SPCA Mobile App

by

Connor O'Leary, Randy Buka, Alex Bitter

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Connor O'Leary	04/15/2019
Randall Buka	04/15/2019
Alex Bitter	04/15/2019
Abdou Fall	04/15/2019

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

Table of Contents

Section *Page*

Abstract *1*

Introduction *2-3*

Problem **2**

Solution **2-3**

Project Goals/Methodology **3**

Overview **3**

Discussion/Details *4-12*

Project Concept **4**

Design Objectives **4-5**

Methodology **5-6**

User Profile **6-7**

Use Case Diagram **8**

Technical Architecture **9-10**

Testing **10-17**

Budget **17-18**

GANTT Chart **18-19**

Application Screenshots *20*

Problems Encountered *21*

Future Plans/Recommendations *21-22*

Conclusion *23-24*

References ..... *a*

Example Code ..... *b-c*

Expo Poster ..... *d*

List of Illustrations

Tables

Item	Name	Page
Table 1	Testing Reports	17
Table 2	Budget Breakdown	18

Figures

Item	Name	Page
Figure 1	User Profile Form	6-7
Figure 2	Use Case Profile	8
Figure 3	Technical Architecture	9
Figure 4	GANTT Chart	19
Figure 5	Home Screen	13
Figure 6	Animal Listing Screen	13
Figure 7	About Screen	13
Figure 8	Lambda Code	b
Figure 9	React Native Code	b
Figure 10	Expo Poster	c

Abstract

85% of dogs in the local Society for the Prevention of Cruelty to Animals (SPCA) shelters are pit bulls or a pit bull mix and at any given time, all dog cages are filled with 2-3 dogs per cage. 100-150 animals, including cats and exotic pets, are adopted from these shelters weekly. We chose to do this project because one of us is a regular volunteer for the shelters as a dog walker. He sees many animals that don't deserve to be locked up in cages and wanted to help adoptees find the perfect animal for them. The SPCA Mobile App, built with React Native and Java as its core technologies, is the solution to this problem. This app lists all adoptable pets from the convenience of a smart phone along with animal care information. The Find-A-Friend Score will assist in matching adoptees with an adoptable pet using an algorithm based on key animal traits. Animal Socialization Calendar and a dog distraction system will give volunteers at the shelters an easier time taking care of the animals. This app's goal is to empty out all the cages and ensure that no animal is deprived of a loving home.

Introduction

Problem:

The local SPCA (Society for the Prevention of Cruelty to Animals) shelters (Sharonville and Northside) have very archaic systems to track workflow. Animal information is printed out on paper and given to shelter visitors and tracking what dogs have been walked recently is done all on a whiteboard (which could get easily erased). Clocking in and out for volunteers involves clicking numbers one at a time (not possible to type) and is done through a very old system. Since SPCA are classified as no-kill shelters, animals can remain in the shelters for months or even years without being adopted. For dog socialization volunteers, it is tough getting a single dog out of a cage to go for a walk (2-3 dogs are in each cage) since all the dogs in the cage are excited and wanting to get out.

Solution:

An app that can be downloaded on a phone. From a visitor perspective, this app would hold all information about an animal that a visitor would find on the current paper prints. It will also have personality traits of the different animals (which can be gathered by volunteers) and a quiz that visitors can take to see which animal they would match up best with. From a volunteer perspective, this could help track what dogs have been walked recently, which can replace the whiteboard. It will also let volunteers clock in and out for volunteer hours. We will also use the app to connect to speakers and play

sounds to distract the dogs in the cage, making it easier to take one out of the cage at a time.

Project Goals/Methodology:

The goal of this project is to make the adoption process easier for the SPCA volunteers and the people that come in looking to adopt an animal. Connor got the idea by working as a volunteer and noticing several things that were outdated, such as paper handouts for dog descriptions and a whiteboard that tracks animal socialization. We chose, through group brainstorming, to use React Native as our front end, Java as our backend, and MySQL as a database, along with AWS cloud technology to host an API. React Native was chosen because it can be deployed to both Android and iOS phones, which was a requirement. AWS Cloud Technology was chosen because the free tier of AWS is more than enough to host all of the data that the SPCA would need. There are 2 additional systems that the SPCA board requested that we use that will be incorporated at a future date.

Overview:

The remainder of this report contains details about methodology, budget, design objectives, timeline, problems encountered and future objectives.

Discussion/Details

Project Concept:

As stated previously, the concept for this idea came from Connor, who works at the SPCA as a volunteer. He noticed several archaic processes that slowed the other volunteers down and saw the senior design project as an opportunity to build something to help the SPCA. This is where the idea for an app came from. The app works for both Android and iOS devices and features a pet listing for all adoptable pets. The big feature for this app is the Find-A-Friend score, where a user takes a quiz and is matched up with the best possible options for adoption. There will be alerts for new available pets and other small features for typical users. Volunteers will have extra features, including access to speaker controls for dog distraction and an electronic socialization calendar to help track pet socialization easier. Admin volunteers will be able to customize alerts and edit specific parts of the app.

Design Objectives:

Project goals include:

- A mobile app that works on Android and iOS
- A match score to pair users with animals
- A pet listing for all adoptable pets
- Easy to use interface that is quick and attractive

Stretch goals include:

- Integrating external SPCA systems to the app
- Several volunteer only features

The biggest goal we could not include was a volunteer hour tracker, which was at the request of the SPCA board. In the initial meeting, several features were offered to the SPCA, but the goal was simplicity for them. The stretch goals we have will most likely not be included, but after graduation, the plan is to continue to work on this app.

Methodology

React native was chosen as the front end for a few reasons. The first was that it is a backend agnostic technology, so we are free to use Java, which all three group members know. Second, a group member had previous experience with Xamarin and was not a fan of it, so we stayed away from that, which is also cross platform. Third, it is similar to Ember.js, which our front-end programmer knows very well, so it is an easy transition.

Java was chosen as the backend because, as stated above, everyone knew it. The same can be said for choosing MySQL. All of these technologies are open-source and free to use. React also works off of components, making repeat objects very easy to integrate into any page.

For designing the app, Picular was used to help select a color scheme that would appeal to users and keep the signature SPCA red color. The app is centered around the Find-A-Friend score and pet listings, so making these appealing and functional was the top priority. Any extra everyday user features were built around those two things. Volunteer features came from an initial list of things that Connor thought would be useful. In the initial meeting with the board, each feature was brought up and given an OK or a no with some additional features being suggested until a final list was confirmed.

Project goals were also discussed in the board meeting and a final list was also confirmed for that.

User Profile:

Figure 1 below contains the User Profile Form. In total, there are 3 different types of users that could use this application: standard users, volunteer users, and volunteer administrator users. Standard users will only have access to the pet listing, pet information, and Find-A-Friend features of the app. Volunteer users will have access to the above plus some volunteer specific features, like the socialization calendar. Volunteer admins will have access to all of the above plus tools to customize alerts and edit the socialization calendar.

User Profile Form
PROJECT: SPCA Mobile App
POTENTIAL USERS: - Standard Users: anyone who can download the app - Volunteer Users: Standard users who have been identified as volunteers - Volunteer Admin: Select management of the SPCA
SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: This project will be aimed towards anyone with a smartphone who wishes to get more information about the SPCA shelter animals. Volunteers can use the app to coordinate care efforts and volunteer admins can track and help coordinate regular volunteer work.

EXPERIENCE WITH SIMILAR APPLICATIONS:

- No prior experience needed (excluding smartphone experience)

TASK EXPERIENCE:

- Signing up for an online account

FREQUENCY OF USE:

- Standard Users: However often they visit the shelter, or at home when looking for themselves or a friend
- Volunteers: anytime they volunteer at the shelter
- Volunteer Admin: anytime they are needed for managerial tasks

KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS:

- Fast-loading pet listing with pet detail pages
- Grid calendar for socialization
- Alert editing for admins

Figure 1: User Profile Form

Use Case Diagram:

Figure 2 illustrates the Use Case Diagram for the SPCA Mobile app. Basic functionality will be given to standard users/users that do not log in. The main purpose for basic users is to view animals and take the Find-A-Friend quiz, which will lead to animal adoptions. Volunteer users will get access to more features that help them in day to day tasks along with the basic features. Volunteer admins will get access to all volunteer features, plus the ability to edit certain features of the app like the announcements.

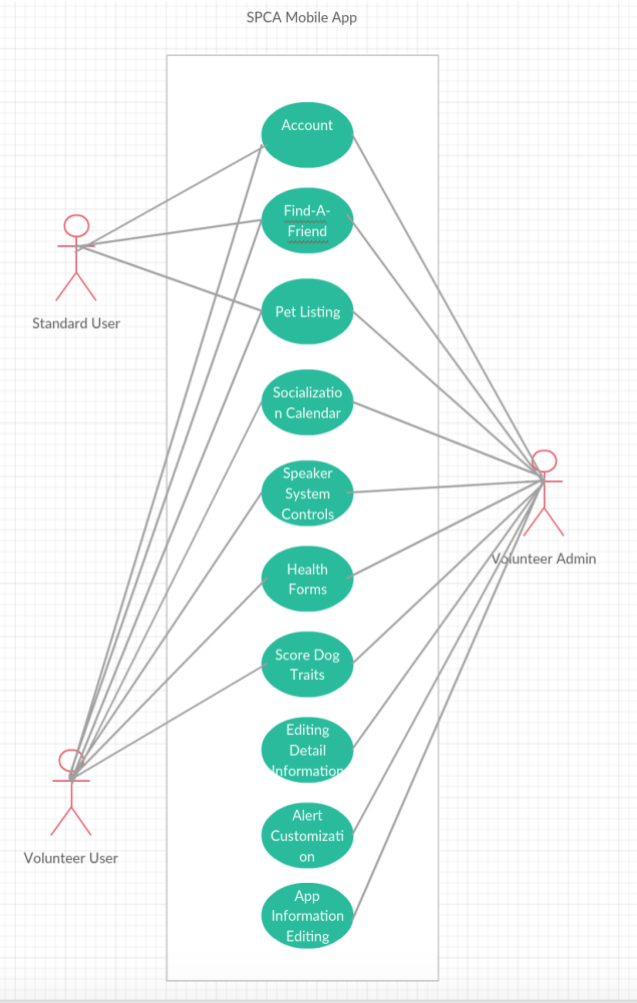

Figure 2: Use case diagram

Technical Architecture:

Figure 3 contains the technical architecture. React Native makes HTTP calls to AWS, which grabs data from a number of different sources, which are explained below.

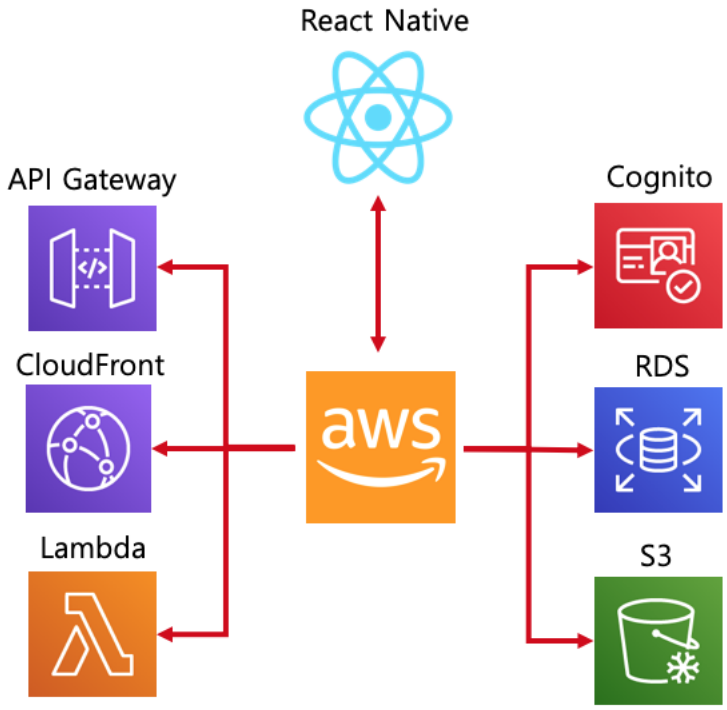

Figure 3: Technical Architecture Diagram

Architecturally, the SPCA App is designed to be very simple. React Native contains a fetch() method, which makes RESTful HTTP calls to a provided URL. In this case, the fetch method will call a URL hosted by API Gateway. API Gateway processes the request and sends data to a Lambda function. Each Lambda function will handle the request further and make data requests to MySQL hosted by RDS. The data is then passed back to the front-end in a JSON string, which React Native can handle by default. All images displayed on the front-end are stored in an S3 bucket, which has a CloudFront service sitting in front of it to host the images without giving up the S3 bucket security.

The only front-end architecture is really the structure of react-navigation, which is what enables users to hop between each of the 4 main screens and any sub-screens. React Native handles this on its own without much needed from the developer.

The MySQL database contains 4 tables; 3 for animal information and 1 for announcements. 1 table of animal information is an enumeration table for the animal type, while another holds all of the dog scores for the quiz. Each table is tied together by a primary key id, which is an auto-incrementing number to prevent ids from getting complex.

Testing:

Overview

This section is intended to guide relevant users on proper testing of the SPCA Mobile Application. The app is intended to run on all iOS mobile devices (iPhones) and Android mobile devices. The following users with these roles should use this section: Developers, Project Managers, QA Testers, UX Designers, Volunteer/User Assurance Testers.

Scope

The scope of the testing is to test the design and functionality of all user screens on the SPCA Mobile App. Tests are organized by tech stack architecture (Front end, back end, server).

Objective

The objective of testing varies based on test section. For front end, the objective is to verify each screen displays as they were designed. For back end, data should transfer through the proper services and display data correctly on the front end. For the server, minimal testing is

needed, but data should be stored in the proper tables and all services should be running properly.

Entry and Exit Criteria

Entry:

- React Native compiles and builds
- iOS simulator starts
- Android simulator starts
- AWS Console is open

Exit:

- All Tests are run (Front, Back, Server)
- Bugs are documented in Slack Channel

Logging Test and Reporting

In the event a bug is found during testing, the tester (if not a team member) will report the bug to a team member. The team member will document the bug and reproduction steps in the spca-bug channel in the team's Slack. A team member will investigate the bug and, depending on the area, the proper team member will be assigned to fix the bug.

System Testing

The SPCA Mobile Application will be tested first at the server level to ensure that all services are working and running. Testing will then move to back end to ensure that data is being

fetched properly so the front end can handle the data. Finally, with both iOS and Android simulators, the front-end screens will be tested, ensuring that the design is correct, and data is returning from the API. This covers the full application tech stack, while still keeping it semi-modular to figure out the source of bugs easier.

Testing Procedures

Steps needed for testing:

- Create/Edit Regression Testing document
- Check off all test cases in Regression doc
- Document bugs in right-hand column of Regression doc
- Document bugs in spca-bug channel in Slack

Below are high-level sections of testing.

Frontend Test 1 (FT1) - proper CSS displays

FT1.2 - Purpose: the purpose of this test is to ensure that all of the CSS definitions are rendered correctly for each page within the app frontend.

FT1.3 - Description: the frontend features will be tested for both Android (testing with Android Studio) and IOS (testing with Expo). This test involves pulling up each page of the React interface and making sure that the CSS and JavaScript for each page is working as expected.

FT1.4 - Input: because we are developing a mobile app, the inputs will be driven by the users touch screen.

FT 1.5 - Expected Output: The in-app display will match our CSS definitions in the source code for each page.

Frontend Test 2 (FT2) - load testing on CSS/JS framework

FT2.2 - Purpose: to test the limitations of the JavaScript underlying the React Framework.

FT2.3 - Description: for each of the dynamic in-app pages (aka the pages that will be making backend calls), we will be testing both the upper and lower limits of the page displays. We will test the displays for cases where 0 records are pulled from the database, and for cases with large amounts of data being pulled in. This will be tested with JSON objects sent through a LAN to mimic successful database calls.

FT2.4 - Input: The input will be json files designed for each of the tests highlighted. The messages will be sent to the frontend, and displayed within the app.

FT2.5 - Expected Output: the pages will render properly in both cases. The app will only pull down as much data as it can handle within each database call.

Frontend Test 3 (FT3) - testing personality quiz for potential adopters

FT3.2 - Purpose: to test the functionality of the personality quiz, which will point prospective adopters towards pets that cater to them

FT3.3 - Description: In order to test the quiz, we will be checking to make sure that the algorithm we have chosen points individuals towards a pet that suits their personality. This will be hard to test, as multiple perspectives are required for accurate testing. As such, we will administer the quiz to multiple individuals, even outside of our project group, and will get their opinions on how closely the app matches them to pets they like.

FT3.4 - Input: the input for this case will be administered by touch within the ‘personality page’ of the app.

FT3.5 - Expected Output: adopters are able to use the app to reliably find pets that cater to their own personality/living situation.

Backend Test 1 (BT1) - test backend page updates

BT1.2 - Purpose: to test backend calls to the AWS RDB database. This will make sure data can be pulled reliably once the app is in production

BT1.3 - Description: test the backend calls by running the latest version of our AWS code and making sure each page update is working properly.

BT1.4 - Input: the input for this call will be the react native JSON objects being sent through the API gateway. The calls for each page are static in format, so no mock data is necessary to test this.

BT1.5 - Expected Output: JSON objects returned through Lambda/Gateway that can be correctly parsed into the App’s page formats.

Backend Test 2 (BT2) - test correct handling of invalid requests.

BT2.2 - Purpose: to make sure the app runs properly and handles errors for corrupted input data.

BT2.3 - Description: will mockup JSON objects with missing fields/invalid field headers and make sure that AWS can handle the calls without interrupting the frontend interface. Meaningful error handling will be added to the frontend as well.

BT2.4 - Input: mockup JSON objects with data corruption.

BT2.5 - Expected Output: error messages returned from AWS that are accompanied with meaningful in-app display.

Backend Test 3 (BT3) - test the handling of our backend for large/small database pulls.

BT3.2 - Purpose: to make sure the frontend is capable of handling database calls with no data, and with more data than it is capable of handling within app memory. To guarantee app reliability in the cases described.

BT3.3 - Description: we will test database pulls in the case when the database contains no data for the request, and in the case where the database contains more data than the frontend can handle. Make sure either the frontend or database has the necessary restrictions to ensure that this doesn't interrupted the in-app experience.

BT3.4 - Input: database interface

BT3.5 - Expected Output: uninterrupted frontend experience in both cases.

Backend Test 4 (BT4) - test the handling of the backend for simultaneous calls

BT4.2 - Purpose: to make sure the app works properly if multiple people are making the database calls at the same time.

BT4.3 - Description: this should be handled automatically by AWS, but we will be testing this to make sure everything works properly. We will setup the app and send in multiple JSON objects from multiple different devices concurrently - to make sure everything works as expected.

BT4.4 - Input: mockup JSON objects with multiple calls each (pulling from the same RDB table), being sent from multiple devices at the same time

BT4.5 - Expected Output: the apps will not throw any errors when trying to access the same data as the other connected devices.

Pass/Fail Conditions

All tests in the regression testing document must pass for any build of the SPCA Mobile App to be pushed to production. Any test that does not pass must be fixed by the proper team member before deploying.

Schedule of Team Member Testing

Team members should be spot checking features and testing whenever they commit new code to a git branch. Full regression test will be done before every deployment to production.

Schedule of Round Table Testing

Basic user assurance testing is scheduled for 3/1/2019. Depending on the results of this, another round of user assurance testing might be needed 3/7/2019. All user assurance testing will be complete by 3/10/2019.

SPCA sponsors will get access to the app sometime between 3/3/2019 and 3/7/2019. A meeting invitation has been sent via email to sponsors to test the application.

Risks

- Regression testing impact development lifecycle time
- Delays in bugs being fixed
- SPCA sponsors not able to meet during needed time

Testing Reports

Table 1 gives the results of the testing that was performed for front and back end. Initial tests were conducted by each team member before User Assurance (UA) testing. Based on feedback/bugs from the first round of UA, bugs were fixed, and a second UA phase was done.

Date	Test	Pass/Fail	Resolution
2/29/2019	Lambda Functions	Pass	-
2/29/2019	Server Uptime	Pass	-
2/29/2019	iOS Front End	Pass	-
3/1/2019	Android UA	Fail	Observe bug feedback and resolve. Test again 3/7/2019
3/1/2019	iOS UA	Fail	Add more padding to each page. Test again 3/7/2019
3/7/2019	Android UA	Pass	-
3/7/2019	iOS UA	Pass	-

Table 1: Testing reports table.

Budget:

No money is needed in the fall semester for the front or backend technologies. React Native is free and open source to use as well as Expo. AWS Cloud Services are free up to a certain number of hours, and we will not hit above free hours for the duration of the project if we run just a single instance. For hardware, we need a test speaker and Raspberry Pi. If the speaker works out in testing, we plan to purchase 3 or 4 more units. We only need one raspberry pi to host and connect the speaker system.

Table 2 contains a budget breakdown for our project for the Fall of 2018. In the spring semester, we chose to not do any hardware additions for the volunteers since it was a bit outside of scope. Therefore, an updated budget of $0 is the total, since all technologies used were free.

Item	Unit Count	Cost	Total
Bose Speaker	4	$50	$200
Raspberry Pi	1	$30	$30
Total			$230

Table 2: Budget Breakdown (Fall 2018)

GANTT Chart:

Figure 4 on the next page contains our GANTT chart. Compared to previous iterations of the GANTT chart and a few setbacks, we have chosen to push back the hardware side of the project. Since all hardware aspects are volunteer related, and the main focus of the app is to build something for everyday users with the pet listings, we felt that it was best to focus on the software to get that done and then connect the hardware to it, given enough time.

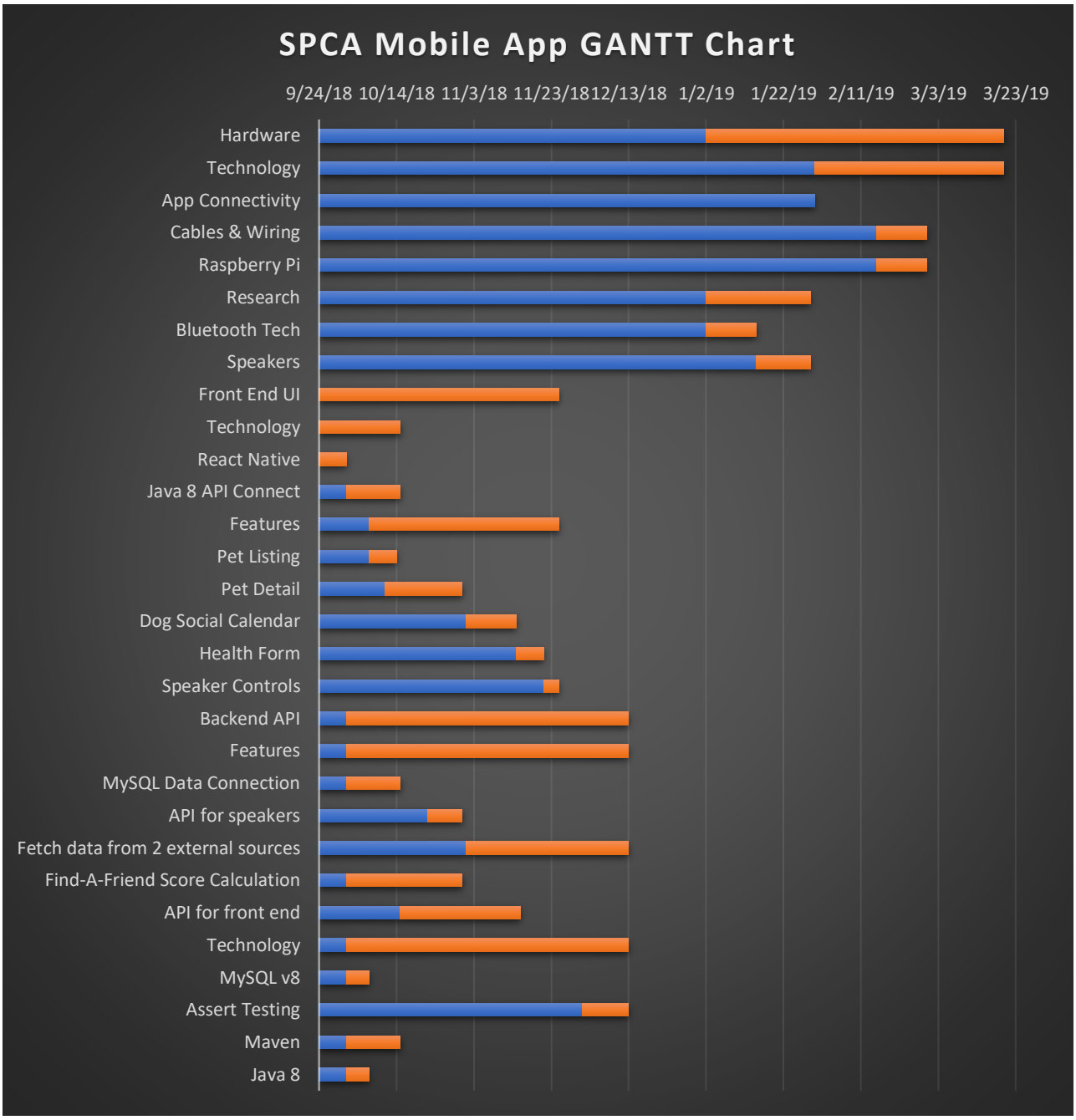

Figure 4: GANTT Chart

Application Screenshots:

Here are some screenshots of the app. The colors needed to be light enough for the app to look good and still give off a playful vibe. Colors also had to work with SPCA Cincinnati's signature red color, which is why a white background was chosen. *Figure 4* shows the home page where all users can find announcements that the SPCA puts out. *Figure 5* shows the animals listing page where users can go to see cats, dogs, and exotic pets. *Figure 6* shows the about page where users can go to find the history and mission statement of the SPCA.

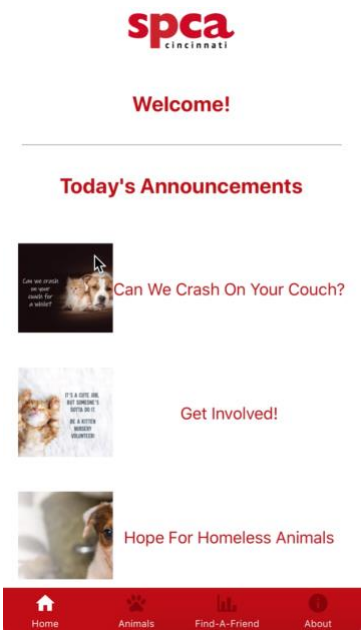

Figure 5: Announcements

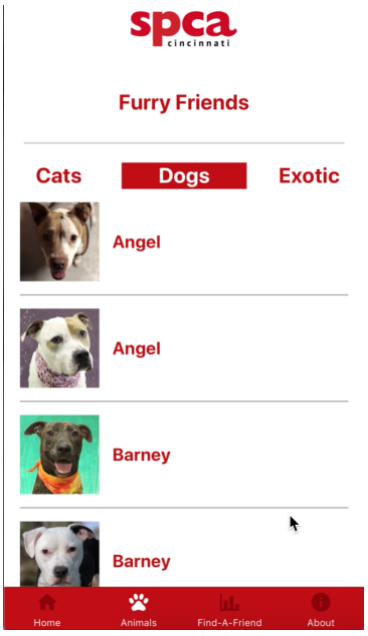

Figure 6: Animal Listing

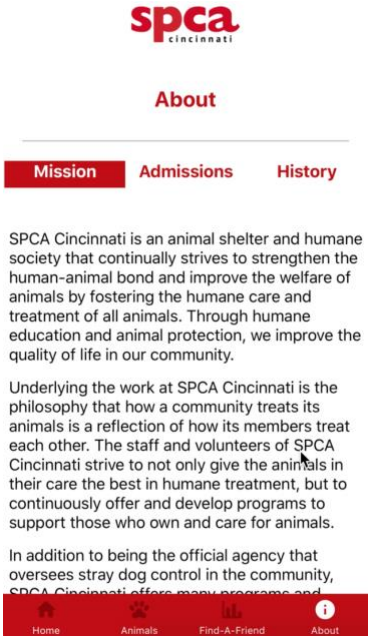

Figure 7: About page

Problems Encountered

Some initial problems we struggled with was what front-end technology to use. We went back and forth between 3 of them: React Native, Xamarin, and Meteor. This caused a delay in starting the app until we came upon an agreed decision through research of each technology.

Another problem is attempting to integrate the 3rd party systems that the SPCA wants. They have been nonresponsive in our attempts to contact them. Our solution for this is to build the app and not include any 3rd party system until senior design is complete, then work with the SPCA once the prototype is finished to deliver a final product.

Future Recommendations/Improvement

We feel that if we were to do this app all over again, that nothing would change. We set out with the best technologies we had available and came up with a simple design for a front-end and backend architecture that worked very well together. It would have been nice if the SPCA was more directly involved in this project instead of occasional emails.

If we had more time, we would have implemented some of the volunteer specific features like the dog socialization calendar and also probably moved it to the app stores. SPCA did not respond to give us permission to move them to the app stores, which would have been the last thing we would have done.

Big suggestions were on the color choices on the front-end, since none of us were very color-savvy. Other than that, mainly just small design suggestions were recommended, and we implemented several of them to make the app very clean.

The plans for this project are to shelve it for now, as we all have greater app ambitions in the future. Pending that SPCA responds to our requests to demo the app to them, we would gladly continue working to improve the app.

Conclusion

Fall 2018:

Some initial lessons we have learned are that it is difficult keeping on track with tasks with all the other schoolwork going on. Connor has so far learned to use React Native and Randy has been doing some initial Bluetooth technology integration as well. Coming in the Spring, we will have to test and set up the final application, which will involve some developer operations work with AWS. All of the hardware stuff will also be coming in the Spring.

Spring 2019:

Spring 2019 was not nearly as challenging as Fall 2018. A majority of the work was done in the fall so that spring semester would be a relatively light workload. Many of the technologies used between React Native and AWS worked seamlessly together, and other than brushing up on AWS documentation, there were really no challenges.

Connor's React Native skills, and JavaScript skills in general, have greatly improved from the start of this project. Alex and Randy both were introduced to AWS technologies and project management skills, respectively.

Since the fall semester was all front-end work, the backend was started and completed in the spring. Several improvements and color changes were made to the front-end in the spring as well.

Expo went exactly how was planned. Everything worked as it should, however we would have liked to have had more demo phones available for people to use. The judges were not very reactive, so it was very hard to read if they liked the app or not, which is what I would expect to

happen in important meetings in the future. Expo taught us that preparation is very critical to having a wonderful experience.

References

O'Leary, Connor M, and Lee Ann Luxemberger. “SPCA Board Meeting.” 5 Sept. 2018.
React Native - https://facebook.github.io/react-native/

Example Programming Code

Figure 8 has a screenshot of a successful test of a Lambda function in the AWS console. This particular call retrieved all of the cats and cat information out of our RDS MySQL database.

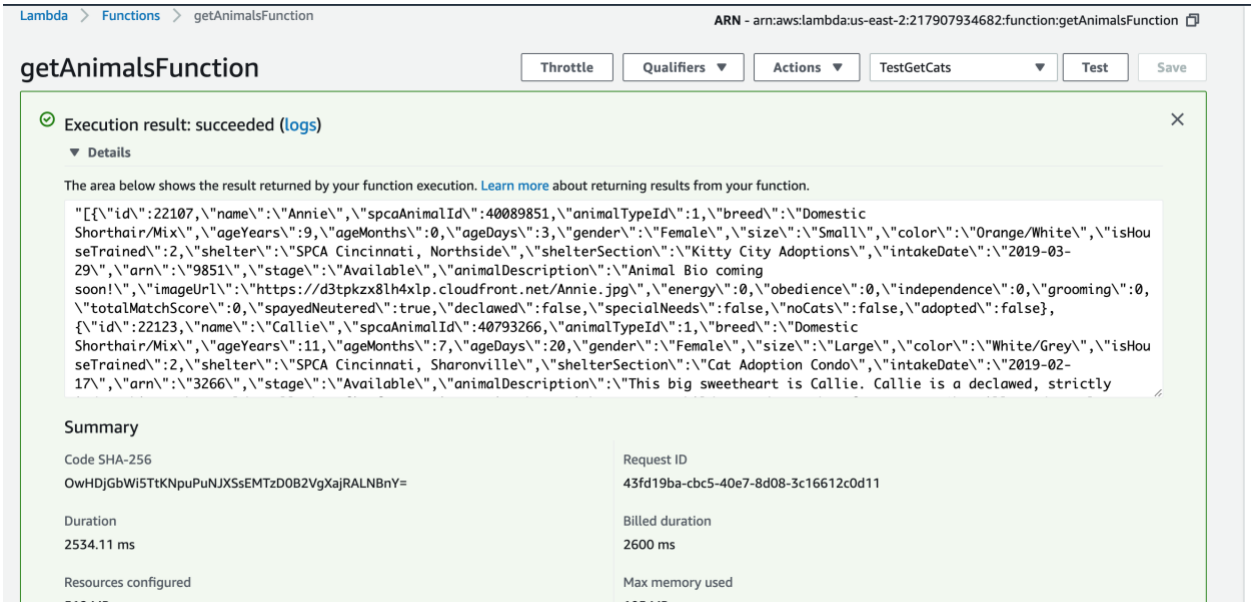

Figure 8: Lambda function for fetching Animal Listing data (test)

Figure 9 has a screenshot of the React Native code for the front end. On the left side is the scss sheet used to control all of the colors and padding for every screen. On the right is a screenshot of the render() function for the About page. It shows an example of the Handlebars syntax, which is similar to Bootstrap.

```
const React = require('react-native');
const {
  StyleSheet,
  Platform,
} = React;

import colors from '../constants/Colors';

module.exports = StyleSheet.create({
  logoContainer: {
    flex: .1,
    backgroundColor: '#ffffff',
    alignItems: 'center',
    height: 40,
    paddingTop: '5%',
  },
  logoContainerBack: {
    flex: .1,
    flexDirection: 'row',
    backgroundColor: '#ffffff',
    alignItems: 'center',
    height: 40,
    paddingTop: '5%',
  },
  spcaImageBack: {
    width: '57%',
    height: '90%',
    resizeMode: 'contain',
    marginTop: 10,
    alignItems: 'center',
  },
  container: {
    flex: 1,
    backgroundColor: colors.background,
```

```
      aboutSection: 'Mission'
    }
  }
  render() {
    const section = this.state.aboutSection;
    let currentSection;
    if (section === 'Mission') {
      currentSection = 'Mission'
    } else if(section === 'History') {
      currentSection = 'History'
    } else {
      currentSection = 'Admissions'
    }
    return (
      <View style={styles.container}>
      <HeaderLogo />
        <ScrollView style={styles.container} contentContainerStyle
          <View style={styles.getStartedContainer}>
            <HeaderText title='About' />
            <View style={styles.headerTextUnderline}/>
            <View style={styles.aboutSectionHeaders}>
              <TouchableOpacity style={this.state.aboutSection ===
                onPress={this.setMissionStatement}><Text style={th
              <TouchableOpacity style={this.state.aboutSection ===
                onPress={this.setAdmissions}><Text style={this.sta
              <TouchableOpacity style={this.state.aboutSection ===
                onPress={this.setHistory}><Text style={this.state.
            </View>
            <View style={styles.aboutScreenUnderline}/>
          </View>
          <AboutWrapper section={currentSection} />
        </ScrollView>
      </View>
    );
```

Figure 9: React Native code for Style sheet (left) and About screen (right)

IT Expo 2019 Poster

Figure 10 has a rendering of the poster used at IT Expo 2019. On the left is the technical stack seen above and some acknowledgements on groups/people who helped the group with this project. In the center are screenshots of the app (Announcements and Animal List screens). On the right are the group member names and roles, technical advisor, and small paragraphs about the application, the problem statement, and the solution the group came up with. The red is the same shade of red the SPCA uses.

Figure 10: IT 2019 SPCA Mobile App Poster