Unified Butterfly Recorder

iOS Development

Design Document

Senior Design Team Dec14-16

ADVISOR: DR. DIANE ROVER CLIENT: NATHAN BROCKMAN
TEAM MEMBERS: ERIC LARSSEN (TEAM LEADER, SCRUM MASTER), SEAN SHICKELL (COMMUNICATOR, DOCUMENTS, GUI), CHARLES MANKIN (WEBMASTER, DATABASE)

Project Charter

Project Name	Unified Butterfly Recorder iOS Development
Performing Division	iOS Development Division
Performing Group	Senior Design (EE/CprE/SE 491) Group Dec14-16
Product	UBR App for the Apple App Store

Prepared By

Document Owner(s)	Project / Organization Role
Eric Larssen	Team Leader, Scrum Master
Sean Shickell	Communications, Documents, GUI
Charles Mankin	Webmaster, Database

Project Charter Version Control

Version	Date	Author	Change Description
0.0	3/12/14	Sean Shickell	Document Created
0.1	3/25/14	Sean Shickell	Document formatted and properly filled with any pertaining information regarding this project and the Design Document Rubric

Table of Contents

Project Overview

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 functionalities on their existing Apple devices. Devices like the iPhone (5S, 5C, 5, 4S, 4), iPad (3rd & 4th Generations), iPad 2, iPad mini, and iPod touch (5th Generation) equipped with iOS 7 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.

Background

Butterflies are great indicator species; from monitoring climate change to determining 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 of different backgrounds, volunteers thru professionals. The app and results of this study will be presented at various conferences in the entomology and conservation fields.

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 analysis (mapping, graphing, other visuals for presentation) to benefit research and enhance conservation efforts.

Objective

Our goal for this project is to deliver a fully working iOS application compactible with Apple devices running iOS 7.0 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 7 app should look like.

One of our primary concerns while working on this project is to get an Alpha (maybe a Beta) version out before the last day of the Spring Semester, which would allow for the staff at Reiman Gardens to start testing and giving us much needed feedback. The main reason we need an early release is because the time frame for butterfly sightings doesn’t align well with the time frame set out by our Senior Design class. Since our goal in this project ‘should’ be to meet the customer’s needs, we are going to try to keep to a schedule better aligned with what the customer wants rather than what our class determines.

System Level Design

System Requirements

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]

Functional Requirements

Android and iOS Applications

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.

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.

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

- GPS
- waypoint of routes
- date and time
- start/stop times
- light levels
- walking pace

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, # of survey takers - recorders, observers)
- taxonomic tree-based classification
- miscellaneous comment section (animal life, plants present)
- differentiation of sections along route, distance/habitat category

Web services

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

- temperature
- wind speed
- wind direction
- percent cloud cover

Database

Allow authenticated submissions through Drupal's XML-RPC interface

Web interface

Allow exporting of structured data (csv) based on queries

The following was transferred from the previous group's website. (Check References) [2]

Nonfunctional Requirements

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

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

Block Diagram

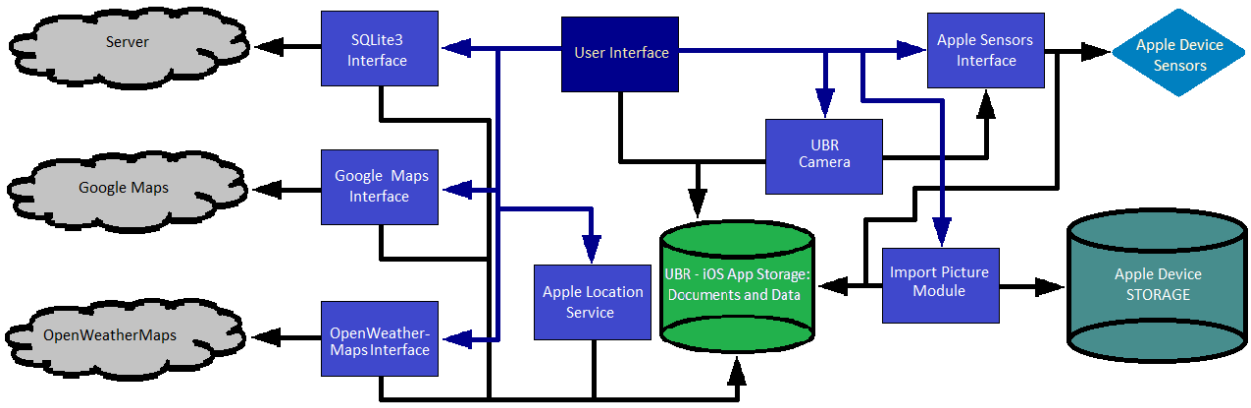

Fig. 1 UBR iOS App Sytem Architecture

UBR iOS Sytem Architecture: User Interface Controller (Dark Blue), data interfaces (Blue), App Storage (Green), Device Sensors (Sky Blue), Device Storage (Teal), Network-Connected Devices/Providers

The following (Mobile applications, Web server, Web application) were transferred from the previous group's website. (Check References) [3-5]

Functional Decomposition

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. 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.

Web server

The web server was set up by the previous group and will expose a CRUD interface for the mobile applications. This means that the web server will provide a means through which the applications will be able to authenticate users, upload survey records. The server will also provide a means through which the users would be able to access and export the survey records in the database. This functionality will not be provided to the mobile application. The users will interact with the web application on the server to submit queries and export the data that is returned from those queries.

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.

Web application

The web application was developed in PHP by the last 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.

Weather service

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

Mapping service

Google Maps API will be used as our mapping system.

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.

System Analysis

iOS app

We were assembled to build an iOS version of this app, and so we didn't have much choice there. However, since there was already a lot of research into mobile platforms done by the previous group, and they covered the Android portion of the mobile platform devices (and it having such a great response) it makes sense to cover Apple devices running iOS (since it's the next biggest, after devices running Android).

On the other hand, we chose iOS 7 to be our target operating system, because of many complex, yet very simple reasons. One being iBeacon, which is a Bluetooth Low Energy (BLE) location software/hardware that is just getting released with iOS 7 (along with a few others that have great potential with our app in more than one way). Another, is the fact that our app will be released in the Fall, when iOS 7 will be a familiar operating system in the mobile realm, and other mobile services (such as iBeacon) will have had time to mature and better integrate with 64-bit Apple Devices running iOS 7. Moreover,

iOS 8 will be getting released at some point with its hardware compliment, the iPhone 6 (which could possibly get released in June or July). This would improve/add functionality to our app correlating with improvements in the hardware & software coming from Apple and third-party companies.

We have had talks about some of the hardware available and that a lot of Android devices have more sensors than Apple products. However, a lot of the main things are available on most smart phones. Those being, GPS, date & time, and camera. Using these in combination with the OpenWeatherMaps and Google Maps, we can store crucial information such as approximate locations, time-stamps, and weather conditions during sightings.

Mapping service

We chose to use Google Maps over Apple's Maps, because we feel that Google keeps better updated with road construction and other physical changes.

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. This felt like an obvious choice to get weather information from.

Web component, BAMONA, etc.

The web application and the arrangement with BAMONA were set up when the previous group was developing the Android portion of this project. We plan on keeping them the same and making the proper interface to properly communicate with them.

Detail Description

I/O Specifications

Arrows indicate the direction of information flow. Check references [6]

Apple device → Server
We are currently testing with a SQLite 3. The data will be exported as CSV files to the database.
Web server → Database
The web application will communicate with the database using Drupal's database abstraction layer in PHP. Both of these were set up by the previous group. (Check References)
Weather service → Apple device
We will interface with OpenWeatherMaps periodically to get weather information, we will have basic information (temp, wind speed & direction, …) displayed in a widget on our app.
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.

User Interface Specifications

The user will use the GUI to create, edit, and submit a survey to the database. We plan on many different types of people using this app, from Researchers and conservation workers to people who just started butterfly tracking as a hobby. Therefore, we need to make the app easy to use and understand (which is the "Apple" way), and yet with enough capability and functionality to support the advanced users. Currently we don't have a concrete Screen Flow Diagram that we have decided on, and so we might make a different UI to support different users (e.g. – "Basic UI", "UI with most used tools and features", "UI full version"). At this point we are leaning to creating just one UI (which would be the "UI full version"), and then waiting until later to see about creating the others ("Basic UI" and "UI with most used tools and features"). By waiting until later, we could see what percentage of the users would prefer a simpler UBR UI (something like "Basic UI") after using the current UI we will have implemented.

As mentioned previously, one of our main goals is making our app appear as "Apple" as possible. Since the functionality might also be getting updated (as per user feedback of the Android version), we are planning to develop the iOS version with the user feedback in mind.

Moreover, simplicity and beauty are other fundamental goals were trying to accomplish. We plan to conform to Apple's UI Design Basics: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/, while developing our app to keep it simple, and use proper color, graphics, text, and object size to effectively show content without overcrowding the UI screen.

Hardware Specifications

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.

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.

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).

Software Specifications

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

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]

Database *(Schema)*

Main Table

- key - Record_ID
- Columns
  - Record_ID → Links into Environment and Pictures
  - Surveyor_ID → Links into surveyor Information
  - Survey_ID
  - Start_Time - datetime
  - End_Time - datetime
  - Site_Name / Route_Name
  - Latitude - GPS
  - Longitude - GPS
  - Amount_of_Butterflys_Seen

Weather Service Environment Table

- key - Record_ID
- Columns
  - Record_ID
  - Temperature
  - Wind_Speed
  - Wind_Direction
  - Cloud_Cover

User Environment Table

- key - Record_ID
- Columns
  - Record_ID
  - Habitat_Type/Category
  - Section_Number
  - Temperature
  - Wind_Speed
  - Wind_Direction
  - Light_Level
  - Cloud_Cover
  - Comments

Surveyor Table

- key - Surveyor_ID
- Columns

- Surveyor_ID
- Surveyor_Name
- Surveyor_Email
- Surveyor_Password
- Surveyor_Phone
- Surveyor_Organization
- Surveyor_Comments

Pictures Storage

- key – Record_ID
- Columns
  - Record_ID
  - Path_to_Photo

Third party software

We will interface with Google Maps, Open Weather Map, and BOMONA using APIs and a similar method to the previous group (in regards to BOMONA).

Simulations and Modeling

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.

Database simulation

We currently have the SQLite3 database part (which includes being able to import data from a CSV file) ready and will soon be integrating it with a working iOS application that allows users to modify the data, and create tables (which is needed for the UBR database). We are also looking into Core Data, the main Apple database framework, as a possible replacement.

OpenWeatherMap 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.

Alpha Release (for demoing UI) OR Screen Flow Diagram with layout structure

We plan to have at the bare minimum an Alpha, or 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 Summer to tinker with the GUI and develop a Beta of the app by July.

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. Furthermore, the learning curve for iOS development is a lot steeper than learning Android (and that Android is written in Java, which most ISU students in CprE/SE know). This, coupled with the fact that no one in the group knew Objective-C prior to this project, is going to add more time requirements each week for learning purposes. Another challenge being that we have a group of 3, not 4, and so, each person on the team has to pick up the extra slack of not having the recommended 4th person.

Another implementation issue we're facing is the fact that Apple doesn't put many sensors in their phones and tablets as some of their Android competitors. Which might lead to decreased functionality for the iOS version vs the Android version. This could also mean that they would not be identical for their functional and nonfunctional requirements.

In addition, with this being the second project (the first group never started the iOS app) the web server, web application, database, and one mobile application interface have already been set up, which means we won't have any freedom to choose what would best fit our needs in this project. Furthermore, these items were set up while the previous team was developing the Android app, meaning that these items are probably more suited to the Android app.

Testing, Procedures and Specifications

Unit tests

We plan to write unit test suites using the Xcode Unit Test target (and possibly some third party programs), and use tools provided by Xcode's to measure the code coverage of our unit test suites. Xcode provides extensive coverage features, trying to provide all a developer needs to create amazing iOS apps.

Beta testing

If we don't have the app ready at the end of this semester, then we will plan to have it out by no later than the beginning of July, allowing Reiman Gardens to perform surveys in the field using their Apple devices and giving us our first user feedback.

Conclusion

The production of this app is demand driven. There was (is) so much excitement about the production of the Android version of UBR, it was only a matter of time before this app was expanded to include iOS devices.

This app will definitely deliver more than expected. Our team will develop this app with all iOS has to offer and integrate Google Maps and Open Weather Map for up to date mapping and weather information.

We will also be in contact with the previous group, so that even before we're finished, we can make adjustments to menu layouts, settings, tools, and more based on current user feedback of the Android version and what the previous team sees fit.

This, combined with implementing Apple's UI Design Basics, will help us keep the design simple, intuitive, and elegant, while still giving all the functionality available to Android users.

References

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.

1, 2. Functional & Nonfunctional Requirements

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

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

These were transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-design-doc.pdf → System Design → Functional Decomposition → Mobile applications & … → Web server & … → 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.

6. I/O Specifications

These were transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-design-doc.pdf → Detailed Design → Component Interface. For this portion I mimicked the layout and transferred the "Web server→Database" as is since both of those were set up by the previous group.

7. Database (*Schema)*

This part was transferred directly from http://butterflies.ece.iastate.edu/files/butterfly-design-doc.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.