Poison Ivy Appalachian Trail
Mega-Transect Data Collection Application
Final Report

Project Manager:
Dr. John Jelesko

Project Group:
Anthony Barbee
Alisher Pazylbekov
Bing Qi
Nathan Rosa

Executive Summary

This document contains the details and specifications for the Poison Ivy Appalachian Trail Mega-Transect Data Collection Application; referred to as PIAT. Included in this report are the software requirements, design specification, implementation process, as well as the prototyping and the refinement process for this application. Additionally, the testing of said application and two manuals for the users and developers will be included at the end of the report. This is so that any future users and developers will be able to interact with and properly use this application as well as understand how it works.

This project is a collaboration between Virginia Tech undergraduate students and Dr. John Jelesko, an associate professor working in Virginia Tech's Department of Plant Pathology, Physiology, and Weed Science. The initial details and expected goals were drafted up after meeting with Dr. Jelesko and hearing his expectations about the PIAT application. The primary objective of this application, as outlined by Dr. Jelesko, is for him and his team to track the growth of poison ivy along the appalachian trail. Users will be able to use the application to make entries about poison ivy's presence or absence as they see it along the trail. The application will have a local database on the phone that can be synced with a larger database. This will allow for multiple users to track data and upload it to the main server. Alongside the application, there will be an external device that pairs with the user's Android device; which will allow for users to log entries without interacting with the phone itself.

Table of Contents

User Manual 5
1) Introduction 5
a) Background 5
b) Concept Statement 5
c) Project Scope 5
2) Requirements 6
a) Project Description 6
b) Technologies Used 7
c) User's Role 7
3) Tutorials 8
a) Application Layout 8
b) Hardware Interface 12
c) Hardware Button Event Flow Instruction 12
d) Dim and Undim Screen Before and After Using Hardware 13
e) Migrating Application to External SD Card 13
f) Zipping and Unzipping Project 13
g) Accessing Exported Files 14
Developer Manual 15
1) Design 15
a) Application Design 15
b) Database Design 16
c) Hardware Interface 19
d) Explanation of Design Choices 20
2) Implementation 21
a) Implementation Phased Approach 21
b) Major Tasks 22
c) Implementing Testing/ Version Control 22
d) Application Technologies (Referencing Table in Section 2b.) 23

e) Personnel 24
f) Proposed Schedule 24
g) Risks and Contingencies 24
3) Prototyping 25
a) User Interface 25
b) GPS Location 26
c) External Hardware Interaction Hardware 27
d) MySQL and Android Integration 29
4) Refinement 31
a) Database Updates 31
b) Functionality Changes 31
c) Schedule Changes 31
5) Testing 33
a) SQL Database Migration & Security 33
b) External Hardware 33
c) GPS Tracker 34
d) External SD Card Migration 35
6) Tutorials 36
a) Setting up Server PHP files 36
Acknowledgements 38
Bibliography 39

User Manual

1) Introduction

a) Background

Dr. John G. Jelesko is looking for a solution that will provide his team of plant pathologists the ability to record information on current poison ivy trends along the Appalachian Trail. Dr. Jelesko, the founder of the Virginia Tech Jelesko Laboratory, is interested in recording the collection of text and image data about poison ivy: growth, leaf shape, herbivory, microbial disease symptoms, and geospatial data. His laboratory solicits poison ivy observations so that they can make new discoveries about poison ivy that are beyond the scope of the personnel in the Jelesko lab. These findings will hopefully lead to new insights into poison ivy ecology, biogeography, and future novel poison ivy management methods. [1]

b) Concept Statement

Poison Ivy Appalachian Trail Mega-Transect Data Collection, or PIAT, should help Dr. Jelesko and his team of researchers, especially those who fit the persona of experienced backpackers/hikers who have a knowledgeable background in regards to poison ivy plant pathology. Throughout the Appalachian Trail, there is more to worry about than just having to record poison ivy observations, and the backpackers need to be focusing on their footing, miles covered, environmental hazards, self-wellness, and each other. PIAT aims to alleviate the complexity of data collection, as well as to simplify the whole process. The Android application provides a transparent and efficient experience each time they use it to help them focus on their backpacking rather than cumbersome data collection.

c) Project Scope

PIAT will be developed as an Android application. This way, Dr. Jelesko and his team of researchers can use it on the trail. The application interface should be intuitive enough that the intended users are capable of logging their data with ease. We can also use terminology relevant to poison ivy and the Appalachian Trial since our users are knowledgeable, at least to some degree, of the subject matter.

2) Requirements

a) Project Description

Develop an Android cell phone application and external button rig to facilitate further the ease of data collection for various types of data collection along the Poison Ivy AT Mega Transect. This application will collect data about poison ivy:

1) Absence/presence & growth habit with
2) Corresponding GPS coordinates
3) Assigning unique identification codes
4) Timestamp of collection

Approximately 1,200 poison ivy leaf samples will be collected along the Appalachian Trail. These leaf samples will just be data points collected along the way with different codes representing the type of poison ivy found. These samples will be collected every 2 miles along the Appalachian Trail recording the poison ivy absence/presence, growth habit, time of collection, and GPS coordinate data which will be uploaded to a Virginia Tech server which will be hosting the MySQL database. The goal is to simplify the data collection of the most frequent data, to minimize handling the cell phone's battery usage, by designing and implementing a solution utilizing a micro-USB or audio jack interfaced button actuation device.

The Appalachian Trail from Maine to Georgia

Dr. Jelesko and his team intend to use the PIAT application to log their findings and data collections every two miles and when reaching the white trail blazes. Along the trail white trail blazes are frequently found indicating to hikers that they are still indeed on that trail. To the right is a picture of one such white trail blaze.

b) Technologies Used

The project will require multiple technologies to be utilized in the design, development, implementation, and prototyping of PIAT. Below are the following hardware and software applications that will be used.

Software	Hardware
MySQL (Primary Database)	Local Server (MySQL and PHP prototype server)
Android Studio (Java IDE for Android)	Selfie Stick (Possible hardware actuator)
Java (Primary Language)	Headphones w/ Volume Control (Possible hardware actuator)
PHP (Java and MySQL connection layer)	Android Phone (Run and test the PIAT application)

c) User's Role

The users for the PIAT application will be members of Dr. Jelesko's research team. These members will be hikers using the PIAT app amongst several groups starting at different trailheads on the Appalachian Trail. They will be using the application the whole time while on the trail. They will interact most of the time with the application through a hard-wired button, which interacts with PIAT. For most of their observations, they will just have to click a button to create a recording. Otherwise, they will have to make any recordings that require more detail by physically opening the app on the smartphone.

3) Tutorials

a) Application Layout

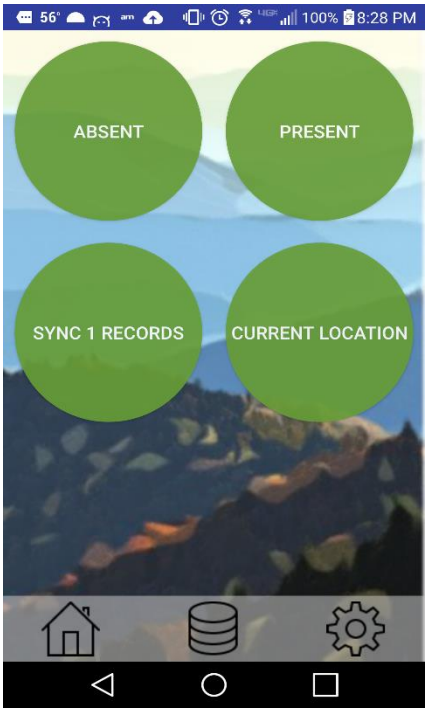


Figure 1: Current User Interface

Upon starting the application and accepting the permission the user is taken to the home screen. There are four buttons and a navigation bar at the bottom of the application.

Buttons:

- **Absent**: Creates an absent record in the local database
- **Present**: Opens a dialog for the user to provide an optional plant id and allows the user to choose the type (figure x).
- **Sync**: Will display the number of unsynced files and will sync to the online database if a proper network connection is available.
- **Current Location**: Displays the most recently calculated GPS coordinate (figure x).

Navigation Bar:

Home Button - Brings user back to the homepage (*Figure 1*).

Database Button - Displays the local database in a table format the current records and their respective properties. The table has two separate views, portrait (*Figure 3*) and landscape (*Figure 4*).

Settings Button - Displays the input fields for the user to input their respective teams, MySQL username, and password. (*Figure 5)*

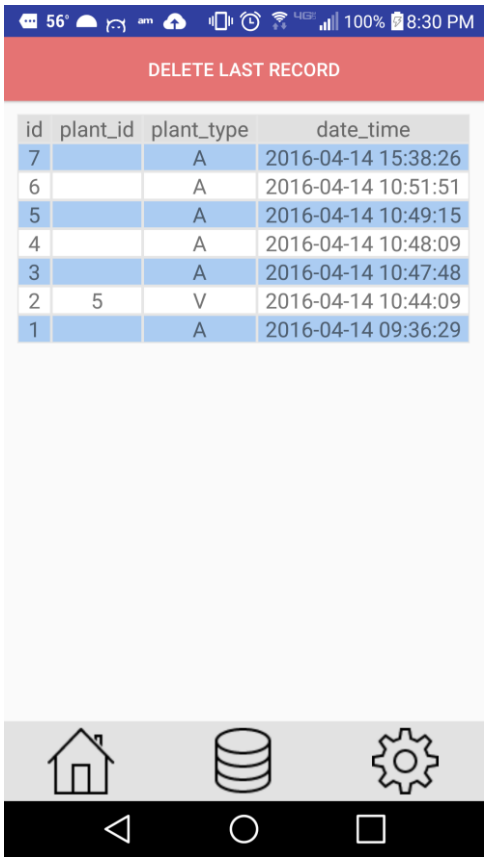
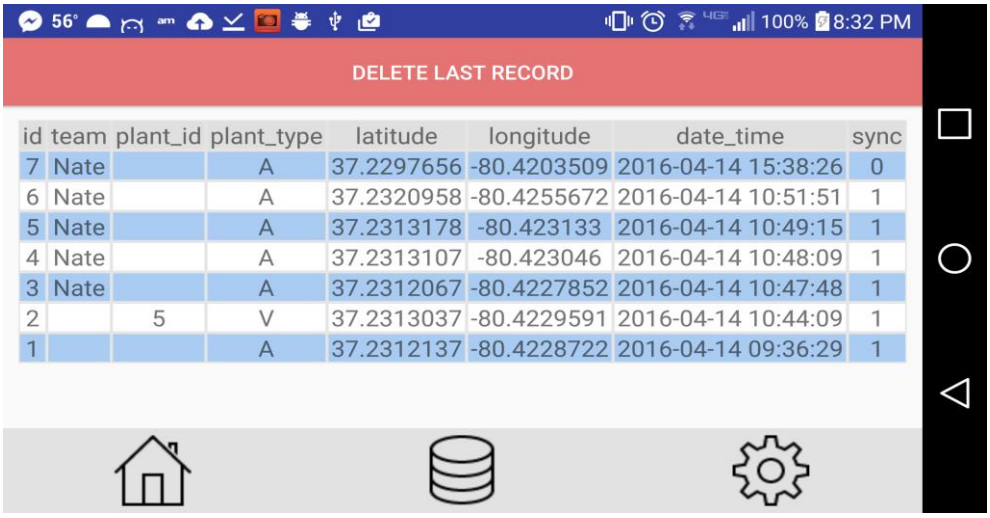


Figure 3: Local Database Portrait View



id	team	plant_id	plant_type	latitude	longitude	date_time	sync
7	Nate		A	37.2297656	-80.4203509	2016-04-14 15:38:26	0
6	Nate		A	37.2320958	-80.4255672	2016-04-14 10:51:51	1
5	Nate		A	37.2313178	-80.423133	2016-04-14 10:49:15	1
4	Nate		A	37.2313107	-80.423046	2016-04-14 10:48:09	1
3	Nate		A	37.2312067	-80.4227852	2016-04-14 10:47:48	1
2		5	V	37.2313037	-80.4229591	2016-04-14 10:44:09	1
1			A	37.2312137	-80.4228722	2016-04-14 09:36:29	1

Figure 4: Local Database Landscape View

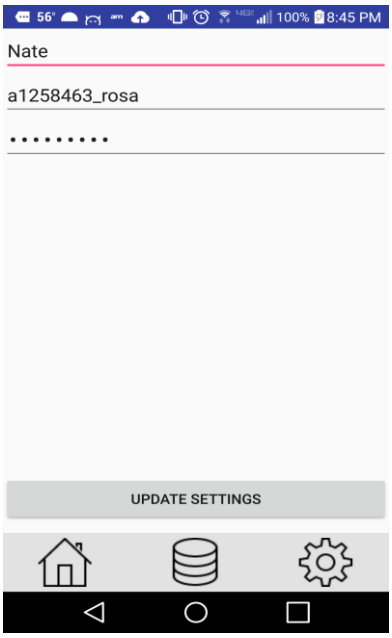


Figure 5: Settings Screen

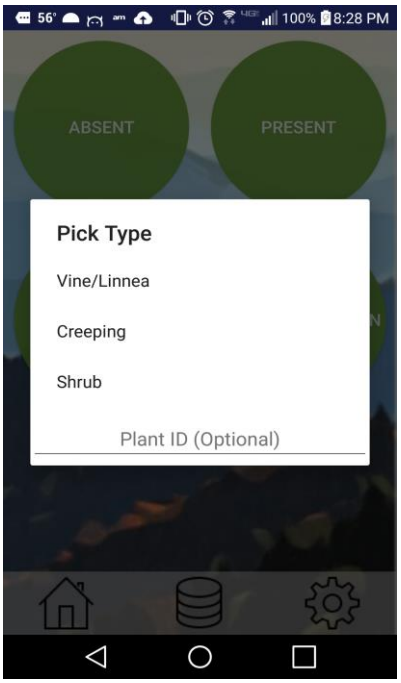


Figure 6: Poison Ivy Presence Options

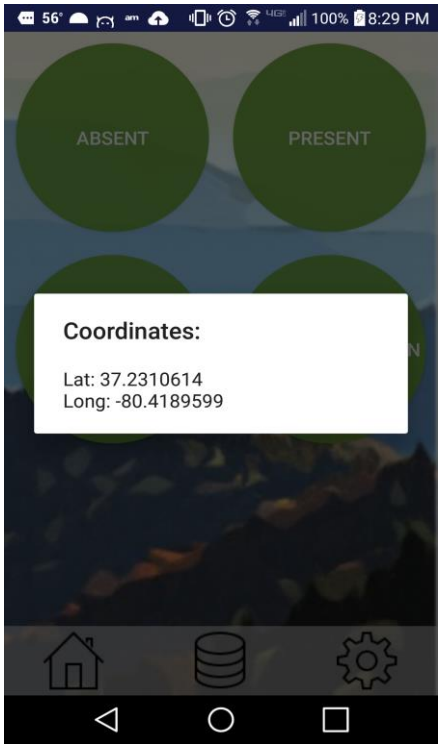


Figure 7: Logged Coordinates Feedback

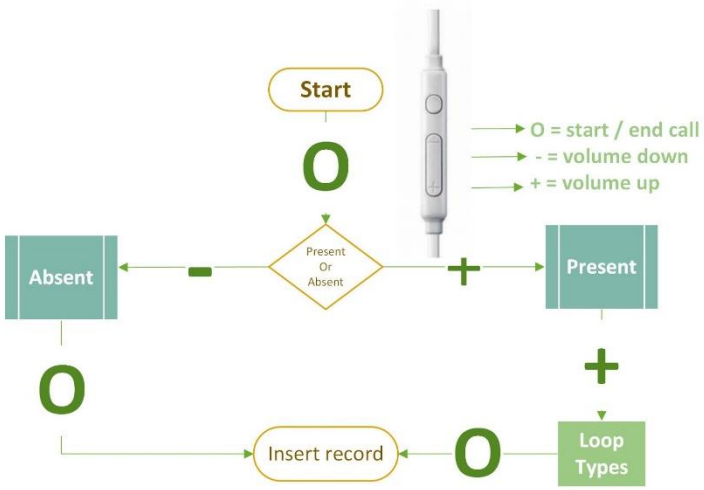
b) Hardware Interface

Records can be recorded with the appropriate hardware. The user tutorial on how to use the PIAT app accordingly can be found at:

https://www.youtube.com/watch?v=tpzp9H9hXF8

Figure 2: Hardware Button Usage

c) Hardware Button Event Flow Instruction



d) Dim and Undim Screen Before and After Using Hardware

Because most of the time user will be logging the data with the headphones the screen will need to be on all the time. This will use a lot of battery, we included a dimming functionality embedded into the application.

By performing **a long press gesture** on the screen anywhere except on a button. The screen will gradually dim to very low brightness for the sake of reducing energy usage.

By performing a **flick gesture** on the screen while it is on the low brightness state, the brightness will increase to the original level.

e) Migrating Application to External SD Card

It is recommended that the PIAT application be moved over to an external SD prior to running and/ or logging data into the local database. This is so that the user can move the SD card and the data to different Android devices if need be. As such, we will include a tutorial below that will help the user transfer the application over to the SD card. [2]

Steps:

1.) Go to Settings on your Android device
2.) Select the Application Manager underneath Settings
3.) Select the desired application to be moved to the external SD card
4.) Select either "Change" or "Move to SD Card"
5.) Choose the SD Card (if prompted)

f) Zipping and Unzipping Project

Unzipping:

Once the compressed application is downloaded, unzip/decompress it and the actual folder for the application should be downloaded to the same folder as the zipped file. There are two ways to relocate it to the folder. In either operating system, go to the "Users" folder and look for the one with the right name of either the computer or the owner of the computer. Click on this folder and look for the folder named "AndroidStudioProjects". Now from the source folder, either right-click on the unzipped folder and copy it to the "AndroidStudioProjects" folder or simply drag and drop. Once the unzipped folder is in this folder, open Android Studio and select "Open an existing Android Studio project" and the application should open up.

Zipping:

Once the application is done, click on "Project" in the top left corner area and look for the folder named "app". Right click this folder and if on an OS X machine, click "Reveal in Finder" and if on a Windows machine, click "Show in Explorer". Once Finder or Explorer opens, simply right click on the "app" folder and select the compress option. The last step would be to rename the zipped file to a desired name and then relocate it to the destination folder.

g) Accessing Exported Files

Access the File manager and navigate to:

Android>data>com.cs4624.poison.ivymap>files

Once in the files folder you can find poison_ivy.db and poison_ivy.csv files.

Developer Manual

1) Design

a) Application Design

1) Goals

PIAT will provide a simple and efficient data collection interface that will use low amounts of battery and require minimal user interaction. Also, once a proper internet connection is made it will give the user the ability to connect and upload to a MySQL.

2) Wireframes

Below are the initial wireframes for our application. In (*Figure 1)* the representation of the User Interface that we initially proposed to Dr. Jelesko. With his feedback, we developed a functional prototype that can be seen later on in this documentation.
(See Section 5.a)

Figure 2 Preliminary Wireframes

3) Concept Map and Configuration

This diagram demonstrates the different components and how they interact in the PIAT application

Android Application configuration:

PIAT is currently developing in Android Studio IDE.

Compile SDK version: 23

Minimum SDK version: 18

Build tool version: 22.0.1

Voice feedback source audio files are inside the raw folder under "res". Audio file format is feasible both Mp3 and ogg.

b) Database Design

The Android application will interact with PHP files stored on the server which will execute the MYSQL queries. The PHP files will allow us to insert, fetch, and delete data on the MySQL server. [2]

Virginia Tech's servers will host the MySQL database; the table in the database will consist of six columns that capture all the data needed for Dr. Jelesko and his team of researchers. There will be a massive amount of records being uploaded to the database. Since Dr. Jelesko and his team intend to log the poison ivy, if any is found, every white trail marker and every two miles when leaves are collected. Due to this expected high frequency of entries, the

database will also need to be highly scalable. Below are the intended attributes to be associated with the logged data.

Intended Columns for database with examples:

ID	Leaf ID	PI Type	Lat	Long	Date
1	A_1231	V	80.1123456	111.1234567	2016-03-29 21:15:32

ID:

The first column is an auto-incremented unique identifier that will go with each data record upon uploading it to the MySQL server.

Leaf ID:

The second column represents the poison ivy id for each leaf collected at 2-mile increments. The column will be a string that is nullable because it won't be recorded at white blaze markers. This id will be generated based on the team name and a specified user inputted, which will be concatenated together. This is done so that when multiple users upload their data to the MySQL server there is no confusion between whose leaf sample belongs to which team.

PI Type:

The third column represents the types of poison ivy that will be recorded along the Appalachian Trail. There will only be four types of poison ivy placeholders being recorded that will use a specific character either V, S, C, or A for vine/liana, shrub, creeping, or absence respectively.

Table 1: Examples of what each type of poison ivy looks like.

Code	Type of Poison Ivy	Picture
1	Vine/Liana	 http://www.poison-ivy.org/climbing-poison-ivy
2	Shrub	 http://errinv.smugmug.com/Hiking/Grizzly-Flat-Road/i-3BMqJnB/0/L/P4210178-untitled-L.jpg
3	Creeping	 http://static.howstuffworks.com/gif/poison-ivy-4.jpg

Latitude:

The fourth column will hold the latitude for the Global Positioning System (GPS) coordinates of where each data collection record has taken place.

Longitude:

The fifth column will hold the longitude for the Global Positioning System (GPS) coordinates of where each data collection record has taken place.

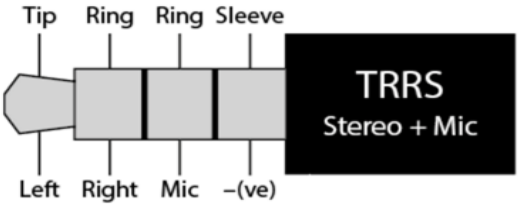
Date:

The final column will contain the time and date of when the sample is recorded by a user. This way, if the sample cannot be uploaded immediately, Dr. Jelesko and his team will

be able to approximate where the sample was recorded on the trail based on the samples recorded before and after the current one.

c) Hardware Interface

The Android application will make use of an outside interface that will enable users to efficiently record and collect data through a series of button presses. There are two possible options both of which integrate with the phone's audio jack to interact with the PIAT application.



http://www.epanorama.net/newepa/2014/09/15/android-device-external-mic-wiring/

The selfie stick controller or the headphones displayed below could be used as an input for the Android device. This is required so that Dr. Jelesko and his staff can record their observations hands-free. He and his team will be clicking a button connected via the auxiliary port to communicate with the PIAT application. The current design for recording data has Dr. Jelesko and his team of researchers clicking the button once to record and absence of poison ivy or holding the button for one, two, or three seconds to indicate vine/liana, creeping, or shrub respectively.

http://swapscart.com/p/48086-plus-extendable.html

d) Explanation of Design Choices

1) Wired over Bluetooth for Connectivity

We decided on using a wired auxiliary cord with buttons to interface with our application. This will reduce the amount of battery usage in comparison to using a Bluetooth connection. This is key because Dr. Jelesko and his team will have a limited power supply on the Appalachian Trail.

2) Location Accuracy vs. Battery Consumption

When on the trail, the hikers will need to have their phone's GPS function on so that the application can properly detect them. We need to pinpoint an appropriate time for the refresh rate of the GPS location so as not to cause too much battery consumption while still maintaining accuracy. If the GPS sampling rate is too high, the battery life of the phone will be drained quicker with each location call. [3] However, if the sampling frequency is too low, then the location in the application will not update quick enough; which will result in inaccuracy when registering the presence of poison ivy.

3) Local Server vs. Virginia Tech Servers

We decide on using a local server for developing and testing our application's ability to upload to an online MySQL server because it is easy to set up and suitable for creating a quick prototype. Also, since there is a short time frame for this project to be completed we don't have to wait for Virginia Tech to set up any of the services.

4) Scalability and Limitations of MySQL

According to MySQL's website, scalability, in regards to databases, refers to the ability to spread both the database and the load of the application queries across multiple MySQL servers. MySQL is designed in such a way that it can be scaled to large sized databases as well as smaller sized ones. [9] Thus, there should not be a difference or any major complications that arise when migrating from our local server to Virginia Tech's server.

As of now, the localized application is reflecting this concept of scalability. The current server we are using has a maximum bandwidth of 100 GB and can handle 100,000 queries per second. This estimate was calculated based on the average time for a query (0.0000001 seconds). However, as we are not at the point of transferring the database over to be hosted on Virginia Tech's server, we do not have firsthand experience with the limitations and boundaries. We have contacted VT Hosting Services to obtain more information about getting a subdomain at www.ppws.piat.vt.edu. However, since Dr.Jelesko already has a subdomain, we will have to discuss the migration of the database at our next meeting on April 6th.

2) Implementation

a) Implementation Phased Approach

1) Initiation Phase

The purpose of the Initiation phase is to mobilize Dr. Jelesko's lab research out into the Appalachian. To start off, we reached out to Dr. Jelesko to discuss our involvement with this project. We also had to ensure that Dr. Jelesko deemed us qualified enough to complete the task to his specifications. Once we had the approval of Dr. Jelesko, we began planning out the project via wireframe and some base level functionality coding, such as: obtaining GPS location and the interaction with the external device.

2) Project-Level Installation Phase - (*This is the current phase of our development as of 3/31/2016)*

During the project-level installation phase, all targeted tasks for the project will be implemented and tested. This will be done locally using similar databases layouts and software. A functional prototype will be developed and presented to Dr. Jelesko for feedback purposes. The external hardware device will also be implemented alongside the software application.

3) Enterprise-Level Installation Phase

During the enterprise-level installation phase, the project will be migrated onto the Virginia Tech dedicated servers and thoroughly tested for the MySQL database functionality. Debugging and testing on the Virginia Tech server will be conducted to assure that the application is fully functional and compatible. We will also be exploring the addition of new features, under the supervision of Dr. Jelesko, such as the storing of photos; which will also transition our project into the next phase.

4) Maintenance Phase

In the maintenance phase, the final refinement of the PIAT application will occur. We will be providing any and all necessary support to Dr. Jelesko and his team up until the final turnover of the project and documentation. This includes, but is not limited to, making sure that the project code has been thoroughly commented so that Dr. Jelesko and any future users can understand it, testing the application in the field, and improving efficiency as necessary. This is also the phase where we will test out other viable options for external hardware.

5) Conclusion

A phased approach to implementing project management can dramatically increase the probability of success because it helps to overcome resistance to change, creates an opportunity to incorporate lessons learned into the design of the project

management infrastructure, and ensures that enterprise-level information is available to our client.

b) Major Tasks

1) Setup MySQL

Deploy my MySQL to a free hosting service and create the necessary tables that will be used to store the required information.

2) Create Android Studio Project and Setup GitHub

Create an Android project, which will use GitHub for version control and to compile all our files into one location, so that the team will have access to the project and be able to make edits remotely with everyone being able to see them. We will grant Dr. Jelesko access to this account, as well as comment our code so that he and any other future developers on this project will understand our work.

(Github Project Link: https://github.com/McBrosa/IvyMap)

3) Develop the Application

Create the user interface with Google Maps layout and GPS capabilities. Next, develop an interface using PHP and Java to interact with MySQL database.

4) Implement Hardware Interface

Create a way to interface with the app using a hardwired button

5) Debug the Application

Create test cases and use the application out in the field to find any bug or unforeseen errors. Apply patches and continue testing.

c) Implementing Testing/ Version Control

As the PIAT application is being developed, it will constantly be undergoing various adaptations and updates. As such, there will need to be a sufficient amount of testing for the application to ensure the intended functionality. Each stage and goal met for PIAT will be documented as its own version. Testing will be done in Android Studio as well as through hands-on testing of the fundamentals in the application.

Initially, we will be developing base level Android applications to reach each of the individual functionalities, such as GPS location, MySQL interactions, and PHP querying. Once we have the base functionalities, we will work towards getting them to interface with each other. For our local instance, we will be using a local server which we have set up to host the MySQL database entries. This is a limitation on our final goal of using the Virginia

Tech server to host the MySQL database. Thus, we will be testing smaller samples of data on our local server to make sure that the program capabilities are at the desired level, then we will be moving on to implementing the PIAT application while connected to the Virginia Tech servers.

The final steps, once testing is complete, and the desired capabilities are met, will be to finalize the user interface so that the user experience is intuitive and simple to use. It is also in our best interest to maintain previous versions of the application in case we reach a dead end. To accomplish this, we will be implementing version control for the PIAT application via GitHub. GitHub will allow each group member to have access to the most recent, as well as all previous, versions of the PIAT application. For example, the GitHub project will include the localized version of the application as well as the versions that are compatible with the Virginia Tech server. This way, we can revert or refer to the local instance if we come across any errors when we are attempting to deploy with the Virginia Tech server.

d) Application Technologies (Referencing Table in Section 2b.)

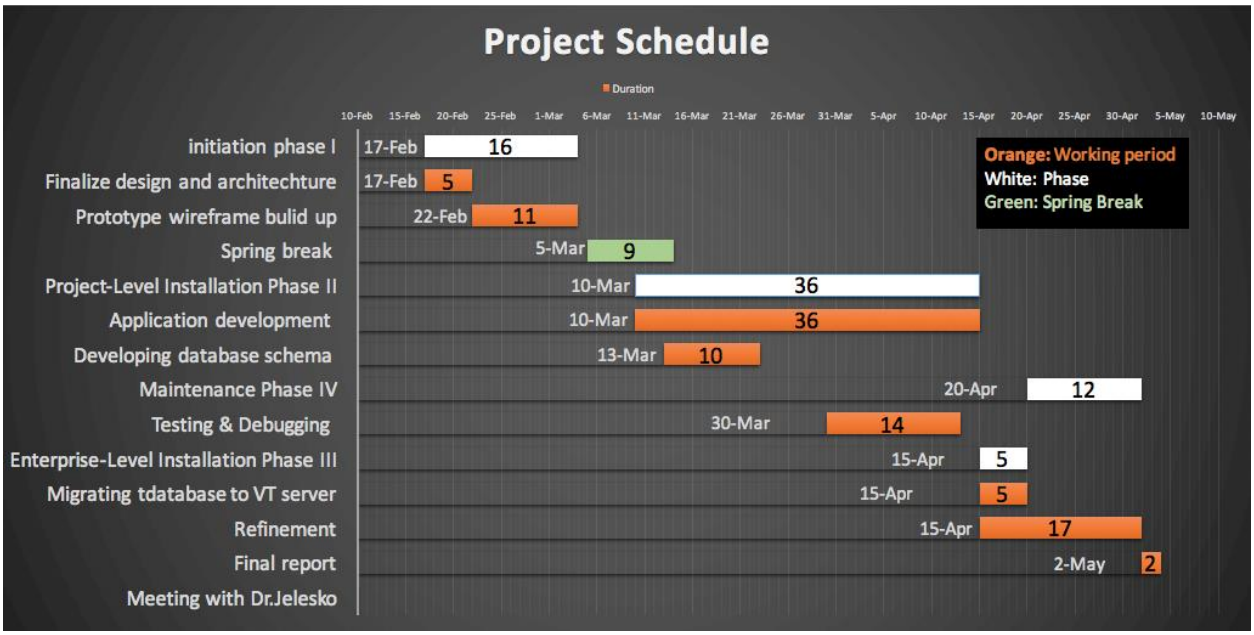
Regarding software, we will be using MySQL; which is a relational database management system. This will be the primary database employed in the PIAT application. The database will need to store a leaf id, leaf type, longitude, latitude, and a timestamp of the data points recorded; these records represent information about the poison ivy samples being observed. To interact with the MySQL database via the Android application, we will be using PHP query files. The use of PHP was a request by Dr. Jelesko as he states the Virginia Tech server the final application will be hosted on uses a combination of PHP and MySQL. This way, we can send a POST request from the mobile application to the server hosting the database and PHP files. The application will request multiple PHP files that will be used for creating a connection to the MySQL database, inserting a record, retrieving the records stored in the database, and deleting a record from the table that may contain an error.

Regarding hardware, we will be implementing this application on Android. Therefore, any potential users will, naturally, need to have an Android mobile device and have the application correctly installed to access the PIAT application. We will also be implementing some external device that will be connected to the phone so that the user can send a request to the application to store the current location without having to open up the application itself. Finally, the MySQL database will need to be hosted somewhere for the application to have reference to it without having to store everything on its own. Therefore, for a local and small-scale implementation, we will be using a local server where we will host our database and PHP files. For the final implementation, we will be running on the Virginia Tech server.

e) Personnel

This project requires three software developers and one hardware developer throughout a 3 ½ month period. The necessary software and hardware expertise for this project implementation include MySQL database management, Java/Android Studio, PHP querying, and auxiliary cord control interactions with an Android device.

f) Proposed Schedule



g) Risks and Contingencies

There are particular challenges that we will encounter that may not be resolved within the given period. These risks will be taken into account, and a backup plan will be used in the worst-case scenario.

Risk 1: External device is not functioning as intended while on trail

Contingency: The application will be fully usable through the PIAT application on the phone screen

Risk 2: Loss of connection between phone and MySQL database

Contingency: Store the data on the phone's SD card as a backup until a proper connection can be made or until the trail is complete.

3) Prototyping

a) User Interface

The user interface has been simplified a great deal to improve flow and functionality (*Figure 2*). In the wireframe implementation, there were extra screens and activities where they did not need to be. In the new and refined prototype, it was found that individual pieces of data could be saved on a button click instead of going on to a confirmation page just for a few pieces of data. Data like GPS coordinates are always saved so clicking any of the buttons that record information should automatically save the coordinates. Also, after meeting with Dr. Jelesko, it was decided that there needs to be an input field where the user inputs what team they are on once, and it is saved as that for the entire collection session. This team ID would play a crucial part in separating bits of data in the database as it allows for a simpler grouping of data.

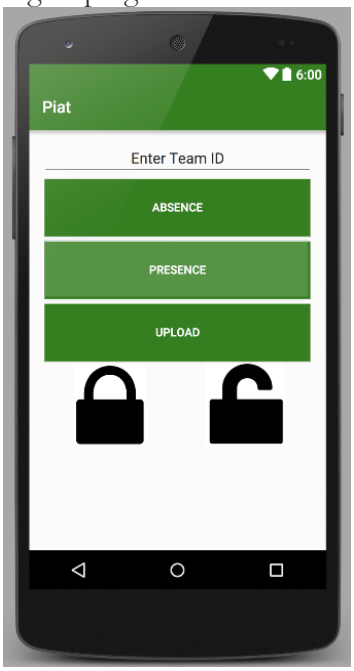


Figure 2: Prototype Home Page

Although the user interface has been simplified a great deal, it still needs to be tested through outside parties. The current plan for after the user interface is determined as final is to find participants to go through our application and ask them to do various tasks. We will gather feedback from these interviews and polish our design so that it is intuitive and easy to follow. Above is the current user interface for our prototype. We have added a lock and unlock button into the prototype user interface so that the application can be active (with the screen on) while in the user's pocket without accidentally pressing any of the buttons. This is because the external device reverts back to controlling the volume instead of interacting with the application when the screen is turned off.

b) GPS Location

Obtaining the GPS coordinates without a network connection is one to the basic functionalities of the PIAT application. To do so, we implemented a Java class called 'GPSTracker' that has a method called getLocation() which returns a Java Location object. With the location object, we can extract the latitude and longitude as doubles and then store them in the MySQL database. The 'GPSTracker' uses two important static variables called 'MIN_DISTANCE_CHANGE_FOR_UPDATES', which is the minimum distance in meters before the application will require a new GPS coordinate to be calculated. Also, 'MIN_TIME_BW_UPDATES' is the minimum time between GPS updates measure in milliseconds.

There is a balance that needs to be meet in regards to how often the PIAT application tries to obtain the GPS location and how power intensive the application will be. We want to maintain a certain level of accuracy; however, the phone's battery will drain too fast if it is constantly trying to ping its location. Therefore, for the prototype the 'MIN_DISTANCE_CHANGE_FOR_UPDATES' was set to 3 (3 meters before the change) and the 'MIN_TIME_BW_UPDATES' was set to 1000 * 60 * 2 milliseconds or 2 minutes. These variables may change during the testing and debugging stage where the application will be undergoing further refinement.

For the application to obtain the GPS location, the correct permissions need to be set. The first permission is the **android.permission.ACCESS_FINE_LOCATION** which allows the app to access precise locations. The second permission is **android.permission.ACCESS_COARSE_LOCATION** which allows the app to access approximate locations. Both GPS and Wi-Fi network locations are considered precise locations. The Wi-Fi network location can get a location in a few milliseconds and extremely accurate down to 1 - 2 meters. However, GPS location is variable and can range from an accuracy of 6 - 12 meters if there are four clear satellite signals. More accurate coordinates down to 1 - 2 meters can be obtained if the phones have Wide Area Augmentation System or WAAS; however, no Android device has this hardware built in. [4]

It all comes down to determining how to configure the GPS requests, which is a matter of trade-offs in accuracy, speed, and battery efficiency. The current prototype's accuracy while using GPS will be within the 6-12 meter range in the clear; however this will not be a quick, instantaneous process. It will take anywhere between 2-10 minutes depending on the current environment that may be obstructing satellite communications. The GPS location service will be battery efficient; this is our primary goal because there will be a lack of charging capabilities on the Appalachian Trail.

These are some pictures of the prototyping for the GPSTracker class. On the left in (*Figure 3*) one can see that that app requires the user to allow the application to access the device's location. This then allows the application to use the permissions mentioned above to provide the required functionality. One the right in (*Figure 3*) the GPSTracker's calculated GPS coordinates with the current latitude and longitude is displayed.

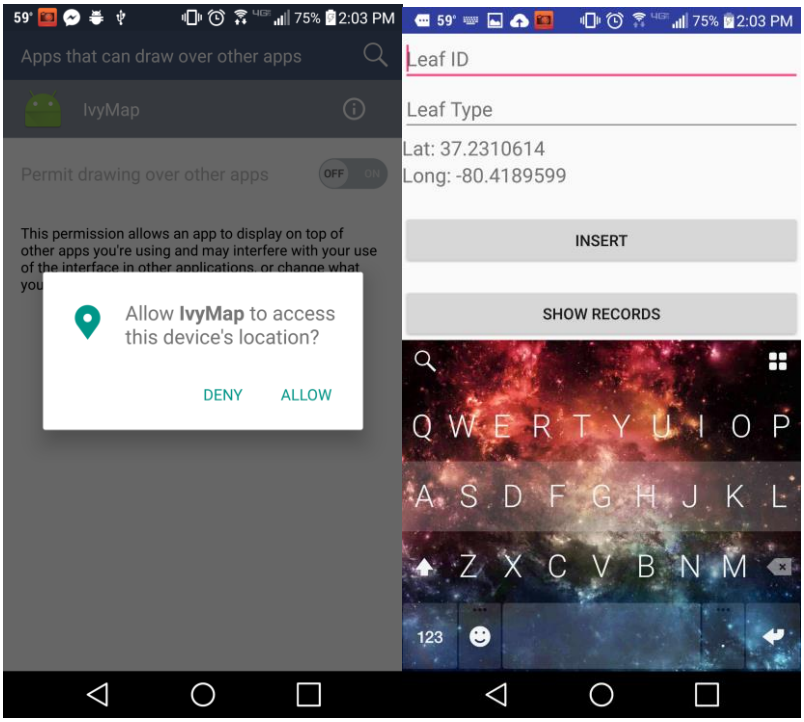


Figure 3: GPS permissions and location

c) External Hardware Interaction Hardware

For our external device, we have elected to use the standard Samsung Galaxy S6 headphones (*Figure 4*). These headphones have a button controller on them that contains functionality for volume up, volume down and start/end call. In Android Studio, there is a *OnkeyDown()* callback function that will start a service based off a pressed button. We use this method to reach the functionality shown in Figure 1. Three corresponding key events should be implemented to get the anticipated response from the key press. Each of these buttons' functions and key events is described as follows. (*Figure 5*)

Figure 4: Samsung Galaxy S6 Headphone.

Physical Key Name	Keyevent Code Variable	Designed Functionality
Volume up button	KEYCODE_VOLUME_UP	Present/Next
Volume down button	KEYCODE_VOLUME_DOWN	Absent
Start/End call button	KEYCODE_HEADSETHOOK	Start/Confirmed

Figure 5: Key Event Relationship

Software:

We used a finite state machine model (*Figure 6*) to represent the state transaction that occurs when the hardware interacts with the software. The initial idle state indicates that none of the buttons are pressed yet. When the user presses the call start button, the state will transfer to "PorA" (Present or Absent) state and ask the user to input whether poison ivy was present or absent. If the user presses volume up button, they will be logging poison ivy presence and will then transfer the state to "Choose." In this state, the user will indicate what type of poison ivy that they are currently logging in the database. The user can use volume up button to rotate through the different kinds of poison ivy and finally press the call start button to confirm the final result. Alternatively, if the user presses the volume down button in the "PorA" state, they just log an absence of poison ivy to their current location. Once a presence or absence has been recorded, the state will return to the initial idle case and await further entries.

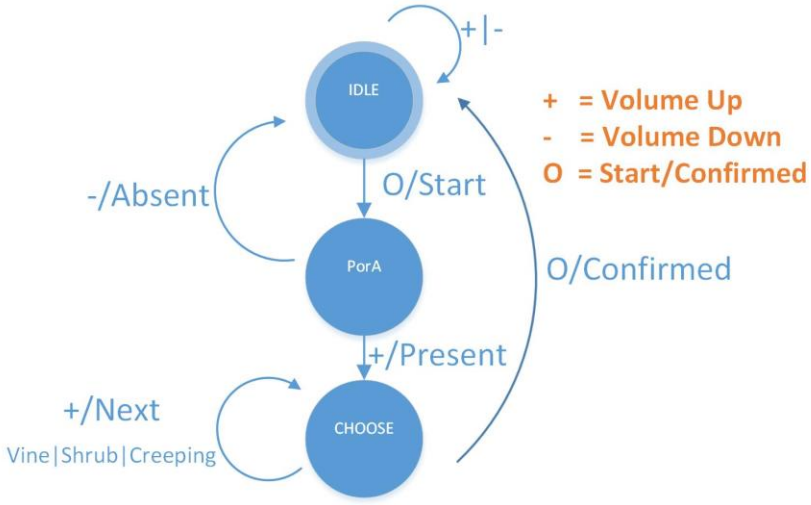


Figure 6: Finite State Model

Additionally, a voice feedback will respond to each state to indicate to the user what type of data they are currently entering. This functionality was accomplished by using the Mediaplayer class in Android Studio and providing text-to-speech audio files that were generated by an online website called http://www.fromtexttospeech.com/.

PIAT Demo Video: https://www.youtube.com/watch?v=tpzp9H9hXF8

d) MySQL and Android Integration

An essential backend services for this application is having a working connection between the Android application and the MySQL database. The best way we found to go about it was calling and passing the recorded parameters to a PHP request script from within Android. The PHP scripts would then interact with the MySQL database on the server side. From here it will at this point either insert a single record or show all the records currently in the database.

The current system uses a library called volley which offers the following benefits [6]:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP cache coherence.
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.

- Ease of customization, for example, for retry and back-off.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

This platform allows us to make an Android POST request to the server with the MySQL database and PHP running on it. Right now for prototyping purposes the MySQL and PHP server is being hosted through www.000webhost.com on the domain vtpiat.netau.net. Here we created the posion_ivy database and uploaded the request PHP files connection.php, insert.php, and show.php. The connection.php establishes a connection to the MySQL database. The insert.php takes the JSON formatted text from the Android POST request and inserts it into the database. Lastly, the show.php sends the Android POST request, and a JSON formatted file is returned with all records currently stored in the database. [5]

The pictures below demonstrate the process for inserting a record into the database. The first picture shows the user entering a 'Leaf ID' and 'Leaf Type' and the second picture displays 'Record Inserted' and clears the text for both fields upon the user clicking the 'INSERT' button. Lastly, the third picture shows the current records stored in the database when the 'SHOW RECORDS' button is clicked. (*Figure 7*)

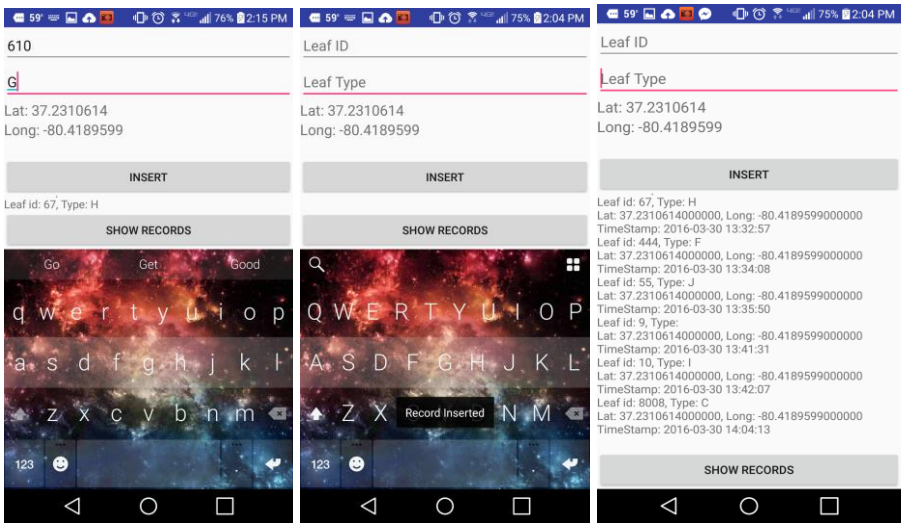


Figure 7: Finite State Model

There is still allot more work that needs to be done on the Android and MySQL connection because right now it can only insert one record at a time. Eventually, we want to save the data record locally and then when the user has an internet connection available they

can upload multiple records all at once. We also want to implement functionality that will allow the user to delete misreported records from the database. Right now we are looking into the Android SQLiteDatabase class for storing the records locally, but do not have it currently working.

4) Refinement

a) Database Updates

Over the course of development for the PIAT application, we have made subtle changes to the layout of the database. The Leaf ID was changed only to be used when a sample is taken and is denoted by the team and the unique tag specified by the uploader. This is done as opposed to its original identifying tag for each leaf which was denoted by the uploader's phone ID concatenated with the record ID number in the database. The Date header was also added into the database so that, if the location cannot be found at the time the poison ivy presence is recorded, Dr. Jelesko and his team will be able to approximate where along the Appalachian Trail the poison ivy record was recorded. These updates were made to tailor the data being stored more towards Dr. Jelesko's desired functionality for the application.

b) Functionality Changes

The PIAT application is moving to a more hardware-focused approach. The user interface will now primarily be used only when the user is collecting a sample. The plan now is to have the white blaze observations recorded through the external hardware linked to the phone via the audio jack, since Dr. Jelesko and his team will be leaving their phones in their pocket or backpack. To make this possible, we will be incorporating multiple functions into the button.

Dr. Jelesko wants the application to function so that the users can specify the type of poison ivy based on the number of button presses or the duration the button is held. This way, the hikers can quickly record data without having to physically interact with the phone screen. The user will also need to receive feedback to assure that the data they log with the external device is the correct data. If it is not, there will also be a functionality incorporated to clear the most recent entry to assure that only accurate and intentional data is being recorded.

c) Schedule Changes

Throughout development, we have had to update our initial schedule to dedicate more time to certain tasks that were more or less than originally estimated in the outlined plan. For instance, after meeting with Dr. Jelesko, we found that the external device was going to need much more functionality than originally imagined. Dr. Jelesko wants the users to be able to log data more in depth when using this external device and, as a result, more

time and functionality needs to be spent on incorporating the hardware to meet the desired requirements.

The project has also been broken up into four phases, Initiation, Project-Level Installation, Enterprise-Level Installation, and Maintenance. These phases are listed on our schedule to provide an overarching structure that shows a more generalized idea of where we are along the application process. We have added in more specific accomplishments to our schedule instead of leaving it to general cases so as to provide more deadlines and keep on track. Additionally, we have had to divide up the labor of work more to get the application up to the desired standards. We have also removed the Finalize Debugging section as we realized what we were shooting for in this time slot was more along the lines of refinement and finishing up the final touches. Furthermore, the base team roles are still the same, but the group is now working more individually to accomplish smaller scale tasks and reconvening with each other to compile those into the PIAT application.

The current schedule can be found earlier on in the report in Section 4f.

d) Future Adaptations

1) Next Steps

The next steps for PIAT will include moving from a prototype to a fully functional application on a local device-specific scale. The local application will include local MySQL Lite storage of recorded data, the user interface, and multiple records uploading to MySQL database, and functionality with the external device. Getting this localized version to work is necessary before moving further in development since the user's entries will be stored locally until a time where they can be uploaded successfully to the server.

Once we have reached the desired functionality on a local scale, we can then move towards migrating the application onto the Virginia Tech servers. Under Dr. Jelesko's recommendation and his advice about the university's servers, we will be using PHP to connect to and manipulate our database on the server. After successfully migrating the application to the servers, we will need to test this application in the field to troubleshoot any issues that Dr. Jelesko and this team may encounter while using PIAT on their hike. Testing the application will take some time as we need to be as thorough as possible, but once we are confident that the application is within working condition we will turn the application, this report, and the user/developer manuals over to Dr. Jelesko.

2) Potential Future Features

Through our meetings with Dr. Jelesko and our inspiration, we have come up with possible features to be implemented after the PIAT application has reached the desired functionality. In our initial meeting with Dr. Jelesko, he stated that he would like to see support for photograph storage in the database, but this is a secondary task. We determined that it would be best first to implement the other functionality points before photographs so as to better deliver a working application

within the timeframe. When we explained the complications behind storing photos directly into the database, Dr. Jelesko agreed that it would be better to hold off on this feature until a later date.

On our own, we came up with some ideas to help improve the application, but for the sake of meeting Dr. Jelesko's requirements, we have elected to postpone these features until we get approval from Dr. Jelesko and satisfy the initial scope of tasks accomplished. We decided that using a timestamp to approximate the location of a poison ivy presence, while sufficient, is not as accurate as if we were to use a pedometer like function to estimate the hikers' course along the trail itself. This way we could come up with a GPS coordinate approximation based on the number of steps the uploader took before recording the presence. Another idea was to create a Smartwatch counterpart to the PIAT application where the Smartwatch would function as the external advice. The common problems that may arise with both of these ideas are that battery life may be strained even more with additional application functionality. However, these ideas still could aid in the accuracy and ease of recording data for the PIAT application.

5) Testing

a) SQL Database Migration & Security

Upon successfully being able to upload records to our MySQL database being hosted on www.000webhost.com we progressed to move the back end of this project to Dr.Jeslko's private server. Once we got the credentials to upload the proper PHP files and access the MySQL database we needed to try an insert records from the PIAT app to the new database. However, there was an unexpected issue that we had come across. When sending the request to the new address at https://oak.ppws.vt.edu/~cs4624 we began receiving "Error 412 - Precondition Failed" errors as a response back from Dr. Jelesko's server.

This problem is persisting and we have contacted the server admin Andrew Mike about the situation. In the meantime we are beginning to think the problem could be with the SSL, so to try and circumvent this for now we tried implementing a certificate handler on the PIAT app that would accept all server certificates. However, this was unsuccessful, we are still looking for the issue that continues to persist. This problem will be solved but we need to meet with the system administrator first so we can see exactly what conditions our HTTP Post request is not meeting

b) External Hardware

After successfully implementing the hardware interface functionality in Android Studio, we tested it with the MySQL database insertion feature. At first, we migrated the insertion code to the "type confirmed" state. Due to this migration, some of the old code was not properly adjusted, so that when other developers on our team began working on it small glitches and/or bugs would occur. This malfunction was not a serious issue and was

quickly cleaned up during our team meeting. We thoroughly documented the code in this section for future developers, so that they do not run into the same confusion.

Early on, we had issues getting the hardware functionality to work properly when the screen was turned off. However, having the screen turned off while the application was running would save some energy when users record their data using only the external hardware. We found out we have to implement a service class to accomplish this functionality. We did not include this part of design initially; therefore, we came up with idea that the user could dim the screen with a long press and turn it back on with a flick/sliding gesture. We tested this feature, and it worked fine, later we will add a lock button to make sure user will not accidentally touch the screen while it is in the pocket.

Additionally, we tested all the combinations of the input key options to assure that the results are valid as well as the feedback sounds. During these tests, it was found that after thirty to forty times of clicking the headphone buttons, the application would crash. Furthermore, an error would arise when pressing the back and delete button on the screen that would also cause it to crash immediately. The problem found was with the *onKeyPressed()* callback function, the function should return *super.onKeyPressed()* instead of directly returning *onKeyPressed(),* which results in a recursive function call that causes a crash as a result.

Another bug was about the mediaplayer class instantiation. It should be constructed in each state transaction instead of directly under the callback function. Instantiated a lot of raw sound file directly under the *onKeyPressed()* function will cause the sound delay and occasionally crashes the app possibly because of heap size overflow.

c) GPS Tracker

Once the front end was successfully connected to the back end, it was important to test our application in its actual environment. We decided to go up to the McAfee's Knob trail since it is located along the Appalachian Trail. We selected a variety of combinations for inputs along parts of the trail and tried to sync them at different connection levels as well as different altitudes.

One of the main errors that was encountered was connectivity. When the phone had poor service, the user interface and basic operations worked perfectly, however, the GPS coordinates were consistently stuck at one value for extended periods of time. There is not much that can be done about this because the application depends on the GPSTracker class to connect to the phone's service provider. For the most part, service was not a big issue. There was good enough connectivity during the majority of the test run so that accurate test results were acquired. After collecting the data, the GPS coordinates were looked up on Google Maps to see if they collected the accurate spots along the trail and the majority of the points were spot on.

Aside from GPS coordinates, information was accurately saved and loaded into the local table. The only other error encountered was when we attempted to sync the local table with the remote database. Just like the GPS coordinates, when there was quality service such

as LTE and even lower parts of 3G, the sync was successful and completed in a timely manner. When the phone was roaming, syncing took much longer.

This first test run on the Appalachian Trail was a smaller scale run to see if the application would crash prematurely. We intend to go on either a full McAfee's Knob trip or even longer to see if the application can take the stress in the actual field where it would be used. Aside from service provider connectivity, there are not many other situations which can cause problems for Dr. Jelesko and his team. While conducting these field tests, the battery on the phone did not drop too quickly and was sparingly used thanks to the local table. Also, the application was able to be closed whether it was killed or the home button was killed and it would still have the local table saved.

d) External SD Card Migration

Being able to back up the local database on the phone to an SD card was a feature that would put Dr. Jelesko and his team at ease if a phone got damaged at any point along the trail. This was something that we were able to implement and get to working on a virtual machine running a rooted Android 5.0 operating system; however, was not successful on a phone running Android 6.0. This is because, in Android 6.0, third-party apps don't have access to the sdcard_r and sdcard_rw GIDs. Instead, access is controlled by mounting only the appropriate runtime view in place for that app. Cross-user interactions are blocked using the everybody GID. [10]

The next step would be to test this on unrooted phones running a different version of Android other than 6.0. Then we can begin further testing to check and see if the local SQLite database was exported correctly. This has be done using Android Studio's built-in Android Device Monitor that allows a user to view the external file storage to confirm the successful record transfer to the external SD card.

There is an alternative solution to these problems that could be taken care of on the user's side of the application. All the user would have to do is open up the "Settings > Application Manager > PIAT" path to access further options. From here the user can click "Move to SD card", which would then store the application and all of its data files (including the SQLite database) on the SD card. This may be a naive way to going about and solving the issue, but it is the best solution we can provide right now because of Android 6.0 and the lack of physical unrooted phones we have to test with.

e.) Results of Testing

We have been testing our application over the course of development to assure that everything works as intended. During prototype development, we were focused on testing the features on a smaller scale and an individual basis, so that we could make sure that the individual features were working properly. After the prototype was successfully implemented, we began testing all of these components and their interactions with one another to check for efficiency and edge cases; such as with connectivity issues, stress testing and more. All the different parts needed to be sufficiently tested, including: the database,

server, GPS functionality, hardware, and the mobile application itself. We broke down testing as much as possible in order to cover as many cases as we could think of.

From testing the system directly, we were able to conclude several things. The first and most noticeable is that the overall user interface is responding as expected. From preliminary run-throughs of the applications, the back end functionality appears to be working in order as well. The local and server-hosted databases were responsive and logging entries and the syncing between the two was operating correctly. In the case that the user lost connection during a sync, the application would respond and provide feedback to the user; thereby letting them know that the application stopped syncing entries and they should try again once connection was restored. However, through field testing on a local trail, we found that the application would still sync, albeit slightly slower than normal, when the connection was minimal.

To stress test the application, we used Android Developer's UI/Application Exerciser Monkey [11]. The Monkey uses a command-line to allow testers to input the desired quantity of events to the application on any emulator or Android device. As such, our tests involved entering large quantities of data to the local database and, in turn, syncing these entries to the server, in order to push the application to its limits. Through our stress tests, we found that the application works as expected. This means that, even if the user has hundreds of records in their local database, syncing should still be possible without causing any issue. That being said, we still highly recommend that the user sync records as often as possible to avoid losing connection during syncing. On the same note, when syncing entries to the Virginia Tech hosted server, the user should stay in the same location.

6) Tutorials

a) Setting up Server PHP files

Make sure that the database has been initialized first and then you can configure the server to receive the HTTP POST request from the PIAT application follow these steps.

1) On the server you are trying to configure access the **public_html** folder and create a new folder called **post.**
2) In the post folder you will copy and paste the following PHP files into the post directory just created.
    a) connection.php

```php
<?php
define ('hostname', 'mysql7.000webhost.com');
define('databaseName', 'a1258463_poison');

$username = $_POST["username"];
$password = $_POST["password"];

$connect = mysqli_connect(hostname, $username, $password, databaseName);
?>
```

This file will have to be edited to work on the server it is being migrated too. The following must be changed:

define('hostname', 'MySQL Server URL')
define(databaseName, 'MySQL database name')

b) insert.php

```php
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
    require "connection.php";
    createPI();
}

function createPI()
{
    global $connect;

    $team = $_POST["team"];
    $plant_id = $_POST["plant_id"];
    $plant_type = $_POST["plant_type"];
    $latitude = $_POST["latitude"];
    $longitude = $_POST["longitude"];
    $date_time = $_POST["date_time"];

    $query = " Insert into poison_ivy(team, plant_id, plant_type,
        latitude, longitude, date_time) values ('$team', '$plant_id', '$
        plant_type', '$latitude', '$longitude', '$date_time');";

    mysqli_query($connect, $query) or die (mysqli_error($connect));
    mysqli_close($connect);
}
?>
```

This file contains the record elements to be retrieved from the HTTP Post request and the MySQL command to insert that record into the database. This file would change if there is a new column in the database. For instance if you wanted to include a 'color' in the database the following changes will be added.

$color = $_POST["color"]

$query = " Insert into poison_ivy(team, plant_id, plant_type, latitude, longitude, date_time, color) values ('$team', '$plant_id', '$plant_type', '$latitude', '$longitude', '$date_time', '$color');""

3) Once the PHP files are setup in accordance with the MySQL database structure and configuration the application will be able to remotely sync the records.

Acknowledgements

Dr. John G. Jelesko

Office Phone: (540) 231-3728
Lab Phone: (540) 231-5899
E-mail: jelesko@vt.edu
Office Address:
548 Latham Hall
Blacksburg, VA 24061
United States

Bibliography

1) “Poison Ivy Citizen Scientist Project 2014." Poison Ivy Citizen Scientist Project 2014. Web. 17 Feb. 2016.
2) Piltchm Avram. “How to Move Android Apps to an SD Card.” Tom’s Guide. N.p., 25 Aug. 2014.
3) "Android PHP/MYSQL Tutorial." w*ww.tutorialspoint.com*. Web. 19 Feb. 2016.
4) "Android Location Tradeoff between Accuracy and Battery Consumption." *Gps*. Web. 19 Feb. 2016.
5) "Location Strategies." *Location Strategies*. Web. 28 Mar. 2016.
6) Miskoajkula94. "Insert Data Into a Database Using Android, PHP, MySQL, JSON Part 1/2." *YouTube*. YouTube, 2015. Web. 28 Mar. 2016.
7) "Transmitting Network Data Using Volley." *Transmitting Network Data Using Volley*. Web. 28 Mar. 2016.
8) "Android PHP/MYSQL Tutorial." *Www.tutorialspoint.com*. Web. 28 Mar. 2016.
9) "Saving Data in SQL Databases." *Saving Data in SQL Databases*. Web. 28 Mar. 2016.
10) "Chapter 16 High Availability and Scalability." MySQL. N.p., n.d. Web. 30 Mar. 2016.
11) "Storage." Storage. N.p., n.d. Web. 14 Apr. 2016. <https://source.android.com/devices/storage/index.html>.
12) "UI/Application Exerciser Monkey." *Android Developers*. N.p., n.d. Web. 13 Apr. 2016. <http://developer.android.com/tools/help/monkey.html>.