Design Document

Unified Butterfly Recorder iOS — May1614

Matt McKillip
Mason Berhenke
Blake Burns
DJ Todd
Kyle Long
Yu Jin
Eric Soland

Advisor: Dr. Diane Rover
Client: Nathan Brockman
Team Website: http://may1614.sd.ece.iastate.edu/

Contents

1 Project Overview

1.1 Abstract

There is a great app for recording butterfly sightings available for Android devices (Google Play store: Unified Butterfly Recorder), and now it's time for one to be created for all of those who love to use their beautiful Apple devices. Our team has been assembled so that researchers, graduate students, conservation workers, citizen scientists, hobbyists, and volunteers with minimal experience won't have to purchase an Android device to use this amazing new app. Which will record essentials like time (in Real-time), exact coordinates (within inches from help with technologies such as Cellular and Wi-Fi networks, GPS, and iBeacon), and local weather conditions at the time of the butterfly sighting (using more than 40,000 Weather Stations around the world) automatically using built-in functionality on their existing Apple devices. Devices like the iPhone (6s, 6, 5S, 5C, 5,), iPad (3rd and 4th Generations), iPad 2, iPad mini, and iPod touch (5th Generation) equipped with iOS 8 will be able to utilize all this app has to offer, making this app a must have for any butterfly enthusiast on an Apple device.

1.2 Background

Butterflies are great indicator species; from monitoring climate change to determining the quality of habitat restoration, butterflies are used a lot to help answer scientific questions. Across the nation and around the world, however, there is a lack of information on native butterfly species, their annual dispersal and numbers. To help with this task the scientific community has begun relying on citizen scientists, which are volunteers of different backgrounds, to go out in the field and do the surveys. In January 2013, Team Butterfly, Senior Design Team Dec13-08, took on the task of creating the Unified Butterfly Recorder (UBR) Android app. Throughout 2013, with the collaboration of the Reiman Gardens Entomology staff, UBR was designed, developed, and released on the Google Play store. This app has the potential to significantly and positively alter the course of global conservation research. It is being tested currently by researchers in the United States, Canada, Germany, the Netherlands and other countries around the world. In the spring of 2014, a comprehensive in-the-field study will be conducted, by RG Entomology staff, to test all aspects of the current application, as well as gathering information as to the usability of the app by researchers from different backgrounds, volunteers thru professionals. The app and results of this study will be presented at various conferences in the entomology and conservation fields. UBR iOS Development — Project Design Document Page 5 The release and demonstration of UBR in the conservation community has produced excitement about the ease of data collection, ability to standardize survey results despite variable collection methods and the capability to manipulate the analysis (mapping, graphing, other visuals for presentation) to benefit research and enhance conservation efforts.

1.3 Objective

Our goal for this project is to deliver a fully working iOS application compatible with Apple devices running iOS 8.0 or newer, with network connectivity. The app will be an updated, Apple version of the Android Unified Butterfly Recorder App, http://www.reimangardens.com/collections/insects/unified-butterfly-recorder-app/ with help from user feedback on the Android UBR App. As with many Apple apps, we want to make the iOS UBR App to be a beautiful as possible, and give it the "Apple" look and feel. There are tutorials and plenty of App Store examples of what an iOS 8 app should look like. One of our primary concerns while working on this project is to get a simple Beta version out before the last day of the Fall Semester, which would allow for the staff at Reiman Gardens to start testing and giving us feedback on the look and feel of the application in regards to the existing Android one.

2 System Level Design

2.1 System Requirements

2.1.1 App

The system will include an iOS app that will be identical in functional requirements to that of its Android counterpart, so far as the iOS hardware allows. One of our main goals is to make it appear as "Apple" as possible. The Apple device will need to communicate with the weather service, OpenWeatherMap, along with being able to interface with Google Maps. The app will only allow users to submit data to the database. The following was transferred from the previous group's website. (Check References) [1]

2.2 Functional Requirements

2.2.1 Android and iOS Applications

2.2.1.1 Synopsis

Allow users to record all relevant information of a butterfly sighting during a survey and submit that information to a database. The Android and iOS applications will conform to identical functional requirements, non-functional requirements, and output interfaces.

2.2.1.2 List of requirements

Must create a default list of common butterfly species based on the location of the survey. The user must also have the ability to submit anomalous sightings.

2.2.1.3 Data Collection

The following data points will be collected or calculated automatically by the app

- GPS
- Way point of routes
- Date and time
- Start/stop times
- Light levels
- Walking pace

2.2.1.4 User Input

The app will facilitate user input of the following data

- Tally of sightings for each species
- Habitat category
- Habitat conditions
- Data entry for mark recapture details
- Categorized behavior notes: Mating, nectaring, basking, puddling, perching, patrolling
- Site name
- Level of correct identification certainty
- Surveyor information (level of expertise, contact info, number of survey takers - recorders, observers)
- Taxonomic tree-based classification
- Miscellaneous comment section (animal life, plants present)
- Differentiation of sections along a route, distance/habitat category

2.2.2 Web services

The following data will be imported by the server for each record

- Temperature
- Wind speed
- Direction
- Percent cloud cover

2.2.3 Web Interface

Allow exporting of structured data (csv) based on queries

2.3 Nonfunctional Requirements

2.3.1 Android and iOS Applications

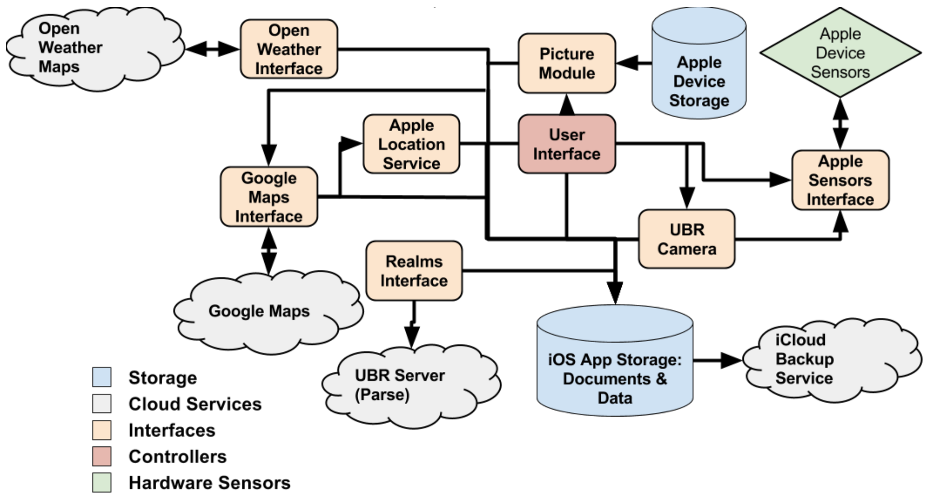
- Performance: The battery must last a minimum of 3 hours of surveying time on a recent mobile device with a quality battery.
- Ease of use: Users should require no training to record a sighting and perform a survey with the app
- Security: The authentication system must protect users' passwords and private information
- Graceful failure: The application should exit cleanly in the face of exceptional conditions, saving user data prior to exit, and not causing the device to hang or restart
- Form factor adaptability: The apps should function well on devices of all standard form factors
- Offline usability: The app should allow users to record data in the absence of a network connection
- Hardware adaptability: The app should still be usable on a device with a subset of the supported hardware features and sensors
- Minimal data transfer: The app should minimize the amount of network traffic necessary for submitting data to the server
- Network agnosticism: The app should submit data over wifi connections or cell networks
- Transactional data submissions: In the event of a lost network connection or otherwise failed data transmission, the app should retain all data locally and report a failed upload.
- Multi-task capability: The app should allow the user to move to another app in the middle of a survey or sighting and return without losing data
- Minimal resource usage: The app will minimize CPU and memory usage where possible
- Security: The app should not leak users' BAMONA account information during authentication, or store account information in plain text

2.3.2 Web server

- Availability
  - The database should handle up to 10,000 requests per day
  - The database should store up to 30,000,000 records

- Security
  - The server should not allow submissions from a mobile app to override data in the database
  - The server should not allow mobile apps to query the database
- Maintainability
  - The database should allow inclusion of new data fields

2.4 Block Diagram



2.5 Functional Decomposition

2.5.1 User Interface

The mobile applications will provide a graphical interface to the user of the application. The interface allows the user to add butterfly sightings into the system. Our main focus with the UI is to make data entry faster than paper and pencil. The application will collect numerous data points from the devices on board sensors, such as GPS location. The data will be stored locally until the user decides to upload it to the central database. The application will only allow for uploading new records to the database; as such, a user will not be able to pull data from database through the application.

2.5.2 Web Server

The web server is currently in development by the third group, UBR3. It will be using Parse in order to communicate easily between both Android and iOS and they are developing an API that we can use to implement the server calls we will need.

2.5.3 Database

We are currently testing with a SQLite 3 database, and investigating Core Data, the main Apple database framework, as a possible replacement. The data will imported and exported as CSV files.

2.5.4 Web Application

The web application was developed in PHP by the first UBR group using Drupal as a backend and database abstraction layer. The application provides users with the ability to perform queries on the survey data and export those results in a common format, such as CSV.

2.5.5 Weather Service

We will be using OpenWeatherMap, a free weather data and forecast API.

2.5.6 Mapping Service

Google Maps API will be used as our mapping system.

2.5.7 Storage Devices

We will store data in the App's "Documents and Data" storage. Items such as pictures, thumbnails, and CSV files. Pictures can be imported from the Apple Device storage into our App's storage.

2.6 System Analysis

2.6.1 Apple iOS Devices

When compared to Android, iOS devices don't have built-in GPS so we will have to use Apple's Core Location framework. Determining when to update location and turn on/off the location features is very important for saving battery life. Battery life is an important resource to consider when developing for iOS because Apple devices often have much smaller batteries than their Android counterparts. Since we'll still be implementing all of the features on the Android application, and hopefully a few more, we will need to be wary of our energy consumption and try to keep battery life at a maximum.

2.6.2 Mapping Service

We chose to use Google Maps over Apple's Maps because we have more experience with Google Maps and feel that it provides a better user experience.

2.6.3 Weather Service

OpenWeatherMap is a free service that provides weather data and has more than 40,000 weather stations worldwide (openweathermap.org), and is designed for web and smartphone applications.

3 Detail Description

3.1 I/O Specifications

Arrows indicate the direction of information flow

3.1.1 Apple device → Server

The iOS app will connect with a server code that will be written in a yet to be determined language.

3.1.2 Web server → Database

The web server will then connect with the backend server created by UBR3.

3.1.3 Weather service → Apple device

We will interface with OpenWeatherMaps periodically to get weather information, we will have basic information (temp, wind speed and direction, . . .) displayed in a widget on our app.

3.1.4 Mapping service → Apple device

We will interface with Google Maps to get mapping information, we will have basic information (small map display) displayed in a widget on our app. Suggestion from Nathan, our client to implement butterfly flight pattern, need further clarification.

3.1.5 User → Apple Device

User will be able to create a favorite list. User would be able to update sightings faster and simpler.

3.2 User Interface Specifications

Using the UI the user should be able to create , edit, and submit a survey to the local database and the project created by UBR3. With several different people expected to use the app, ranging from scientists to hobbiest, we will need to create an app that is easy to use while supporting a wide range of users. We do not yet have a concrete Screen Flow Diagram however we plan to follow many of the same screens and interfaces used by the Android app from UBR1 while following current designed standards given from Apple. This is in the hopes that an Android user would still be able to help an iPhone user navigate the app. We will continue to work on creating a basic screen flow over the next two weeks using screenshots from UBR for android, and then follow up with a new design for each screen following Apple's UI Design Basics whenever possible.

3.3 Hardware Specifications

3.3.1 Apple Device

The user must have an Apple Device that can run iOS 7.0 or newer, preferably on a 64-bit device. The device must have network connectivity, either wifi or cellular connection capability. We will assume the smallest screen size to be the size of the iPhone 5 (or 5S, SC). Our layout will change appropriately to incorporate more and give more space between items as the screen size of the device being used is larger.

3.3.2 Server

The server will interface with the phone and the BOMONA database and will need to always be on, this was set up by the previous group and is something that we will leave in their control until other arrangements are made. Depending on user size, we will probably need to increase the amount of servers over time to meet load needs.

3.3.3 Third Party Hardware

Google Maps and Open Weather Maps provide servers, weather stations, network devices, and more that will all be used to get information for sightings. These are very large providers which have excellent track records for always being up and running. The same can be said of BOMONA (based on the previous group's choice to work with them).

3.4 Software Specifications

3.4.1 Operating system

This app will be designed to run on iOS 7.0 or newer.

3.4.2 Server

The server will interface with the phone and the BOMONA database and will need to always be on, this was set up by the previous group and is something that we will leave in their control until other arrangements are made. We will interface with the Server using SQLite3, and we are currently testing a SQLite3 database with a test app. (Check References) [7]

3.4.3 Database (Schema)

3.4.4 Third Party Software

We will interface with Google Maps, Open Weather Map, and Parse using API's and functionality similar to that of UBR1.

3.5 Simulations and Modeling

3.5.1 Unit tests

We will write unit test suites for the app using the Xcode Unit Test target. We will also use Xcode 5's coverage features to measure the code coverage of our unit test suites.

3.5.2 Database simulation

UBR3 is currently working on a database interface which will use Parse to easily communicate between Android and iOS. For our own testing purposes we will be able to use SQLite to simulate a sample database of butterfly information locally on the iOS device. O

3.5.3 penWeatherMap simulation

We also have an online tutorial that walks through how to integrate OpenWeatherMap API into an iOS app. We plan on creating the example app provided in the tutorial to see all the coding needed to integrate Open Weather Map into our app.

3.5.4 UI Mockup

We plan to have mockups of the UI complete by the end of this semester in order to get much needed feedback on the look and feel. This will give us essential time during the spring semester to develop the rest of the application and get it released successfully on the AppStore.

3.6 Implementation Issues and Challenges

One of our bigger challenges is getting familiar enough with iOS programming to have a prototype out by the end of the semester and communicating within our large group of 7 members. Our biggest implementation issue may arise while working with UBR3's API which we must use in order to upload butterfly surveys. There is also a very looming danger of the time crunch of finishing development within the timeframe of the rest of this semester and next semester. Since we are all involved in other classes, work and campus organizations, we must consistently keep our development on track and keep working on it every week if we are going to succeed. Our second semester will be following the Agile sprint methodology where we set bi-weekly sprint goals that we must reach in order to stay on track.

Other issues may arise with the use of smart watches, particularly in the lack of documentation due to smart watches being relatively new. With this new technology, we may face more hurdles with testing and being unfamiliar with certain technologies with the smart watch.

3.7 Testing, Procedures and Specifications

3.7.1 Unit Tests

We plan to create and implement unit testing for all separate modules of our code. This will allow us to create a high quality app, decrease development time, and decrease debugging time. We plan to use Apple's XCTest unit testing framework. This framework will work natively in Xcode, which will minimize the testing setup for each developer.

3.7.2 Continuous Integration

We plan to use our unit tests so we can follow a continuous integration workflow. The goal of continuous integration is to catch problems quickly and easily, enhance collaboration, and broaden test coverage. We plan to use the continuous integration native to Xcode or the Xcode Jenkins plugin to minimize startup time, and allow the developers to use a single platform for all of the developing.

3.7.3 Integration Testing

Since our application will have many different communication paths to outside of the application. We will need to implement integration tests as well to test our modules with a connection to UBR3's server, GPS location, weather location, and others.

3.7.4 Beta testing

We plan to have a rudimentary application by December 2015. We aim to load the application on a few devices and allow the clients and developers to do informal beta testing over Winter Break. After the first run of beta testing, we will gather results, and make changes. Hopefully we will be able to launch a second beta test to gather feedback once more before launching our application.

4 Conclusion

Our application is in high demand from the client. The Android version is very popular, and organizations are waiting to adopt the application until both iOS and Android versions are published. Our team will develop an application with all of the functionality of the Android application with an iOS feel to it. We hope to go above and beyond the current application and add additional features by request from the client. We will also be working with the UBR3 team to store the data collected from our application. We believe we have a good plan and design in place to create a consistent, yet innovative experience when compared to the Android version.

5 References

5.1 Continuation

This project is a continuation from the Butterfly Population Survey app (butterflies.ece.iastate.edu), and therefore we are working with the previous team on several things. I found a few things overlapping between our projects and thus a few things and been moved over from their Design Document to ours. Below is what was transferred and the rationale behind why these items have been incorporated into our Design Document.

5.2 1, 2. Functional and Nonfunctional Requirements

These were transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-designdoc.pdf Background Functional Requirements and . . . Nonfunctional Requirements and not modified as we are trying to have the same functionality regardless of platform.

5.3 3-5. Mobile applications, Web server, and Web application

These were transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-designdoc.pdf System Design Functional Decomposition Mobile applications and . . . Web server and . . . Web application. For Mobile applications we changed the title to “User interface” to better match our block diagram. For the Web server and Web application, we simply inserted the text “was set up by the previous group and” and “by the last group” and alter some text to insure the inserted text matched properly with the syntax. The reason for this being that it gives credit to the previous group and lets the reader know that those items have been set up, and that our job then becomes interfacing with these items properly.

5.4 6. I/O Specifications

These were transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-designdoc.pdf Detailed Design Component Interface. For this portion we mimicked the layout and transferred the “Web serverDatabase” as is since both of those were set up by the previous group.

5.5 7. Database (Schema)

This part was transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-designdoc.pdf Detailed Design Database. Since they set up the database and we plan on interfacing with the one they set up, we have to abide by the schema they set up.