DESIGN DOCUMENT
CS 461 CAPSTONE - FALL TERM

DECEMBER 4, 2018

ECOLOGICAL FOOTPRINT APP

PREPARED FOR

OREGON STATE UNIVERSITY SUSTAINABILITY DOUBLE DEGREE

ANN SCHEERER

PREPARED BY

GROUP 77
ECOLOGICAL FOOTPRINT TEAM

ROHAN BARVE
SATHYA RAMANATHAN
DOMINIC WASKO

Abstract

This design document serves to provide information on the development of a mobile application focused on spreading awareness of individual impact on the environment. We outline the various functions, technologies, processes, and design decisions required for the application to proceed with development. The deployment and management of this software will be left for our project sponsor to decide after development has concluded.

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

For our Senior Capstone project, we decided to work with the sustainability department here at Oregon State University to create a greener solution for the community. We have agreed with our client on the basis of creating an ecological footprint calculator specifically for the city of Corvallis. An ecological footprint is essentially a measurement of land area that is required to sustain a given population. Through this calculator, we can measure individually how much each person consumes the overall available land/water resources. In this document, we will be focusing on the design overview of our project.

1.1 Scope

The software outlined in this document is a Mobile application to be developed for use on smart phones that will allow users to calculate and track their environmental impact by means of an ecological footprint score. Our goal is to develop an application meeting the following specifications and requirements along with a well designed interface to make absorbing the information and learning more easy and simple for the end users.

1.2 Purpose

The purpose of this document is to provide planning information that will guide the development of this mobile application. The document will describe the processes that we intend to use to complete this task. Functionality and interface design will be discussed. The intended purpose of this application is to raise awareness and inform users about their impact on the environment. The application's goal is to inform and educate users, and also to encourage them to take measures on their own in order to reduce their total impact. Certain aspects of the application will be focused on informing and other aspects will be focused on persuasion and encouraging the desired behavior change.

1.3 Intended Audience

The purpose of this application is to spread awareness to as many people as possible. It would be ideal for as many people as possible to find and use this application. Since this application will be developed for Android smart phone devices the software will be available for any person to download via the Google play store or an APK download link from a website. The entirety of the targeted user group is anyone who has an android smart phone, although without marketing and advertisement it is very difficult to estimate the number of downloads and installs at this point.

2 GLOSSARY

Update as a group

TABLE 1
Glossary Table

Term	Definition
User Interface (UI)	Objects in the program that allow the user to interact with the application. Examples are pages, screens, images, buttons, etc.
Application Program Interface (API)	Code that allows two software programs to communicate with each other.
Bio capacity	The capacity of an area of land to produce renewable resources and to absorb waste.
Yield	The full amount of an agricultural or industrial product.
Local Storage	The storage of files on the device that the piece of software is running on
Integrated Development Environment (IDE)	A piece of software that consists of tools for creating software
Android Studio	An IDE for creating applications for android devices
JSON	A method of storing data that makes reading and writing easier
Library	A collection of code that can be imported and used

3 System Architecture

3.1 Design Rationale

The integrated development environment we are planning to use is Android Studio. It is based on the community created IntelliJIDEA. Some of the benefits of using this platform include faster deployment of fresh builds, more accurate programming, faster programming and testing, inclusive application development, better app indexing to name a few. To elaborate further on these benefits, for example, bringing incremental changes to an existing app code or resource is now easier and faster due to a feature called Instant Run. Code changes can be witnessed in the emulator or physical device on real-time without restarting the app or building a new APK (Android Application Package file) every time. The platform is also equipped with an intelligent code editor which makes code writing and analysis faster, easier and more accurate. In addition, the newly introduced emulator is 3x faster in CPU, RAM,and I/O in comparison to its predecessor. The virtual testing environment is faster than a real device and has a user-friendly UI. Sensor controls are effective to read every move of the developers. Developers can drag and drop APKs for quick installation, resize and rescale the window, use multi-touch actions (pinch and zoom, pan, rotate, tilt) and much more. Promoting is an important component of the app marketing, and Android Studio 2.0 takes it to a new high. The App Indexing feature available in the IDE helps in creating and adding indexable URL links to the app. Due to the reasons described above, we have chosen Android Studio as our preferred platform of choice over others available such as Xamarin and IntelliJ

4 DATA GENERATION

In order to calculate the ecological footprint score certain type of data will be pulled from a third party application programming interface. This data is crucial in allowing us to compute the ecological footprint score for the mobile application we are developing. Finding reliable data that can be utilized to calculate the footprint score is a challenging task. The ecological footprint score is calculated using a complex equation that requires several inputs of data. Some of this data will be entered by the user via a questionnaire that we will be displayed and the other pieces of data will be gathered by pulling the relevant data from the application programming interface. Given the complexity of the equation and time constraints we face to build a usable version of the application our plan is to use a simplified equation to calculate the footprint score. This will enable us to focus our efforts on other features of the application so that at the end of the project we hope to have at least five usable features for the mobile application we are developing.

4.1 Example API Endpoints

We will be using the Global Footprint Network application programming interface to pull the necessary data in order to calculate the ecological footprint score. Denoted below are a few specific API endpoints that we will utilize to pull data.

The Base URL:http://api.footprintnetwork.org/v1/[endpoint]

Endpoints: http://api.footprintnetwork.org/v1/data/countrycode/year http://api.footprintnetwork.org/v1/data

http://api.footprintnetwork.org/v1/countries

http://api.footprintnetwork.org/v1/types

http://api.footprintnetwork.org/v1/types/count

http://api.footprintnetwork.org/v1/data/[countryCode]/[year]/[record]

The various endpoints above allow us to gather the data we will need to plug in our equation for calculating the ecological footprint score. The Global Footprint Network makes this data freely available for us to use. All we need to do this is to register an API key which will be used when we make API calls.

4.2 Example Code for Mobile App

Here is an example piece of code that could be used to make an HTTP request after the request URL has been constructed. The URL will specify which resource is being requested and the code below specifically makes the request by using OkHttpclient.

```
public class NetworkUtils {
    private static final OkHttpClient mHTTPClient = new OkHttpClient();
    public static String doHTTPGet(String url) throws IOException {
        Request request = new Request.Builder()
                .url(url)
                .build();
        Response response = mHTTPClient.newCall(request).execute();

        try {
            return response.body().string();
        } finally {
```

```
            response.close();
        }
    }
}
```

Here is an example of how data could be parsed after receiving the response body back from the server.The JSON object contains the relevant information that needs to be parsed and made ready to be displayed to the user. The code snippet below systematically extracts the pieces of data that had been requested earlier and returns it as a list to be used at a later time.

```
public static ArrayList<SearchResult> parseGitHubSearchResultsJSON(String searchResultsJSON) {
        try {
            JSONObject searchResultsObj = new JSONObject(searchResultsJSON);
            JSONArray searchResultsItems = searchResultsObj.getJSONArray("items");

            ArrayList<SearchResult> searchResultsList = new ArrayList<SearchResult>();
            for (int i = 0; i < searchResultsItems.length(); i++) {
                SearchResult searchResult = new SearchResult();
                JSONObject searchResultItem = searchResultsItems.getJSONObject(i);
                searchResult.fullName = searchResultItem.getString("full_name");
                searchResult.description = searchResultItem.getString("description");
                searchResult.htmlURL = searchResultItem.getString("html_url");
                searchResult.stars = searchResultItem.getInt("stargazers_count");
                searchResultsList.add(searchResult);
            }
            return searchResultsList;
        } catch (JSONException e) {
            return null;
        }
    }
```

5 Collecting Data From The User

The way we plan on collecting data from the user is by asking them a set of questions that we will use to produce their ecological footprint score. To get a stable set of questions, we aim to gauge our questions with other footprint calculators; also, to get a good sense of where we are. We believe this will also help us in generating certain questions either to target specific areas or simply ask more questions to improve accuracy. In discussion with our client, we identified the five major areas that need to be addressed in order to produce an accurate score. The five areas are: energy, water, food, transportation, and waste. For the purpose of our application, we will be addressing one question from each of these five areas. The first application we looked at was Islandwoods Ecological Footprint Calculator. This web-app had an elegant user interface containing various questions revolving around the five major areas we identified. Something else we took away from this app is their live graphic of the users footprint as they progress through the quiz. We also noticed the structure of the questions they asked. The questions were split into three categories: Morning, daytime, and evening routine. For example, one of the questions asked was, How do you get from place to place? with answers listing walk, bicycle, car, or bus. Another question was After eating, what do you discard? with answers listing plastic, paper, or nothing[5]. With these questions in mind, we believe it will guide us in coming up with our own and better question set. Overall, the point in researching the questions is so we can produce an accurate score for the user based off a limited set of questions. Since we aim to have around 8 questions, we need to maximize the best results from each question. once we collect this data, we can then move on to the next stage of calculating the score. The more precise data we have, the more accurate the score will be. Likewise, we plan on researching more calculators to gain proper insight on the questions we need to ask.

6 Displaying Data In A Meaningful Way

6.1 Graph View

Since we will be developing our application on Android Studio, we plan on creating some graphs so that users can easily understand their score, with respect to certain criteria. This idea stems from the question: Would providing just the score be sufficient to the user or would a graph in correlation with other scores be better? Also, what would the tradeoffs be in terms of development. GraphView meanwhile is a popular library that offers highly customizable graphs and charts[2]. A great feature about this is that it can be synced directly with the SQLite database, making plotting data easier. With the database in hand, we can then create graphs that display the users score in comparison with other users on a line-graph. Other ideas include showing the users score over a period of time, as well as for other users. Even during the questioning phase of our app, we can create a live graph that displays the users current footprint and how each question impacts it. For example, if a user answers yes to the question Do you eat meat frequently?, the bar would increase significantly. Though we can simply represent this change by a number, we find it more interactive if we could display it on a bar-graph. It would also make the user-interface more intuitive and less static. Below is a snippet of the code:

```
GraphView graph = (GraphView) findViewById(R.id.graph);
LineGraphSeries<DataPoint> series = new LineGraphSeries<>(new DataPoint[] {
        new DataPoint(0, 1),
        new DataPoint(1, 5),
        new DataPoint(2, 3)
});
graph.addSeries(series);
```

6.2 Android Studio

With the other libraries involved, all the data displays will be carried out using Android Studio. After researching the kit, we were also able to come up with features that are built in, such as List View. These simple features will allow us to create visuals such as progress bars, bar graphs, and other useful visuals. With various activities and pages, we can create a usable interface where we can display/collect data from and to the user. On another note, Android Studio development uses java. Since we are determined to create our application for android devices, we will not be using any other languages. After discussing with our client and TA, we came across the idea of coding on a universal base so that are code can be reflected on both android and IOS devices. However, for the scope of our project we decided to focus on android devices alone.

7 Local Recommendations

One of the exciting features our application will include is a local recommendations system. The goal of this is to provide the user with suggestions on how they can go about reducing their own footprint using local resources. For example, if a users carbon emission is identified as dragging their score due to heavy transportation by car; the app will suggest a nearby bus route they can take. Another example is displaying local produce, so the user can shop sustaining their respective community. In terms of development, we need a solid foundation to look up nearby places and services in relation to the users location. Google offers an API with their map service and location feature. After researching on their developer documentation, we discovered a step-by-step procedure on how to integrate this very method. We will need three main files: google_maps_api.xml, MapsActivity.java, and AndroidManifest.xml[4]. With this in place and the proper configuration, we will be able to pull up nearby locations and services for the user. The next step will be to work on the logic behind when we would need to pull data, since we now know the how. Our focus will then shift to creating an algorithm that will allow us to pick specific locations that will be catered toward what the user needs. We will also need to consider situations where for example there are no bus routes near where the user is situated.

We believe this method of interaction will show the user exactly what they need to do to lower their footprint. In terms of how we will deliver these suggestions to the user, we are working towards integrating the suggestions with Amazon Alexa, via voice. This way, we can set quick reminders and adjust with an individuals daily schedule to see if there are any ways of improving their score. However, we will still implement the suggestions in the old fashioned-manner as just displaying them on the app. The goal of the Alexa platform is mainly to integrate our local recommendations into the users lifestyle as efficiently as possible.

8 DATA STORAGE

8.1 Storage Mediums

With the exception of data obtained through API calls to the global footprint network, all of the other data for our application will need to be stored locally on the end user's device. Most elements will be stored as simple file types such as .txt or .png for large chunks of text and pictures respectively. Android Studio helps with the storage of data for certain things. For example, in the Android Studio IDE, if you were to create a text box and fill it with text, android studio handles saving and loading that text into the text box for you. Other elements will have to read data from files and reference the file locations when the data is required. The other method of storing data that we plan to take advantage of is JSON. JSON files allow us to store data in an organized way and read specific bits of the information when necessary. The benefits of using JSON are that you can store information in files that are easily readable and editable by humans, as well as organizing the information in a way that makes accessing certain parts of it easier. We can take advantage of this to store a lot of the data that needs to be calculated such as the user's footprint score. Instead of recalculating values such as this, we can store them in a JSON file and tweak the organization of the JSON file to make it very simple and understandable to store variables and attributes of objects.

8.2 Example JSON Contents

Here is an example of the contents of a JSON file that stores data on a UI element. The words with quotations next to them can be used to search through the file to find the data you are looking for and read or write it.

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1";
    }
}}
```

8.3 Saving Data

Writing data to the device is relatively simple for most datatypes. At this point, we can use JSON to store information about objects that we want to reconstruct when the application is launched again. Writing to JSON files and editing them is something that can be done within android studio without the need for any aditional technologies; however, there are libraries that exist that help with reading and writing of JSON in order to make less work for programmers. Here is a small segment of code that would allow us to edit a specific bit of information within a JSON file.

```
JSONArray myJson = new JSONArray(str);
for(int i = 0; i < myJson.length(); i++)
{
    JSONObject jsonObj = (JSONObject)myJson.get(i); // gets the JSON Object
    if(jsonObj.getString("name").equals("Value")) // check the attribute name
        ((JSONObject)myJson.get(i)).put("id", 200); // change the value for the key
    textview.setText(myJson.toString());// checking for updated value
}
```

9 Data Processing

9.1 Loading Data

Some code is required for reading from JSON files. Android studio has plenty of documentation for how to perform this. Depending on how the JSON file was organized will change then method that is required to read and write from it. Several functions that can be called from anywhere in the application will be required to read from JSON. The use of these static functions will allow us to speed up development instead of relying on re-coding a new function in every page that requires reading from JSON. The following is a segment of code that will allow us to read information from .JSON files stored on the end user's device.

```
public String loadJsonFromAssets()
{
    String myJson = null;
    try
    {
        InputStream jsonStream = getActivity().getAssets().open("filename.json"); // specifying th
        int size = jsonStream.available(); // checking if the file is available for writing
        byte[] buffer = new byte[size];
        jsonStream.read(buffer); // read into the buffer
        jsonStream.close();
        myJson = new String(buffer, "UTF-8");
    }
    catch (IOException ex)
    {
        ex.printStackTrace();
        return null;
    }
    return myJson;
}
```

9.2 Calculations

After the data has been loaded into the application memory, we will need to perform various calculations on the data in order to make it displayable in a meaningful way to the user. Calculations on data will be performed only when necessary or when data has been updated. Similar to the idea of caching, we only want to re-calculate the data when it is either needed and not available, or when the data is updated. An example of this would be saving the user's ecological footprint score and questionnaire responses after they are collected. This data would then be accessed when needed, and only recalculated if the user were to change their answers. Within the code, data calculations will occur directly within the section of code that requires the data.

9.3 Passing Data

Certain elements of the application will require data that may be in use by other elements at the same time. Whenever data needs to be accessed, it is important to check if that data is available first. The majority of data passing will occur in reading and writing to data files. Whenever data is requested, we will first perform a check on the specified file to ensure that it is not currently in use.

10 UI Organization

10.1 Screens

The layout that we plan to use for this project revolves around screens. Screens are a way of referring to the various pages within the application and the content that belongs on each screen. A screen will encompass all of the visual elements aside from UI elements required for navigation.Screens are used in order to group and categorize elements that relate to each other. Each screen has a primary goal or purpose. Anything that is not relevant to that purpose will not be placed on that screen. Similarly, content should always be placed on the screen that its purpose relates to most. For example, a ”user score” screen would have information about the user’s footprint score and any other information or elements that we decide to show the user relating to their score.

10.2 Screen Objects and Interaction

Depending on the screen, content may vary in size, shape, and design. The purpose of the varying content from screen to screen is to create different UI elements that we believe will best allow the user to explore and interact with the data that we have collected in a meaningful and stimulating way. Every screen will have a few The following are user stories that we have generated when thinking of all of the possible screens that a user may want to be able to interact with.

10.3 User Stories

TABLE 2
User Stories

Story Title	Story Description	Priority
Main Screen (UI)	A user wants to be able to navigate to other pages within the application	High
Menu (UI)	A user wants to to open a menu to view the pages they can visit	High
Questionnaire(UI)	A user wants to be able to provide information about their lifestyle and habits by pressing buttons and selecting answers	High
Information (UI)	A user wants to view a page containing information about what the application is	Med
Statistics (UI)	A user wants to be able to view information in a neat and easy to understand page	Med
Personalized Stats (UI)	A user wants to view the personalized data in a easy to understand page	Low
Feedback (UI)	A user wants to see a visual breakdown of their score and which factors contribute most	Low
Resources (UI)	A user wants to view resources to help them reduce specific resource use	Low

10.4 Screen Navigation

Navigation from screen to screen will be accomplished through the implementation of an expanding navigation menu. This design is common across mobile applications because it is very simple and easy for people with any level of technology experience to pick up and learn. Every screen will have a button in the top left corner that will cause a menu to slide out from the left side of the screen. This menu will will be a vertically scrolling list of screens that the user can navigate to. Each screen name within the list will be it's own button that will cause the user to be taken to the screen corresponding to the button pressed. This allows navigation between any two screens quickly, and using the same process regardless of where you are in the application. By organizing the content into groups called screens, this allows the user to find information in a very natural way as well by encouraging information foraging. Information foraging is the theory that users are best able to find the information that they are looking for by starting general and becoming more specific. This design encourages information foraging by having an easily accessible menu that shows screens where similar information is grouped. Once the user understands what kind of information they want, they can navigate to that page, and from there they can explore the page to find the more specific bit of information that they are looking for.

11 TESTING

We will perform several types of testing to ensure the quality of this application. The three primary types of testing that we will perform are user testing on selected friends and family members, random testing, and keystroke-level-method testing. Primarily, we will perform several types of testing on test users. There is no technological experience required to use this application, and we would like for people who are as technologically illiterate as possible to still be able to figure out how to use our application. We will use test users to evaluate the learn ability of the application. We will give test users a task that they must accomplish. This task will be something that we expect most users will want to be able to do. The test users will attempt to perform that action without being given instructions. Afterward, we will take notes of anywhere the user was unsure of what to do as that may be a flaw in our design. We will also interview the users and ask them a series of standard questions to figure out if they experienced any frustrations or problems with using the application. The next type of testing we will perform is random testing. During our random testing phase, we will write segments of code to generate use cases with random numbers in order to check for bugs in our program. An example of this type of testing would be generating thousands of footprint scores by writing code that randomly answers the questions. We can compare the output of this test to the expected output to understand if our functions are behaving as expected, or if there is a logic error. The final type of testing that we will perform is called keystroke-level-method testing. In this type of testing, we do not require users or code. We simply count the number of keystrokes, taps, swipes, etc. required to perform certain actions. The general idea of this type of testing is that any action that the user will be repeating constantly, should require with a small number of actions. This allows us to notice if there are any actions that require a large number of actions to accomplish. This is strong indication that the specified feature needs to be moved to a place that is more accessible to the user.

12 TIMELINE

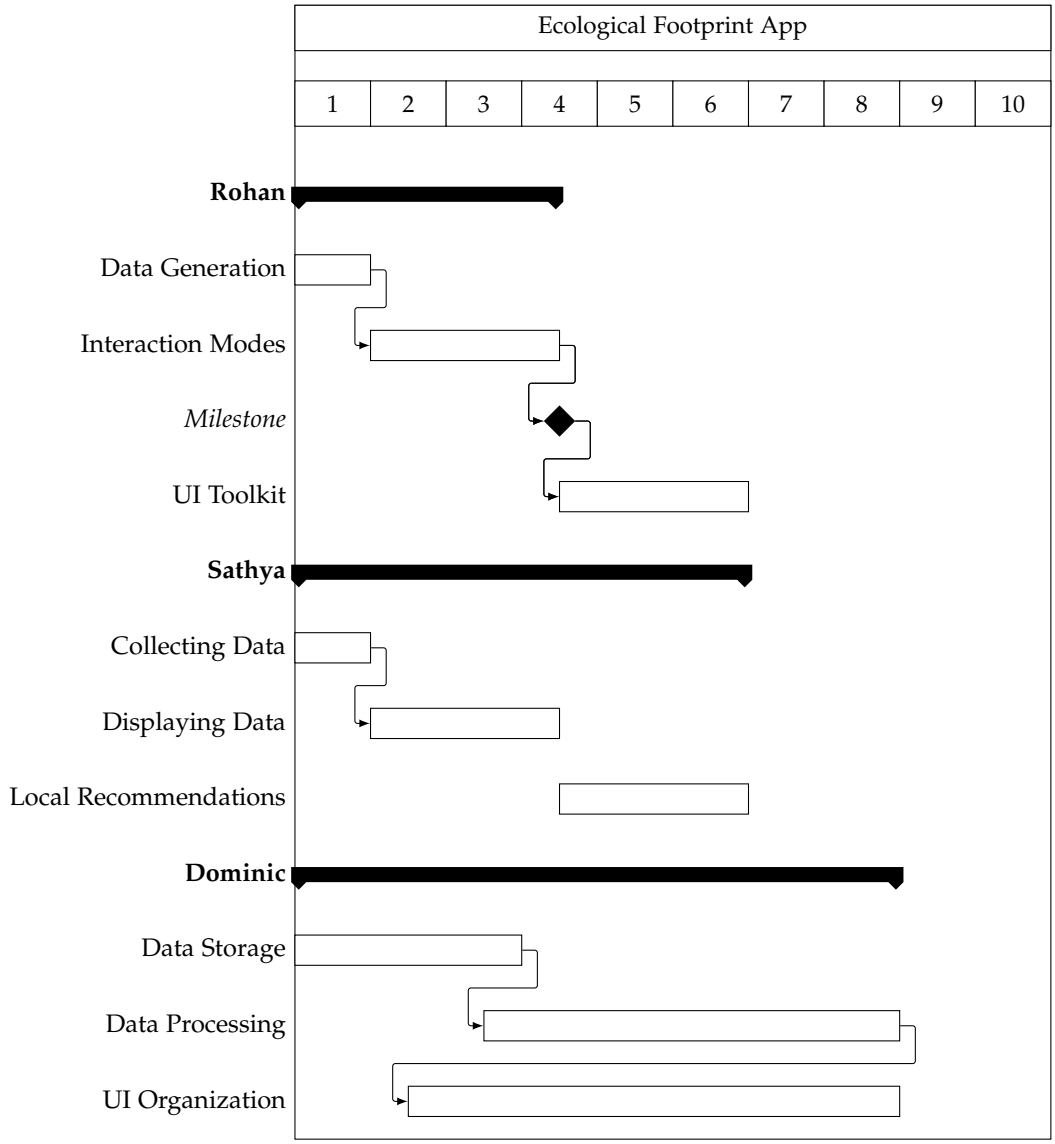

Fig. 1. Gantt Chart

13 Conclusion

In conclusion, with the design approaches listed above, we aim to produce an easy to use ecological footprint calculator app. The main development focus will be centered around Android Studio with features added on such as SQLite for data storage and GraphView for displaying data in a meaningful way. With these in place alongside other additional features stated above, we believe our application will not only produce a footprint score for the user, but provide them with insight on how they can minimize it.

REFERENCES

[1] ”SQLite Documentation” SQLite, n.d. Web. 1 November 2018
Available: https://www.sqlite.org/docs.html

[2] ”GraphView - open source graph plotting library for Android” Graph View, n.d. Web. 1 November 2018
Available: http://www.android-graphview.org

[3] ”Open Data Platform” Global Footprint Network, n.d. Web. 4 November 2018
Available: http://data.footprintnetwork.org/#/api

[4] ”Google Maps Search Nearby” Android Tutorial Point, n.d. Web. 5 November 2018
Available: https://www.androidtutorialpoint.com

[5] ”Ecological Footprint Calculator” Islandwood, n.d. Web. 24 November 2018
Available: https://islandwood.org/footprint-calculator/

[6] ”Top 5 benefits of Studio” ”Android Studio”, n.d. Web. 26 November 2018
Available: https://www.rootinfosol.com/top-5-benefits-from-android-studio-2-0-for-an-android-app-development-company

[7] ”JSON Documentation”, n.d. Web. 26 November 2018
Available: https://www.json.org/

[8] ”Information Foraging Theory”, n.d. Web. 26 November 2018
Available: https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/information-foraging-theory