to put in the time and work with us to do so. Although it seems that the bite we made was bigger than we had initially thought, we are excited to continue to work on it after our work is done on the app.

In the upcoming weeks, when the client gives us the go ahead, we will publicly release our application to the App Store with the full functionality of the Android version. Due to the fact that our application is only available to a small set of users at this time, we are unaware of how well liked the application will become in the community. Our client plans on taking the application to conventions and show the world how easy and powerful it is to use effectively.

Hopefully, adoption will become widespread enough to truly realize the idea of the Unified Butterfly Recorder.

Future Forward

Resources and Contributions

We are encouraging our clients to create their own Apple ID and development profile so that our team can continue to develop features that they requested. Because of the time frame it takes to release a product to the Apple App Store, we plan on becoming the maintainers of both the code repository and the applications. Any major features to the iOS application will be ran through CJ for now or until someone else comes along.

Improvements

As discussed in the document above, exporting to external databases is still a work in progress and will need continuous communication with each organization. Eric will continue his work on this following graduation in his free time.

Codebase and New Owners

All contributions to the UBR platform are under the Reiman Gardens Github organization. Each repository in the organization works differently, while maintaining excellent documentation. If another design team is chosen for the UBR platform or if the client hires other developers, we will relinquish our roles upon request and do our best to support their efforts.

Appendix A: User Manual

Getting the App

The Unified Butterfly Recorder (iOS version) will be available on the App Store.

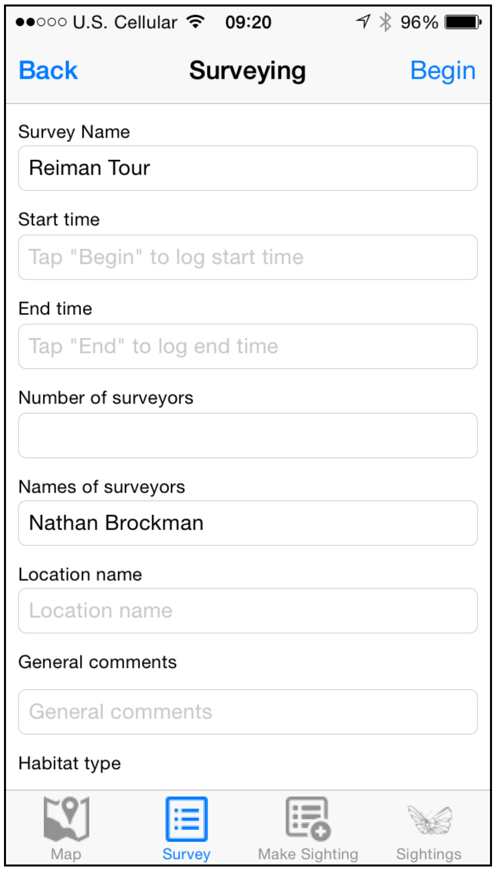
Basic Surveying

When UBR is first opened, it will begin on the Survey Selection page. If you have not taken a survey yet, the list of surveys will be blank. Tap the "+" button at the top of the screen and choose your survey type from menu that appears

Creating a New Survey

Incidental Protocol

If you selected Incidental as your protocol type, you will not be taken to the following Survey page. Skip to the "Adding a Sighting" section



Survey Name
Reiman Tour
Start time
Tap "Begin" to log start time
End time
Tap "End" to log end time
Number of surveyors
Names of surveyors
Nathan Brockman
Location name
Location name
General comments
General comments
Habitat type

Other Protocols

For all other protocols, you will be taken to the Survey page. Fill in as much information about the survey as you'd like.

- Survey Name - what the survey will be saved as (could be a name, a route on a location, etc.)
- Start/End Time - recorded automatically when you start and stop a survey, based on system time. These can be edited later if need be
- (Pollard protocol only) Transect Division Format - how transects are distinguished. Is each habitat a transect or is it determined by x meters.
- Viewing Radius (meters) - maximum distance from a surveyor that an individual will be recorded. This can be infinite if desired
- Location Name - name of the location being surveyed
- Number of Surveyors - the number of people participating in the survey

- Names of Surveyors - name of each of the participants
- General Comments - any other information to be logged about the survey
- Habitat Type - description of the habitat being surveyed (i.e. forest, prairie, etc.
- Wind Speed - current wind speed
    - Calm: 0 mph (smoke rising vertically)
    - Relatively Still: 1-7 mph (wind felt on face)
    - Moderately Windy: 8-17 mph (leaves and small branches move)
    - Windy: 18+ mph (small trees or large branches sway)
- Cloud Cover - current sky conditions
    - Clear - no clouds
    - Mostly Clear - less than half cloud cover
    - Mostly Cloudy - more than half cloud cover
    - Cloudy - full cloud cover
- Temperature - current temperature. Default units (Fahrenheit or Celsius) can be changed in settings
- Level of Engagement - amount of user attention given to the survey
    - 5: You are out just for the survey. No distractions
    - 4: Your main reason for being out is the survey, but there are some distractions (i.e. conversations with other surveyors)
    - 3: You are performing the survey alongside another activity (i.e. walking a dog)
    - 2: You are performing the survey as a secondary activity
    - 1: The survey was a last second decision that occurred upon seeing some interesting butterflies while out

Adding Sightings

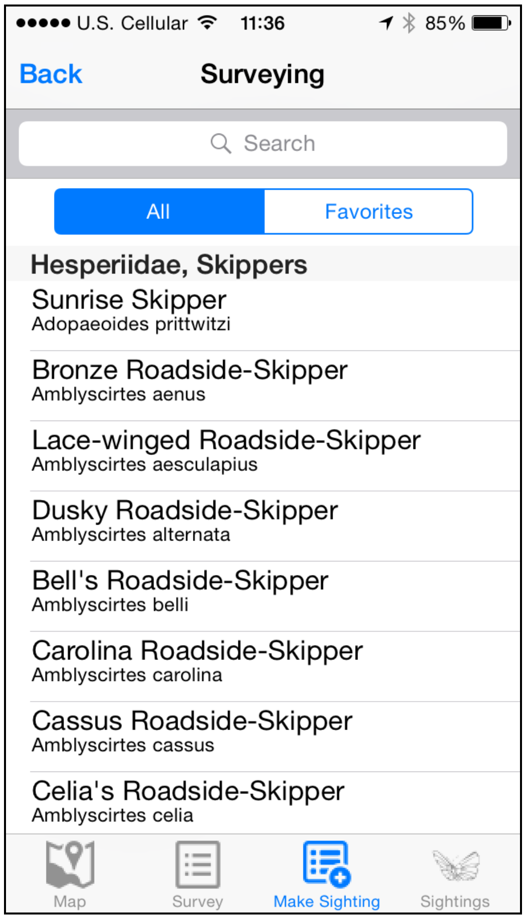
Incidental Protocol

Once you have selected the Incidental protocol, you will be brought to the Make Sighting tab. Making a sighting is the same as described below except that once you have selected a butterfly, you will be immediately taken to the Edit Sighting page, described later.

Other Protocols

Once all survey information has been logged, tap the “Make Sighting” tab from the tab bar located at the bottom of the screen. If you remember to log more survey information later, you can return to the Survey tab by tapping it’s tab from the tab bar.

On the Make Sighting tab, you can use the search bar at the top to search for a particular species or look through the full list. Each table section, accompanied by it's own header, groups butterflies of a common family. You can choose whether to list species by their common name or scientific name in Settings



Hesperiidae, Skippers

Sunrise Skipper
Adopaeoides prittwitzi

Bronze Roadside-Skipper
Amblyscirtes aenus

Lace-winged Roadside-Skipper
Amblyscirtes aesculapius

Dusky Roadside-Skipper
Amblyscirtes alternata

Bell's Roadside-Skipper
Amblyscirtes belli

Carolina Roadside-Skipper
Amblyscirtes carolina

Cassus Roadside-Skipper
Amblyscirtes cassus

Celia's Roadside-Skipper
Amblyscirtes celia

You can use table tabs to switch between viewing All butterflies and Favorite butterflies. To add a favorite, long press the table row for that butterfly in the All section and select "Add to Favorites." Favorites are typically used for butterflies that are seen frequently.

Tap a species in the list to record a sighting. Subsequent taps on the same species will record additional sightings. When tapped, the table row will expand to include a count stepper. Tap the "+" and "-" buttons to increment or decrement the count for that sighting. Incrementing can also be achieved by tapping anywhere inside the row that is expanded. After a period of inactivity, the row will shrink to it's original size. You do not have to wait for this to occur before making another sighting; tapping another row will automatically log the count for the previously made sighting. Each sighting is automatically created with the system time and GPS location, if available.

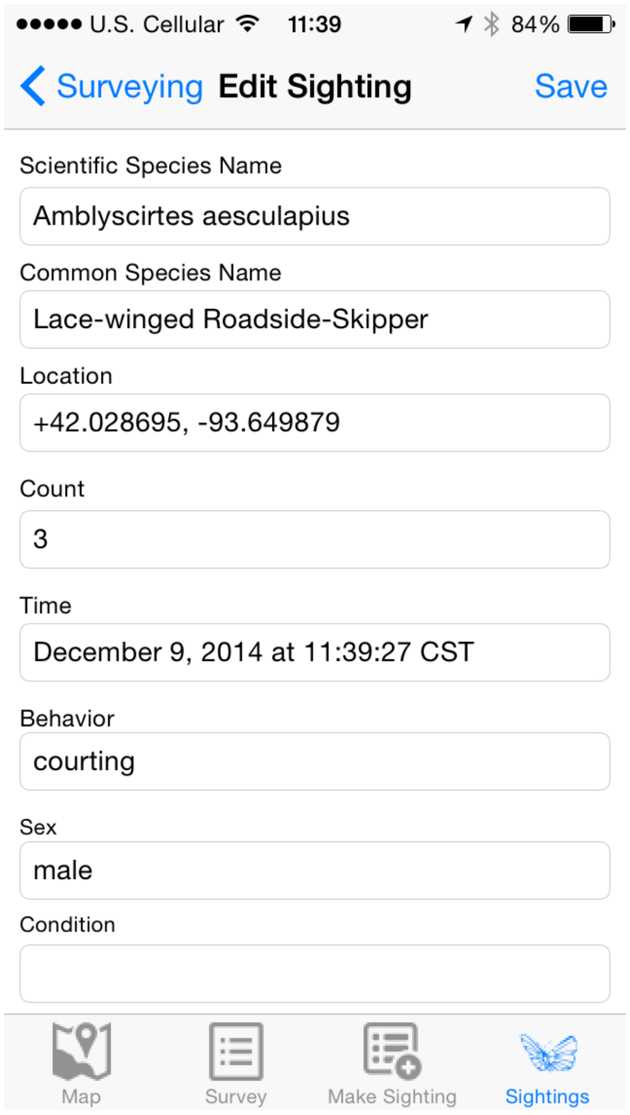
If a butterfly cannot be identified, there are Unknown butterfly species you can select instead. There is a general Unknown, as well as an Unknown for each family if you are able to narrow it down to a family.

(Pollard protocols only) The transect button at the top will let you determine which section or habitat on your route the sightings were recorded in. Be sure to update the selected transect when navigating between transects while surveying.

Editing Sightings

Incidental Protocol

Because Incidental protocols are used to record one species of butterfly, you will immediately be brought to this page once you have logged the sighting. You will not be able to view the list of sightings because of the nature of this protocol. The Edit Sighting page is the same as the other protocols



Scientific Species Name: Amblyscirtes aesculapius

Common Species Name: Lace-winged Roadside-Skipper

Location: +42.028695, -93.649879

Count: 3

Time: December 9, 2014 at 11:39:27 CST

Behavior: courting

Sex: male

Condition:

Other Protocols

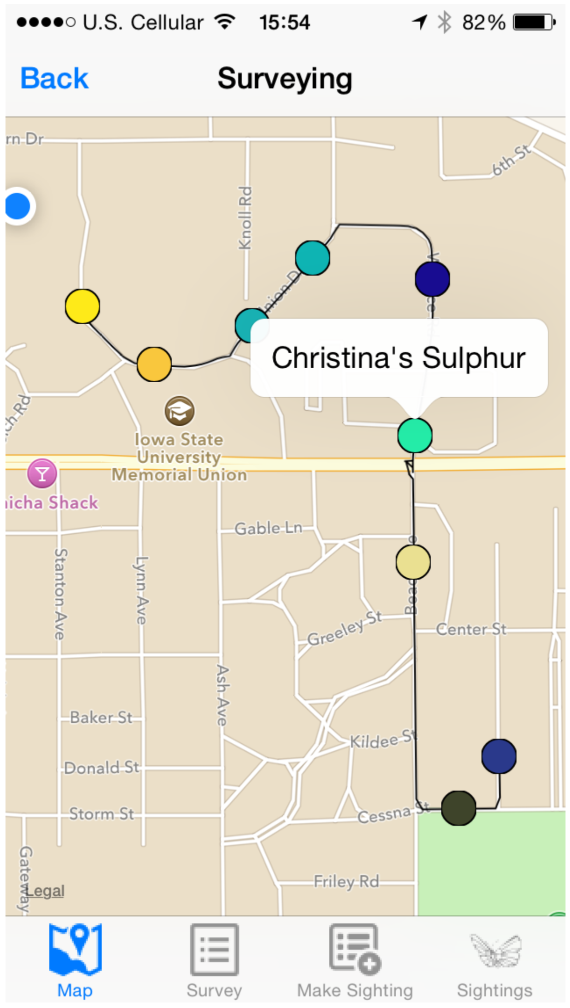
Tapping the “Sightings” tab from the tab bar will bring you to the Sightings page, which lists all the sightings associated with your survey. Each table row briefly summarizes the information contained. Long pressing a row will allow you to delete that sighting if it was made in error.

Tap on a sighting to view the Edit Sighting page. Editing allows you to change the species name, the number of individuals seen, as well as adding extra information. You can note interesting or unusual behavior, butterfly gender, or it’s physical condition.

Tap the “Take Photo” button to take a picture of this sighting with your device’s camera. This picture will be displayed next to your sighting and on the sighting’s edit page.

When you are finished editing details, tap either the “Back” button or the “Save” button to save and write your changes.

Mapping



The final tab to investigate is the Map tab, which is available for all protocols **except** Incidental. This displays a map of your current location (marked as a blue dot) and uses the GPS gridpoints of your sightings to plot their location.

The map view also uses your breadcrumb data to plot the path you've walked during your survey. This is done in real time as you view it or whenever the view is loaded. Use traditional pinch-to-zoom gestures to zoom in and out and slide your finger across the view to change the displayed region.

Ending a Survey

For all protocols **except** Incidental, survey's must be ended. When you have completed your route, bring up the Survey tab and tap "End" at the top. This will automatically log the end time and discontinue breadcrumbing operations to conserve battery life. All details are still able to be edited. You can even add additional sightings if you wish.