Group 6:
Aryan Aziz (aziza); Eric Rouse (rousee); Lacie Wilson (wilslaci)

1. Introduction

Executive Summary

The three of us collaborated to build the Corvallis Reuse and Repair Directory for people in the community to easily determine which business will take their reusable or repairable items. This was accomplished with a mobile application and an api-accessed database backend. Also included was a web-application to allow administrators to update and manipulate the databases.

Importance

This application makes it easier for people to find ways to recycle their reusable items, simultaneously reducing landfill use and benefiting local businesses.

Clients

The primary client is Republic Services, but their client, and thus our final customer, is the general public in Corvallis and the surrounding community.

Roles

The role of the client was to provide a general description of what they wanted and the data that they currently use.

2. Changes Since Design

Since the original design document, we decided to switch the Xcode project template from a Single View Application to a Master-Detail Application. This allowed our initial category list to be our master view, and trickle into the subsequent categories and business details from there.

In addition, we decided to make use of core data to store and dynamically retrieve key information (category and category id) for the initial category list. This worked to make the flow of information from screen to screen more fluid.

For the business list screen, we decided to add a few more details than we had originally planned, namely the numbering of businesses and the calculation of distance between the user and the business. With the numbering of businesses came changes to the map. We used custom pins with corresponding numbers to more easily locate a certain business on the list. There was also a last minute decision to add another screen that is launched by clicking on the detail disclosure button on a map pin annotation.

This new screen gives the user the opportunity to view business details from both the list of businesses and the map of businesses.

3. Project Documentation

How Does it Work

Basically the mobile application works as a step-by-step search tool. It guides a person to find businesses for the items they want to give away.

Let’s say a member of the community wants to get rid of a microwave. That user opens app and selects the “Appliances - small” category. From the subsequent list, she chooses “Microwaves”. A list of all the business that will accept the microwave is listed along with the distance of the business from the user’s current location. From there, the user can choose to look at a map to see where all the businesses are located, or they can click on a business to get the specific details such as address, website, phone number, hours and directions. The phone number is linked to call the business upon clicking, and the website is hyperlinked as well. The directions button will launch the Maps application and give the user directions to the business from their current location. Should the user choose the view the businesses on the map, they will be able to view these same details with the same features by clicking on the pin annotation info button. The user will use these features and details to find the appropriate and most convenient place to recycle their microwave.

All of the information, including the category and subcategory lists as well as the businesses and their information is dynamically loaded from the database using API keys.

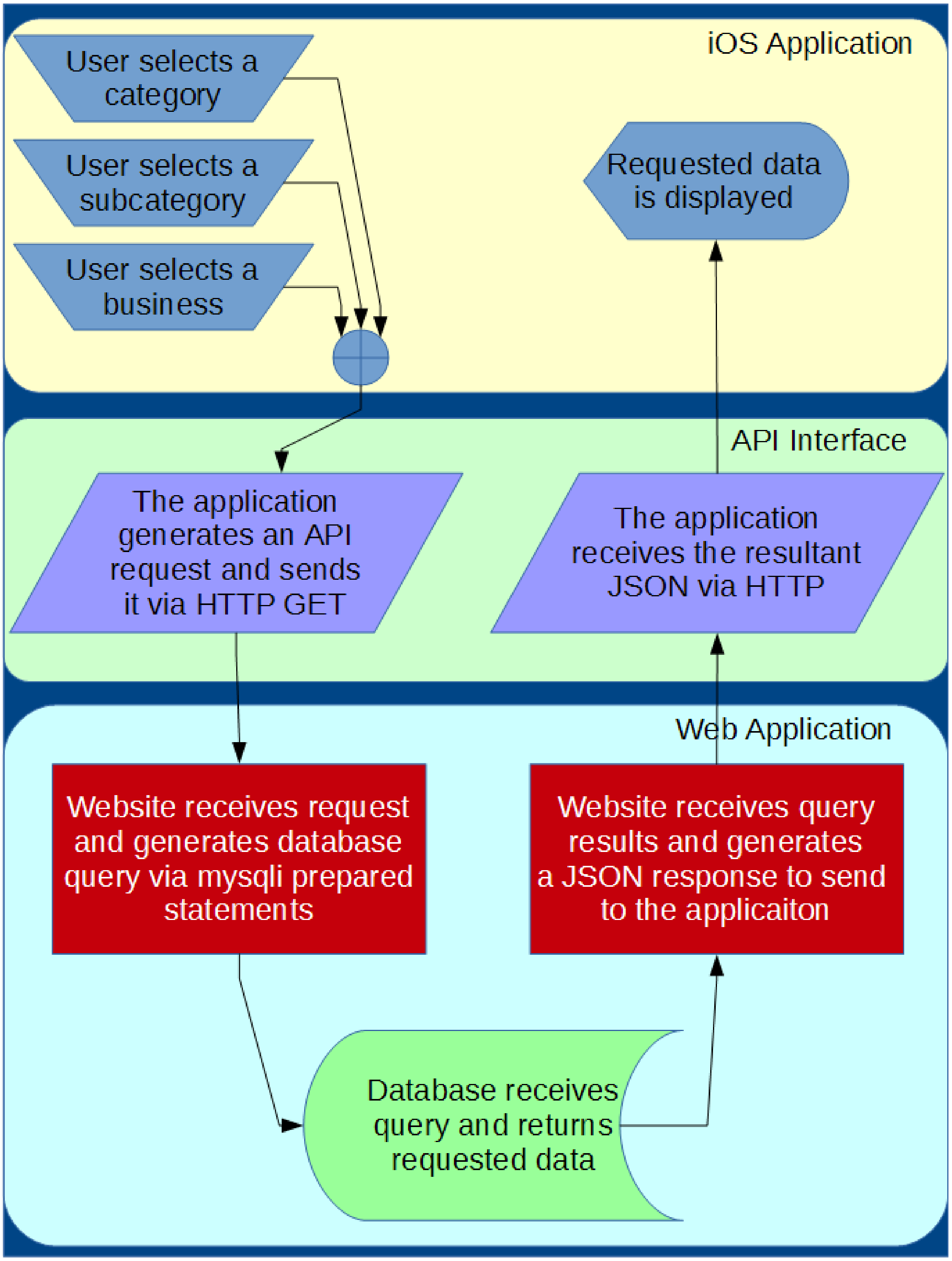
Structure

Corvallis Reuse and Repair Directory is an application composed of three parts, the mobile application, a database and an API to connect those two.

The mobile application was built using Xcode and Swift for iOS 8, while the database is MariaDB, accessed through an API built with PHP. Thus, database access is restricted to ensure security and data integrity. No user is allowed to query the database directly, the API is the only method for getting database info.

Also, there is an administrator's web interface is also almost entirely PHP and accesses the database through mysqli prepared statements. A username and password are required for Administrative access, also for security reasons. And, through use of prepared statements, malicious or poorly sanitized inputs are thwarted.

Theory of Operation

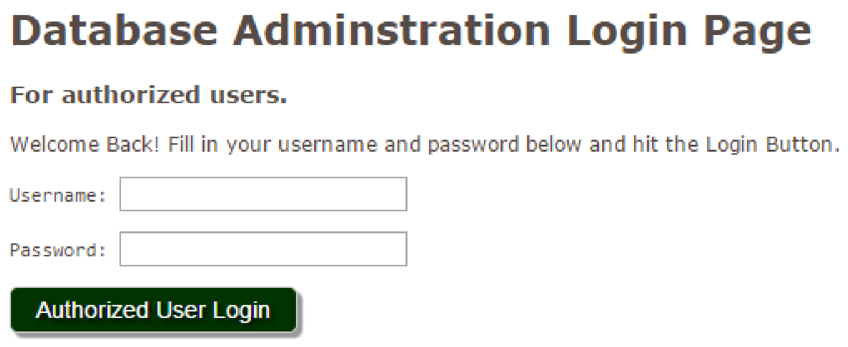

iOS Application
User selects a category
User selects a subcategory
User selects a business
Requested data is displayed
API Interface
The application generates an API request and sends it via HTTP GET
The application receives the resultant JSON via HTTP
Web Application
Website receives request and generates database query via mysqli prepared statements
Website receives query results and generates a JSON response to send to the applicaiton
Database receives query and returns requested data

Installation

Connect a registered device attached to a corresponding Apple Developer membership to a laptop running Xcode with the project. The steps to enroll in the Apple Developer program, get the necessary certificates, register your device, and set up Xcode to run your app on your registered device are all outlined on this website: http://codewithchris.com/deploy-your-app-on-an-iphone/.

Once these steps are completed, you just need to find the .xcodeproj file in the project folder, and click it so that the project opens in Xcode. From there, you should be able to choose to run the application on your device, and it will install and automatically open the application on your phone.

Of course, for the the administration web site no installation is required, just the use of a modern browser like Chrome or Firefox.


Database Adminstration Login Page

For authorized users.

Welcome Back! Fill in your username and password below and hit the Login Button.

Username:

Password:

Run

Running the application is as easy as tapping the icon. It will also automatically run on the phone by running deploying the project to your device from XCode.

To access the web administration database, one must navigate to the URL: http://web.engr.oregonstate.edu/~rousee/CS419/. The **username** is "*Admin*" and the **password** is "*cs419dbp@ss*" without quotes.

Hardware Requirements

To run the mobile application we used OS X Yosemite 10.10 with Xcode 6.3.2. The application was tested on an iPhone with the 8.3 software. However, the deployment target is 8.2, so it will run on software of version 8.2+.

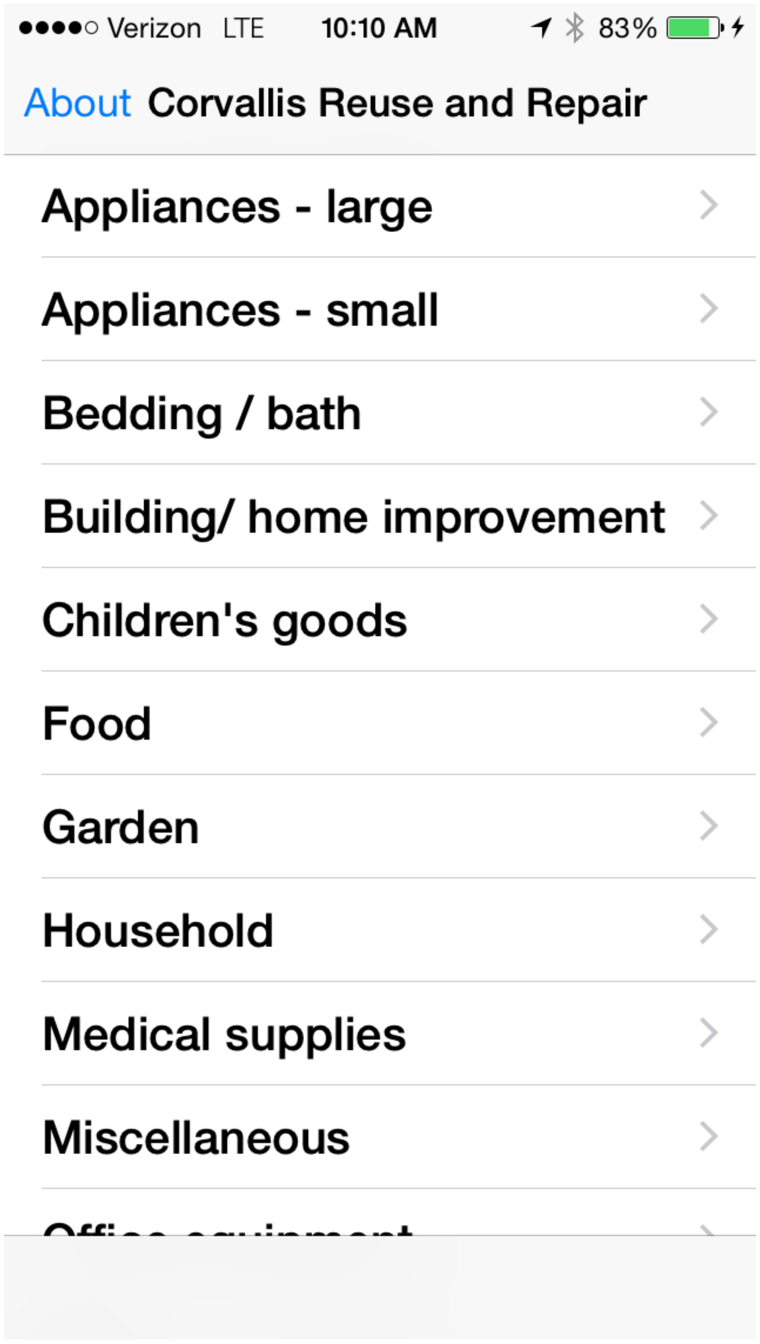
For database administration it is strongly recommended to use either Mozilla Firefox or Google Chrome.

Getting Started Guide

To begin, your phone software must be up to date with the version 8.2 or higher. The XCode and Mac OS X software compatible with your version of software is necessary in order to get the application on your device. Once you have your phone software updated to the necessary level, and the corresponding XCode and Mac OS X software, you can continue on to the steps needed to deploy the app on your iPhone. The steps to complete this are detailed at this website: http://codewithchris.com/deploy-your-app-on-an-iphone/. One difference to note is that, depending on the version of XCode you are using, you might need to find the “devices” tab under Window -> Devices rather than Windows -> Organizer. In addition, ensure that you download and open each certificate. Much of the steps can be completed from the “Certificates” section of the Apple Developer Member Center after you enroll in the program.

Once you have your device prepared for deployment, attach it to your computer via a USB cord, and you should see it come up as a run option. Click “Run” and ensure that your phone is open to the home screen. Shortly after, you should see the application install and open on your device. The application is now installed on your phone and ready for use.

The first screen that you see will be the list of the top level categories of recyclables. All lists are maintained and updated dynamically from our backend database. From this screen, the user can move in one of two directions.


●●●●○ Verizon LTE 10:10 AM 83%

About Corvallis Reuse and Repair

Appliances - large

Appliances - small

Bedding / bath

Building/ home improvement

Children's goods

Food

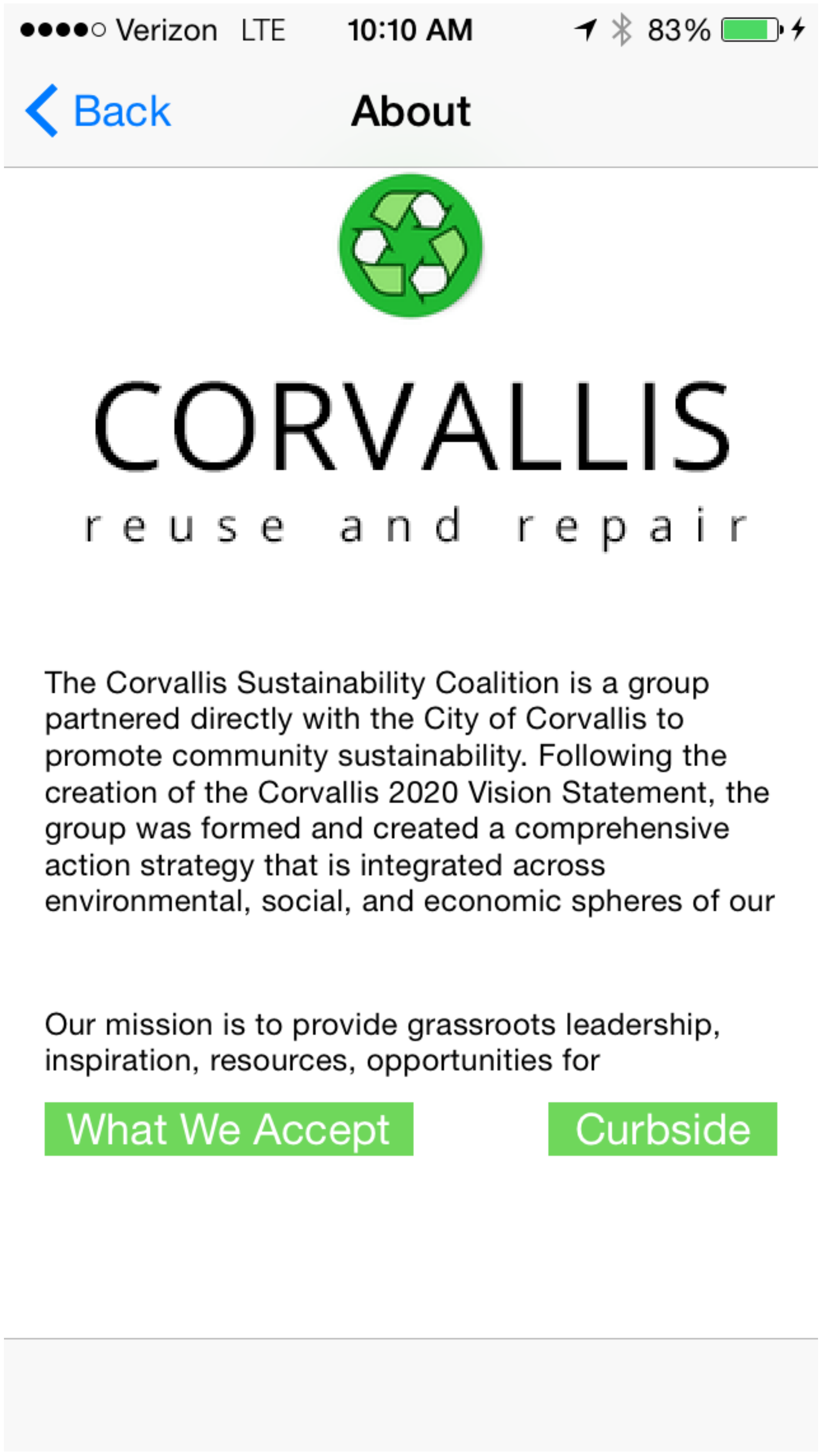
Garden

Household

Medical supplies

Miscellaneous

The user can click the “About” button to take them to an informative screen that talks about Corvallis Reuse and Repair Directory and provides a link to the local company Republic Services responsible for collecting recyclables so the public knows what items can be recycled in case their items are unusable and unrepairable.


●●●●○ Verizon LTE 10:10 AM 83%

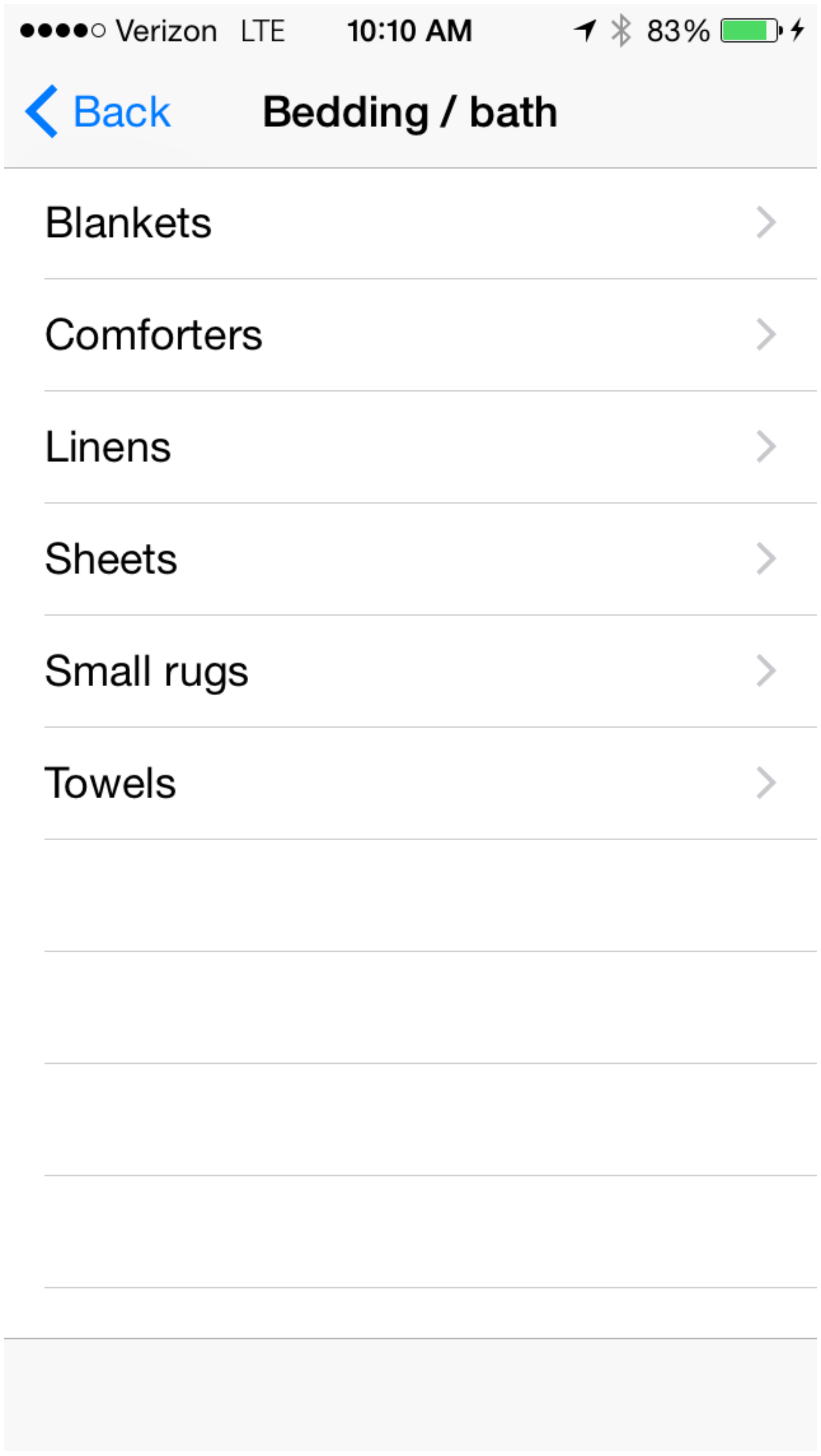
Back **About**

CORVALLIS
reuse and repair

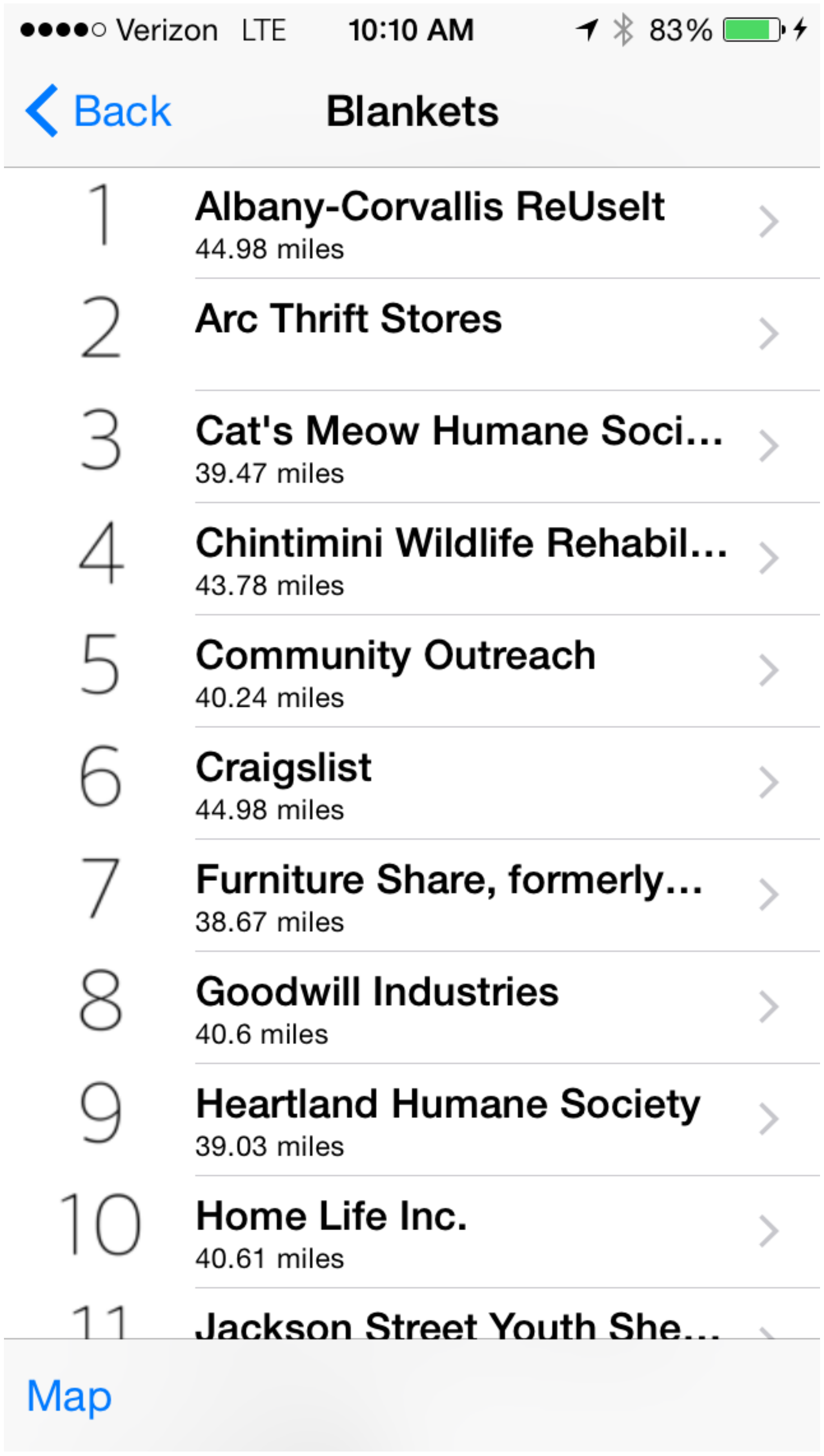
The Corvallis Sustainability Coalition is a group partnered directly with the City of Corvallis to promote community sustainability. Following the creation of the Corvallis 2020 Vision Statement, the group was formed and created a comprehensive action strategy that is integrated across environmental, social, and economic spheres of our

Our mission is to provide grassroots leadership, inspiration, resources, opportunities for

The user can also choose a top level category of interest to view the relevant subcategories.



From here, the user will choose the correct subcategory, and then be taken to a numbered list of businesses that accept their item. This list will also indicate the distance of the business from the user's current location (provided that the business has an address entered in the database).


●●●●○ Verizon LTE 10:10 AM 83%

Back **Blankets**

1 **Albany-Corvallis ReUseIt**
44.98 miles

2 **Arc Thrift Stores**

3 **Cat's Meow Humane Soci…**
39.47 miles

4 **Chintimini Wildlife Rehabil…**
43.78 miles

5 **Community Outreach**
40.24 miles

6 **Craigslist**
44.98 miles

7 **Furniture Share, formerly…**
38.67 miles

8 **Goodwill Industries**
40.6 miles

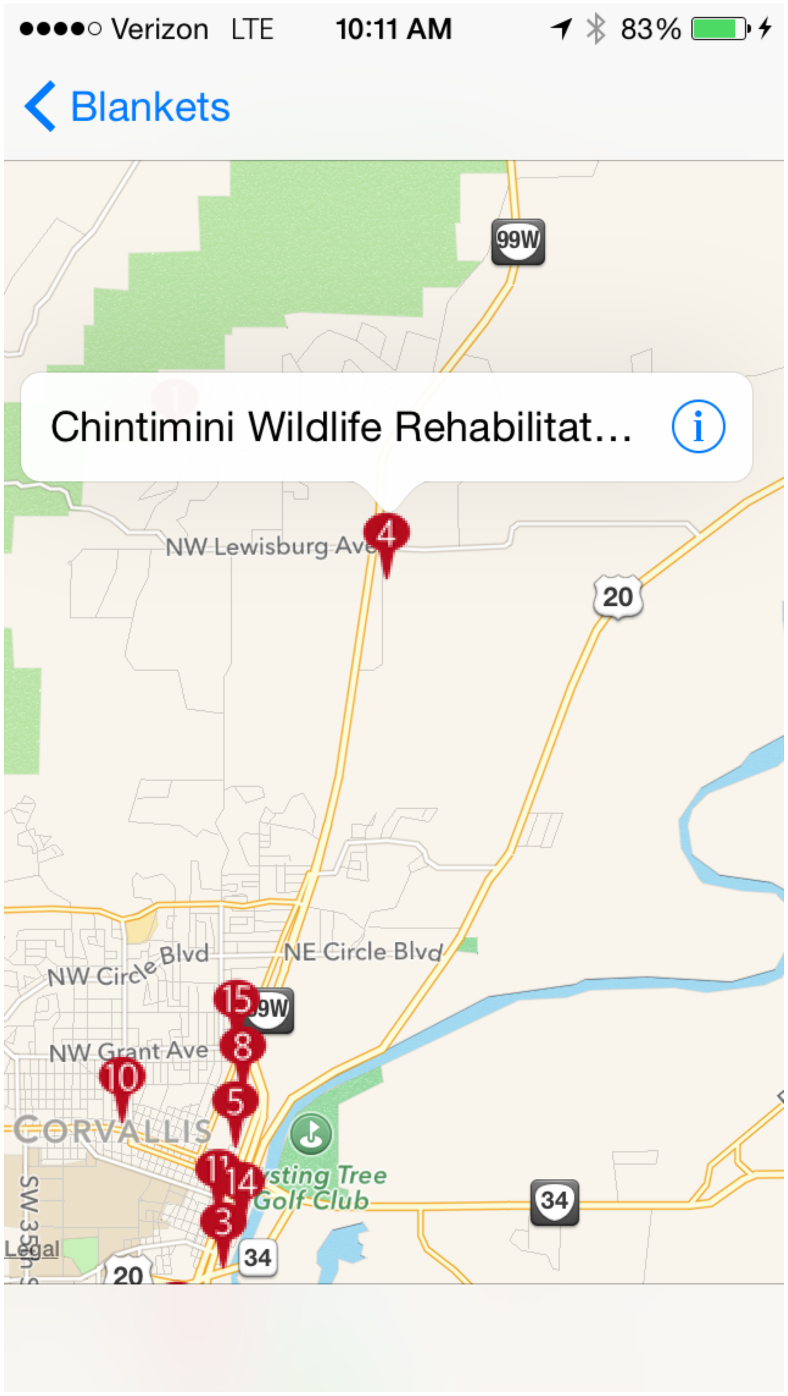
9 **Heartland Humane Society**
39.03 miles

10 **Home Life Inc.**
40.61 miles

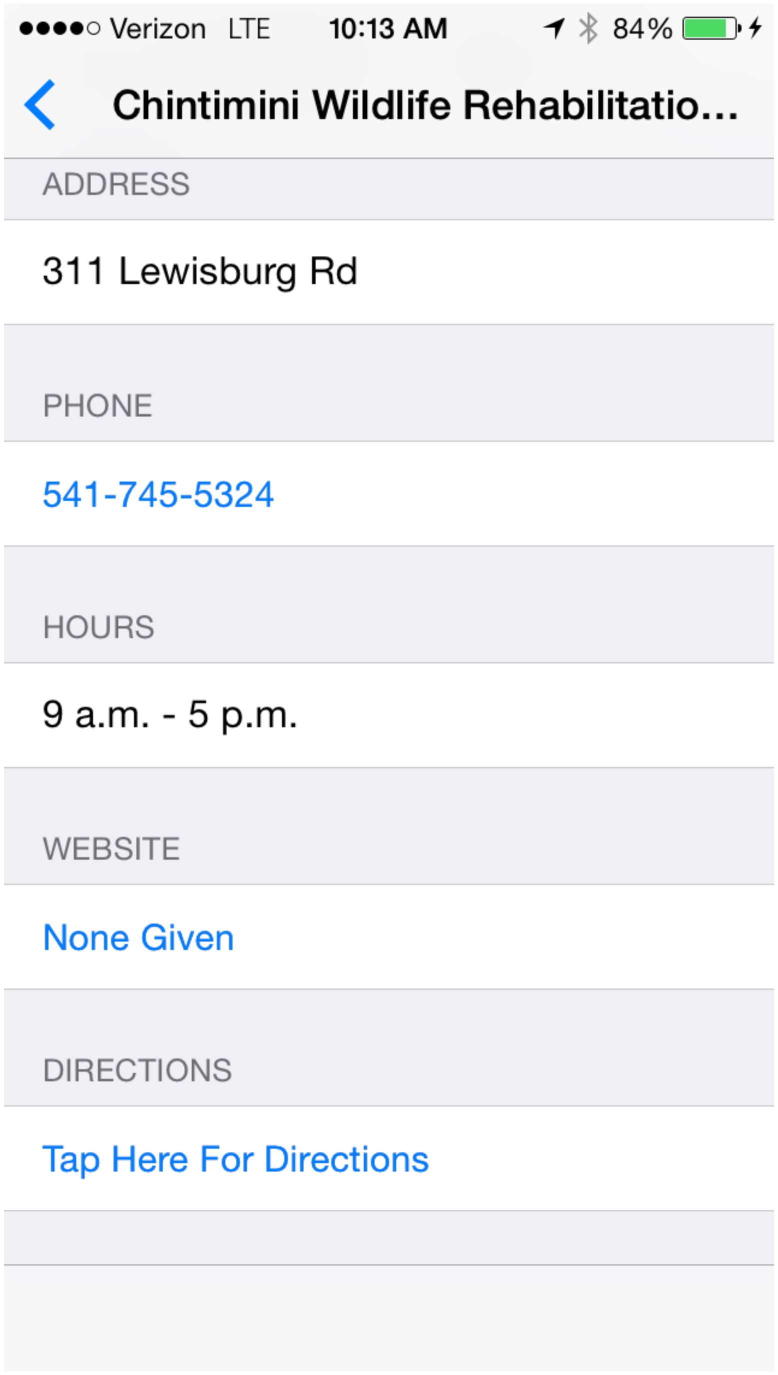
11 **Jackson Street Youth She…**

Map

On this screen, there is also a “Map” button that will take the user to a map of the businesses that have provided addresses.



On the map screen, these businesses will be represented by numbered pins that correspond to their numbers on the previous list of businesses. The user’s location will also appear on the map as a blue dot. Upon tapping a pin, an annotation will appear with the business’s name and an information button. Clicking the information button will take the user to a screen with details about the business.


●●●●○ Verizon LTE 10:13 AM 84%

Chintimini Wildlife Rehabilitatio...

ADDRESS

311 Lewisburg Rd

PHONE

541-745-5324

HOURS

9 a.m. - 5 p.m.

WEBSITE

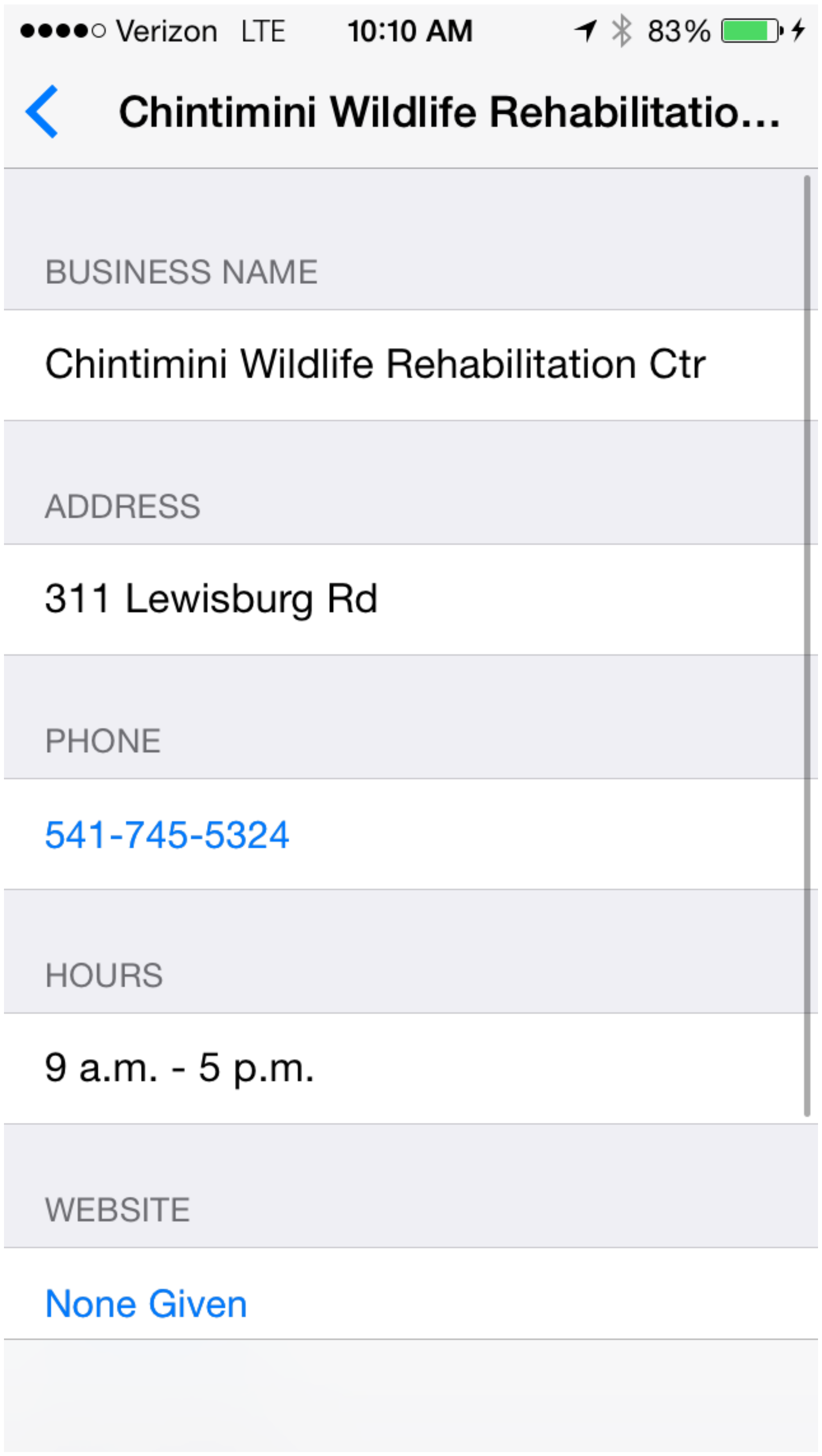
None Given

DIRECTIONS

Tap Here For Directions

The business detail screen provides essential information about the business. Depending on the information provided by the business, this screen will include the business name, address, phone number, website, and hours. The phone number link will immediately call the business about tapping the number. The website link will open a browser and take the user to the website. There is also a directions button that will launch the Maps application and give the user directions from their current location to the business.

The user can also be taken to a screen with this same business information by clicking on the business name from the business list screen.


●●●●○ Verizon LTE 10:10 AM 83%
Chintimini Wildlife Rehabilitatio...

BUSINESS NAME

Chintimini Wildlife Rehabilitation Ctr

ADDRESS

311 Lewisburg Rd

PHONE

541-745-5324

HOURS

9 a.m. - 5 p.m.

WEBSITE

None Given

4. Learning New Technology

Helpful Websites

1. Stack Overflow: http://stackoverflow.com/
2. Udemy Tutorials: https://www.udemy.com/
3. How to Deploy App on iPhone: http://codewithchris.com/deploy-your-app-on-an-iphone/
4. PHP Documentation: http://php.net/docs.php
5. MySQL Documentation: https://dev.mysql.com/doc/

No books or people used as resources

5. What We Learned

Aryan Aziz:

What technical information did you learn?

Since our iOS application was built completely on top of Swift; this project allowed me to learn both the usage and debugging of Xcode, as well as Apple's new programming language, Swift. Surprisingly, it was more difficult to use Xcode and it's finicky requirements for rendering the app (modifying the plist file, introducing of locations, managing the permissions/targets, and designing on the storyboard) than it was to actually learn and understand Swift (which seems much more like an extension of the languages we previously learned).

What non-technical information did you learn?

The importance of planning and having everyone on the same track, and how hugely important it is. This is something you don't necessarily think about when you're working on your own, and I am especially guilty of having terrible pre-planning for the code I am going to write. But the best graphic I can relate this to is to imagine 2 groups of people building a bridge from different sides; hoping to meet in the middle. Planning is the most important step because everyone should know which way the bridge is being built and at what style and at what height and angle and what materials, etc etc etc. If one side if under a different impression of the plans than the other, or they change plans mid way, it can have disastrous results (bridge not meeting, one side higher than the other, etc). Before anything, there must be strong planning with everyone having knowledge of where they are, what they should be doing, and where they (and the whole project) should be going.

What have you learned about project work?

I think I covered the bulk of this in the above section, but the most important parts of a project are the planning and making sure everyone is on the same page. In terms of this specific project, the difficulties of getting everyone together for meetings (due to conflicting schedules) definitely showed; however we were able to overcome this and work independently to produce what I feel is a great product.

What have you learned about project management?

Changes/issues will always popup along the way, and being able to move on from them is crucial. As important as planning is, there will always be things that you miss or things that are planned one way but must be done in a different way. The best thing is to get all of the pros/cons of the changes, and everything else that it would effect and in what way, and determine where the changes in development should occur. Maybe it’s easier to change something else, which would fix the current issue, rather than changing this item; for example.

What have you learned about working in teams?

When it comes to teamwork, being in a team is all about having each other’s backs and working together to solve a problem. Teams will always have weak links somewhere, and it’s important for team members to try to cover those weak links and instead play to their strengths. For example, someone might be strong in MySQL/PHP but weak in iOS development. Someone else might be strong at design and web development but weak in the back end api. And someone else could be strong at iOS development and weak at the others. The team should work together to exploit each other’s strengths and produce the best products.

If you could do it all over, what would you do differently?

Better planning and staging upfront of what we are trying to accomplish. Early on we got into the notion that the project was going to be simpler than we thought, and we skirted around some of the planning and designing and just dove into development; which really came back to bite us in the later stages. Regardless, I think we all did a great job in working independently towards a single common goal with the minimal meetings we had; but obviously working more closely and meeting together more often would have definitely helped.

Lacie Wilson:

What technical information did you learn?

I learned how to use and navigate Xcode. I also learned the Swift programming language, and how to use swift to develop an iOS application. In addition, I learned a little bit about the use of photoshop to create images of the appropriate size/pixels for an iOS application icon and splash screen.

What non-technical information did you learn?

I learned that having a drawn out outline and “plan of attack” is very important. I also found it to be necessary to keep an updated checklist of things that need to be added/fixed/updated in my code. The “learn as you go” approach was eventually effective, but I found myself needing or wanting to start over as I learned new things that would work better for the purposes of the project. Overall, the non-technical information that I learned was largely in the category of planning and organizing. However, I also learned that it can be difficult to find information and resources for a new programming language such as Swift.

What have you learned about project work?

I’ve learned that a large part of it is actually the planning, design, and organization that occurs before any of the implementation comes into play. Having a plan, a timeline, and an updated list of things complete for the project is key. It can be more difficult to keep yourself on task for a larger project, but it’s also more rewarding to have a substantial product at the end.

What have you learned about project management?

It can be difficult to try to manage both yourself and other project members. People complete tasks at different times that might not perfectly coincide with when you would like them to be done, and vice versa. The progression of a project depends on the pace that all of the team members are moving, not just your own. It is hard for individual members to design their own work around what you are doing when they don’t know the specifics of your work. Open, accessible lines of communication are key. Clear, common goals for the project are also essential. It is also very convenient to manage a project when there are different people who can specialize in different parts of the project. Each member can manage themselves in their specialized area, and then reconnect with the group members to integrate the different parts.

What have you learned about working in teams?

Working in teams can make things much less stressful at times because when you are stuck or wanting advice, you have team members to turn to. Working in teams remotely does make thing more difficult. Scheduling meetings can be challenging when everyone has a different schedule, and it is also not the ideal situation to have a meeting via Google Hangouts. Working in teams would be a greater benefit if in person pair-programming could be an option. Relying heavily on communication via emails can really slow the process. As mentioned in the management section, having team members with their own personal knowledge of a topic that you are less brushed up on is very useful. It is great to be able to split up tasks and have people to bounce ideas off of or ask questions.

If you could do it all over, what would you do differently?

I would have been more prepared with a very solid plan of action. I would understand all the necessary bits and pieces of making the dynamic parts of the application before I began coding. I would draw out every screen that I wanted to have, have a list of features for each one, and research each of those features before project coding happened. Having a plan and knowing how to achieve it beforehand reduces the stress and the need to alter things or start over. I would also have had my Apple Developer information linked up to Xcode from the beginning to make for simpler and more accurate testing of the app. Having all of my graphics done from the beginning would have also been convenient. Knowing more about advanced screen formatting would have been beneficial as well.

Eric Rouse:

What technical information did you learn?

I became more familiar with msqli and SQL.

What non-technical information did you learn?

Databases are never done, there is always something to add or modify. Also, customer data is not always that good.

What have you learned about working in teams?

Distributed teams are often kind of hard, (especially when the members have other jobs). Communication delays can slow development significantly, but we all worked on our stuff pretty independently. I was surprised by how effective we were individually with minimal update meetings.

If you could do it all over, what would you do differently?

I'd rethink the database schema to make it more adaptive.

Appendix 1: Essential Code Listings

Database Schema

```
+--------------------+
| Tables_in_cs419-g6 |
+--------------------+
| administrators     |
| businesses         |
| categories         |
+--------------------+

+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| username | varchar(255) | NO   | UNI | NULL    |                |
| password | varchar(255) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+


+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| name    | varchar(255) | NO   |     | NULL    |                |
| address | varchar(255) | YES  |     | NULL    |                |
| phone   | varchar(255) | YES  |     | NULL    |                |
| url     | varchar(255) | YES  |     | NULL    |                |
| hours   | varchar(255) | YES  |     | NULL    |                |
| zip     | varchar(255) | YES  |     | NULL    |                |
| accepts | text         | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+

+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| item_1  | varchar(255) | YES  |     | NULL    |                |
| item_2  | varchar(255) | YES  |     | NULL    |                |
| item_3  | varchar(255) | YES  |     | NULL    |                |
| item_4  | varchar(255) | YES  |     | NULL    |                |
| item_5  | varchar(255) | YES  |     | NULL    |                |
| item_6  | varchar(255) | YES  |     | NULL    |                |
| item_7  | varchar(255) | YES  |     | NULL    |                |
| item_8  | varchar(255) | YES  |     | NULL    |                |
| item_9  | varchar(255) | YES  |     | NULL    |                |
| item_10 | varchar(255) | YES  |     | NULL    |                |
| item_11 | varchar(255) | YES  |     | NULL    |                |
| item_12 | varchar(255) | YES  |     | NULL    |                |
| item_13 | varchar(255) | YES  |     | NULL    |                |
| item_14 | varchar(255) | YES  |     | NULL    |                |
| item_15 | varchar(255) | YES  |     | NULL    |                |
| item_16 | varchar(255) | YES  |     | NULL    |                |
| item_17 | varchar(255) | YES  |     | NULL    |                |
| item_18 | varchar(255) | YES  |     | NULL    |                |
| item_19 | varchar(255) | YES  |     | NULL    |                |
| item_20 | varchar(255) | YES  |     | NULL    |                |
| item_21 | varchar(255) | YES  |     | NULL    |                |
| item_22 | varchar(255) | YES  |     | NULL    |                |
| item_23 | varchar(255) | YES  |     | NULL    |                |
| item_24 | varchar(255) | YES  |     | NULL    |                |
| name    | varchar(255) | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
```

API Listener/Responder Code

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 'On');
//declares some useful vars
$hdr = "{\n";
$ftr = "\n}";
```

```
$item = array(26);

include '../cats.php';
include '../pass.php';
$mysqli = new mysqli("mysql.eecs.oregonstate.edu", "cs419-g6",
$db_pass, "cs419-g6");
if ($mysqli->connect_errno){
    echo "Database connection failed: (" . $mysqli->connect_errno .
")" . $mysqli->connect_error;
}

echo $hdr;

//a function to pull out the accept data from that data structure
function findaccept($accepts, $cat_id, $sub_id){
    $items = explode("]],[[", $accepts);
    $iter = 0;
    $seperator = "";
    $output = "";
    foreach($items as $item){
        $item = explode("],[", $item);
        if (!isset($item[1])){
            return null;
        }
        $item[0] = str_replace("[", "", $item[0]);
        $item[1] = str_replace("]", "", $item[1]);
        $item[1] = explode(",", $item[1]);

        if ($item[0] == $cat_id && in_array($sub_id, $item[1])){
            return true;
        }

    }

    return false;
}

//check request method
if ($_SERVER['REQUEST_METHOD'] === 'GET'){
    //return JSON based on businesses by ID
    //otherwise return them all
    if (isset($_GET['BusinessByID'])){
        if(!($stmt = $mysqli->prepare("SELECT name, address, phone,
url, accepts, zip, hours FROM businesses
            WHERE id = ?"))){
            echo "Prepare failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_param("s", $_GET['BusinessByID'])){
            echo "Bind failed: " .$stmt->errno." ".$stmt->error;
        }
        if(!$stmt->execute()){
            echo "Execute failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_result($name, $address, $phone, $url,
$accepts, $zip, $hours)){
            echo "Bind failed: "  . $stmt->errno . " " . $stmt->error;
        }
        echo "\"Business\":\n";
        while($stmt->fetch()){
            echo "{\n \"id\": \"$_GET[BusinessByID]\",\n";
            echo "\"name\": \"$name\",\n";
            echo "\"address\": \"$address\",\n";
            echo "\"phone\": \"$phone\",\n";
            echo "\"url\": \"$url\",\n";
            echo "\"hours\": \"$hours\",\n";
            echo "\"zip\": \"$zip\",\n";
            echo "\"accepts\": \"".$accepts."\"\n";
            echo "}\n";

        }
        $stmt->close();
    //all buisnesses
    } else if (isset($_GET['BusinessAll'])){
        if(!($stmt = $mysqli->prepare("SELECT name, address, phone,
url, accepts, zip, hours, id FROM businesses
            "))){
            echo "Prepare failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->execute()){
            echo "Execute failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_result($name, $address, $phone, $url,
$accepts, $zip, $hours, $id)){
            echo "Bind failed: "  . $stmt->errno . " " . $stmt->error;
        }
        $seperator = "";
        echo "\"Businesses\": [\n";
        while($stmt->fetch()){
            echo $seperator;
            echo "{\n \"id\": \"$id\",\n";
            echo "\"name\": \"$name\",\n";
            echo "\"address\": \"$address\",\n";
            echo "\"phone\": \"$phone\",\n";
            echo "\"url\": \"$url\",\n";
            echo "\"hours\": \"$hours\",\n";
            echo "\"zip\": \"$zip\",\n";
            echo "\"accepts\": \"".$accepts."\"\n";
            echo "}";
            $seperator = ",\n";
        }
        echo "\n]";
        $stmt->close();
    //category by id
    } else if (isset($_GET['CategoryByID'])){
        if(!($stmt = $mysqli->prepare("SELECT name, id,
            item_1, item_2, item_3, item_4, item_5,
            item_6, item_7, item_8, item_9, item_10,
            item_11, item_12, item_13, item_14, item_15,
            item_16, item_17, item_18, item_19, item_20,
            item_21, item_22, item_23, item_24, item_25
            FROM categories
            WHERE id = ?"))){
            echo "Prepare failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_param("s", $_GET['CategoryByID'])){
            echo "Bind failed: " .$stmt->errno." ".$stmt->error;
        }
        if(!$stmt->execute()){
            echo "Execute failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_result($name, $id,
            $item[1], $item[2], $item[3], $item[4], $item[5],
            $item[6], $item[7], $item[8], $item[9], $item[10],
            $item[11], $item[12], $item[13], $item[14], $item[15],
            $item[16], $item[17], $item[18], $item[19], $item[20],
            $item[21], $item[22], $item[23], $item[24], $item[25]
            )){
            echo "Bind failed: "  . $stmt->errno . " " . $stmt->error;
        }
        echo "\"Category\": ";
        $seperator = "";
        while($stmt->fetch()){
            echo $seperator;
            $seperator = "";
            echo "{\n \"id\": \"$id\",\n";
            echo "\"name\": \"$name\",\n";
            echo "\"items\": [\n";
            for ($i = 1; $i <26; $i++){
                if ($item[$i] == "")
                    break;
                echo $seperator;
                echo "\"".$item[$i]."\"";
                $seperator = ",\n";
            }
            echo "]}";
        }
        $stmt->close();
    //all categories
    } else if (isset($_GET['CategoryAll'])){
        if(!($stmt = $mysqli->prepare("SELECT name, id,
            item_1, item_2, item_3, item_4, item_5,
            item_6, item_7, item_8, item_9, item_10,
            item_11, item_12, item_13, item_14, item_15,
            item_16, item_17, item_18, item_19, item_20,
            item_21, item_22, item_23, item_24, item_25
            FROM categories
            "))){
            echo "Prepare failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->execute()){
            echo "Execute failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_result($name, $id,
            $item[1], $item[2], $item[3], $item[4], $item[5],
            $item[6], $item[7], $item[8], $item[9], $item[10],
            $item[11], $item[12], $item[13], $item[14], $item[15],
            $item[16], $item[17], $item[18], $item[19], $item[20],
            $item[21], $item[22], $item[23], $item[24], $item[25]
            )){
            echo "Bind failed: "  . $stmt->errno . " " . $stmt->error;
        }
        $seperator = "";
        echo "\"Categories\": [\n";
        while($stmt->fetch()){
            echo $seperator;
            $seperator = "";
            echo "{\n \"id\": \"$id\",\n";
            echo "\"name\": \"$name\",\n";
            echo "\"items\": [\n";
            for ($i = 1; $i <26; $i++){
                if ($item[$i] == "")
                    break;
                echo $seperator;
                echo "\"".$item[$i]."\"";
                $seperator = ",\n";
            }
            echo "]}";
            $seperator = ",\n";
        }
        echo "\n]";
        $stmt->close();
    //return JSON of businesses that match a category/sub-category
pair
    } else if (isset($_GET['BusinessesByCatID']) &&
isset($_GET['SubcategoryID'])) {

        if(!($stmt = $mysqli->prepare("SELECT name, address, phone,
url, accepts, zip, hours, id FROM businesses
            "))){
            echo "Prepare failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->execute()){
            echo "Execute failed: "  . $stmt->errno . " " .
$stmt->error;
        }
        if(!$stmt->bind_result($name, $address, $phone, $url,
$accepts, $zip, $hours, $id)){
            echo "Bind failed: "  . $stmt->errno . " " . $stmt->error;
        }
        $seperator = "";
        echo "\"Businesses\": [\n";
        while($stmt->fetch()){
            if (findaccept($accepts, $_GET['BusinessesByCatID'],
$_GET['SubcategoryID'])){
                echo $seperator;
                echo "{\n \"id\": \"$id\",\n";
                echo "\"name\": \"$name\",\n";
                echo "\"address\": \"$address\",\n";
                echo "\"phone\": \"$phone\",\n";
                echo "\"url\": \"$url\",\n";
                echo "\"hours\": \"$hours\",\n";
                echo "\"zip\": \"$zip\",\n";
                echo "\"accepts\": \"".$accepts."\"\n";
                echo "}";
                $seperator = ",\n";
            }
        }
        echo "\n]";
        $stmt->close();
    }
}
    echo $ftr;
?>
```

iOS Applicaton Code

```
/*This code makes it possible for us to use a custom annotation for
our map pins*/
class CustomAnnotation: NSObject, MKAnnotation {

    var coordinate: CLLocationCoordinate2D
    var title: String!
    var subtitle: String!
    var image: UIImage?

    init(coordinate: CLLocationCoordinate2D, title: String!, subtitle:
String!) {

        self.coordinate = coordinate
        self.title = title
        self.subtitle = subtitle
    }
}

/*This code takes in a JSON object and stores the categories and
category ids in core data, making it our initial point to store info
and pass to subsequent screens*/
        var appDel: AppDelegate =
UIApplication.sharedApplication().delegate as! AppDelegate
        var context: NSManagedObjectContext =
appDel.managedObjectContext!
        let url = NSURL(string:
"http://web.engr.oregonstate.edu/~rousee/CS419/api/?CategoryAll")
        let session = NSURLSession.sharedSession()
        let task = session.dataTaskWithURL(url!, completionHandler: {
(data, response, error) -> Void in

            if error != nil {

                println(error)

            }

            else {
                dispatch_async(dispatch_get_main_queue()) {

                let jsonResult =
NSJSONSerialization.JSONObjectWithData(data, options:
NSJSONReadingOptions.MutableContainers, error: nil) as! NSDictionary

                if jsonResult.count > 0 {

                    if let categories = jsonResult["Categories"] as?
NSArray {

                        var request = NSFetchRequest(entityName:
"Categories")
                        request.returnsObjectsAsFaults = false

                        var results =
context.executeFetchRequest(request, error: nil)!
                        if results.count > 0 {
                            }




                            for result in results {

                                context.deleteObject(result as!
NSManagedObject)
                                context.save(nil)

                            }

                        }

                        for category in categories {

                            if let name = category["name"] as? String
{

                                if let id = category["id"] as? String{
                                var newPost: NSManagedObject =
NSEntityDescription.insertNewObjectForEntityForName("Categories",
inManagedObjectContext: context) as! NSManagedObject

                                newPost.setValue(name, forKey:
"category")

                                newPost.setValue(id, forKey: "id")

                                context.save(nil)

                                }

                            }

                        }

                    }

                }

                self.tableView.reloadData()

            }
        }
    })
```

```
        task.resume()

    }
/*Here is our segue to go to the second screen, the subcategory
screen. We pass the core data info to the next screen so that it can
be used there to populate the table with the correct subcategories*/
override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "showSubcategories" {

            if let indexPath =
self.tableView.indexPathForSelectedRow() {
                let object =
self.fetchedResultsController.objectAtIndexPath(indexPath) as!
NSManagedObject
                var destinationViewController =
segue.destinationViewController as! TableViewController1
                destinationViewController.title =
object.valueForKey("category")?.description
                let theObject = object
                let keys =
Array(theObject.entity.attributesByName.keys)
                let dict = theObject.dictionaryWithValuesForKeys(keys)

                destinationViewController.dataPassed = dict
            }

        }
    }
/*This code dictates what the category table will look like, and how
it will get populated from the core data*/
override func numberOfSectionsInTableView(tableView: UITableView) ->
Int {
        return self.fetchedResultsController.sections?.count ?? 0
    }

    override func tableView(tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
        let sectionInfo =
self.fetchedResultsController.sections![section] as!
NSFetchedResultsSectionInfo
        return sectionInfo.numberOfObjects
    }

    override func tableView(tableView: UITableView,
cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("Cell",
forIndexPath: indexPath)as! UITableViewCell

        self.configureCell(cell, atIndexPath: indexPath)

        return cell
    }

    func configureCell(cell: UITableViewCell, atIndexPath indexPath:
NSIndexPath) {
        let object =
self.fetchedResultsController.objectAtIndexPath(indexPath) as!
NSManagedObject
        cell.textLabel!.text =
object.valueForKey("category")!.description
    }

/*This allows core data to be fetched, and it retrieves and sorts the
data as we want it*/
var fetchedResultsController: NSFetchedResultsController {
        if _fetchedResultsController != nil {
            return _fetchedResultsController!
        }

        let fetchRequest = NSFetchRequest()
        // Edit the entity name as appropriate.
        let entity = NSEntityDescription.entityForName("Categories",
inManagedObjectContext: self.managedObjectContext!)
        fetchRequest.entity = entity

        // Set the batch size to a suitable number.
        fetchRequest.fetchBatchSize = 20

        // Edit the sort key as appropriate.
        let sortDescriptor = NSSortDescriptor(key: "category",
ascending: true)
        let sortDescriptors = [sortDescriptor]

        fetchRequest.sortDescriptors = [sortDescriptor]
        // Edit the section name key path and cache name if
appropriate.
        // nil for section name key path means "no sections".
        let aFetchedResultsController =
NSFetchedResultsController(fetchRequest: fetchRequest,
managedObjectContext: self.managedObjectContext!, sectionNameKeyPath:
nil, cacheName: "Master")
        aFetchedResultsController.delegate = self
        _fetchedResultsController = aFetchedResultsController

        var error: NSError? = nil
        if !_fetchedResultsController!.performFetch(&error) {
            // Replace this implementation with code to handle the
error appropriately.
            // abort() causes the application to generate a crash log
and terminate. You should not use this function in a shipping
application, although it may be useful during development.
            //println("Unresolved error \(error), \(error.userInfo)")
            abort()
        }

        return _fetchedResultsController!
    }
    var _fetchedResultsController: NSFetchedResultsController? = nil

    func controllerWillChangeContent(controller:
NSFetchedResultsController) {
        self.tableView.beginUpdates()
    }

    func controller(controller: NSFetchedResultsController,
didChangeSection sectionInfo: NSFetchedResultsSectionInfo, atIndex
sectionIndex: Int, forChangeType type: NSFetchedResultsChangeType) {
        switch type {
        case .Insert:
            self.tableView.insertSections(NSIndexSet(index:
sectionIndex), withRowAnimation: .Fade)
        case .Delete:
            self.tableView.deleteSections(NSIndexSet(index:
sectionIndex), withRowAnimation: .Fade)
        default:
            return
        }
    }

    func controller(controller: NSFetchedResultsController,
didChangeObject anObject: AnyObject, atIndexPath indexPath:
NSIndexPath?, forChangeType type: NSFetchedResultsChangeType,
newIndexPath: NSIndexPath?) {
        switch type {
        case .Insert:
            tableView.insertRowsAtIndexPaths([newIndexPath!],
withRowAnimation: .Fade)
        case .Delete:
            tableView.deleteRowsAtIndexPaths([indexPath!],
withRowAnimation: .Fade)
        case .Update:

self.configureCell(tableView.cellForRowAtIndexPath(indexPath!)!,
atIndexPath: indexPath!)
        case .Move:
            tableView.deleteRowsAtIndexPaths([indexPath!],
withRowAnimation: .Fade)
            tableView.insertRowsAtIndexPaths([newIndexPath!],
withRowAnimation: .Fade)
        default:
            return
        }
    }

    func controllerDidChangeContent(controller:
NSFetchedResultsController) {
        self.tableView.endUpdates()
    }
}

/*necessary variables to receive from the last screen, and pass
location info to the next screen*/
var managedObjectContext: NSManagedObjectContext? = nil

    var dataPassed: NSDictionary!
    var itemsReceived : NSArray = []

    var location: CLLocation!
/*requests location services to use on the next screen*/
```

```
var locationManager: CLLocationManager = {
        let _locationManager = CLLocationManager()
        _locationManager.requestWhenInUseAuthorization()
        return _locationManager
        }()

/*gets location, and then gets the subcategories corresponding to the
passed over category that was clicked, stores them*/
      locationManager.startUpdatingLocation()

        var jsonResult: NSDictionary
        var appDel: AppDelegate =
UIApplication.sharedApplication().delegate as! AppDelegate
        var context: NSManagedObjectContext =
appDel.managedObjectContext!
        let url = NSURL(string:
"http://web.engr.oregonstate.edu/~rousee/CS419/api/?CategoryAll")
        let session = NSURLSession.sharedSession()
        let task = session.dataTaskWithURL(url!, completionHandler: {
(data, response, error) -> Void in

            if error != nil {

                println(error)

            } else {

                let jsonResult =
NSJSONSerialization.JSONObjectWithData(data, options:
NSJSONReadingOptions.MutableContainers, error: nil) as! NSDictionary

                if jsonResult.count > 0 {

                    if let categories = jsonResult["Categories"] as?
NSArray {

                        for category in categories {

                            if let cat = category["id"] as? String {

                                var dat = self.dataPassed["id"] as?
String
```

```
                                if cat == dat {

                                    self.itemsReceived =
category["items"] as! NSArray!
                                }
                            }

                        }

                    }

                    dispatch_async(dispatch_get_main_queue()){

                        self.tableView.reloadData()
                    }
                }
            }

        })

        task.resume()
    }

/*segue that passes the subcategory id and category id to the business
list screen, also sends over the user’s location*/
override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "showBusinessList" {

            if let indexPath =
self.tableView.indexPathForSelectedRow() {

                var destinationViewController =
segue.destinationViewController as! TableViewController2
                destinationViewController.title =
itemsReceived[indexPath.row] as? String

                var categoryId = dataPassed["id"] as! String
                let subCategoryId =
itemsReceived.indexOfObject(itemsReceived[indexPath.row]) + 1
                var subCategoryIdString = String(subCategoryId)

                let location = locationManager.location!
                destinationViewController.currentLocation = location

                destinationViewController.passedCategoryId =
categoryId
                destinationViewController.passedSubCategoryId =
subCategoryIdString

            }

        }
    }

/*populates the subcategory list with info received above*/
 override func numberOfSectionsInTableView(tableView: UITableView) ->
Int {
        // #warning Potentially incomplete method implementation.
        // Return the number of sections.
        return 1
    }

    override func tableView(tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
        // #warning Incomplete method implementation.
        // Return the number of rows in the section.
        return itemsReceived.count
    }


    override func tableView(tableView: UITableView,
cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        let cell =
tableView.dequeueReusableCellWithIdentifier("subcells", forIndexPath:
indexPath) as! UITableViewCell

        cell.textLabel!.text = itemsReceived[indexPath.row] as? String
        cell.accessoryType =
UITableViewCellAccessoryType.DisclosureIndicator

        return cell
    }
/*variables for received info, and info that will be further used on
this screen*/
```

```
var passedCategoryId: String = ""
    var passedSubCategoryId: String = ""

    var businessList : [AnyObject] = []
    var businessAddressList : [AnyObject] = []
    var businessWebsiteList : [AnyObject] = []
    var businessPhoneList : [AnyObject] = []
    var businessZipList : [AnyObject] = []

    var currentLocation: CLLocation!

/*gets info that will be used for this business list table, and that
will passed on to the business detail screen*/
var jsonResult: NSDictionary
            var appDel: AppDelegate =
UIApplication.sharedApplication().delegate as! AppDelegate
            var context: NSManagedObjectContext =
appDel.managedObjectContext!
            let url = NSURL(string:
"http://web.engr.oregonstate.edu/~rousee/CS419/api?BusinessesByCatID=\
(self.passedCategoryId)&SubcategoryID=\(self.passedSubCategoryId)")
            let session = NSURLSession.sharedSession()
            let task = session.dataTaskWithURL(url!,
completionHandler: { (data, response, error) -> Void in

                if error != nil {

                    println(error)

                } else {

                    let jsonResult =
NSJSONSerialization.JSONObjectWithData(data, options:
NSJSONReadingOptions.MutableContainers, error: nil) as! NSDictionary

                    if jsonResult.count > 0 {

                        if let businesses = jsonResult["Businesses"]
as? NSArray {

                            for business in businesses {
                                if let businessName = business["name"]
as? String {

self.businessList.append(businessName)

                                }

                                if let businessAddress =
business["address"] as? String {

self.businessAddressList.append(businessAddress)

                                }
                                if let businessPhone =
business["phone"] as? String {

self.businessPhoneList.append(businessPhone)

                                }
                                if let businessWebsite =
business["url"] as? String {

self.businessWebsiteList.append(businessWebsite)

                                }
                                if let businessZip = business["zip"]
as? String {

self.businessZipList.append(businessZip)

                                }
                            }
                        }

                        dispatch_async(dispatch_get_main_queue()){

                            self.tableView.reloadData()
                        }
                    }
                }

            })

            task.resume()

        }

/*Sends info that we collected on this screen to the business detail
screen for the clicked on business, also sends business info to the
map so pins for them can be created*/
override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "showBusinessDetail" {

            if let indexPath =
self.tableView.indexPathForSelectedRow() {

                var destinationViewController =
segue.destinationViewController as! BusinessTableView
                destinationViewController.title =
businessList[indexPath.row] as? String

                var passedBusinessTitle =
businessList[indexPath.row].description

                var passedPhone =
businessPhoneList[indexPath.row].description
                destinationViewController.receivedPhone = passedPhone

                var passedWebsite =
businessWebsiteList[indexPath.row].description
                destinationViewController.receivedWebsite =
passedWebsite

                var passedZip =
businessZipList[indexPath.row].description
                destinationViewController.receivedZip = passedZip

                destinationViewController.receivedBusinessTitle =
passedBusinessTitle
```

```
                var passedBusinessAddresses = businessAddressList
                destinationViewController.receivedBusinessAddresses =
passedBusinessAddresses
                var passedBusinessAddress =
businessAddressList[indexPath.row].description
                destinationViewController.receivedBusinessAddress =
passedBusinessAddress

            }

        }
        if segue.identifier == "showMap" {

            var destinationViewController =
segue.destinationViewController as! ViewControllerMap
            var passedBusinessAddresses = businessAddressList
            var passedBusinessNames = businessList

            var passedZip = businessZipList
            destinationViewController.receivedZip = passedZip

            destinationViewController.receivedBusinessAddresses =
passedBusinessAddresses
            destinationViewController.receivedBusinessNames =
passedBusinessNames
        }


    }

/*Sets up the business list table. This one is more detailed because
there are images in the cells to number the businesses. There are also
distances calculated for businesses with address. The distance in
miles is calculated between the user’s current location and the
business*/
override func numberOfSectionsInTableView(tableView: UITableView) ->
Int {
        // #warning Potentially incomplete method implementation.
        // Return the number of sections.
        return 1
    }
    override func tableView(tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
        // #warning Incomplete method implementation.
        // Return the number of rows in the section.
        return businessList.count
    }

    override func tableView(tableView: UITableView,
cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {

        let cell =
tableView.dequeueReusableCellWithIdentifier("businessCell",
forIndexPath: indexPath) as! UITableViewCell

        var businessListArray = businessList as NSArray
        var number =
businessListArray.indexOfObject(businessListArray[indexPath.row]) + 1
        var numberString = String(number)
        cell.detailTextLabel?.text = " "
        cell.textLabel!.text = businessList[indexPath.row] as? String
        cell.accessoryType =
UITableViewCellAccessoryType.DisclosureIndicator
        var imageName = UIImage(named: numberString)
        cell.imageView?.image = imageName

        var distanceInMiles: Double
        var distanceInMilesString: String = ""
        let location = "\(businessAddressList[indexPath.row]) +
\(businessZipList[indexPath.row])"

        var geocoder:CLGeocoder = CLGeocoder()
        geocoder.geocodeAddressString(location, completionHandler:
{(placemarks, error) -> Void in

            if((error) != nil){

                println("Error", error)
            }

            else if let placemark = placemarks?[0] as? CLPlacemark {

                dispatch_async(dispatch_get_main_queue()) {
```

```
                var placemark:CLPlacemark = placemarks[0] as!
CLPlacemark
                var coordinates:CLLocationCoordinate2D =
placemark.location.coordinate

                var destination: CLLocation = CLLocation(latitude:
coordinates.latitude, longitude: coordinates.longitude)

                let distance =
destination.distanceFromLocation(self.currentLocation)
                let distanceInMiles = distance * 0.000621371
                var distanceInMilesRounded = round(100.0 *
distanceInMiles) / 100.0
                var distanceInMilesString = "\(distanceInMilesRounded)
miles"

                cell.detailTextLabel?.text = distanceInMilesString

                }
            }

        })

        return cell

    }
  /*Connects our map and other needed variables*/
 @IBOutlet var map: MKMapView!

    var manager: CLLocationManager!
    var managedObjectContext: NSManagedObjectContext? = nil

    var currentAnnotation : MKAnnotation!

    var receivedBusinessAddresses : [AnyObject] = []
    var receivedBusinessNames : [AnyObject] = []
    var receivedZip : [AnyObject] = []

    var businessName: String!

/*Holds our custom pin images*/
  var globalImageArray: [UIImage] = [
        UIImage(named: "pin1.png")!,
        UIImage(named: "pin2.png")!, UIImage(named: "pin3.png")!,
UIImage(named: "pin4.png")!, UIImage(named: "pin5.png")!,
UIImage(named: "pin6.png")!, UIImage(named: "pin7.png")!,
UIImage(named: "pin8.png")!, UIImage(named: "pin9.png")!,
UIImage(named: "pin10.png")!,  UIImage(named: "pin11.png")!,
UIImage(named: "pin12.png")!,  UIImage(named: "pin13.png")!,
UIImage(named: "pin14.png")!,  UIImage(named: "pin15.png")!,
UIImage(named: "pin16.png")!,  UIImage(named: "pin17.png")!,
UIImage(named: "pin18.png")!,  UIImage(named: "pin19.png")!,
UIImage(named: "pin20.png")!,  UIImage(named: "pin21.png")!,
UIImage(named: "pin22.png")!, UIImage(named: "pin23.png")!,
UIImage(named: "pin24.png")!, UIImage(named: "pin25.png")!,
UIImage(named: "pin26.png")!, UIImage(named: "pin27.png")!,
UIImage(named: "pin28.png")!, UIImage(named: "pin29.png")!,
UIImage(named: "pin30.png")!, UIImage(named: "pin31.png")!,
UIImage(named: "pin32.png")!, UIImage(named: "pin33.png")!,
UIImage(named: "pin34.png")!, UIImage(named: "pin35.png")!,
UIImage(named: "pin36.png")!, UIImage(named: "pin37.png")!,
UIImage(named: "pin38.png")!, UIImage(named: "pin39.png")!,
UIImage(named: "pin40.png")!
    ]

/*Starts getting user’s location, and creates pins for each business
with an address, adds the custom annotation for each one*/
view.addSubview(map)

        manager = CLLocationManager()
        manager.delegate = self
        manager.desiredAccuracy = kCLLocationAccuracyBest
        manager.requestWhenInUseAuthorization()
        manager.startUpdatingLocation()

        for var i = 0; i < receivedBusinessAddresses.count; i++ {
            let location = "\(receivedBusinessAddresses[i]) +
\(receivedZip[i])"
            let title = receivedBusinessNames[i] as! String
            let subtitle = ""
            var geocoder:CLGeocoder = CLGeocoder()
            var imageName = globalImageArray[i]
            geocoder.geocodeAddressString(location, completionHandler:
{(placemarks, error) -> Void in
```

```
                if((error) != nil){

                    println("Error", error)
                }

                else if let placemark = placemarks?[0] as? CLPlacemark
{

                    var placemark:CLPlacemark = placemarks[0] as!
CLPlacemark
                    var coordinate:CLLocationCoordinate2D =
placemark.location.coordinate

                    var annotation = CustomAnnotation(coordinate:
coordinate, title: title, subtitle: subtitle)
                    annotation.image = imageName
                    self.map.addAnnotation(annotation)

                    var destination: CLLocation = CLLocation(latitude:
coordinate.latitude, longitude: coordinate.longitude)

                    var locManager = CLLocationManager()
                    locManager.requestWhenInUseAuthorization()

                    var currentLocation = CLLocation()
                    currentLocation = locManager.location
                    let distance =
destination.distanceFromLocation(currentLocation)
                    let distanceInMiles = distance * 0.000621371

                }

            })

        }

        // Do any additional setup after loading the view.
    }

/*Our location manager to update locations, and show user’s lcoation*/
func locationManager(manager: CLLocationManager!, didUpdateLocations
locations: [AnyObject]!) {
        var userLocation: CLLocation = locations[0] as! CLLocation
        var latitude = userLocation.coordinate.latitude
        var longitude = userLocation.coordinate.longitude
        var coordinate = CLLocationCoordinate2DMake(latitude,
longitude)


        self.map.showsUserLocation = true

    }

   /*centers map around user’s location intially*/
    func mapView(mapView: MKMapView!, didUpdateUserLocation
        userLocation: MKUserLocation!) {
            let userLocation = mapView.userLocation

            let region = MKCoordinateRegionMakeWithDistance(
                userLocation.location.coordinate, 2000, 2000)

            mapView.setRegion(region, animated: true)
    }

 /*connects the detail disclosure button to a specific segue*/
    func mapView(mapView: MKMapView!, annotationView view:
MKAnnotationView!, calloutAccessoryControlTapped control: UIControl!)
{
        if control == view.rightCalloutAccessoryView {

            businessName = view.annotation.title

            performSegueWithIdentifier("showBusinessButton", sender:
self)

        }
    }
/*connects the detail disclosure to a new screen, provides that screen
with the business name of the tapped annotation*/
    override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "showBusinessButton" {

            var destination = segue.destinationViewController as!
MapBusinessTableView
```

```
            destination.receivedBusinessTitle = businessName
            destination.title = businessName
    }
    }
/*sets up each custom annotation*/
    func mapView(mapView: MKMapView!, viewForAnnotation annotation:
MKAnnotation!) -> MKAnnotationView! {
        if !(annotation is CustomAnnotation) {
            return nil
        }

        let reuseId = "business"
        var anView =
mapView.dequeueReusableAnnotationViewWithIdentifier(reuseId)
        if anView == nil {
            anView = MKAnnotationView(annotation: annotation,
reuseIdentifier: reuseId)
            anView.canShowCallout = true

        }
        else {
            anView.annotation = annotation
        }

        var button =
UIButton.buttonWithType(UIButtonType.DetailDisclosure) as! UIButton
        anView.rightCalloutAccessoryView = button

        let customAnnotation = annotation as! CustomAnnotation

        if (customAnnotation.image != nil) {
            anView.image = customAnnotation.image!

        }
        else {

            anView.image = UIImage(named: "defaultpin.png")
            //set default image
        }

        return anView
    }
```

```
//We have two business details screens that both do the exact same
//thing. One is accessed by tapping a business list name, the other by
//tapping the info button on the map.

/*Sets up variables and labels for the business details screens*/
var managedObjectContext: NSManagedObjectContext? = nil

    var receivedBusinessAddresses : [AnyObject] = []
    var receivedBusinessTitle: String = ""
    var receivedBusinessAddress: String = ""
    var receivedZip: String = ""
    var receivedPhone: String = ""
    var receivedWebsite: String = ""

    var coords: CLLocationCoordinate2D?


    @IBOutlet var directionsLabel: UIButton!
    @IBOutlet var businessNameLabel: UILabel!
    @IBOutlet var businessAddressLabel: UILabel!
    @IBOutlet var businessPhoneLabel: UILabel!
    @IBOutlet var businessHoursLabel: UILabel!
    @IBOutlet var website: UIButton!
    @IBOutlet var phone: UIButton!
    @IBOutlet var errorLabel: UIButton!

/*Connects the phone numbers button to call the numbers, connects the
website link to open in Safari*/
 @IBAction func websiteButton(sender: AnyObject) {
        if let url = NSURL(string: "http://\(receivedWebsite)") {
            UIApplication.sharedApplication().openURL(url)
        }
    }
    @IBAction func phoneButton(sender: AnyObject) {
        if let url = NSURL(string: "tel://\(receivedPhone)") {
            UIApplication.sharedApplication().openURL(url)
        }
    }

/*Launches the map app when the directions button is pressed to give
direcitons from the user's current location to the business*/
@IBAction func directionsPressed(sender: AnyObject) {
        let geoCoder = CLGeocoder()
        println("address string: \(receivedBusinessAddress)
\(receivedZip)")
        let addressString = "\(receivedBusinessAddress)
\(receivedZip)"

        geoCoder.geocodeAddressString(addressString,
completionHandler:
            {(placemarks: [AnyObject]!, error: NSError!) in

                if error != nil {
                    //self.errorLabel.setTitle = "Address"
                    self.directionsLabel.setTitle("Directions
Unavailable", forState: UIControlState.Normal)

                } else if placemarks.count > 0 {
                    let placemark = placemarks[0] as! CLPlacemark
                    let location = placemark.location

                    self.coords = location.coordinate
                    let addressDict =
                    [kABPersonAddressStreetKey as NSString:
self.receivedBusinessAddress,
                        kABPersonAddressZIPKey: self.receivedZip
                    ]
                    let place = MKPlacemark(coordinate: self.coords!,
addressDictionary: addressDict)
                    let mapItem = MKMapItem(placemark: place)
                    println("map item: \(mapItem)")
                    let options = [MKLaunchOptionsDirectionsModeKey:
                    MKLaunchOptionsDirectionsModeDriving]

                    mapItem.openInMapsWithLaunchOptions(options)

                }
        })
    }

/*Gets all the necessary business info to set the labels and text of
the contact screen*/
var jsonResult: NSDictionary
        var appDel: AppDelegate =
UIApplication.sharedApplication().delegate as! AppDelegate
```

```
        var context: NSManagedObjectContext =
appDel.managedObjectContext!
        let url = NSURL(string:
"http://web.engr.oregonstate.edu/~rousee/CS419/api?BusinessAll")
        let session = NSURLSession.sharedSession()
        let task = session.dataTaskWithURL(url!, completionHandler: {
(data, response, error) -> Void in

            if error != nil {

                println(error)

            } else {

                let jsonResult =
NSJSONSerialization.JSONObjectWithData(data, options:
NSJSONReadingOptions.MutableContainers, error: nil) as! NSDictionary

                if jsonResult.count > 0 {

                    if let businesses = jsonResult["Businesses"] as?
NSArray {

                        for business in businesses {
                            if let businessName = business["name"] as?
String {
                                if let businessAddress =
business["address"] as? String {
                                    if let businessHours =
business["hours"] as? String {

                                        if businessName ==
self.receivedBusinessTitle {


dispatch_async(dispatch_get_main_queue()) {
                                                if businessName != ""
{

self.businessNameLabel.text = self.receivedBusinessTitle
                                                } else {

self.businessNameLabel.text = "None Given"
```

```
                                                        }
                                                        if businessAddress !=
"" {

self.businessAddressLabel.text = self.receivedBusinessAddress
                                                        } else {

self.businessAddressLabel.text = "None Given"
                                                        }
                                                        if businessHours != ""
{

self.businessHoursLabel.text = businessHours
                                                        } else {

self.businessHoursLabel.text = "None Given"
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }

                                }
                            }

                        }
                    }

        })

        task.resume()

        if receivedPhone != "" {
            self.phone.setTitle(self.receivedPhone, forState:
UIControlState.Normal)
        } else {

            self.phone.setTitle("None Given", forState:
UIControlState.Normal)
        }
        if receivedWebsite != "" {
            self.website.setTitle(self.receivedWebsite, forState:
UIControlState.Normal)
        } else {
            self.website.setTitle("None Given", forState:
UIControlState.Normal)
        }

        // Do any additional setup after loading the view.
    }


}
```