

CS 480 Final Project

A BARCODE SCANNING APPLICATION

ZACHARY STUCK

Introduction/Overview	p. 2
Technologies	p. 3
Organization	p. 6
Retrospective	p. 7
Conclusion	p. 8

Introduction

As a prelude to my project, I will first formally introduce myself. My name is Zachary Stuck, and I am a Mobile Web & App Development major with a Math minor. I chose to come to NMU to after taking a tour of campus with Professor Appleton four years ago. It was a cold February and there was still snow and sand everywhere, but the visit was good. Most people told me I was crazy for going so far north into the wilderness to learn about computers, but I knew it would be the right fit for me. My reason for choosing Mobile Web & App Development, or MWeb for short, was specifically due to my interest in mobile technology, and how prevalent it had started becoming in modern society. I had always wondered what it took to create mobile counterparts to major applications, such as Facebook. So, I pursued that interest as my college career, and I couldn't be happier with where it lead me.

Overview

With that out of the way, I can introduce my application. For my project, I wished to create an application that could scan barcodes and associate various information with that barcode. This way, barcodes could be applied to various objects such as personal boxes, totes, or anything you could think of. Then, when you want to see when or where a specific thing has been, you can look up the history of that particular object. The easiest comparison that can be drawn is the same sort of technology that allows customers to track packages through USPS, FedEx, or UPS. With that in mind, I designed the project with the intent of use in a business setting. However, it could be easily adapted for personal use, such as keeping track of stuff that may be in different locations, such as storage units.

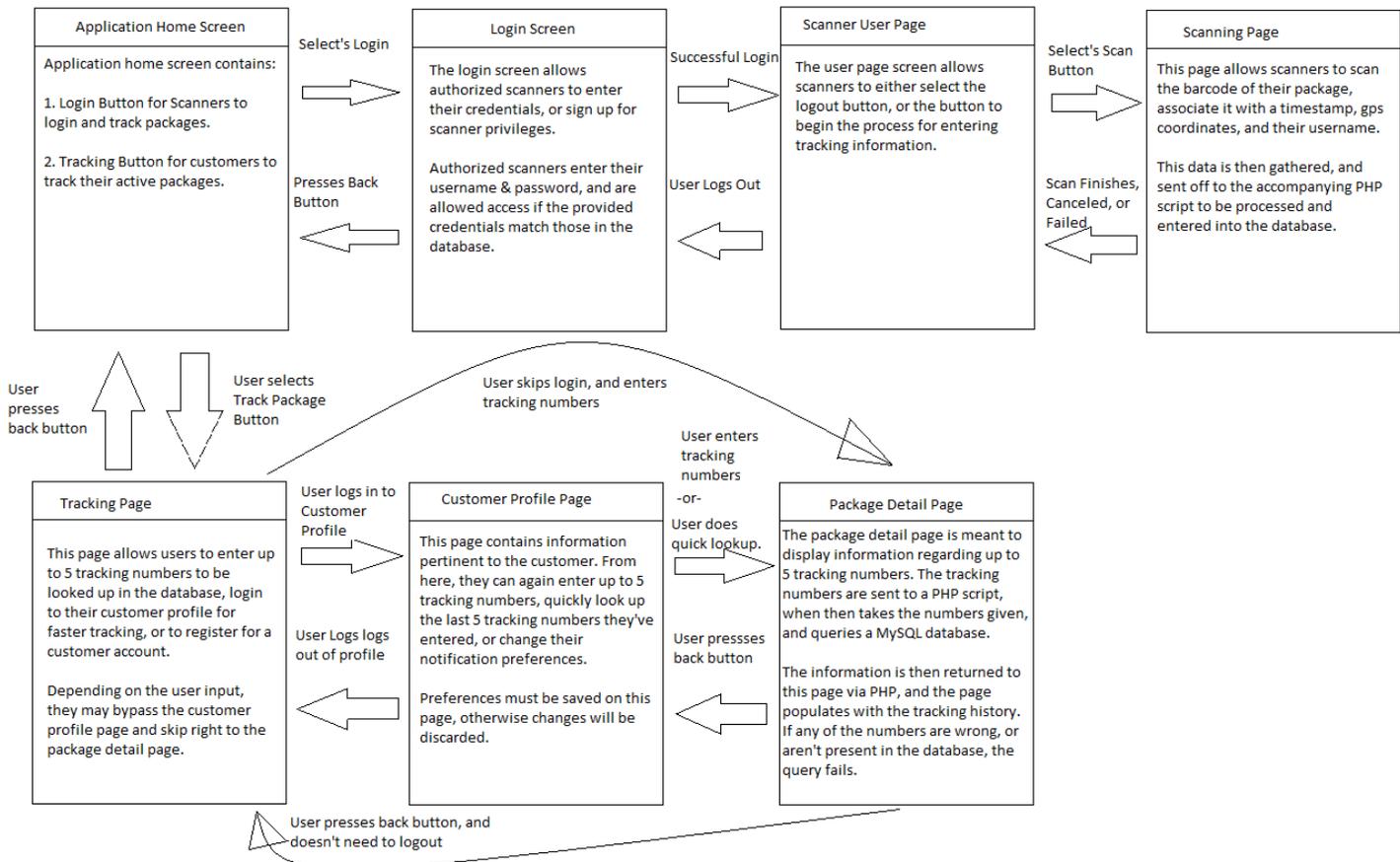
My main motivation for this project was to gain the experience of what goes into developing an application that utilizes a database from start to finish. I had worked on small applications during school involving the iOS and Android platform, but it left much to be desired about the development process as a whole. At the same time, I had also gained quite a bit of knowledge in other aspects of Computer Science, like basic website programming and client to

server communication. With that in mind, I set out to create an application implementing features and technologies about the mobile platform that interested me.

Technologies

As for how it functions, my project is an application designed for use on Android Devices, mobile phones specifically. To store the data, I send the information gathered on the mobile device to PHP scripts hosted on my Euclid account. These PHP scripts then take the information, and insert it into a MySQL table, also hosted on Euclid. To retrieve the data, users can enter the numbers of the barcodes that they wish to know about, that data gets sent to different PHP scripts on Euclid, the scripts poll the MySQL database, and return the information found. As a supplementary way to view the data, a simple website can also be visited that provides the same functionality of looking up barcode history. A diagram of the app's usage can be seen below.

Current Application Flow



To comment on the diagram shown on the previous page, the program can seem a bit confusing when referring to GUI elements present in the application. Simply put, the buttons are tied to fire off functions when clicked that transition the application to the next state. These functions can also be tied to a physical back button if the phone supports it. The application utilizes activities in the Android environment, which act similar to web pages. Each one loads independently of the other, and that means transitioning between the two means also passing along data.

For information to be retained across these different views, the application makes use of Intents. Before transitioning to a screen that needs data from the previous, I store the data into a specific Intent, using a string I select as the key, and the information I need as the value. This Intent also targets the next activity that will be called, which means I can retrieve the data once in that new activity. All I have to do at that point is request the information by providing the key necessary. If one were to transition to a different screen than what is associated in the Intent, the information would no longer exist. An example of this intent can be seen below.

```
scanButton.setOnClickListener((view) → {
    Intent scanIntent = new Intent( packageContext: UserPageActivity.this, ScanActivity.class);
    scanIntent.putExtra( name: "userKey", userfield.getText());
    startActivity(scanIntent);
});

try {
    username = extras.getString( key: "userKey");
    userSlot.setText( username );
}
catch (NullPointerException npe){
    userSlot.setText( "Error" );
}
```

In the top image, I am transitioning from the user page of an authorized scanner to the page in which barcodes can be scanned. As I need the username on the next screen, I store that information into the Intent that is targeting the next activity, ScanActivity. Once I am inside the ScanActivity, I attempt to retrieve the information by using the same key. As this information could potentially be null, it must be wrapped within a try/catch block. Multiple views in the project utilize this feature, with the largest one being the transition between entering tracking numbers and viewing the tracking history.

The scanning part of the application is provided by utilizing the ZXing library. The source for this library can be found at this URL: <https://github.com/zxing/zxing>. This library allows for many ways to implement a barcode scanner with multiple format support. The implementation that I chose to go with is the integrated variation, allowing users to scan barcodes without leaving my application. Snippets of code and a brief description of their interaction can be seen on the following page.

```
scanPkgButton.setOnClickListener((view) → {
    //IntentIntegrator derives from ZXing integrated library
    new IntentIntegrator(activity: ScanActivity.this).initiateScan();
});

//Handles the results returned from starting the scanner activity
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    if(result != null) {
        if(result.getContents() == null) {
            Toast.makeText(context: ScanActivity.this, text: "Cancelled", Toast.LENGTH_LONG).show();
        } else {
            pkgSlot.setText(result.getContents());
        }
    } else {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Once the ZXing library is included in the build, the project can access request access to the camera features of a phone, and launch a new activity using the camera. The line that initiates this process is located in the topmost image, where it begins the scan. This line closely resembles the previous images regarding Intents. In the second snippet, this is where the program checks the end result of the scan. If it managed to scan a barcode, it will be able to access that information from within the result variable, and then manipulate it. If the scan was canceled, a pop-up is called on the current view, and the user can attempt to scan again.

Organization

As this project is mainly focused on the mobile aspect, a great majority of it is within the application itself. However, there are still portions outside of the application that are just as necessary as the app itself. Without the PHP scripts, it would be rather difficult for the application to send data quickly to the MySQL database. In addition, the way in which the app would communicate would be concrete, and would require modifying that particular code in the app itself. By keeping that portion outside of the application, I can guarantee that the bandwidth needed is minimal, as it is simply text data being transmitted. To go further down that path, I could have implemented a web page that would have further reduced the amount of code required for communication between the application and server. If I were to do this part of the project again, I probably would consider doing it this way, since I had already planned on implementing a web page to do this from a computer. However, I know from personal experience that transitioning from an app to a web page can be slow, and can feel tacky. In that regard, I'm happier with utilizing PHP scripts.

Once the information is sent to the PHP scripts, all that is necessary is for the computer running those scripts to finish processing and send a return message of either a success, failure, or information that was requested. The primary advantage of keeping the PHP separate is the modularity that affords me. This way, every source that would be sending information to and receiving information from the server is a single path through the PHP scripts. With the scripts all available in one place, I can guarantee the information needed for the scripts stays consistent, the messages going to the MySQL are the same, and updating the script doesn't require updating the applications, unless the data necessary for the scripts change.

In terms of code organization, I've been utilizing GitHub as my main form of source control. While the project may not have been quite big enough to fully utilize all that source control offered, being able to easily pull the code base back down again after a bad code spike was a godsend. I was able to implement two branches, master and development, with a majority of the coding happening within the dev branch. Whenever I achieved a major milestone, I would switch to master, and pull in new changes. This allowed for me to see just

how much progress I was achieving, and providing a sense of satisfaction and direction to my project. While there were days that I didn't accomplish as much as I would have liked, there were others where I achieved far more than expected in a given timeframe. On that, I can discuss in greater detail in my section on retrospection.

Retrospection

Overall, I thoroughly enjoyed working on this project. I was able to learn more about the app development process, and how technologies can integrate and become something much more. Even some of the features in the application that can be deemed "simple" can require a lot of careful thought and planning in order to not be exploitable or weak. When designed right, though, applications can be extremely simple and incredibly useful, providing very easy access to tools that may have only been available on desktops or laptop computers. When most people think of building an app for smartphones or tablets, they assume it is in reference to making the next cash grab mobile game or bloatware. However, when utilized for business or practical purposes, handheld technology is definitely something that is way more involved than just trying to earn some quick cash through in-app advertisements and micro-transactions.

Thinking back on the project as a whole, there are definitely parts that I would have done differently, or not at all. When first designing the project, I focused more on the application itself, and not the back-end which would provide the majority of the functionality to the app. If I had focused on that front first, I would've been able to have a better idea of what to expect from the application, and what the application could expect to be sent back from the server. Many of the adjustments and modifications in order to expand functionality or correct errors involved me having to alter multiple files, when it felt that simple planning could've avoided those types of issues.

The other big issue that has plagued this project is properly debugging PHP code. For any query involving the MySQL database, I was easily able to test syntax to ensure proper operations. Unfortunately, with no way to send the data from a computer to view the errors in a web browser, I had to debug error messages being returned from PHP script to the android

device. In addition, without a standalone Android device, making sure the scanner library was properly implemented would have been a nightmare.

Early on in development, I was able to easily program and debug the application from within Android Studio using their emulator. Upon reaching the stage where integrating the scanner library was necessary, I realized that debugging that part of the application would be harder than anticipated. The main predicament was how could I test the camera feature of an application, when the device running the application was being emulated? I looked into using a webcam as an external camera for the emulator, but that seemed too difficult to figure out in the given timeframe. Fortunately, my daily phone is a Samsung Galaxy S7 running on Android, and setting it up for debugging purposes went relatively quick. I can attest that this project would not have run nearly as smooth without a physical android device to debug and test with, and would highly recommend picking one up if you were to program for Android devices at all.

Conclusion

To summarize my project, I believe I was able to successfully build the application that I set out to create, and even include features I hadn't thought about prior to starting the project. I regret not being able to work on it nearly as much as I had hoped to, but the lessons learned from it will definitely stick with me for a long time. Being able to stay motivated while working through difficult portions of the project, such as getting communication between the application and PHP, was a big part of this project.

I feel that it was about as difficult as I expected it to be, with how many different technologies I was coordinating together. Learning how to properly gauge and estimate complexity is a task that I feel I will never perfect, but should strive to become more accurate in all the same. That lesson, along with the experience of combining multiple class's knowledge, are two of the biggest takeaways from this project. With that in mind, I will now shift focus to my grading scale.

Proposed Features:

1. Create an application capable of scanning barcodes 20pts.
 - a. Users can search for tracking info, or log in to gain scanner privileges 5pts.
 - b. Scanner can properly read barcode information 5pts.
 - c. User can successfully enter data manually 5pts.
 - d. Using sessions, users can quickly resume activity within the application 5pts.
2. Upload barcode and location data to remote server from app 25pts.
 - a. App can store information and upload when a connection is available 10pts.
 - b. Data can be queued to upload while not interrupting additional scans 5pts.
 - c. Data is submitted in unique key style to prevent conflicts 10pts.
3. Create a mysql server capable of handling and storing data 5pts.
4. Using PHP, process data to be stored in the mysql server 25pts.
 - a. Check for errors before modifying table 10pts.
 - i. Data uploaded later should have time of scan, not upload.
 - ii. Duplicate uploads should not overwrite the original
 - b. Parse uploaded tracking info for proper storage 5pts.
 - c. Handle errors and exceptions accordingly, without interrupting app 10pts.
5. Create a website to view tracking data from server 25pts.
 - a. Allow tracking number lookup to show progress of a package 15pts.
 - b. Users can look up multiple tracking numbers at the same time 5pts.
 - c. Allow entry and editing of tracking info via admin login for website 5pts.
6. Allow users to lookup tracking information from app 10pts.
7. Users can create a profile to save tracking information under 10pts.
8. Users with a profile can receive push notifications about tracked packages 10pts.
9. Application is compatible across multiple versions of android 5pts.
10. Create a companion iPhone app 25pts.

Grading Scale (Maximum of 160 pts.)

+140 pts.	= A
139 – 125 pts.	= B
124 – 110 pts.	= C
< 109 pts.	= D

In terms of grading scale and point distribution, I feel that this is where my senior project was the weakest. Parts of the scale make sense, while others can seem confusing. For example, 2.b is worth 5 points for being able to queue data for upload without additional scans. Due to the size of information being sent to and from the server, scans will never have to worry about queueing to be uploaded. Similarly, with how android apps are suspended, 1.d doesn't seem nearly as useful. In the same vein, push notifications are easily worth more than 10

points, due to the fact that I had not foresaw the difficulty implementing REST API can be with no prior experience, and could easily have been worth 15 or even 20 points. I was also able to complete additional features, like implementing e-mail notifications as well. With this grading scale, I am looking at about 115 points, which would be a mid C per my grading scale. That grade is definitely not one that I would be proud of, because I do not feel the letter grade reflects the work that went into the project. This dissonance, I believe, stems from how my grading scale was designed.

If I was to redo this grading scale, I would consider dropping the iPhone objective completely, as it a bit unclear just how much is needed to achieve that objective. In addition, the amount of allocated points for that particular objective overshadows major parts of the application that I would deem too necessary to ignore. While it was a good stretch goal to keep in mind, with the allotted time that I had this semester, the rest of the project seemed far more important. Without the iPhone objective in my allotted total, my 115 of 160 points would instead be out of 135, and would be a mid B per my grading scale. This letter grade, while still not the A that I had hoped for, is far more acceptable for the work that I completed.