# Arboretum Tree Walk Mobile App 2014 Home

Tools

Added by Xiaodan Zhang, last edited by Sairam Sasank Chundi on May 08, 2015

## Project Links

GitHub repository (contains all project code): https://github.com/kfishster/ArboretumSeniorProject

Trello (keeps track of work being done): https://trello.com/b/7rUKLdFg/arboretum-senior-project

Arboretum Survey (asking visitors for input): https://docs.google.com/forms/d/1hMjDKvfcobPbA_FOpyReDJ59WGNZKp-SnU6v7UFVxI0/viewform

## Client Profile

Our client is the Arboretum staff at the University of Illinois Arboretum and Japan House. They are responsible for maintaining a cornucopia of plants and trees along with a database of all the species within the Arboretum grounds. They want visitors to have more information about the various plants and enhance enhance visitors' Arboretum touring experience.

### Liaison Profile

Our Liaison, Dr. Diane Anderson, is a biologist, horticulturist in charge of the plants at the Arboretum. She does not have an extensive technical background, and along with her colleagues, are leaning on us to provide them with the best implementation of their ideas.

### Other Stakeholders

- Gary Kling - Associate Professor of Horticulture
- Chad Kupferschmid - GIS specialist
- Kevin McSweeney - Professor of Soil Science
- Chris Lindsey - Original author of the Woody Plants database,  REST API

## Problem Statement

The University of Illinois at Urbana Champaign Arboretum, although a beautiful place for spending your time, cannot provide essential information about the species of plants in the various gardens without cluttering up the gorgeous landscapes. This information can be helpful to the frequent visitors of the gardens and also to the students studying the diversity of the plots. Additionally, casual visitors would love to know about various trivia tidbits about the plants and structures in the gardens such as the plants' practical use and the origins of the statues and gazebos. Also, there is no backbone in place to create cohesive tours around the gardens.

## Requirements

### Functional

The app needs to have a GPS tracking aspect such that the visitor is able to identify where in the gardens they are. The app also needs to provide the visitor with the pertinent information about the plants and structures in the vicinity and all relevant information pertaining to them. Another feature the app must have is to provide a means to conduct tours around the gardens pointing out various landmarks and features.

We will also need to develop a web application for their staff to add information to the app. This information includes, but is not limited to, seasonal alerts, trivia about plants,

### Non-functional

A survey will be created to gather input from potential users of the app. The survey will ask questions such as what platform is the user most likely to use, and what specific features they would like to see in an app for the Arboretum. It will be distributed via email and also physically. The physical survey will direct the user to a website where they can answer the questions, and the email will simply link the potential user to the same survey.

A mock up of the User Interface for the app: First UI Design

Using a quadcopter, a remotely controlled drone, aerial photos and video of the gardens will be collected.

## Proposed Solution

The Arboretum app will be completed using the following tools:

- Xcode for iOS programming
- Apple Maps framework for in-app maps
- NCSA WoodyPlants database
- GitHub for code repository
- Trello for work management
- Google Docs for the first part of editing the wiki page, so multiple people could edit at the same time without causing conflicts.

This app is similar to other applications created for tours of gardens, however, we are catering this implementation to be fully featured but also modular such that it could be used for other purposes.

## Scope

Since essentially all of the items that we will be doing are already in the Functional/Non-functional sections, this will simply include items we will NOT be doing, or items that we are not sure yet.

We will not be providing them a way to modify their plant database. We also will not be redesigning their map of the arboretum. Since we are planning on making this app as abstract as possible, we will not provide support for updating information once we are finished with the project.

As for features that we are not sure of yet, since we aren't entirely sure what access we will have to their database, or how the database is even organized, it's unclear as to whether or not we will be able to query it or if we will have to scrape information off their webpage. A possible solution to this problem is described in the **Semester Narrative** section.

## Feature Breakdown

Feature breakdown is edited frequently, currently resides at the trello page mentioned at the top. A short summary can be found below

1. Application
    a. Plant dictionary/reference
        i. Will include non-plant attractions such as the Gazebo and other structures
    b. GPS Capabilities
    c. Tour options
    d. Seasonal notifications
    e. Event notifications
    f. Links to wikipedia/woodyplants database
        i. Pull information such as a picture of the plant
    g. Timer for runners
2. Data Input Mechanism

    a. Allow client to input own data, app changes accordingly
    b. Ability to send notifications
    c. Ability to edit trivia/links
    d. Serialization to JSON
    e. JSON config defines app behavior

3. Additional visuals
    a. Quadcopter photography for zoom in animations
    b. Seasonal photography

Proposed Deliverable Timeline

We will update it as we make progress. The ultimate deadline is that the app should be completed by the end of the Spring 2015 semester, however, we plan to have it done before then.

Deadline #1 - November 12th (Meet with Client)

- Show first full mockup of the app to client
- Have a completed and up to date Wiki page

Deadline #2 - April 15th (Demo app)

- Application completed, show client for various modifications
- Plan out data input mechanism
- Data input mechanism working
- App bases all its data off of the custom input data
- Application pings server to get config updates

Deadline #3 - May 6th (Final Tweaks)

- Make any changes necessary or as requested by our clients

## Fall Semester Narrative

Our first official group meeting took place on October 15, during which we were acquainted with our (at the time) full group and clients. During this meeting we discussed ideas for the app and got a very rough idea of the features the clients were going to expect from it.

After getting an idea for what the app would require, we decided that it would be best for us to tour the Arboretum by ourselves to get an idea of what a visitor might experience on their first trip there. We went to visit the Arboretum on October 20. This was extremely helpful, as it allowed us to get an idea of the scale of the gardens, and gave us an idea of how accurate phone GPS was. A few days later, on October 22, we visited the Arboretum again, but this time with our clients. We used this time to ask questions about the gardens and features they wanted to highlight in the app. During this visit, our clients informed us that they were planning on expanding one of the gardens, reinforcing the need to have the app be very abstract in how the data is fetched.

Our next group meeting was on October 29. Here we got our new member, Sasank, up to speed. We also used this time to develop our first mock up of the app's UI.

Once we had an idea of how we wanted the app to look, our next step was to actually learn how to go about implementing this. Fortunately, we were able to schedule a meeting on November 5 with an iOS professional to give us a brief tutorial on how to develop in xCode.

Now that we had an outline for how the app should look and the knowledge on how to create it, it was time to create the first version of the app. We set aside a block of time on November 9 to prepare a demo app for our clients. Since it was impractical for everyone in the group to work on the app at the same time, we split into several different groups. While some of us worked on the app itself, others used the time to update our wiki page. We also formalized our roles in the group, as well as sketched out a rough timeline for our project. All in all, the day was very productive, as everyone in the group worked well together. At this point, we feel that we are setting a good pace for ourselves to meet our deadline goals.

We were able to show our first prototype to our clients on November 12. The demonstration went well and we were able to get some good feedback about the app. At this point, however, we are pretty much stuck until we are able to meet with their contact at NCSA to discuss what type of database access we can get. Our next steps will be to set up a meeting with him to decide on the best way to add the relevant information to the app.

The week before Thanksgiving Break, we were able to schedule an impromptu meeting on November 20 to discuss database access. We were also able to meet Chris, who maintains the plant database. While there was nothing definitive decided at the meeting, it was nonetheless helpful simply because it allowed for some good conversation about issues that would be faced getting the information from the NCSA servers, as well as matching data from Diane and Chad's databases with Gary's. Security is obviously a big concern for Chris and Gary, as they do not want someone to simply scrape the information and steal all of their work. This is something we will need to discuss as a group on the best way to avoid that from happening.

Following the meeting described above, Harjas was in contact with Chris Lindsey to iron out specifics in terms of database access. Chris Lindsey consulted Gary Kling and came to a conclusion that it would be better to host the UI Plant Database on a server other than the one currently hosted via the NCSA. This would allow Chris to put a RESTful service in place which would allow us to access the data using simple GET/POST requests. The data can be queried by hitting the end point server over HTTPS and returns a JSON which can be parsed to output plant information data. Data at this point can be queried using plat type or scientific name. Harjas is still working with Chris Lindsey to iron out exact details but an alpha version of the REST service is already in place.

## The Whole Gang

- Client point of contact - Harjas Singh
- Project supervisor point of contact - Kirill Varshavskiy
- Artistic point of contact - Drew Metz
- Security point of contact - Sasank Chundi
- Wiki contact - James Beck

Breakdown of specific work each member has done to date:

1. Harjas Singh
    a. Initial app design, interface
    b. Made Survey
    c. Liaison to client
2. Kirill Varshavskiy
    a. Wiki duties
    b. Made Survey
    c. Initial app design, button design
3. Drew Metz
    a. Initial app design, visual mockup, button design
    b. Made Survey
    c. Wiki duties
4. Sasank Chundi
    a. Initial app design, interface
    b. Made Survey
5. James Beck
    a. Wiki Duties
        i. Maintain Semester Narrative
    b. Made Survey
    c. Created Splash Screen

## Lessons Learned

- XCode is **prettyrad**

- Grainger doesn't let us reserve rooms with monitors without warning them ahead of time
- Google Docs are **much** better for group editing than this wiki
- The Arboretum gardens are **prettyrad**
- Woodley has a mythical quadcopter that resides in a mythical box from NASA (probably came from space)

## End Of Semester Status

- At this point, we have a functioning bare bones application. However, we can't move much beyond this point until the database holding all the plant information is set up and we are given access to it.
- Chris Lindsey (Woody Plants Database developer) has moved the hosting server from the NCSA to an independent server allowing for a backend RESTful service to be developed. He has developed the REST API that we can use to get the plant information.
- Diane Anderson is working on updating the plant coordinates database and merging it with the woody plants database. This will be ready by early next semester.
- We will be presenting what we have so far at the Plant Sciences Lab Board Meeting.
- Our clients are happy with the progress we've made this semester.

- Here is the End of the semester presentation we gave in class (confluence doesn't let us embed it):
  - https://docs.google.com/presentation/d/1S55gdQg4qSBr8-Os0KfbIXVHEKz7K7F2Yr1do-9gXB8/pub?start=false&loop=false&delayms=3000

## Plans for Completion

- In order to complete, we just need to have the application fetch relevant information from the plant database
- We also need to design a simple web app to allow our clients to update information in the app
- Continue meeting with the client on a weekly/bi-weekly basis to present progress we've made. We will be relying on constant feedback (from the client) to complete the application to our clients' satisfaction.

# Spring Semester Narrative

# Group Writeup

## Client Organization Profile

Our client is the Arboretum staff at the University of Illinois Arboretum and Japan House. They are responsible for maintaining a cornucopia of plants and trees along with a database of all the species of plants found within the Arboretum grounds. What they want out of our project is a way to provide visitors of the arboretum with a method of learning about the various plants hosted at the arboretum as well as a way to explore and make discoveries with the help of a devoted application. Part of the purpose of the arboretum is to grant visitors access to a diverse set of plants as well as an education on species they may have never seen before. As such, the collaboration we are participating in with the arboretum seeks to enhance the effectiveness of their organization and reach out to people who have an interest in utilizing technology to educate themselves.

#### Liaison Profile

Our Liaison, Dr. Diane Anderson, is a biologist and horticulturist working with the various plants at the Arboretum. She does not have an extensive technical background, and along with her colleagues, is reaching out to us to cooperate on a functional implementation of their ideas.

#### Other Stakeholders

- Gary Kling - Associate Professor of Horticulture
- Chad Kupferschmid - GIS specialist

- Kevin McSweeney - Professor of Soil Science

## Overview Original Problem

The University of Illinois at Urbana Champaign Arboretum, although a beautiful place for spending your time, cannot provide essential information about the species of plants in its various gardens without cluttering up the gorgeous landscapes. This information can be helpful to the frequent visitors of the gardens and also to the students studying the diversity of the plots. Additionally, casual visitors would love to know about various trivia tidbits about the plants and structures in the gardens such as the plants' practical use and the origins of the statues and gazebos. In addition, there is no backbone in place to create cohesive tours around the gardens.

## Communication with your Client

So far in the spring semester our group has just been working on the project as time allows us. Instead of bi weekly meeting with our clients, we instead only schedule meetings as needed, as having regular meetings isn't needed at this point of the project.

Most of the communication with the client were done through our client point of contact, Harjas, and the internal group communication was done through the fantastic technology otherwise known as Facebook Messenger. This way, Harjas relayed to us the information about planned meetings and we would discuss the topic at hand in the Facebook chat.

Nearly every meeting we had with the client were attended by the majority of the technical team, Diane Anderson, the project lead, Gary Kling, professor helping with data integration, and Kevin McSweeney, the director of the Arboretum. This way, we were able to either project the progress of our app onto the big screen in a classroom at the Plant Sciences Laboratory, or show the clients the application running on a native iOS device as we added on to it. Some people say one should not show an unfinished product, but we're lucky that the Arboretum staff were very patient with the buggy/partially working prototypes we presented to them.

Although we provided the links to all the Wiki and trello and github sites to the Arboretum team, the usage of these sites and their relevance to the actual work being done has diminished significantly as we proceeded with the project. The trello, although updated fairly regularly, was not as informative of a resource as it should have been, and the wiki space was more for us to reflect on our work rather than for the clients to read about what we have done. Most of the communication was through in person meetings or through our point of contact.

## Non-functional Requirements

Beyond developing the application and meeting regularly with our client, there were few non-functional requirements place on our team. However, through fortunate timing, an Arboretum Board Meeting was scheduled to occur during the Spring semester. At this point in development our application was functional enough to demo to interested parties. Our clients expressed interest in our team sending a representative, Kirill, to present what we had been working on to members of the board.

## Functional Requirements

After attending a tour of the grounds, we realized that the app needed to provide several functionalities. Our app’s first requirement is to provide the visitors information about the Arboretum: hours of operation, directions, parking information, ways to donate, and general announcements. Its second requirement is to provide visitors to learn about their surroundings. That is, users should be able to locate themselves and any landmarks in their vicinity. More specifically, they must be able to see a map of the Arboretum and easily identify their positions and any nearby points of interest (such as the Hosta Gardens, Japan House, etc.). The third requirement is to intuitively provide information about each plant they discover. Our team must decide whether to provide this functionality by placing QR-codes for all the plants or by retrieving plant information based on a user’s GPS location. The fourth requirement is to provide the visitors an enriching resource to learn about all things arboreal. Other requirements include providing recommended tours, seasonal alerts, trivia about plants, surveys, and counting the number of visitors. We must build the application to provide these features for the users.

After a discussion with Dr. Diane Anderson, a biologist and horticulturist at the Arboretum, our team realized that a majority

of the Arboreturm staff has limited technical background. What we need to build is an application that is as abstract and flexible as possible. Plants' locations and the app's miscellaneous configuration details should be adjustable without the staff's interference with any code. For example, gardeners and researchers plant different flowers each season. Users must perceive these changes without installing app updates. In order to meet this requirement, we are building an intuitive user interface (web app) to allow our client to externally modify the app. It is important that only existing technical infrastructure is used by our team to complete the project so that no additional systems have to be maintained for the sake of the app. Essentially, our client needs to be able to manage the utility of the app with minimal technical involvement.

# The App

## iOS Application

The design layout of the app itself is a standard iOS tabbed application. The first tab is a map of the Arboretum with markers indicating spots of interest. This is the screen that will be presented to the user when the app first starts. This screen will also have GPS tracking enabled to show the user's current location within the Arboretum. This first screen serves as an anytime available map of the Arboretum and what it has to offer, even when the user is not at the Arboretum. The second view of the app is a dictionary list of all the plants present at the Arboretum. Clicking on any of these plant names will take the user to a detailed view of the plant, which would include images, plant details. The plant dictionary can be sorted alphabetically, based on plant category and plant type. The detailed view of the plants contains plant details rendered from the WoodyPlants REST API. Styling of the text was automated using HTML tags sent along with the plant data. The third view is Events - detailing any events taking place at the Arboretum. The fourth view is the Trivia view. This contains any plant-related trivia that the Arboretum staff would want users to know. This view can be updated via the configuration file web app (details below). The fifth view is the About page that entails the Arboretum's mission, it's patrons and donors and a link to donate to the Arboretum. This view can be updated using the configuration file web app too.

## Configuration File Web App

To address application maintainability after we graduate, we decided to create a web app that would allow editing of the configuration file. Since the app data is primarily generated from the configuration file when the app starts, giving our clients the ability to manipulate the configuration file to their liking adds an additional layer of abstraction to the app. The clients can change the map latitudes and longitudes which would update the Map view on the app. They can also add content for Trivia and About pages in a rich text editor. The web app also includes a link to the plant database and the ability to add notifications to be sent out the users. Changing any of these fields on the web app, reflects the changes on the app itself.

## Overall Lifecycle

The overall lifecycle of the app looks something like this.

1. Client edits configuration file on the web app, the details of which are stored in the database.
2. When the iOS app is opened, the app makes a GET request to retrieve the configuration file (returned as JSON)
3. The app parses the configuration file and stores all config data in CoreData for availability throughout the application.
4. App views generated using config values from CoreData - including the map view, directory (using API URL), the trivia page and the about page.
5. As long as app remains open, a call is not made to the config file again to regenerate data reducing constant dependence on the server.
6. User enjoys Arboretum experience on his/her fingertips.

# Target Users and User Stories

One interesting part of working with the arboretum was considering who visits it and how that would impact the development of our project. The target audience for this app is not limited, it could be used by anyone. It's great for younger kids to learn more information about the arboretum on a grade school field trip, to older generations simply interested in trees while they are walking through the gardens. The app's interface is extremely intuitive to use, so it's

useful to even the most technologically unsavvy. Another great feature about the app is that it's not limited to use in the arboretum. Users will be able to use the app to identify plants anywhere by using the database. One of our group members has even used the app himself. While walking around Siebel Center, after a debate with some friends about what species a tree was, he was able to correctly identify the tree by searching the database for it.

## Team Decisions

Part of creating an app that is easy to use by the public is to develop it on the platform that the majority of people use. This is part of the reason that we decided to develop the app in iOS. From the feedback we got from a survey that was created at the beginning of the fall semester, the majority of people in this area use iPhones, so iOS was the logical choice. Another reason we chose to develop on this platform is because none of our group had any experience developing for iOS, and since part of this class is about learning new things it made sense for us to use it. This meant we really had two choices for what language to use: Objective C or Swift. We ended up using Swift for two reasons main reasons. The first of which is that it's much newer, and will be much more forward compatible compared to Objective C. The other reason is because early in the fall semester we met with a professional iOS developer who suggested that Swift would be better for our needs. However, we ended up using a mixture of the two, as some of the libraries we used were written in Objective C.

## Team Research

After outlining the concepts and features of the application with our client, we made the decision that pursuing IOS as our platform was the best course of action. Partly we made this decision out of the assumption that IOS would be the most popular platform with the demographic of arboretum visitors, and also out of a personal desire of our team to learn a development language we were not familiar with. After receiving MacBooks from the University, we were provided with the Xcode IDE, but little experience with the language Swift. As part of our team research, we contacted an acquaintance of our member Kirill's that has a background in IOS development. We scheduled a meeting with this contact, and he provided our team with a tutorial on the basics of setting up a project in Xcode as well as the essentials of Xcode and its storyboard component. Through utilizing this opportunity we were able to any avoid early stallings on setting up the development environment and as a result we had more time to work on the actual implementation.

## Designing Reliability and Performance

From early on in development our team was faced with making design decisions with reliability
and maintaining performance as a priority. Considering our app handles information from quite a large database, performance is very important for us. One way we help mitigate issues with such a large database is we only pull new information when the MD5hash provided to us by the database changes. This allows us to only have to query the database and pull new plant information when it's changed, instead of every time the app loads. Another big concern for us is the app's reliability. Since we won't be able to maintain the app too much after the semester is over, our clients needed a way to change the app's configuration file without accessing the code. Our solution to this is to store the configuration file on a web server, which the app then loads information from. This makes it much easier for them to change stuff, as well as fix problems.

## Unfinished Work

At the time of writing this fabulous account of work done in the spring semester (May 7th), there are still some things left unfinished in the application. The app and the internal website used for updating the app's information is about 90% completed. Some things that still remain to be done are:

- Tie all the functionality together in a cohesive manner - make sure the changes made to the config file propagate completely to the application
- Add push notifications to the app
- Add a password change page to the website
- Add "Config file last updated by <username>" feature on the website
- Add an admin account that can add or delete users, allowing them to edit the config file
- Get on demand GPS plant information from the server instead of hitting the same URL at all times
- Load picture in the plant directory

- Include non-plant specimens in the catalog
- Sort directory by type of object

We plan to complete all of these bullet points within a month or so and hopefully submit this app for store certifications shortly thereafter.

There was also some talk of how exactly we will be publishing this app, through the university or by ourselves. This goes into a lot of legalese mumbo jumbo, but the bottom point being, we have to submit the app from the university if a professor/university staff had direct contribution to the app (written code for example), or we can submit it from our own developer accounts if the app was done completely by us. We are in this legal limbo where we can argue both sides of the story, so we might either hand it over to the university and jump through some hoops to get it approved first by the university and then by Apple, or just publish it and maintain it on our own.

The configuration file web app was originally hosted on the web.engr.illinois.edu servers under Harjas' web space. Since that web space will no longer be accessible after graduation, we ported the web app on an Azure server running MySQL and PHP. As we continue to move forward, we'll need to find a more permanent home for the website.

# Individual Contribution: Kirill Varshavskiy

## Lessons learned as an individual

I mainly focused on the iOS functionality, tying all the pages together as well as hitting the server when necessary. Most of my time was spent learning Swift and its intricacies. Surprisingly, Swift is a pretty nice language. It feels just like JavaScript and is quite intuitive. XCode definitely helps with its autocomplete functions just like a good IDE should do. That being said, there are several downside to coding in Swift at this point in time.

There are two big deterrents from swift: it’s a very young language and most of the helpful code examples are still in Objective C.

### It’s a young language

Swift was introduced at WWDC only last year (2014) and is now on version 1.2. When it was announced, all the developers at the conference gave a standing ovation, and rightfully so, iOS was the only platform still using a fairly low level language. Swift provides a very accessible and intuitive programming backbone to developers. My high school has even introduced courses in iOS programming due to Swift being super straightforward.

Being a fairly young language, it is still changing very rapidly. There were several times during development when we would update XCode and all of a sudden see compiler errors throughout the entire app. This was due to Swift now enforcing new rules that we have not known before. One such rule was called “Enforced downcasting” which would throw a compiler error if you were trying to cast a superclass type to a certain subclass. The syntax for casting is “X as Y” where X is the object and Y is the class you want to cast to. The new rule forced all downcasting to be “X as! Y” the exclamation point serving as a forced downcast. This means that instead of throwing a InvalidCastException, the code would throw something like an InvalidDowncastException. Big change. About 75% of all casts had to be edited.

### StackOverflow

When programming, one would often search StackOverflow for answers on how to do certain tasks. Unfortunately, iOS has existed for 7 iterations before Swift was introduced, so most of the answers on SO were in Objective C. Adding a “Swift” keyword after the query would not always yield a working answer. Additionally, when dealing with a new feature called AutoLayout, lots of layout questions were not very helpful. Bottom line – because the language and the development environment is rapidly changing, a lot of the available resources out there are helpful just as a high level (semantic) guide rather than a syntactic aid.

I vented my frustrations to my teammates and also my high school computer science teacher who was teaching an intro to Swift class at the time. She was also not very happy with Swift changing some of its enforced rules.

With all of that being said, Swift will be used much more extensively in the future, and thus more resources will be available. For a certain portion of the application, in particular, the code that shows a certain amount of pins on the map, we had to use an objective C library and tie it into the application via a compiler bridge.

### Organizational conclusions

All in all, this has been a fairly efficient use of our time, we were lucky enough to be assigned to a group that is passionate about their work and also because we were allowed to be the technical leads in this collaborative effort. This also came with a certain obligation to satisfy the aforementioned criteria, which we did not end up doing. Thus, we will still continue our efforts to work on this app through the summer so that we can create a finished application and publish it to the app store. At that point, if another Arboretum group takes over this project in the fall, we can hand them an already finished project and let them either create an Android clone or add new features to the app.

The organizational mishap was mainly due to the majority of our workload for the project fell on the second semester where we were fairly busy finishing off our last classes. This made collaboration much more difficult and led to more separated work groups which slightly broke down the communication we had setup previous to this semester. Additionally, with these random scheduling conflicts, we weren't very organized in delegation and making sure people know what they are doing. This led to some confusion and a reduced work ethic, thus an unfinished product. Taken that into consideration, we still ended up presenting an application that looked good and worked well.

### Client interactions

As I said, we were the technical leads in this project and we became a fairly tight knit group with the Arboretum folk. We had regular meetings with the project lead and also the director of the Arboretum, as well as some staff who helped with the data and content of the app. I had a wonderful time presenting to the Arboretum board the plans for the app and its progress, which was welcomed very warmly, with the board members asking various questions and getting in contact with me about the development. It was definitely a great opportunity to see the inner workings of the organization that we are contributing to (not many other groups can brag about that!)

## Solution/Source Code

**Information about the code and how the app is organized**

# Individual Contribution: James Beck

### Research

For this project, the two features I did the most research on was how to store data in the app's persistent memory and creating a feature to allow users to zoom and pan pictures of the plants. Almost all of my research came either from reading various questions other people had asked on stackoverflow or from apple's development library.

I discovered that there are several ways to store data persistently in an app. For our needs, we decided using SQLite was the best option. This allowed us to store configuration information the app needed locally, instead of having to fetch it from the server every time the app started. Instead, we can just query the MD5sum and see if the data is different, and only then download the new information.

The other feature I researched and implemented is the view that allows users to pan and zoom images. This was surprisingly complicated for something that I had originally thought would be a built in feature. The task involved taking two of xCode's views and combining them. Through lots of help from stackoverflow, as well as a fair amount of trial and error, I did manage to create the results I wanted. Interestingly, one of the things I had the most trouble with was simply getting the picture to center. Because of the way we designed the app, the normal way of centering the image based off screen size did not work. Since we built the app using xCode's auto layout feature, the dimensions at compile time were fixed, and then when they would render on a device's screen, those dimensions would often be incorrect. I instead had to utilize some

system features of xCode that probably aren't standard practice, but they worked, so I'm not complaining.

## Solution

While there are indeed many solutions to create the app we did, the one we chose gave us the best modularity and room for growth. One of the biggest issues an app like this will face is that by it's nature, plants change very frequently. The location and variety of plants in the Arboretum can easily change from year to year, and even from season to season. This is why the app needed to be able to handle these changes without needed a programmer to do it. Our solution was to make everything very abstracted. The app itself has very little information in it that is hardcoded, and even then, we've implemented ways to change the default hardcoded values later on. This allows the staff of the Arboretum to change nearly any aspect of the app they want, all without needing to touch the code. The way the app works is that it fetches it's configuration data from a web server. This data can be changed with a web app that was written for this purpose. This includes everything from where the plant database is at to what the app's "About" page should show. This means that if in a year or so, the arboretum adds a whole new garden, it's trivial to reflect that change in the app.

## Your Team Experience/Lessons Learned as an Individual

Throughout this project I didn't have a specific task; rather I focused more on helping out and developing things as needed. One of the key features I worked on was the app's view that allows a user to zoom and pan images. I also designed the splash screen for the application, as well as doing the initial research on how to store information in the app's core data. I was surprised how difficult it was to develop the zoomable view for the pictures. My initial thought was that since this is used in so many different applications, it would be fairly trivial to implement. However, it turned out to be much trickier,, as it involved combining several features of XCode in order to allow for zooming and panning. Interestingly enough, one of the more frustrating aspects of this was simply to get the picture to position itself correctly.

Overall, the thing I've taken away from working on this project hasn't been learning a new language or working with a new technology, but rather how much more productive people are when they are working on a project they actually care about. One of the best things about this project is that our clients didn't really have any specific requirements for the application. They just wanted an app to help display information about the Arboretum. This allowed our group to really take ownership of just about every aspect of the app; from the user interface design to completely abstracting the backend so no values are hardcoded. Because of this, instead of feeling that the app was simply another assignment to meet requirements handed to us, it felt more of a personal challenge to make it work and look good. I've worked with many groups in the past where several members were not interested in the project and their involvement reflected that. Everyone in this group, however, has a personal investment in the project and so we worked very well together to finish it.

To manage our group and the project, we at first assigned specific tasks. However, that more or less fell through the further along the semester we got as people got busy with other classes. However, our team definitely benefited from having Kirill step up as unofficial leader. I don't believe there was any actual decision made regarding this, but he definitely did a great job keeping us on track and making sure things got done. Overall our project went pretty smoothly. There were a couple times the ideas we had or libraries we were using turned out to not be useful, but they were generally easy to revert. Also another nice thing about having a fairly simple project is that our clients never changed their expectations of us; they stayed consistent throughout the two semesters.

From the technical aspect of the project, this is the first time I've worked on developing a mobile application, so this whole ordeal was quite a learning experience for me. I'm still a bit undecided whether I like XCode and swift, but I do admit that XCode's AutoLayout feature is very useful. Instead of having to create views that dynamically adjust to screen sizes for a variety of devices, we just had to create one and XCode did the rest for making it work across devices. Coding in Swift was also challenging because the syntax was unlike any other language I've used. However, like most new languages, all it would take is using it more consistently and it would feel second nature.

All said, these two semesters have been a great learning experience for me and has been very useful in my growth as a developer.

## Communication with your Client

Most of our communication with our client was done through email and by Harjas rather than through the wiki page. We decided early on that it would be easiest to just have one person doing the communicating, otherwise it was likely that information would be lost or confused. Generally what would happen is Harjas would speak with Diane about scheduling meetings, and then let the rest of our group know through facebook. This worked very well, as there were very few times any of our group was surprised with a meeting time or some other such bit of information. In terms of meetings, we generally only held them when there was something worthwhile to present, or something that needed discussing that would require more than email correspondents. We actually had many less meetings during the spring semester than fall, simply because by spring we had gotten most of the app done.

## Options (Individual)

- Coursework - Discuss how specific courses you have taken helped you in this project
- Best New Things Learned
- Best Part of Doing This Project
- Worst Parts...

There were several classes I'm either in or have taken in the past that have helped with this project, of which 411 and 242 probably stand out the most. Database management was definitely helpful when looking into how core data works and how to store information in the app's memory. It allowed me to develop the schema that we used for storing the configuration data. 242 was also very helpful in that it prepared me for picking up a new language quickly. One of the assignments in 242 was to implement the goal in a language I had never used before, and to do it in just a few weeks. Learning how to learn a new language was very useful for picking up Swift.

As a result of this project, I have a new understanding and respect for how much work goes into mobile applications. This was a fairly simple app, yet there are quite a few components to it that a normal user will have no idea is there. It's quite mind boggling to think of how much work goes into apps that are much more advanced and richly featured. That being said, I also find it very interesting that a lot of the work that goes into designing an app isn't actually done programmatically (at least in our app), but rather arranging things on a storyboard. I don't know how it's done with Android, but I have definitely found myself viewing apps on my iPhone differently, in terms of what went into it's development and how I would have done it.

I think the best part of this project was that it was really the first time I've worked on something outside of academia that will be used by other people besides myself. I've very excited to be able to tell friends that I was part of a team that has an app on the app store. While I had already procured a job for after I graduate before this project was over, being able to talk about it in future interviews will be nice I believe. Another part of this project that I've greatly enjoyed with working with my group. A lot of the times after a group is finished, team members won't really interact with each other too much. However, I'm hoping that's not the case with this group, and that we stay in touch. The last thing I would like to mention that was great about this project is that our clients were wonderful. They were very easy to work with and genuinely appreciative of our work and our results.

Overall I feel that there weren't very many bad parts relating to this project. The advanced composition part of the course was not very much fun, but regarding the project nothing was really that bad. It was challenging enough to make it fun and worthwhile, but not so challenging that it got to be annoying. Our clients weren't difficult to work with at all. All in all, it was a very good experience.

## Recommendations for Improvement or Extension

There are a few features that we weren't able to implement that would be good to improve the app with. One of those features is the ability for the app to create tours of the arboretum. It will soon be able to tell a user where they are in the gardens, but in order to show them a path to take to a tree, the app needs to have a data set of where the paths in the arboretum are. This would require someone to map them, or have the paths geotagged. Another aspect of the project our clients have expressed interest in is to have the app ported to Android. This would be a great project for the next group of students that take this course. In terms of other features though, the way the app is designed it should be fairly easy to add new functionality to it. This could also be part of a project for next years students.

# Individual Contribution: Drew Metz

## Lessons Learned as an Individual

This course has provided innumerable circumstances for me to learn lessons, from the sharing of knowledge between teammates and the collective research as part of development to learning from our clients about our university's arboretum. I think one of the biggest lessons I learned during our time working on the application was the organization necessary to go from a conceptualized idea to implementing the actual functionality. What ended up working well for achieving organization was the employment of shared spaces where we could all monitor what tasks we have to implement and how various tasks interact with and are dependent on each other. The tools we used to accomplish this included Trello and the collective Wiki page.

### Utilization of Research, Stackoverflow

The task I worked on that provided me with many lessons was the configuring of the visual design of the user interface. This involved starting from the core functions we desired in our application, and deciding how to visually separate these features into distinct parts of one cohesive application. Our earliest meetings involved long discussions in front of a dry erase board planning out the distinct features and their visual organization. In particular, I found the transition from the pen and paper mockup I drew with input from our group to the implementation of the design with xCode's storyboard feature to be challenging. During this process I encountered a few instances of devoting too much time to minor details in the design that ended up being difficult to the point of being not worth the time to implement, and this meant some sacrifices in design. One instance of this that still bothers me is the discrepancy between the visual cues of our navigations tabs at the bottom of the application and the visual cues of the text throughout the rest of application. In particular I wanted to be able to have a contrast between the selected navigation icon and the unselected navigation icon that would make it clear what was selected while still maintaining readability on the unselected icons. This proved more difficult than it should have been for the version of IOS that we developed on, and I had to compromise with dull unselected icons that are still readable. Stackoverflow ended up being a great resource to see what was a manageable design choice to implement and what might not be worth my time, and seeing other users describing difficulty with certain implementations helped me avoid problems. While our investments in organization kept all of us up to date on who was working on what, I think one area in which I could improve is having a complete understanding of the code written by all other members. Overall this was a very rewarding experience and this course provided a level of diversity of work that was unprecedented for me.

### My Team Experience

The experience of working with my team on this project was enjoyable, and I believe that we started the first semester with a well established group discipline which was maintained throughout our time together. Our method of development and engaging with each other felt successful, and could certainly be influenced by different Agile Methods. Some methods of Agile development such as Adaptive Software Development, ASD, maintain a focus on continuous cycles of learning while developing goals flexible enough to be changed in response to new discoveries. While we didn't set out to exclusively follow this method of development, much of our experience progressing as a team benefits from the advantages of ASD.

After having a few meetings with our clients we were able to develop a list of the core features needed to be included in our arboretum application, as well as a list of features that could be considered accessory. Namely, the core features of the application were the interactive map of the arboretum and the searchable directory of plants in the client's database. These core goals were developed by the clients previous to our meetings as the necessary aspects of the project. Once we were able to have a dialogue and brainstorm with the clients, the accessory ideas we developed were as follows: a system to create guided tours through the arboretum, a method for users to bookmark and save certain plants for later reference, a way to host trivia about specific plants for users to read, and finally a space for the clients to provide the users of our application with the most current news and events happening at the arboretum. Upon constructing a list outlining both the core and accessory features of the application, we were able to organize our efforts towards learning the technologies necessary while laying foundation for the primary aspects. The flexibility that was provided to us by differentiating between core and accessory features was in agreement with the principles of many methods of Agile development.

The organization of tasks during development was maintained through frequent meetings between the whole group that

served the double purpose of acting as a time when we could all sit down and work in close proximity while also ending on a discussion of what we needed to do as individuals before the next meeting. During the early days of the project we would usually designate one member to take informal notations of what we discussed, for the purposes of reflection and refreshing ourselves later on. I would often take my own personal notes to keep myself from forgetting details during the longer gaps between meetings.

For the formal management of tracking, we employed a myriad of different technologies that proved beneficial in terms of organization, pacing and productivity. Our method of tracking the actual programming of the project was the use of github as a code repository. Discussion was frequent between our members when it came to pushing changes to the repository and pulling updates from other members. In addition we utilized the web service Trello, which allows a group to host multiple tasks with details on when they need to be done by and who is currently assigned to them. Having a display of exactly what goals our team had and when they needed to be done proved useful in designating tasks to individual group members. Early in our time together two leadership positions developed naturally, one being the leader of actual programming and development and the other being the primary contact point for our clients; these positions were occupied by Kirill and Harjas respectively. Having a position that monitored the broad development of our application was beneficial in avoiding hang ups and wasted time in development, and having a position to monitor communication provided our client with frequent updates on our progress. Utilizing Trello, we were able to designate different tasks to our members throughout the semester, the main distribution being as follows: Harjas focused on the web application that gave our clients input to our application, Sasank spent his time on the configuration file of the application, Kirill developed our connection between the application and the client's database, and the display of the retrieved information throughout our application, I worked on the user interface design, and visual design of the application and focused on the detail plant views, and James worked on the splash screen, researching core data as well as implementing a zoom feature within our plant detail view.

## Coursework

Over the last two semesters of working on our senior project, I found myself constantly utilizing skills, ideas and disciplines from the various courses I had previously taken in my college career. One of the enriching experiences of working with such a large team was that as separate members we each had some overlap of previous courses from the core Computer Science track, but we also had a wide variety of course experience to draw from.

### CS411

A portion of our work on the project was based around interpreting and interacting with large collections of data on plants and other features of the arboretum. Although this database had existed and operated for years before we ever saw it, understanding the fundamentals of database and relational design was a skillset that I found useful. In particular, my background with the course CS 411: Database Systems provided me with a working understanding of databases to approach the collections of data used by the arboretum. Although there were some gaps in understanding separating the client's knowledge of plants and our knowledge of programming, we had a common ground when it came to how the plant data is actually stored. Concepts such as joining different databases over common keys were mutually understood, and ultimately led to more productive cooperation between our group and the client.

### CS242

One of the most important courses I have taken during my time at the University has been CS 242: Programming Studio. A core lesson from the course was learning to write code that is readable by others, maintainable for future changes, and modular enough to be reused. Another important aspect was presenting the work that you have done to the moderator of the class as well as the other students. This aspect in particular was rewarding, as it provided preparation for explaining complex coding concepts in ways that can be easily understood by other programmers. In my computer science courses taken before CS 242 I rarely had the opportunity or requirement to develop the readability of my code or my ability to vocalise exactly what my code was doing to others. Programming sprints were common practice throughout the course of this project, and as such there was frequent explanation of code between members. I think that all of our members having CS 242 in our past course work experience certainly strengthened our abilities to communicate higher level coding concepts to each other.

### CS498

Once our group established what kind of functionality we wanted our application to have, we were tasked with

conceptualizing a consistent design for our project. I found this aspect of the project to particularly resonate with me, as I have always had an interest in visual design, branding and other disciplines associated with designing an application. I am concurrently enrolled in CS 498 RK1: The Art and Science of Web Programming, and while the web technologies used in this course weren't extensively relevant to our project, I found the visual design aspects of the class to be very helpful in the design of our application.

In determining the theme of our project, I looked to what we had learned about color theory in CS 498 RK1 for reference. What I ended up utilizing from these lessons was the idea of using mostly subdued, flat colors for spaces of the application that should be in the background of the user's attention. Further, a tenet of color theory is the usage of contrast to differentiate between primary and secondary information for the user. A specific evocation of this principle in our application was the usage of text color to highlight what text is interactive and what is meant for consumption: text colored white is in contrast with our mostly dark green backgrounds, indicating to the user that these portions of text were important and interactive.

### Best New Thing Learned

An easy option to pick as my favorite thing I've learned about during the course of this project would be the results of researching Swift and IOS development. As myself, and most of our team, approached this project with little prior experience working with IOS, there is still much that is unknown, and I am left with a motivation to learn more after graduation. In particular, discovering the Storyboard aspect of IOS development peaked my interest. I think what really is interesting about the Storyboard is that it is a visual, top-down approach to creating functionality that appeals to my interests in art, design and user interfacing in general.

On the other hand, some of the most interesting experiences I gained from working on this project had little to do with programming or development. Having the opportunity to work with the arboretum brought with it exposure to all the wonderful sights and information the arboretum has to offer. The early days of our project focused heavily on visiting the grounds of the arboretum and gaining a physical sense of the subject of our project. On some of these visits we were accompanied by the clients and were able to glimpse firsthand the vast understanding that they have of the various trees, shrubs and history of the grounds. As an engineering major, it isn't often that I am afforded the chance to learn about these types of subjects, much less to be surrounded by the very physical things we were talking about. Eventually, hearing enough interesting bits of information about different species of plants from our clients led us to pursue adding a trivia function in our application so that others could also benefit from their knowledge.

### Client Interactions

Personally I felt that the various meetings we had with our clients were some of the most rewarding experiences of working on this project. With regards to my own contributions, I enjoyed presenting our clients with our pen and paper mockup of the user interface and later showing them the implemented design. I believe our meetings were scheduled frequently enough that the feedback they provided could be taken into consideration but with enough time between meetings that we always had something new to show them. Overall the clients were very generous with their time and knowledge and were invested in all of our interactions.

### Recommendations for Improvement or Extensions

Earlier I discussed the separation of core and accessory features that developed when we began figuring out exactly what we wanted to present in our application. Some of these accessory features ended up being removed due to time constraints, while we managed to mostly implement all the core features that we set out to create. One of these scrapped features would have been a tour option, which would allow users to designate specific plants that they would like to see and in response be provided with an optimized path through the arboretum that features the requested plants. While one of the appealing aspects of our application is the freedom it allows users to explore the arboretum at their own pace and make discoveries, I think that the tour feature could be an extension to our project that would provide the users a more structured or educational experience in the arboretum. The only information necessary to implement this feature would be the various locations of different plants for each species, which the plant database we use has, and an implementation of a chosen graph theory algorithm. In addition, I think part of what makes this feature appealing for extension is that fact that as computer science students, every member of our group has a background in graph theory through the CS 173, 373, 473 track. I know that when I took these courses I often asked myself when I would actually utilize some parts of graph theory,

and this feature would be an interesting and useful way to do so.

# Individual Contribution: Harjas Singh

## Work Outline

As a member of the Arboretum Mobile app team, I was responsible for everything web. One of the major concerns we had while outlining the design of the app was maintainability. We would remain points of contact going forward (even after graduation) but the app wouldn't be out first priority. To address maintainability, and give our clients more freedom to interact with the target audience of the app (which primarily involves university students, Arboretum visitors and locals who want to experience the Arboretum), we decided to create a web app that would generate a configuration file. This configuration file could then be used by the app to dynamically generate content on the app. Doing this would also give a level of abstraction higher than we'd originally hoped. This would also help with debugging live app features, since we could isolate the problem to the contents of the configuration file.

## Inspiration and Potential Use

The idea of using a configuration file came from Prof. Woodley. He told us that a Senior Project group in a previous semester, working with mapping out the Adler Planetarium, used a configuration file to make the app work for any building (and not be specific only to the Planetarium). We used that idea as inspiration to model and design our app's hierarchy. Since the idea seemed simple enough to execute, we didn't look at any other sources. We did take the design one step further though by creating a web app that generates the configuration file for the app instead having our clients write a JSON based configuration file. This reduces user-error and aids in automation.

## Configuration File Design, Functionality and Link to iOS App

Before working on the web app, the most important question that needed to be answered was, what were the things we could allow our clients to control? First, since the app involved a map view, it made sense for our clients to be able to change the GPS coordinates and focus on different areas of the Arboretum if need be. Second, the map view contains markers showing the plant locations within the Arboretum. Clicking on these markers takes the user to the detailed view of the plant. This marker and plant detail data is received by the app via a REST API written by the plant database's original author, Chris Lindsey. The URL to the REST API could be set using the web app. This way, if the URL for the API were to ever change, the only change that would need to be made would be on the web app and the iOS app would not crash. Third, to increase engagement with the app we suggested to our clients that the app have a "Trivia of the Week" view. This would give users the incentive to come back to the app even if they weren't physically going to the Arboretum. We wanted to make updating trivia easy, so we included the ability to edit trivia via the web app. Similarly, the "About" page on the application could be updated via the web app. Another feature we decided to include was for our clients to add any notifications they'd like to let the users know of. For instance, "Cherry trees are in bloom!" This could simply be included on the web app. When the iOS application would first start, it would get the configuration file (generated by the web app), parse it and notify the user of the newly added notification.

This design would essentially give our clients all the control they needed to configure the app to their delight. This way we give a platform to the content creators (our clients) and not limit it to the content provider (us). It also makes the data representation of the app very evident and allows future developers to make sense of all the variables being manipulated to generate content. Essentially, a simple JSON file would contain details for representation of data across the entire application.

## Design Improvements and Limitations

Since the JSON file generated by the app is very small (few kilobytes tops), it does not have very big implications when it comes to scalability and performance. Based on the current implementation, every time the app is started afresh, it will make a GET call to retrieve the configuration file. Repeatedly doing this increases overhead and goes against the principle

of DRY (Don't Repeat Yourself). This could be improved in the future by adding a checksum that checks if there is a change to the configuration file. The app would get the configuration file only when a change was made to it, not otherwise. Since the amount of data that the config file holds is not significant, it doesn't affect the performance yet. More robust testing could also be included for future versions to eliminate faulty user inputs. Essentially only generating the configuration file when all validation passes and every input field fits the bill of the configuration file design.

## Technical Details and Documentation

### Source Code

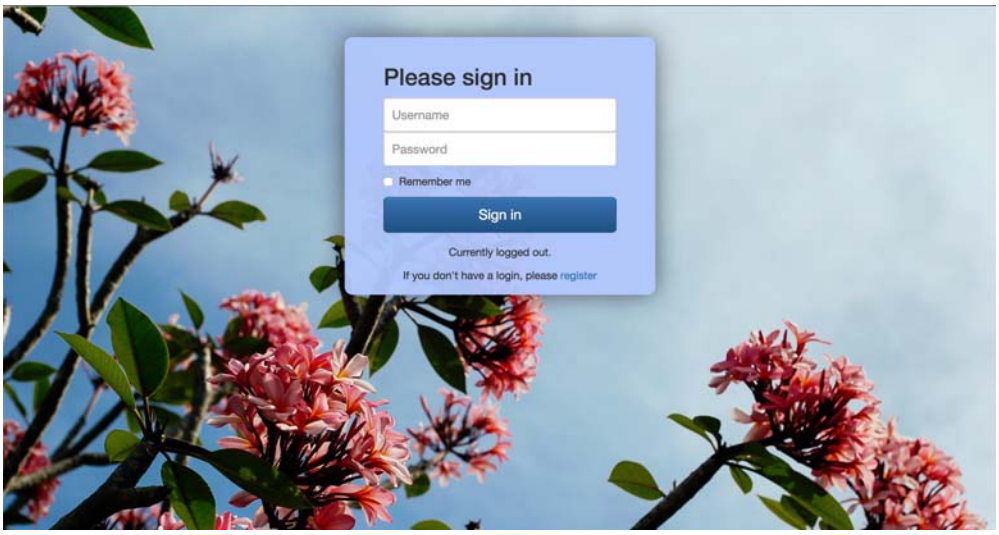
The source code can be found on GitHub. The URL is https://github.com/kfishster/arboretum

The repository is currently private since there are database credentials in some of the files, which we did not want to make public (for obvious reasons).

Contact hsingh10@illinois.edu or varshav2@illinois.edu for access/details to the repo or any further assistance.

### Web App

URL: http://arboretumwebapp.azurewebsites.net/





### How the Web App Works?

There are two main views of the application - the “login” view and the “edit configuration” view. The “edit configuration” view is visible only after login. The page is otherwise restricted (again, for obvious reasons).

**Technology Used**

The web app is written in PHP and MySQL. Since we assumed the app was going to be hosted on university servers (running cPanel and PhpMySql), we decided to write the app in PHP since portability and maintainability would not be an issue. Without that constraint, we considered using the MEAN stack or Flask before having to settle on PHP/MySQL.

**Code Organization**

The folder structure of the code is something like this

- assets
    - flowers.jpg
    - garden.jpg
- CSS
    - signin.css
    - config.css
    - includes
- includes
    - ChromePhp.php
    - db_connect.php
    - functions.php
    - logout.php
    - get_config.php
    - post_config.php
    - process_login.php
    - psl-config.php
- js
    - forms.js
    - sha512.js
    - config_details.js
- config_details.php
- error.php
- index.php

“Assets” includes the two background images used for the two views.

“CSS” includes the CSS files for the two views

“Includes” includes all the PHP scripts for backend processing

- ChromePhp.php – PHP Console logger. Makes debugging a whole lot easier!
- db_connect.php – connects to the db. Included in functions.php for GETting and POSTing data to the database
- functions.php – miscellaneous functions for creating and handling user sessions
- logout.php – closes user session and redirects back to login page (uses helper functions in functions.php)
- get_config.php – used by the iOS app to get the config file. Returns the last inserted row in the database.
- post_config.php – adds new configuration details to the database when the details are edited and submitted in the “edit configuration” view
- process_login.php – processes user login (uses helper functions from functions.php)
- psl-config.php – contains database connection credentials. Change the details of this file when porting

databases/web app

“js” includes all the associated javascript files

- forms.js handles login forms
- config_details.js handles validation and processing of the edit config form before making a POST request
- sha512.js is needed for login since the passwords are hashed with sha512 before being sent to process_login.php where rest of the validation takes place with the password hash and salt.

**Unfinished Parts**

Notifications handling along with the iOS application could use some love.

**Refactoring**

Handling notifications on the web app is such a hack! This could definitely use some refactoring. As of now, you can add as many notifications as you want but before saving the data to the database, I convert the array of notifications to a comma-separated string (because I was too lazy to create a new database to handle notifications). The issue here is that if a notification contains a comma, when parsing the string using a split() function, you’ll end up with half meaningless notifications. For instance, notif1: “This is a notification”, notif2: “hello, goodbye”. This would get stored to a notifications array as [“This is a notification”, “hello, goodbye”]. But then when this array is converted to String, it would look something like “This is a notification, hello, goodbye”. Using a split function on this string renders 3 notifications. This could be fixed in the future versions. This change needs to be made inside **config_details.js**

## Team Experience

Starting in the Spring semester, we decided to do two code sprints every week. We’d meet, discuss progress so far, things we’d work on that day and try to get as much done in 4-5 hour sprints as possible. We used Trello where the boards included “to-do”, “in-progress”, “complete-ish” and “complete”. This allowed us to keep track of our progress over the course of the semester and share our progress with our clients.

We used Facebook group chat for most communication and organization amongst team members. Everyone was available at all times and the communication was quick. Progress, issues and meeting times were discussed here.

Issue tracking was also done using Trello. Blocked progress would be moved out of the work pipeline and pushed back into to-do with descriptions of problems faced. This worked well in the beginning when we were trying to carve out work responsibilities for everyone. Once everyone had found an area they wanted to focus on, we did not use Trello as much.

The responsibilities were divided based on priorities (based on the minimum viable product). The roles we took on during the first meeting (Kirill working on the tabbed view of the app and listing a directory structure, Drew working on design, Shank working on the maps and James working layout and backend functionality) translated into what we ended up being the chunks of the features we took ownership of as the year went on.

There was an unwritten understanding that Kirill would spearhead the team. He was the one to create the Facebook group the day of group assignments and encourage meeting times for the first few weeks. There was always constant input from the rest of the team, which was always fruitful, but at any points when the team productivity would start to slip, he’d pull everyone back together and kind of lay out the goals we had ahead of ourselves. This worked out really well, overall since it gave all of us a sense of accountability and ownership towards the project.

There were times when we’d focus on lavish, stretch features a lot more instead of focusing on the features at hand. This led to some initial wasted time. But constant sprints allowed us to recalculate and chalk out what we really needed our deliverable to be. Scoping out the minimum viable product helped increase productivity manifold.

No client requirements were changed over the course of the year. We kept our clients in the loop for all the progress we made and got constant feedback from them before making any drastic changes. The “work-at-your-own-pace” nature of the project contrasted with an orderly structure of all other classes we were taking was a productivity roadblock at times. It required initiative and motivation on our parts to really carve out time to work on the project in addition to working on all other project assignments. The best part about working with my team was the sense of accountability that everyone had to deliver a good product to our clients. That really pushed us to work harder and get everything done on time. The fact that there are some unfinished parts to the project that we’d still like to focus on (even beyond graduation) is our commitment to the project.

### Communication with Client

Most of our communication with our clients was via email. I was the liaison from our group to schedule meeting times/agendas with our client liaison, Diane Anderson. Throughout the year, most meetings were biweekly. Starting March, we started holding weekly meetings to force ourselves to work towards completion and be held accountable by our clients. Meetings usually had an agenda to let clients know of the progress we’d made. We’d meet at the Plant Sciences Lab and present to them any progress we’d made. After this, our clients would give us feedback on what they liked and what they would want tweaked. We also used this time to address any questions our clients had about what was going on in terms of development. We would end the meeting with steps to take for the future and the potential agenda/progress for the next meeting.

### Lessons Learned as an Individual

Working on something that was going to be used by both our clients and students at the university, there was an impetus is delivering a quality product. When I was working on the web app and creating the login view, the first thing I did was encrypting passwords and use salts for storing in a database. No more plaintext passwords, especially with something that contained sensitive data as people’s life research (the plant data). There was a time when I created an additional view that served the same purpose as the edit config view except only to view data instead of posting config data. I decided to merge that view with the edit config view instead. When logged in, the edit config page makes a GET request to the get_config API call and populates all the fields inside the edit config view with the data from the current config file. This makes it easier for our clients to not have to worry about things they don’t want to change (the map coordinates, for example). This also improves the design to keep things simple instead of having to toggle between multiple views. Learning to work with PHP and MySQL was a newer experience since I’d previously developed web apps with Flask. It was good lesson in adapting to work with resources at our disposal.

### Lessons Learned as a Team

Working on this project was a great lesson in how to work for others and not just for us. Since most projects we worked on in school are projects we’re working on for classes that may never see real users, abandoning those projects is easy. In this case, we HAD to work towards completion and not compromise on quality. If the work wasn’t being done, it wasn’t disappearing either, unlike coursework (which disappears at the end of every semester and we start afresh). StackOverflow is everything you need to program anything! This is a fact that is reinforced every time I start to work on something new. Actively seeking out help from your teammates and talking through issues helps with overall development and boosts your morale. Working on a common project is very dependent on communication. If something is unclear, always ask! There are no stupid questions. Holding each other accountable is the best way to achieve productivity. The work you’re doing is for your clients. Always work to their satisfaction. Constant feedback is absolutely imperative!

## Individual Write Up - schundi2

### Research

My responsibility for this app is to work on the Map view. We first had to do research on iOS programming. When researching the development stack for iOS , we found that we had to Xcode on the OS X. We also had to choose a language for the project: either the community supported Objective C or the recently released Swift. We chose Swift since talks at the Apple Worldwide Developers Conference (WWDC) indicated that it would be the future choice of language to

build iOS applications.

The primary source for learning Swift was the official book called “The Swift Programming Language” written by Apple Inc. This book covers several important features of the Swift language such as Basic Operations, Functions, Classes, Closures, Methods, Inheritance, Generics, Properties, etc. To learn iOS programming with Swift, I used the videos on the “250+ Swift Language Tutorials” playlist on YouTube [1]. Another frequent resource is StackOverflow; questions tagged with ‘swift’ were used a lot.

To implement the map view and other such related tasks, we researched various toolkits for maps on iOS. We found the Apple MapKit, Google Map iOS SDK, and MapBox iOS SDK were the most popular kits to render and integrating maps in iOS applications. In the end, we found that Apple MapKit provided the necessary features to satisfy our app’s requirements. Furthermore, setting up the Apple MapKit was quite easy and was available for our unlimited use without any developer restrictions or a pricing model.

An important problem while adding annotations to a map is to minimize the clutter on the screen. For example, we do not want to load and display 10000 pins on a 4-inch screen. I figured this was a common problem and search for packages or existing solutions to this problem on StackOverflow, Apple iOS Developer Docs, tech blogs, and GitHub.

[1] https://www.youtube.com/playlist?list=PLxwBNxx9j4PUpjCEVwjqFvNecNvQ6Dj6G

## Solution

**Map**

There are two major components to the Map view: the dynamic downloading of points of interest (this includes plants, trees, gardens, etc.) and the dynamic displaying of pins corresponding to those points. Fortunately, the Arboretum staff maintains geographical data by tagging and recording each plant’s GPS (Global Positioning System) coordinates. My responsibility for this application is to build these two components in the Map view.

In order to build the Map view, we used the Apple MapKit framework. As soon as the application loads, the Map view’s controller sends an asynchronous GET request to receive geographical data. Plants’ GPS coordinates, names, descriptions, and corresponding item keys are included in the response. The application initializes the corresponding pins as the data is downloaded asynchronously. This data is also stored locally so that subsequent app loads do not have to download it if no changes have occurred. For example, a piece of data for “Hosta Gardens” contains the garden’s location and a brief description. The MapController will initialize a pin with the corresponding label of “Hosta Gardens” and will place this pin at garden’s coordinates. And when a user selects the pin, MapController will redirect the user to a detailed view that contains more information and photographs of the “Hosta Gardens.” The first component of the Map view, the dynamic loading of geographic data, is implemented in this way.

In order to present an aesthetic design, the pins must be dynamically pruned so that the map does not become cluttered. Several approaches that could work were considered:

At first, I planned on assigning priorities to each piece of geographic data. Then, whether a pin p must be displayed is a function of its priority, given by the function priority(p), and the current map view’s zoom factor, given by zoom. Pins with a higher priority will be displayed when the map is zoomed out to the greatest level. From a practical perspective, when the map is zoomed out, a dense cluster of pins is not valuable; however, a few important pins convey more information. As a user moves through the paths and zooms into the map, the cluster of pins will expand and be more relevant to the user. For example, a garden (for example, “Hartley Gardens”) has a high priority, but a single plant (for example, a rose) will have a lower priority. So, when it is first opened, the Map view presents a zoomed out map so that the entire Arboretum appears. At this point, it only consists of the pins of major gardens (i.e., high priority). However, once a user zooms into the map, the application displays pins of the plants (i.e., lower priority) in the zoom target.

However, after more thought, I figured that this method was not efficient. First, each item had to be assigned a priority. This priority had to be manually given or had to be based on the type of item. Secondly, the details of the function had to derived. There is no way to figure out the zoom threshold without extensive trial and error. The function had to be tweaked and tested for each form factor and each screen size.

After research, I found that this is a very common problem with iOS developers because Apple's MapKit does not provide this specific functionality. The first solution was to use an alternate Map framework. We found that MapBox provided pin clustering. But, setting up MapBox, initializing the map's layers, and placing pins proved to be much more involved than it did with MapKit. Additionally, MapBox required a developer fee and restricted the number of map loads. Thus, we decided not to use MapBox. Then, we found several third-party packages [1][2][3][4] that solve this problem by using QuadTrees. CCHMapClusterController [1], provided under the MIT license, was finally chosen for its licensing, simplicity, and extensibility. Adding CCHMapClusterController to our project was done using CocaoPods [5]. Each initialized pin/annotation is added to the map view's cluster controller. CCHMapClusterController then prunes the pins so that the map is clean and uncluttered. The advantages of this approach are 1) by using an active open-source package, maintaining the clustering feature of the application is easy; 2) the QuadTree implementation provides better performance, even when there are thousands of pins on a small screen.

**Design Patterns**

Model-View-Controller (MVC) is used throughout the application. The map view has a corresponding controller which responds to various events (onLoad, didSelectAnnotation, etc).

**Scalability and Performance**

As described in the functional requirements, we had to make the application as abstract and flexible as possible, while providing a seamless experience for the users. And, I believe we've had made progress towards that goal. Most of the application can be configured remotely. The web application built by Harjas provides several options that can be tweaked. On each load of the application, a checksum of these settings are read to see if any changes have been made. If so, the application will load and reflect the client's updates. The Map view was designed to handle at least several hundred pins and the corresponding data is also loaded remotely. This way, we have made the application quite abstract and flexible.

[1] https://github.com/choefele/CCHMapClusterController

[2] https://robots.thoughtbot.com/how-to-handle-large-amounts-of-data-on-maps

[3] https://www.infinum.co/the-capsized-eight/articles/a-blazingly-fast-open-source-algorithm-for-poi-clustering-on-ios

[4] http://getsuperpin.com/

[5] https://cocoapods.org/

## Source Code

The source code for this project is hosted on GitHub. It is located in a private repository called Arboretum Senior Project owned by a user named kfishster (Kirill Varshavskiy). The source code is organized like a typical Swift project in iOS Xcode. We are also using CocoaPods to manage and install third-party libraries and packages.

More improvements can be made with error handling. What happens if we receive data that's not in the expected format? What happens if network drops in the middle of transferring data or the app is not granted permission to access location and data? More than ensuring that the app doesn't crash, which has been taken care of in most cases, how should the app behave in these situations? The functionality of the app is currently dependent upon connection to the internet. One possible solution is to distribute the application with default data. Plant information in the directory most likely does not change often, the decades-old trees' locations will also not likely change. Is it feasible to include the data corresponding these invariants with the application (i.e., is it reliable, will it occupy too much memory)?

A more thorough discussion of the source code has been developed by Kirill here [1].

[1] https://seniorprojects.cs.illinois.edu/confluence/pages/viewpage.action?pageId=64196848

## Team Experience

During the project, our organization was maintained in two ways. The first is through communication - which was done mostly using a Facebook group chat. The second is through Trello, where we added tasks, ideas, mockups, deadlines,

goals, etc. These two sites helped us communicate and organize our efforts effectively.

We maintained very close contact with our client. During the Spring semester, we had a meeting with the client almost every week. During this meeting, we showcased our progress and received feedback. Since they did not really have a concrete set of requirements or ideas about how our application should look and function, this feedback was incremental and progressive through each update. After each meeting, our application was further refined and tweaked according the received feedback. These meetings also served as a time to ask for resources, such as GPS data or an API to receive plant information.

The division of labor was decided sometime during the Fall semester. After designing our application, we realized that there were five broad components on the iOS side and a web component. The Map and the Directory views were the major parts on the iOS side. Kirill, Drew, and James worked on the Directory and the remaining three views, I worked on the Map, and Harjas worked on the web application. We had periodic code sprints/hackathons to contribute to the project and make progress towards a working application. Although, because of other coursework and projects, there was a decline in our productivity in the middle of the second semester. However, we picked up our pace and built a working, albeit not perfect, application for our presentation.

While it was not explicitly stated, Kirill served as the backbone to our group. He made significant contributions to the codebase and led the timeline of development. Harjas served an extremely effective bridge between our team and the client. Together, they stood out as natural leaders for our group. Meetings with the client were led by them and ideas were most efficiently communicated. Overall, I am grateful to have been part of this group and am glad that it worked well for the group's dynamic.

## Lessons Learned as an Individual

Working for a client is a much different experience than what I've had during an internship or group projects for coursework. There is a stronger yearn to deliver and meet the client's expectations. The fact that our client, in this case, was less technical meant that they sometimes did not understand how much effort something seemingly simple can take. It meant that the onus was on us to effectively inform them about what progress has been made and how long certain things can take.

Technically, I learned a lot about iOS development. Many questions I came across did not have solutions in Swift. It was evident that the iOS development community has not yet transitioned away from Objective C. Naturally, I learned that Objective C libraries and code can be integrated with a Swift application by using an library bridge. This can be done by creating an Objective-C header file and configuring the appropriate values in the build settings. I also learned the benefits of using a mature language instead of a young one. A single update to Xcode/Swift caused our unchanged application to yield around 70 bugs. Most were easy to fix (after realizing what we were doing wrong), but in hindsight, we might have found more help and have had a better time if we chose Objective-C.

I realized that learning by doing a project can be very helpful when learning a new technology. But, I also realized that jumping right away into building the application is extremely unproductive. After struggling and "Google-coding"/"StackOverflow-coding," I decided to peruse the Swift manual and look at a few resources (listed in research). This helped a lot and prevented me from getting stumped by simple bugs.

I also learned that there was an Arboretum in University of Illinois and that it was located towards the south of Florida Avenue Residence Halls.

## Recommendations for Improvement or Extension

I believe that the Map view has plenty of room to improve. I hope to make these changes before the application is released. There are several features that still need to be implemented (in increasing order of importance):

The first such feature is to get the Bookmarks feature working. A user should be able to place a pin anywhere and store the location as a bookmark. This was not a critical feature or a requirement to complete by the demo date. As a result, it was neglected. The way to do this would be to use CoreData to store the bookmarks once placed on the map.

The second feature to add on the Map view is to enable a "Follow mode." In this mode, the map will move along with the

user, much like real-time navigation with directions. This will allow the user to focus on the nature more. They do not have to worry about their location on the map because it will always been in the center of the screen. Finding out more about any nearby plant is as easy as looking at the screen and tapping on a pin.

The third feature on the Map is to load custom icons for the pins. Having custom pins for each pin on the map can help the user find his/her destination quickly, especially because the density of pins is high. For example, if they are looking for some plants nearby, it helps if there are pins that show a leaf icon instead of the default red push-pin. This can be done by extending the AnnotationView and setting the image based on the target's type (plant or tree or building or flower, etc).

The fourth feature to add would be to add links from the detailed view of an item (from the Directory view) to a location on the Map. This is an important feature that allows users who are at the Arboretum to see a specific plant to find it faster. Since each item on the Map shares a key with an item in the directory, the location can be append the detailed view of that item.

The fifth and most important feature to add is to more thoroughly integrate CoreData with the Map. This way, pins aren't loaded each time the application is started.

| User | Edits | Comments | Labels |
|---|---|---|---|
| Kirill Varshavskiy | 20 | 0 | 0 |
| James Beck | 16 | 0 | 0 |
| Drew Metz | 13 | 0 | 0 |
| Harjas Singh | 5 | 0 | 0 |
| Michael Woodley | 5 | 0 | 0 |
| Sairam Sasank Chundi | 3 | 0 | 0 |
| Xiaodan Zhang | 3 | 0 | 0 |

**Recently Updated**

Arboretum Tree Walk Mobile App 2014 Home updated by Sairam Sasank Chundi (view change) May 08, 2015

Arboretum Tree Walk Mobile App 2014 Home updated by Harjas Singh (view change) May 08, 2015

Arboretum Tree Walk Mobile App 2014 Home updated by Drew Metz (view change) May 08, 2015

Screenshot 2015-05-08 17.45.11.png attached by Harjas Singh May 08, 2015

Screenshot 2015-05-08 17.44.38.png attached by Harjas Singh May 08, 2015

Arboretum Tree Walk Mobile App 2014 Home updated by James Beck (view change) May 08, 2015

Arboretum Tree Walk Mobile App 2014 Home updated by Kirill Varshavskiy (view change) May 07, 2015

Application Hierarchy + Development Notes updated by Kirill Varshavskiy (view change) Apr 22, 2015

Screen Shot 2015-04-22 at 6.03.49 PM.png attached by Kirill Varshavskiy Apr 22, 2015

iOS Simulator Screen Shot Apr 22, 2015, 5.54.00 PM.png

**Navigate space**

Search

Application Hierarchy + Development Notes

First UI Design

Ideas

| | |
|---|---|
| iOS Simulator Screen Shot Apr 22, 2015, 5.54.06 PM.png | Apr 22, 2015 |
| attached by Kirill Varshavskiy | Apr 22, 2015 |
| iOS Simulator Screen Shot Apr 22, 2015, 5.54.12 PM.png | |
| attached by Kirill Varshavskiy | Apr 22, 2015 |
| iOS Simulator Screen Shot Apr 22, 2015, 5.54.22 PM.png | |
| attached by Kirill Varshavskiy | Apr 22, 2015 |
| iOS Simulator Screen Shot Apr 22, 2015, 5.54.32 PM.png | |
| attached by Kirill Varshavskiy | Apr 22, 2015 |
| iOS Simulator Screen Shot Apr 22, 2015, 5.54.33 PM.png | |
| attached by Kirill Varshavskiy | Apr 22, 2015 |

Labels None

## 3 Child Pages

- Application Hierarchy + Development Notes
- First UI Design
- Ideas