TECHNOLOGY REVIEW - FINAL
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

DOMINIC WASKO

**Abstract**

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. We will be talking about the ecological problems we face today, the proposed solution, and our projects scope.

## CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# 1 GLOSSARY

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. |

## 2 Introduction

Over the course of the term we have been researching this problem and figuring out which way we think will be the best way to implement this application to the best of out abilities. We have looked into data sources, collection, and discussed overall application structure and format. The purpose of this document is to review the technologies and methods that we have selected for use in this project, and compare and analyze them against other available software and technology solutions. By doing this, we either hope to affirm that we made the correct decisions on which technology to use, or find out about new technology that may be a better solution for us.

### 2.1 Project Breakdown

We have broken down the project into nine subsections. The subsections are:

- **Data Generation:** Where the data that we will need to work with will come from, and how we plan to get it into our application.
- **Data Storage:** What do we need to store, and how we are going to store it.
- **Data Processing:** Statistics
- **Data Display Model:** Creation of custom UI elements to represent data.
- **Data Display Content:** Determining what data we want to show, and how we want to organize it.
- **Data Visualization and Interaction:** How we are going to allow the user to interact with the data.
- **UI Organization:** The overall structure of the UI and navigation.
- **UI Toolkit:** The tools available for creating UI elements.
- **Interaction Modes:** Types of input and interaction that will be supported.

### 2.2 My Role in This Project

For this project, I have selected **Data Storage**, **Data Processing**, and **UI Organization** for the three areas that I plan to specialize in and work primarily on. Due to the nature of this application having a heavy focus on UI and presentation, some of the ”technologies” that will be discussed when talking about certain aspects of this project may be closer to descriptions of possible implementations rather than a separate piece of software or hardware. A large portion of the success of this application rests in how effectively we are able to enable users to engage and interact with the data that we hope to present them.

# 3 BODY

## 3.1 Data Storage

**SQLite**

SQLite is a library that allows for the creation of a local self contained SQL database [1]. This option has a great degree of flexibility, and would be great for projects where there is a large amount of different data that would need to be stored. Having a SQL database would allow the creation of tables to better organize data. This is a great solution for projects that need to keep track of a lot of different kinds of information. One example of a good use case would be needing to hold information about thousands of car types. We plan on only having information that is pre-selected to present to the user, information about general statistics, and information specific to the user. For our project, this does not make very much sense as we will be gathering most of out data via a third party from *The Global Footprint Network [2].* Utilizing their API allows us to only pull in the information that we need, when we need it, and we do not plan on storing enough data to make implementing a database a necessity, or even a priority for us.

**JSON**

I believe that storing files locally with JSON is the option that is the most promising for this project. Any data that we decide to save will be stored locally on the phone of the user. JSON is a way of storing data that is easy to read and write for humans, additionally, there are built in tools in Android Studio that allow us to read and write to JSON files. [4] [5]. Other information that needs to be stored can be stored in simple file types that correspond to the type of data being stored. For example, images and contents that will be rendered on the screens will be saved as images, and accessed and read by the system when it requires the data. Using JSON allows us to more easily reconstruct and read the data, as well as keep it more organized rather than put everything in .dat or .txt files.

**NodeJS Web Server**

The final option that we had considered is a web server. Several members of the group have experience developing web servers in nodeJS [3]. The client would send a request to a database server whenever it needs content so that it isn't always on the user's device. This option is the least beneficial to solving our problem. We have no intention of creating a log-in system, so there is no need to connect to a server for log-in authentication. The benefit of using a Web Server to host content and serve it as needed, is that the content to be displayed can be stored on the server and not the user's device. User accounts is one of the only reasons that a web server would be a good idea for a product such as this as the scope and function do not require many things to be in a database or located somewhere beside the user's phone. The benefits of using a web server over SQLite is that the information does not need to be stored on the device of the user. The amount of data that we plan on storing is not large enough to warrant the time dedication of creating a separate server.

## 3.2 Data Processing

**Class System**

While we do plan on using classes in this assignment anyways as it would be impossible to have all of the code in one file, we could create object classes for representing the different things that would need data calculations. One example of this would be a user class to keep track of all of the information regarding the user, there could be another class for data about statistics and unprocessed data. The classes could have functions to save their information to simple files and then reconstruct themselves at runtime from the data files.

**Calculate When Needed**

One option is to simply perform all of the necessary calculations and statistics whenever the user navigates to a page where that data will be represented. For example, we plan on having a page that will allow the user to view a comparison of their ecological footprint score to the national average. This approach is a waste of resources because every time that that page is clicked on, the users device would need to recalculate their footprint score from the information they entered.

**Google Gson**

Google Gson is a library for android studio that allows you to process .json files much more easily than you would otherwise be able to [6]. While there is built in support for JSON in Android Studio, Google Gson is an alternative library we could use in order to further simplify the process of handling JSON files. Gson's goal is to provide functions that make the process of converting between JSON file data and Java Objects more simple and easy. Each individual user is going to input different data into the application. The users will definitely not want to answer the same questions over and over again whenever they open the application, and it would be a waste of processing resources to calculate the data over and over when it is requested if the inputs have not changed. The use of Gson allows us to store information in .json files and process them in order to extract data that has already been calculated.

## 3.3 UI Organization

**Expanding Menu Bar** The first and most clean feature for UI navigation and overall structure would be to have an expanding menu bar on the side that remains hidden when not in use. Here is a link to an example image depicting a menu that fits this description *Sidebar Navigation Example* [7]. This has become fairly common practice as more and more developers are paying closer attention to UI design and ease of navigation. For the relatively small number of pages that this application will have, this is the perfect solution for UI navigation. It does not clutter the screen when it is not needed, and pages are listed in a menu that will come out from the side when the user presses a button. Implementing this UI also allows users to translate their previous experience using applications into knowledge for our application. This navigation style is very common in applications in the last few years. Most users will be able to distinctively recognize the icon and what it does.

**Launchpad Page** The second option that we had debated was creating a homepage that would serve as the hub and provide links to all of the other pages. This works well only for very small applications, and the lack of a quick way to get to a specific page is a serious drawback to using this method. Most applications that employ the use of a homepage style navigation solve this problem by implementing a very detailed search system with user prediction ability. Without a high quality search function at the top, this style of UI navigation is very easy to get lost in, and many users are unsure of the best way to get to a specific page. The major drawback of this design is how easy it is to forget where you are within the application. By chosing to go with an expanding sidebar, users will be able to navigate to any page within the application and read their names from a button that can be pressed on every page of the application. This way, users always know how to get back to the page they are looking for as the process does not change depending on the page you are on. An example of a quality launchpad style UI can be found at *this address*

**Website Layout** The final option is to use a website layout and have pages nested within other pages. This method of UI can be a great idea for very large and complex applications that have a lot of data to display and many pages. We are focusing on creating a relatively small number of core pages that the user can navigate between. The major benefit of a website style UI is that in most cases, pages that reference other pages link to them. A perfect example of making the most out of this relatively simple design is how Wikipedia articles have links on each page on top of words that Wikipedia has a page for. For example, if you were to read the Wikipedia page for apple, the word fruit would take you to the fruits page. For large applications that have many many pages that share connections and relations with each other, it is a great way for users to find what they are looking for by narrowing their search this way, or by traveling backwards from a more specific page. As the example indicates, hierarchies and classifications can get unmanageable when you have too many, and this can be a good solution. Once again, because of the nature of our application we would not be taking advantage of the main reason one would choose this style of navigation. This function becomes more and more of an inconvenience the smaller the project size, as other design options become a much better choice.

**Final Note on UI Nvaigation** A large portion of this project will be creating UI elements that make the application as professional looking, and as easy and intuitive to use as possible for users. The contents of this application are not as complicated as other projects, the user interface is our opportunity to take what we know about usability and create an application that is very easy to use and figure out for new users. This is going to be very important for accomplishing the goal of spreading awareness of ecological footprints because users will hopefully like the design enough to keep using the application and recommend it to others. I have briefly described three popular navigation designs; however, this does not mean that there are not others, or that you cannot combine elements from different designs in order to create something that works best for your project.

## REFERENCES

[1] SQLite *https://www.sqlite.org/about.html*

[2] The Global Footprint Network *https://www.footprintnetwork.org/about-us/*

[3] NodeJS *https://nodejs.org/en/about/*

[4] JSON *https://www.json.org/*

[5] Android Studio JSON Documentation *https://developer.android.com/reference/org/json/JSONObject*

[6] Google Gson *https://github.com/google/gson*

[7] Sidebar UI Example Image *https://www.designyourway.net/blog/wp-content/uploads/2016/05/dribbble-1.jpg*

[8] Launchpad UI Example Image *http://www.rit.edu/news/lib/filelib/201112/mobile_app_home.jpg*