Unified Butterfly Recorder Design Document

butterflies.ece.iastate.edu | butterflies@iastate.edu

Senior Design Team Dec13-08

Curtis Ullerich (Document Master)
Ryan Scheel (Android master)

Cameron Whipple (Database Master)
Julie Tillman (Communication master)

Client: Reiman Gardens
Nathan Brockman and Anita Westphal

Advisor: Dr. Diane Rover

Table of Contents

Background

Concept Sketch

Fragmentation of collection method and data storage is a very real problem in the worldwide community of butterfly researchers. By designing an Android and iOS mobile app to use during data collection, we aim to help reduce this problem by allowing easy collection of a superset of the data required for each of the many disparate survey protocols. We plan to set up a server for centralized collection of all of this data. By communicating with professionals in the field during the design and development of this system, we hope to achieve wide adoption of our data collection app in the butterfly community and provide a product that matches or exceeds the quality and efficiency of existing survey methods for each protocol.

For a more thorough introduction to our project's background, see the Problem Statement section in our project plan.

Target Devices

The deliverables for this project are an application for both the Android and iOS platforms. The applications will support tablet and phone devices that are equipped with GPS. Some features of the application will not be made available on devices that don't have the required hardware, such as a camera. The applications will still function; however, the data that is reliant on the presence of the hardware will not be collected. An ideal device would have the following components: GPS, light sensor, accelerometer, and a camera. We are currently looking to support Gingerbread (Android 2.3) and higher and iOS 5.0 and higher.

The decision for targeting Android and iOS came about through a series of avenues. We have been conversing with our client as well as professionals in the butterfly community. Through these communications, we were able to determine the platforms that are most commonly used by our potential users. Most of our users are running either Android or iOS devices; thus, we decided to target these platforms.

For Android, we will focus our testing and compatibility based on the current market share of devices, as published at http://developer.android.com/about/dashboards/index.html.

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

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

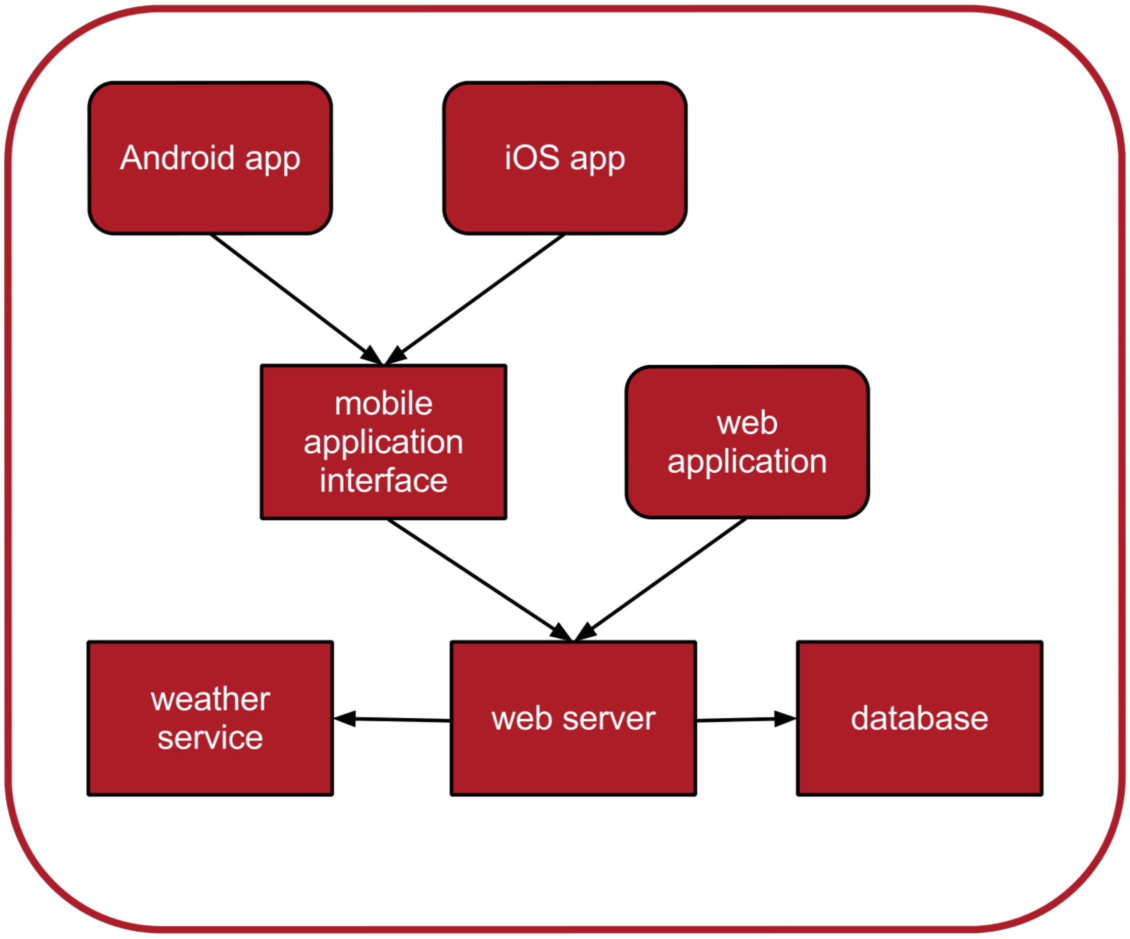
- Availability
  - The database should handle up to 10,000 requests per day
  - The database should store up to 30,000,000 records
- Security
  - The server should not allow submissions from a mobile app to override data in the database
  - The server should not allow mobile apps to query the database
- Maintainability
  - The database should allow inclusion of new data fields

System Design

System Requirements

The system will include an Android app and an iOS app that conform to identical functional requirements, as described below. The web server will expose a platform-agnostic interface for interacting with the database and performing user authentication. Mobile applications will only be able to submit data, and the web application will be able to submit new data, retrieve data, and edit records. The web server will manage the retrieval of all weather service data and insert it into the appropriate records in the database. See the functional decomposition below for details of each system-level component.

Architecture



System architecture

Functional Decomposition

Mobile applications

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 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

Drupal will integrate with a MySQL database.

Web application

The web application will be developed in PHP using Drupal as a backend and database abstraction layer. The application will provide users with the ability to perform queries on the survey data and export those results in a common format, such as csv.

Weather Service

This will be a 3rd party service, queryable through the Drupal server's weather service interface.

System Analysis

Mobile Platforms

Android and iOS are the two dominant mobile operating systems right now, so using them automatically provides a large portion of the market a way to use our system. We investigated the possibility of a Windows Phone implementation, but the current low market share of the OS would not justify the amount of work to develop it. We selected Android as our platform of choice for prototyping because of greater familiarity with the OS and availability of devices for testing.

For the first few weeks of the semester we explored various cross-platform mobile development frameworks like PhoneGap, Appcelerator, and Titanium that allow creation of a single codebase for deploying to multiple platforms. These have the disadvantage of having a common UI, which means that it can match the UI of at most one target platform. There can also be difficulties when working with device sensors, which is a requirement for our implementation. These concerns disqualified such platforms from our design. We then moved on to exploring cross platform design patterns for the possibility of having a shared codebase written in C++ for the core application logic for both platforms. Because of the UI-heavy nature of our design, we decided that this would require *more* work in the end, regardless of the re-implementation time of this core logic, given that the ratio of this code base to the full application would be fairly small.

Web Components

The technologies used to implement the web components remain as proposed by BAMONA. We evaluated their feasibility with respect to our mobile interfacing and found no issues. Drupal is a quality, open source content management system used by tens-of-thousands of projects. It has well-abstracted APIs conducive to modular design. Because it is open source, we would be able to replicate the system at Iowa State if necessary. The database will be MySQL, primarily because of its high level of support in Drupal. The mobile apps will perform authentication and database calls using XML-RPC (remote procedure call) over https. This is a common method of interacting with a Drupal system from a remote mobile device that has been implemented in several existing apps. This will greatly cut down the time needed to implement these app components compared to the time for manual authoring.

We have opted to collect weather data automatically on our server, rather than attempting to collect it on the device. Devices may not have network access, and we want to minimize network traffic to the device anyway, so a server-side implementation allows greater flexibility.

Detailed Design

Component Interfaces

Arrows indicate the direction of information flow.

Phone → Database

The phone will communicate with the database using XML-RPC commands of the Drupal API. XML-RPC will also be used for remote account authentication against Drupal's authorization system.

Web server → Database

The web application will communicate with the database using Drupal's database abstraction layer in PHP.

Weather service → Web server

We will write a weather service abstraction layer through which to periodically get weather information from a 3rd party service. The web server will place this information in the database using Drupal's database abstraction layer.

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

App User Interface

Target Audience

There are several kinds of people that currently perform butterfly surveys: Researchers, graduate students, conservation workers, citizen scientists, hobbyists, and volunteers with minimal experience. We will design this app to be easily understood upon first use, and fluidly functional for each of these users. We will release the app for free on the Google Play Store for anyone to use

Implementation

This discussion of user interface applies to both the Android and iOS implementations. We will conform to platform-specific conventions for navigation, layout, and styling as specified at developer.android.com/design and developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html.

Page descriptions

Launch page

Users will be able to create or log in to their BAMONA account using the app. Once logged in, the user will not have to visit the login page again unless he or she explicitly logs out. Alternatively, users may opt to not create an account. The caveat with this option is that data cannot be submitted to the database without logging in. This app will, however, support export of data to a CSV file on the device, which can be transferred to a computer using an email app, Dropbox, or USB connection. Any data created while not logged in may be submitted to the database at a later time provided the user is logged into the account at the time of upload.

Survey Explorer

The app will have a "survey explorer" that contains a catalog of all surveys performed on the device. Users may view and update any record stored on the device by selecting the record to return to the surveying page (discussed below). Upon upload or export, surveys will be cleared from the app's local storage. This design decision enforces the idea that this app is intended as a data collection interface, and not a data manipulation interface. While filtering of identical submissions from the database is an easy process, managing nearly-identical submission due to editing a record multiple times is a more difficult matter that is outside the scope of this app. The web interface will provide some facility for this operation. Finally, the survey explorer's topmost option will be to create a new survey.

New Survey Page

The new survey page will contain a list of all available protocols. The initial set of implemented protocols will be Pollard walk, mark-recapture, distance sampling, presence-absence, meandering survey, and individual sighting. Selecting a survey type will take the user to the surveying page.

Surveying Page

The surveying page will look a bit different for each protocol, to accommodate the different styles of collection. For each sighting and survey, no matter the protocol, the app will collect the same set of automatic data, as delineated in the requirements-to-deliverables mapping in the project plan. This feature of the system is to promote standardization of data collection without burdening users with the requirement of extra manual recording. In addition to the benefit of automatically collecting a larger amount of data than any one protocol, this automation will speed up the process of surveying for all users. Users will be able to include data outside the scope of the current protocol via an unobtrusive panel.

A survey consists of many sighting and some common data. The user can enter the common data at any point during a sighting. Opening the survey page automatically records the start time, and finishing a survey records a stop time. These values may be manually edited. Common data also includes route waypoints, which are automatically plotted during the survey, at an interval configurable in the app settings, defaulting to every five minutes. The surveying page contains a list of all sightings for this survey, with a button to record a new sighting.

The sighting page will be a single vertical-scrolling interface for inputting all data. The bottom of this page will contain toggles for overriding the automatic collection of any data point with manual entry. This feature is necessary to manage anomalous cases, including correction of faulty sensors and submission of sightings at a time or location other than the physical sighting. The text field for automatic data will display the collected value, but will be uneditable unless toggled for manual collection. This page will include a photo feature, allowing the user to take a voucher photo for each sighting.

Selection of species will allow common name or scientific name specification, using a pre-populated dropdown of likely species based on app history and current location. Users may opt to use taxonomic tree-based specification and select the most precise level from species, genus, and family.

Finally, users will have a field for inputting general comments or task-specific data on both the sighting and surveying pages. This field will be queryable for substring via the web interface to allow users to retrieve data using any custom encoding entered into this field for research purposes.

Settings

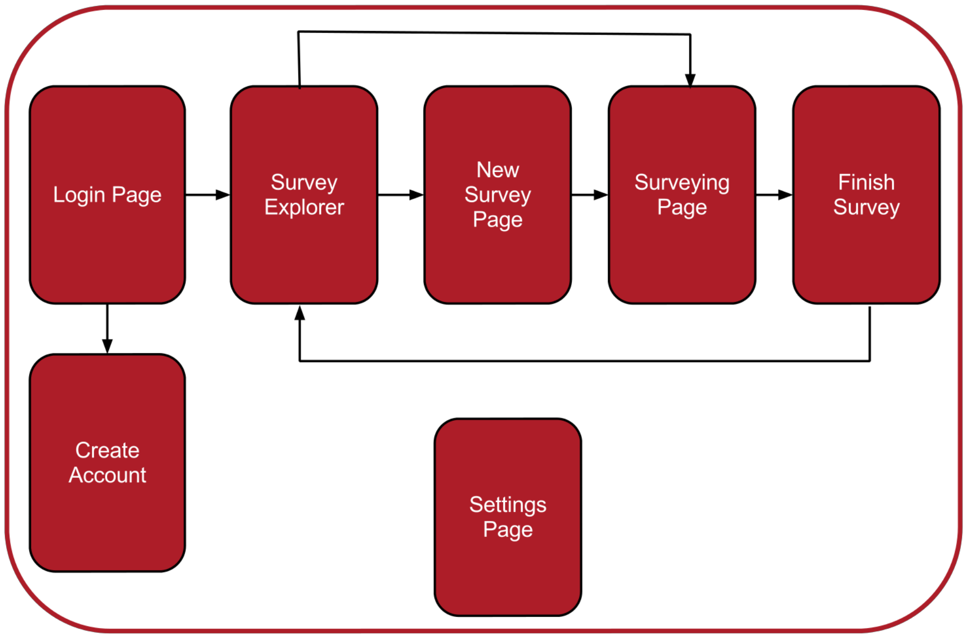
The app will allow users to specify a number of settings:

- Default survey protocol
- User account information
- Automatic upload preferences
- Waypoint plotting interval
- Local export directory
- Wifi-only upload, versus any-network upload

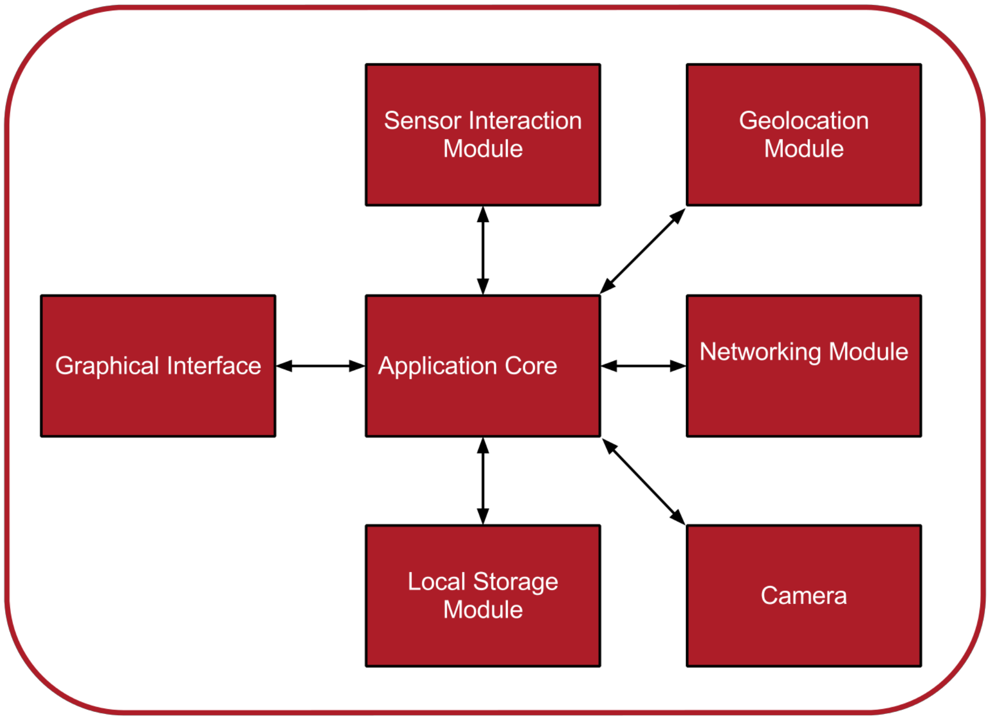
User feedback

For the prototype release, the app will also include a page for submitting user feedback, accessible using the app's context menu. This page will contain fields for providing bug reports and general comments. Each submission will have the option of including usage statistics.

Screen flow diagram



Screenflow diagram



App architecture diagram

Web Server

The web server will be hosted by BAMONA. It will run Drupal 7, which will provide abstract interfaces for the database, web application, weather service, and mobile apps.

The web application will support import and export of data in CSV format. It will support searching and filtering of records by any field.

The weather service abstraction will allow collection of temperature, wind speed, wind direction, percent cloud cover from the third party service. Upon upload of data to the server, this module will use GPS location and timestamp, if available, to access the historical weather data at that place and time and insert it into the database.

Backup Server

The backup server is being hosted by the IT department at ISU. The server is configured to be running throughout the extent of the senior design project. The server's operating system is Redhat and is hosting a couple of services.

The server is running Apache to host our teams web page. PHP is also installed on the system in order to handle the processing of uploaded record data. We are currently able to upload records to the server with the application.

The server is also running a MySQL database that will be used for storing the survey records in the event that we need to switch our fallback into our production server.