IMPLEMENTATION OF IOT BASED FACULTY TRACKING SYSTEM

MS Project under Dr. Anu Bourgeois

Abstract

An IoT system built using Raspberry Pi to ping Mobile Devices to update presence information, making the information accessible from a Web App, Mobile App, and AI based Voice Assistance

Dhruv Gupta

Dgupta3@student.gsu.edu / ddhruvgupta@gmail.com

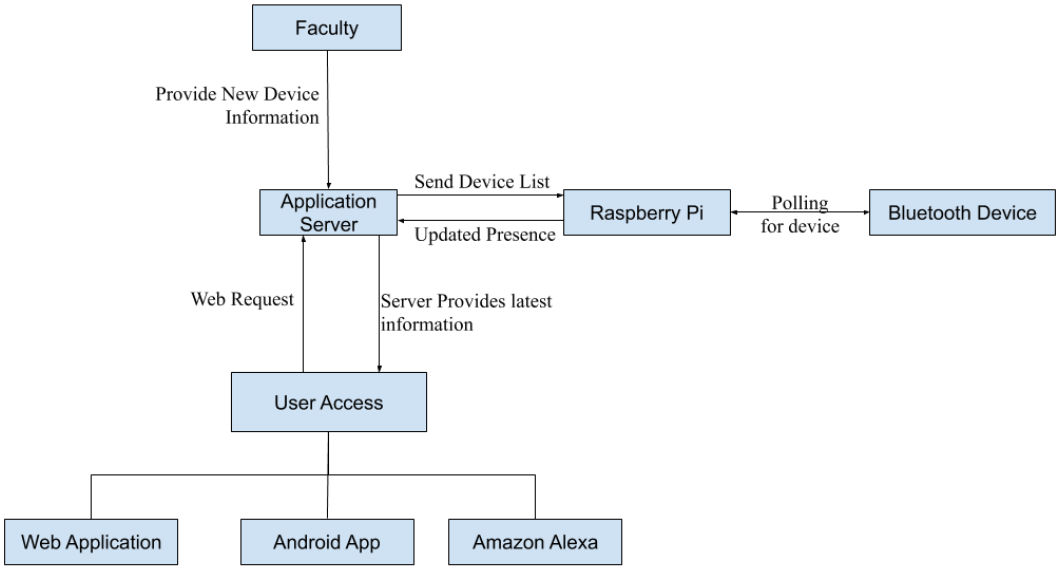
I. INTRODUCTION

The Computer Science department along with other schools in the university at Georgia State share the same issue, where students walk in to the department to either ask a student assistant about the availability of a particular professor or go and check their offices to find out if a faculty member is available. In some instances, students will make the trip from home to come to campus only to find out that a faculty member is not in their office today.

To make students accessibility to educators easier and promote the open door policy that some instructors have, an online application was built which contains an Application Programming Interface (API) which will return the availability of professors registered in the system. Three solutions are proposed to make this information accessible to users:

- Web Application
- Amazon Alexa
- Android Application

This document further details the implementation details of each of these approaches. As well as the implementation details of the IoT infrastructure required to support it and deployment to Amazon Web Services (AWS) via a Continuous Integration Continuous Deployment Pipeline (CI/CD). This document is organized based on the different parts of the application.



II. IMPLEMENTATION

A. Raspberry Pi

Hardware

For the Scope of this project a Raspberry Pi 3 Model B was used. The model requires a Micro SD card for the operating system and secondary storage (Hard Drive), as well as a Micro USB Power Adapter (2.1 A). For the initial set a keyboard and mouse along with a screen are also required. The hardware specification for this mode can be found at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

This device was chosen because of the hardware extensibility provided by the GPIO for future extensions, on board support for Wi-Fi and Ethernet, a Linux based Operating System capable of running headless and where users can SSH into the machine for remote troubleshooting.

Software Requirement

The Operating system being used is Raspbian. There are several version of this Operating System, available, the one used for this implementation was stretch. The OS is optimized to run on the low performance ARM CPU of the Pi. The instructions for installing Raspbian can be found here: https://www.raspberrypi.org/documentation/installation/installing-images/. The method used for this project was based on Windows using BalenaEtcher.

```
pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
```

Figure 1: Raspbian OS Version

```
pi@raspberrypi:~ $ python3 --version
Python 3.5.3
```

Figure 2: Python Version

Solution Architecture

The Raspberry Pi acts as the sensor which will connect to the Bluetooth devices. Since the Raspberry Pi is small, affordable and runs on a Linux operating system distribution called Raspbian, it can be configured to provide high reliability and availability as a distributed sensor bed. The Pi runs a python script at a configured frequency to ping all the devices in its device list and updates their presence in a hash table. The ping is performed thrice every cycle, with the mode of the results being taken as the result to provide accurate results.

There are 2 python modules used for this purpose. The first one, PingTest.py, is responsible for receiving a list of MAC addresses, pinging each device and collating a dictionary of results which is finally sent back to the Web Application. The second module, client_mt.py, was originally a designed as a socket to socket communication module talking to a server-side module to receive the list of MAC addresses to ping and sending the results back, handling all the communications outside of the Raspberry Pi. This module was refactored to communicate with an API instead of using socket based communication. Both modules are easily accessible via the GitHub page for this project.

```
 GNU nano 2.7.4

#@Author: Dhruv Gupta

import os
import sys
import time
import bluetooth
import client_mt

bt_addr = list()
bt_addr= ['48:A9:1C:E7:9A:22']
while True:
    #list of bluetooth addresses to check should be updated with a list from a server every so often


    #dictionary to keep track of status of devices; this should be returned to the server as the result
    status = dict()

    for bt in  bt_addr:
        count = 0
        present = 0
        while count<3:
            count+=1
            result = os.popen('sudo l2ping -s 1 -c 1 ' + bt).read()
            print(result)
            if (result != ''):
                present+=1

        if (present>0):
            status[bt] = 1
        else:
            status[bt] = 0

    print (status)
    bt_addr = client_mt.send(status)
    print("Updated address list: "+ str(bt_addr) )
    time.sleep(60)
```

Figure 3: Module 1, responsible for Bluetooth ping

```
import socket
import json
from requests import Session


def send(inp):
        session = Session()


        val = json.dumps(inp)
        response_raw = session.post(
                url='http://bluetooth-env-test.eba-brqgvwur.us-east-2.elasticbeanstalk.com/WebApp/httpTest.php',
                data={'data': val }
                )
        response = response_raw.content
        response_decode = response.decode()
        response_parsed = json.loads(response_decode)
        print(response_parsed)
        bt_addr = response_parsed['data']
        #print('From server:', modifiedSentence.decode())
        return bt_addr
```

Figure 4: Second Module to communicate with the API

Suggested Deployment

1. To deploy the application, download the code on the Raspberry Pi from:

   https://github.com/ddhruvgupta/Bluetooth/tree/master/Raspberry%20Pi%20Code

2. cd into the directory that the code was downloaded into
3. run the script using python3 pingTest.py
4. Once the script is running, it can be sent to the background using:
    a. >> ctrl+Z;
    b. >> bg
5. To leave the process running and close the connection: ctrl A + ctrl D

B. Web Application

The application is built in PHP with JavaScript and jQuery. The application uses a CDN distribution of jQuery and thus does not require deployment.

Localhost

For the backend, MAMPP can be used on MAC and XAMPP on Windows Machines. XXAMP contains an installation of Apache and mySQL.

Database setup:

After installing XXAMP and starting up the mySQL instance, run the following commands in order in command prompt:

1) mySQL (for this to work, the mySQL bin path needs to be in the environment variables)
2) CREATE DATABASE networking_project;
3) Use networking_project;
4) Take the SQL.DDL file, copy the commands in the run all of them in the SQL instance

Web Application:

1) Copy the files in the GitHub Repo under WebApp and set them up under the htdocs folder.
2) When Apache is running, open a browser and navigate to the directory under the htdocs folder:
    a. C:\xampp\htdocs\project (Windows Explorer Path)
    b. http://localhost/project (Browser URL)
3) This should take you to the login page. The default password setup is Welcome1! For the admin@gsu.edu account.
4) Create GMAIL Account, enable SMTP applications in the settings
    i. Configure the email address and password in the emailVerification.php file under the utils folder

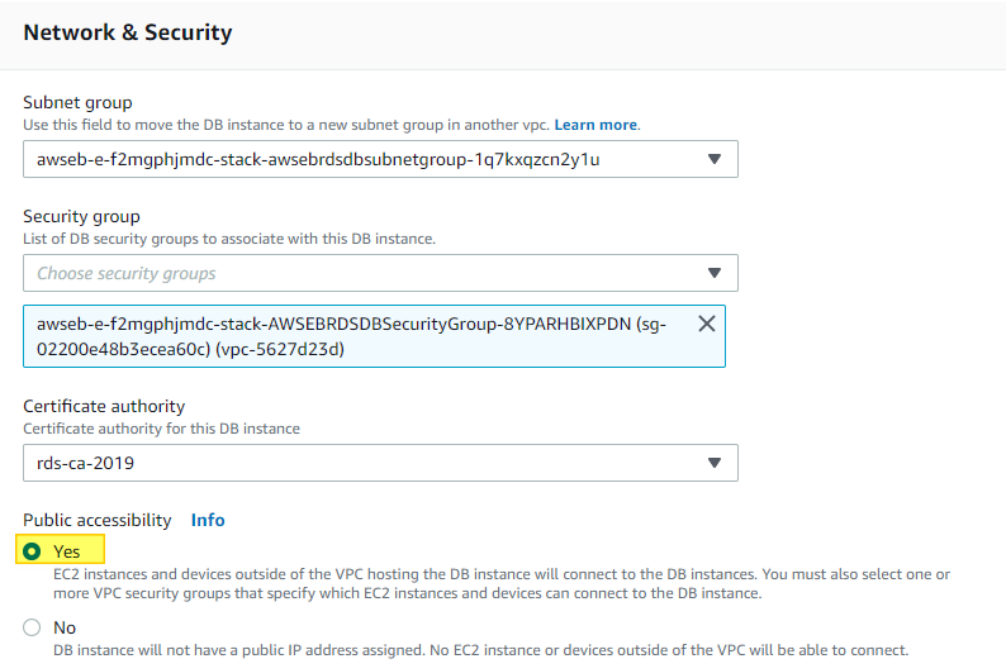
Amazon Web Services:

1) Create AWS Account
2) Create Elastic Beanstalk Instance for PHP
3) Create GMAIL Account, enable SMTP applications in the settings

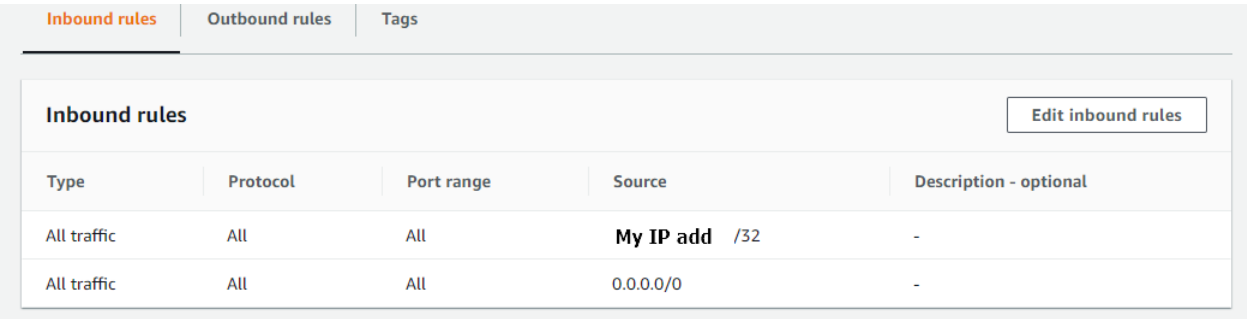
a. Configure the email address and password in the emailVerification.php file under the utils folder

4) Zip the project directory and upload into Elastic Beanstalk
5) In the network settings:

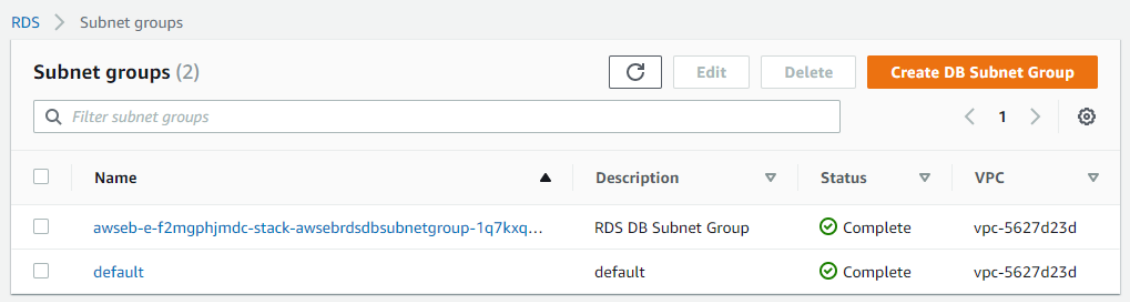
Update Public Accessibility to be able to login to the database:



Update Inbound and Outbound rules:



Subnet Settings:



Subnet group details

VPC ID
VPC-1 (vpc-5627d23d)

ARN
arn:aws:rds:us-east-2:132738084964:subgrp:awseb-e-f2mgphjmdc-stack-awsebrdsdbsubnetgroup-1q7kxqzcn2y1u

Description
RDS DB Subnet Group

Subnets (3)

Availability zone	Subnet ID	CIDR block
us-east-2c	subnet-f5c87ab9	172.31.32.0/20
us-east-2a	subnet-2b9f8243	172.31.0.0/20
us-east-2b	subnet-e0e1aa9a	172.31.16.0/20

After connecting to the database using mySQL workbench or from the command line, run the same steps as the steps under the localhost version.

Deployment via CI/CD Pipeline:

https://aws.amazon.com/getting-started/tutorials/continuous-deployment-pipeline/

C. Web API

There are 2 Application Programming interfaces used in the project and both are part of the web application deployed in the last application. Any Updates to these requires deployment of the web application. Please follow the steps in B for that.

The easiest way to make updates to the APIs is using the CI/CD pipeline. The two APIs are:

- Search API: Serves requests from a mobile application and Alexa devices

- Update API: API for Raspberry Pi devices to update information about devices and get an updated list of devices

The APIs can be tested with Postman as shown in Figs. 5 and 6

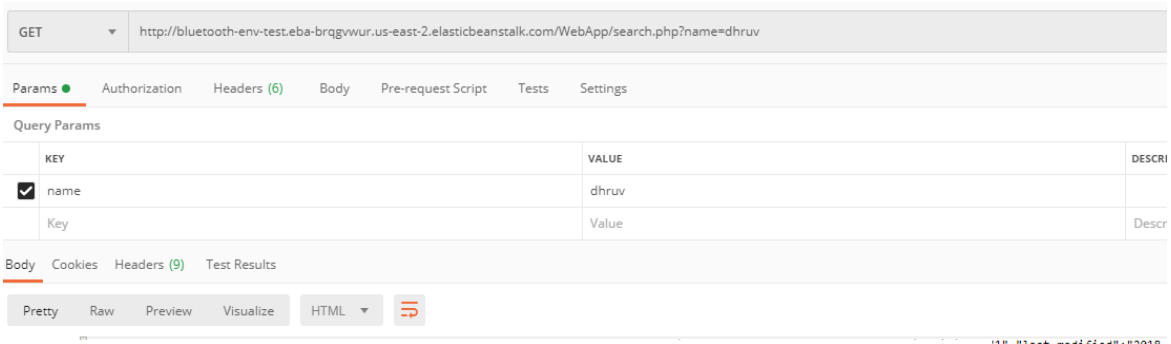


Figure 5: Postman test for search API

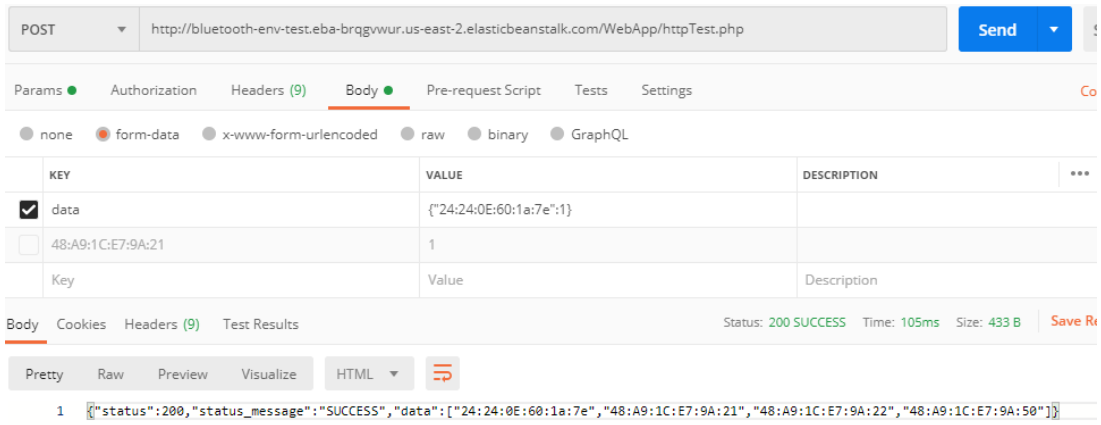
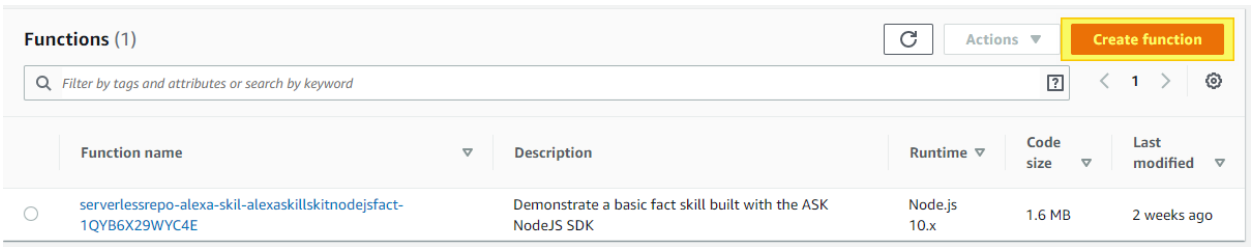


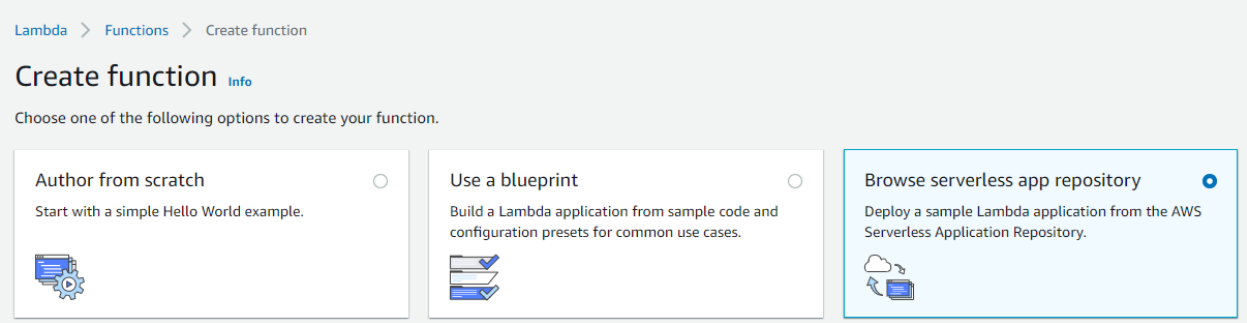
Figure 6: Postman test for httpTest API

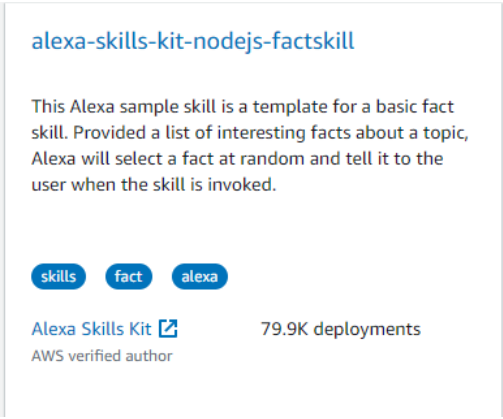
D. Amazon Alexa

The Alexa skill is also built out and available in the GitHub repository, however the steps to create a new skill are detailed here.

To create an Amazon Alexa skill, navigate to Amazon Lambda from the AWS Console and create function.



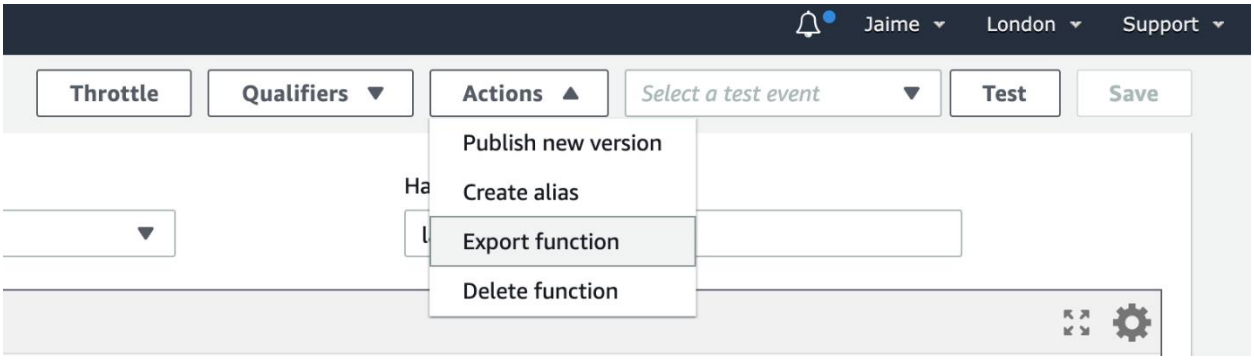


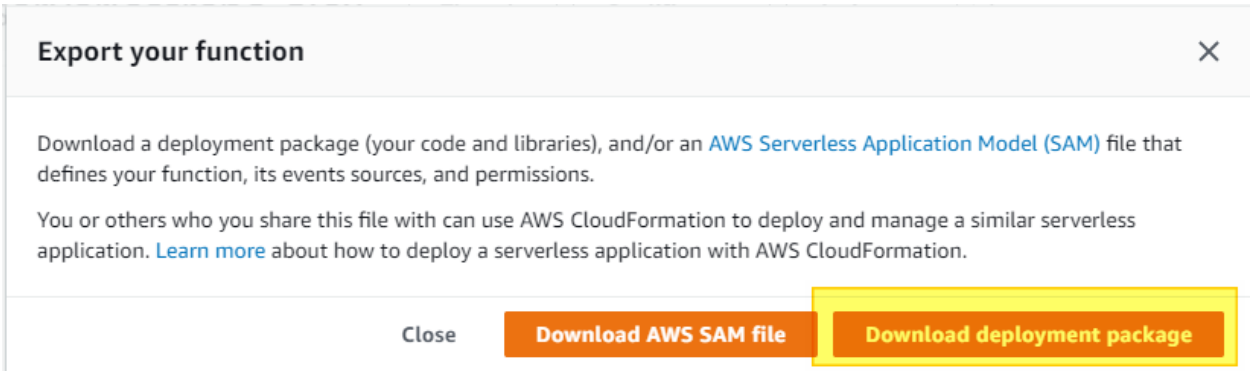


Installing dependencies:

Since this module is making an http request from an API it requires the requests module. To import the module into the lambda environment, the following steps can be followed:

Download your function as a zip, doing click on export function and selecting "Download deployment package":

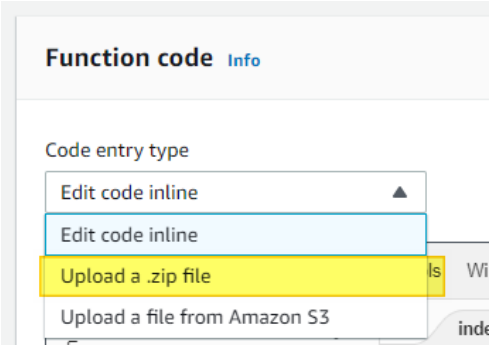




This will download a zip to your local environment, unzip it and inside the folder, open the terminal and install your dependency:

```
npm i request
```

Then the zip can be uploaded again:



Another option is to use the AWS-CLI:

```
aws lambda update-function-code --function-name=your_function_name --zip-file fileb://your_zip_file_name.zip
```

The dependencies can also be added to a lambda layer. However that method was not used in this implementation. To read more about it, this freecodecamp tutorial can be followed.

E. Android Application

An Android based application is built to take advantage of the API so that students have access to this information on their mobile devices and easier access to faculty members. With the information that the application provides, users can make an informed decision about visiting a member of the faculty.

Android Studio

Android studio is the IDE that is provided by Google as a part of the Android SDK. It contains an in-built virtual device manager to simulating code at run time as well as feature to design a front end for the application, back end to process user interactions, troubleshooting support.

Device Emulator

The in-built device emulator in android studio has 2 types of processors to choose from ARM based processors and x86. The x86 emulators are relative well performing however they are constrained to operate only on intel machines. ARM processors have very bad performance. An alternative to using the built in module is using NOX player.

NOX player supports emulating both phone screens and tablets and offers very good performance while running on both Intel and AMD based processors.

Application Life Cycle

Just like an application on a computer operating system, the applications are represented as processes in the android operating system. These processes have a set number of states that they will cycle through over the lifetime of the application's usage.

They include:

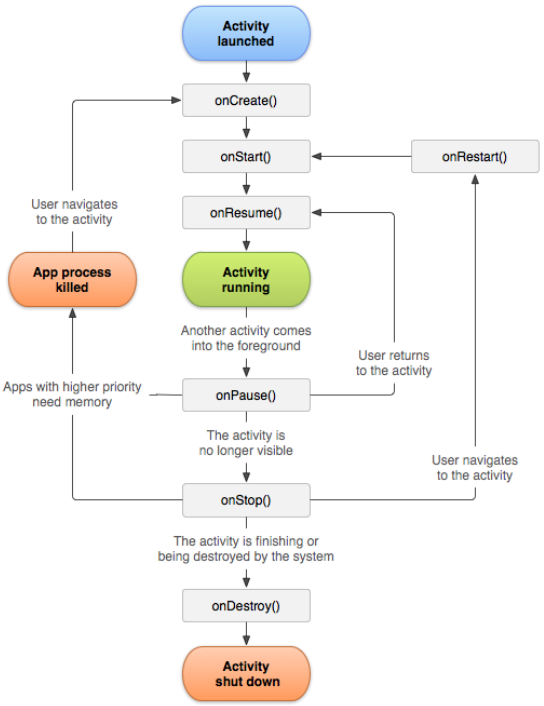
onCreate() – Initial method to setup an Activity

onDestory() – Method to release memory and resources back to the OS

onResume() – Method that is called when an activity is visible in the foreground and read to interact with the user

onPause() – Method that saves current process / activity data, stops CPU intensive tasks when the user switches applications.

onRestart() – Loads the data saved by onPause() when the user switches back to an application that was previously in use



Permissions / Manifest File

An XML file that is required by all Android applications. The AndroidManifest.xml file needs to be present in the root directory of the application and defines the resources and permissions that the application needs to run. It is also used by the system to obtain administrative and organizational information about the application.

Network Access

Connectivity Manager allows applications to check the status of the different access technologies and it informs applications via broadcasted intents about connectivity changers.

Android provides a wide range of packages and classes including:

1. Java.net. * - Standard Java network classes like sockets and simple HTTP and plain packets
2. Android.net. * - Extended java.net capabilities
3. Android.net.http. * - SSL certificate handline
4. Org.apache. * - Specialized HTTP
5. Android.net.wifi. *– WiFi configuration and status

Asynchronous Requests

When a taskHandler is used, the taskHandler needs to return a response to the main thread within a few miliseconds to avoid the application not responding. The main thread is responsible for doing multiple tasks including redrawing the UI every 16ms. Doing a lot of long running computation in the main thread will delay the main thread's normal duties.

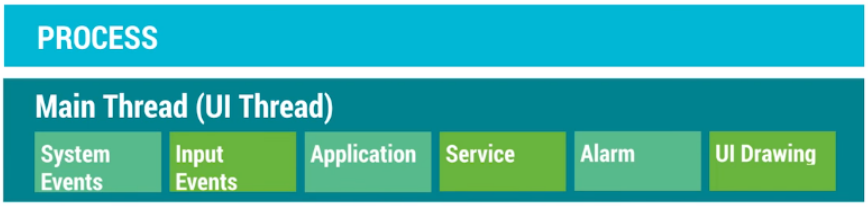


Figure 7: Some of the most common activities performed by the main thread of an android app

In a situation where such a response cannot be guaranteed such as a network call as used in this project a separate thread, known as a worker thread, is used. This worker thread should not call UI elements and should notify the main thread by posting a message to the queue of the main thread.

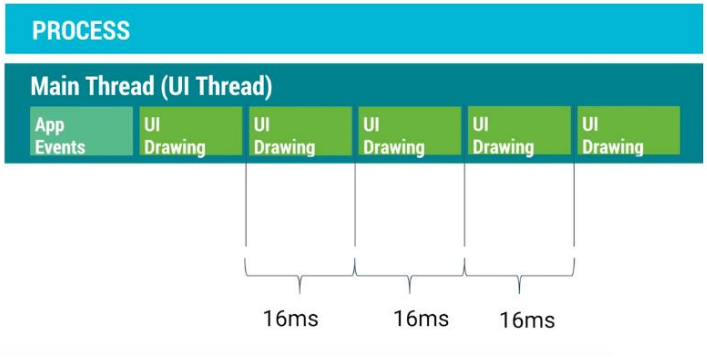


Figure 8: Screen drawn every 16ms, this happens while the user input is still being accepted

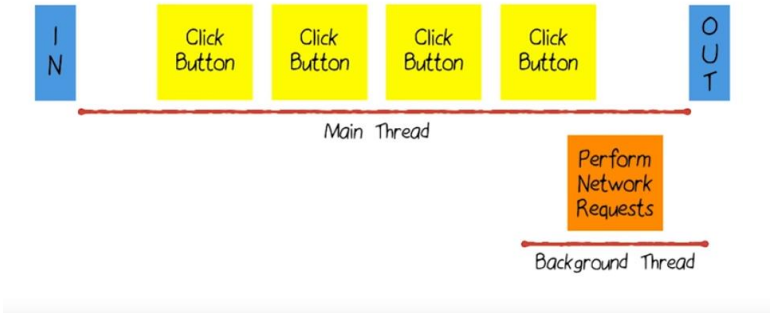


Figure 9: Illustration of the AsyncTask

The best practices for handling this situation as laid out in [4]:

1. Create a handler in the main thread while responding to the menu item. Keep it aside.
2. Create a separate thread (a worker thread) that does the actual work.
    a. Pass the handler from step 1 to the worker thread. This handler allows the worker thread to communicate with the main thread.
3. The worker thread code can now do the actual work for longer than five seconds and, while doing it, can call the handler to send status messages to communicate with the main thread.
4. These status messages now get processed by the main thread, because the handler belonged to the main thread. The main thread can process these messages while the worker thread is doing its work.

An asynchronous task performs computation and other blocking operations on a background worker thread and whose result is published on the UI thread [5]. AsyncTasks should ideally be used for short operations.

An asynchronous task is defined by 3 generic types: Params, Progress and Result, and built on 4 stages:

- OnPreExecute
- doInBackground
- onProgressUpdate
- onPostExecute

- URLConnection

Based on the official android SDK documentation [5]:

The abstract class URLConnection is the superclass of all classes that represent a communications link between the application and a URL. Instances of this class can be used both to read from and to write to the resource referenced by the URL. In general, creating a connection to a URL is a multistep process:

1. The connection object is created by invoking the openConnection method on a URL.
2. The setup parameters and general request properties are manipulated.
3. The actual connection to the remote object is made, using the connect method.
4. The remote object becomes available. The header fields and the contents of the remote object can be accessed.

- Consuming HTTP Services

Android SDK contains a modified version of Apache HttpClient which is a comprehensive HTTP client.

HTTP GET Request pattern [4]:

1. Create an HttpClient (or get an existing reference).

2. Instantiate a new HTTP method, such as PostMethod or GetMethod.
3. Set HTTP parameter names/values.
4. Execute the HTTP call using the HttpClient.
5. Process the HTTP response.

Application Design

UI design

The user-interface makes use of the constrained layout offered by Android Studio and the SDK. Its built using multiple different view components:

- ImageView
- editText View
- Button
- 2 Text Views

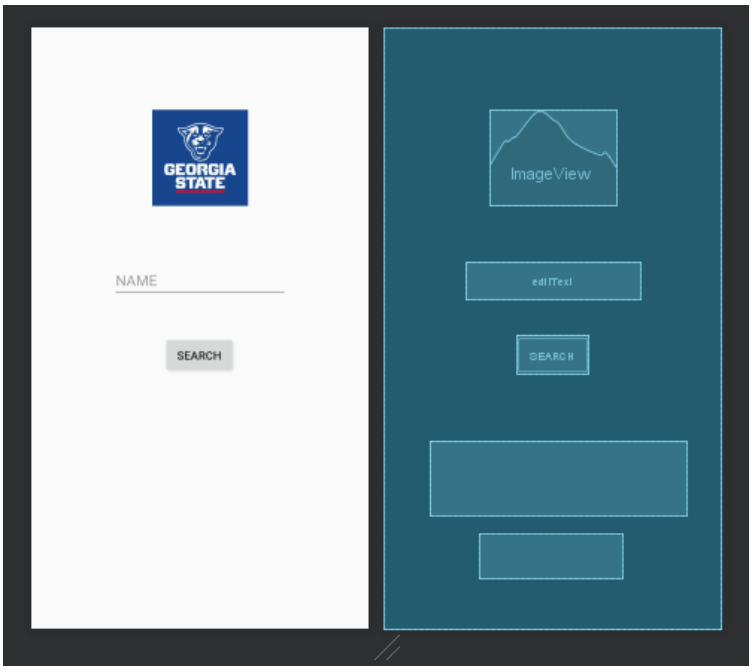


Figure 10: Design of Android App UI

Functionality

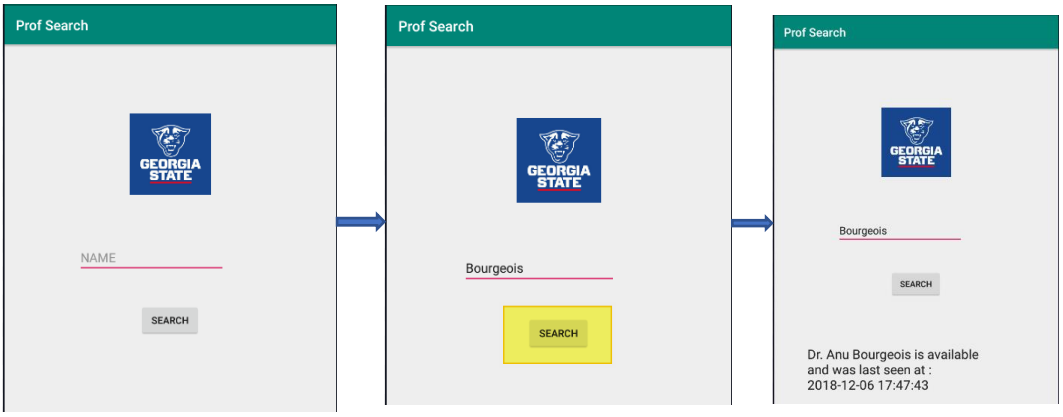


Figure 11: User performing a successful search

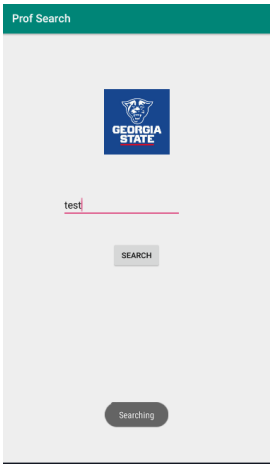


Figure 12: Toast Message Alerts user that search is in progress

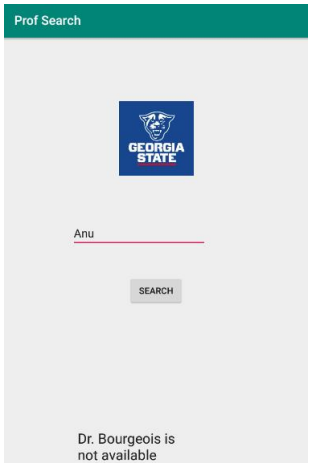


Figure 13: Unsuccessful search result displayed

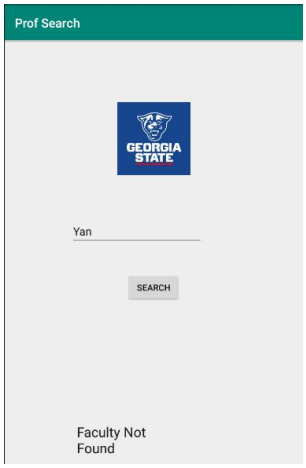


Figure 14: Message when faculty is not registered in the system

Process Flow:

1. User enter the application and the onCreate() method is called
    a. OnCreate() method displays the content that is setup in the activity_main xml file and displays it
    b. The app waits for user input
2. User enters the name of a faculty member they want to search for and clicks SEARCH
    a. The user input is identified based on the id assigned to the textView box

```
searchString = findViewById(R.id.editText);
result = findViewById(R.id.result);
result2 = findViewById(R.id.result2);
search = findViewById(R.id.search_button);
```

3. The Users click is captured by a clickHandler
    a. App logs the click if app is running in debug mode
    b. TOAST message appears alerting the user of the search in progress
    c. Asynchronous task is created and fired (Uses Android.os.AsyncTask)

```
public void clickHandler(View view){
   String str = searchString.getText().toString();
   Log.d("tag","someone pushed button");
   Toast.makeText(this, "Searching", Toast.LENGTH_SHORT).show();
   SearchAsyncTask task = new SearchAsyncTask();
   task.execute();
}
```

4. AsyncTask has a method called doInBackground which needs to be overridden. The overridden method will execute its contents in the background thread
    a. Background thread formats new URL with the search string (Uses the Java.net library)
    b. HTTP request is sent to the URL (Uses HttpURLConnection in the java.net lib)
5. Application points an input stream to the urlConnection and receives the response from the API
6. Response is parsed to JSON format (using libraries in org.json)
7. JSON data structure is parsed and an object of the developer defined data class is populated
8. The AsyncTask is finished and thus calls onPostExecute() method
    a. Return control to main thread and updates UI with the updated information

```
@Override
protected void onPostExecute(data faculty) {
    updateUi(faculty);
}

private void updateUi(data faculty) {
    // Display the earthquake title in the UI
    result.setText("");
    result2.setText("");

    if (faculty != null){
        int ans = faculty.availability;
        if (ans == 1)
            result.setText("Dr. "+ faculty.fname + " "+faculty.lname+" is
available and was last seen at :" + faculty.timeStamp);
        else
            result2.setText("Dr. "+ faculty.lname+" is not available");
```

```
    }else{
        result2.setText("Faculty Not Found");
    }


}
```

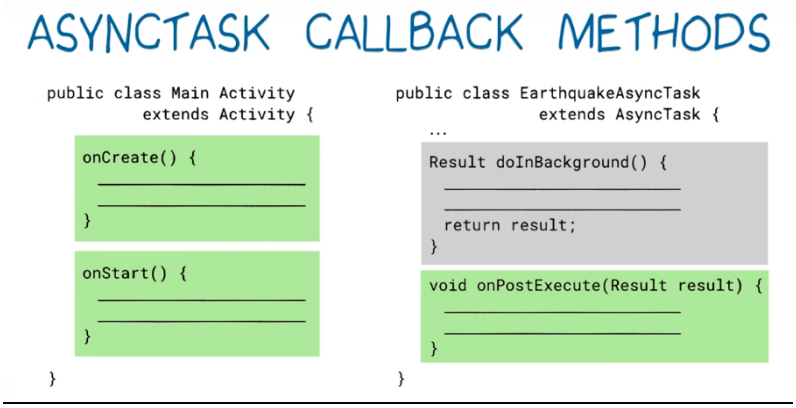


Figure 15: Structure of AsyncTask Call

III. FUTURE WORK

A set of suggested projects based on this project for other under graduates and graduate students could be as follows:

Under Graduates:

- Build an Alexa based Client using available APIs
  - Good OS background knowledge required
  - Eg. Ask Alexa to find Campus Dining Options
- Build an Android Application
  - Requires background in OS and Mobile Apps
  - Eg. Building an application to trigger home automation systems
- Raspberry Pi Application for sensing
  - Python and other tools easily available on Linux, Great community support
  - Good introduction to Linux OS, Networking and working in resources constraints
  - Sensors can interface with GPIO pins
  - Eg. Home Automation Solutions, Health Tracking, Controlling Media Devices
- Database Concepts
  - Explore how to store real time data and database scaling concepts
  - Good introduction for indexing and database partitioning concepts

Graduate Students:

- Add OAuth 2.0 to API
- Implement SSL communication between all components
- Implement Batch Processing Queuing system to enable scaling of Sensor Network
- Implement Caching mechanism for preventing too many calls to database
- Database Sharding based on department
- Mobile application to enable professors to update availability or via Alexa
- Appointment making ability via Alexa

IV. REFERENCES

[1] R. Love, Linux Kernel Development, 3rd ed. Addison Wesley, ISBN 0672329468, 2010

[2] M.E. Russinovich, D.A. Solomon, A.Ionescu, Windows Internals, 5th ed. Microsoft Press, ISBN 0735625301, 2009

[3] Brahler, Stefan. "Analysis of the android architecture." *Karlsruhe institute for technology* 7.8 (2010).

[4] MacLean, Dave, Satya Komatineni, and Grant Allen. "Advanced AsyncTask and Progress Dialogs." *Pro Android 5*. Apress, Berkeley, CA, 2015. 317-341.

[5] https://developer.android.com/reference/java/net/URLConnection