Travel Companion App

Software Engineering Final Report

Colton Coughlin, John Cordwell, and Justin Henley
Travel Companion Github Hyperlink

Abstract—Our team seeks to create a travel companion app that visually displays information about a country's safety using data from the US government.

1 Introduction

1.1 What is our project?

Our project is to make a travel planning app. It will show the travel restrictions placed on every country worldwide by the US government as a way to quickly see which areas of the world are safe to travel. The map will be color coded according to the travel advisory level.

1.2 Motivations

We decided to make this app because we noticed that the US Department of State website with this information was a bit boring and hard to navigate. As visual learners, we would love to find a way to make the information more visually appealing and interactive.

1.3 Market Context

As an app that simply displays publicly available information, there is no money to be made or a market for an app like this. The only source of income possible is through in-app advertisements. We have decided that these types of in-app advertisements will not be included until the product is worthy to stand on its own.

1.4 Team Background

Our team consists of three computer science undergraduate students at the University of Tennessee, Knoxville seeking to make their first real software development. As students, our team has limited first-hand experience in the software engineering industry.

2 Customer Value

There were no changes made to our customer value from the proposal to the final product.

2.1 Customer Need

The primary target audience is people who are interested in traveling internationally as a US citizen. More specifically, we aim to focus on those who are new to traveling abroad and want to research a wide variety of countries based on their safety rating. The customer wants to be able to quickly and easily learn which countries are safer to travel to as well as why this is the case.

2.2 Proposed Solution

Instead of the boring gray scale database on the US Department of State website, we will create a visually appealing map to represent the information with an interactive interface for additional details on each country. By visually displaying the information, customers will be able to assess and compare a countries safety risk more quickly and conveniently.

2.3 Measures of Success

Since we have not released the app onto the Apple Store at this time, we are unable to obtain user reviews through their review method. Instead, we will measure our success based upon how others interact with our project when presented with a live demo. We imagine that there is a lot of potential benefit from showing our product off to as many people as possible to gather feedback on how we can improve.

3 Technology

3.1 Back-end Architecture

For the backend of our project, we used Rust to scrape advisory level from the US Department of State website then store it inside of a JSON file. Figure 1 shows an example of what a country looks like inside of the existing website. Figure 2 shows how the data is stored inside of the JSON file. The gathered fields for every country include the following: country name, advisory level, and advisory text. For the purposes of simplifying the front-end, we decided later in development to leave the advisory text field as an empty string. Once the scrapped JSON file was created, we found a free to use SVG map on MapSVG.com (acknowledgments of use are made inside the app). We were able to conduct a similar JSON file creation process to produce a JSON file version of the countries included in the SVG. Another Rust program was created to simply combine the two JSON files into one by matching the country's name. An example of the combined JSON file which includes the path coordinates can be seen in Figure 3.

3.2 Front-end

The combined JSON file is imported into Xcode where Swift was used to display its contents visually. The first task was

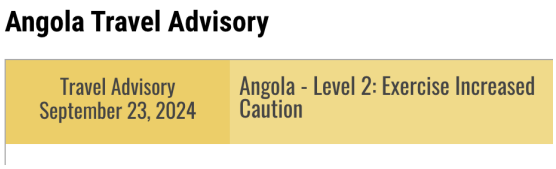

Fig. 1. Example of what the country data (Angola in this case) appears like inside of the US Department of State Travel Advisory site.


```
{
  "country": "Afghanistan",
  "advisory_level": "Level 4",
  "advisory_text": ""
},
{
  "country": "Albania",
  "advisory_level": "Level 2",
  "advisory_text": ""
},
{
  "country": "Algeria",
  "advisory_level": "Level 2",
  "advisory_text": ""
},
```


Fig. 2. Scrapped JSON file organization which includes the country name, advisory level, and advisory text.

to correctly be able to display a country on the screen given its border coordinates. The path given is provided in ”mz” format where ”m” stands for ”move” and ”z” stands for ”end”. Here is how it works:

- **”m” reached**: move to (x,y) coordinates specified immediately following the ”m” and place a black pixel. All country paths begin with an ”m” to signify the start of the country border. If a country’s border is not directly connected then multiple ”m” commands may be present in the path.
- **”z” reached**: border end point reached so place last black pixel. If followed by another ”m” then move to new location, but if not then the country’s path has ended. All country paths end with a ”z”.
- **no ”m” or ”z” reached**: if just (x,y) coordinates are present then there is to be black pixel placed at the relative location of the previous coordinate plus this one. For example, if the previous x location was 479.68 and the next x coordinate is -1.0 then place a black pixel at x = 478.68.

Once the borders for each country are applied to to the map in black pixels, we were able to fill in the area between each border with a color associated with its advisory level in the JSON file. We chose to make white represent advisory level NULL, green represent advisory level 1 (lowest), yellow


```
{
  "code": "AD",
  "name": "Andorra",
  "path": "m 479.68275,331.6274 -0.077,0
  -0.02,-0.067 0.038,-0.181 0.086,-0.097
  "advisory_level": "Level 1",
  "advisory_text": ""
},
```


Fig. 3. Combined JSON file organization which includes the country’s two character ID, country name, coordinate path, advisory level, and advisory text.

represent advisory level 2, orange represent advisory level 3, and red represent advisory level 4 (highest). We then implemented features such as scroll capability, zoom capability, and the ability for each country to be clicked to display a pop up. The final product shown in Figure 4 shows the entire map of clickable countries and Figure 5 shows what the pop-up box looks like.

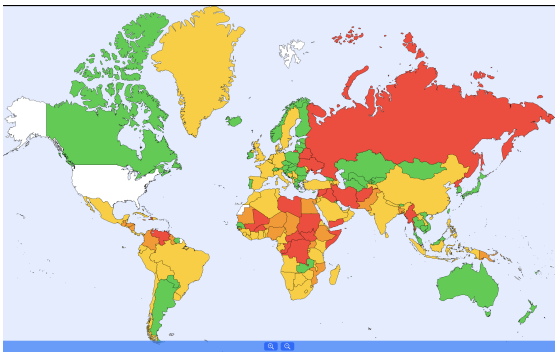

Fig. 4. Final product map view of clickable buttons and color coded according to advisory level.

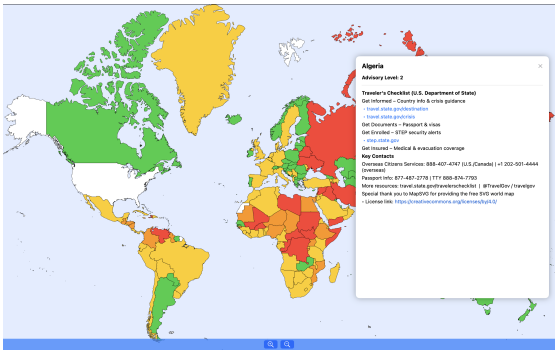

Fig. 5. Final product map view which shows an example of the pop up box which displays on click.

3.3 Testing

For testing the majority of the time was put into finding countries with NULL advisory data (shown in white). These countries all fell into one of the three following categories:

- **Category 1**: The country was not included in the Department of State website and therefore not given an advisory level.

- **Category 2**: There was a name mismatch between the country's name in the Department of State's website and the SVG map data.
- **Category 3**: The country is the United States who did not assign itself an advisory level.

Only countries that fall into category 2 were able to be fixed. This process was done by manually finding the name mismatch (due to outdated name or misspelling) and replacing the advisory level 0 with the appropriate advisory level. An example of this is the country of Myanmar which was listed as "Burma" (an outdated name) in the Department of State's website.

4 Our Team

4.1 Final Team Member Roles

- **Colton**: Designed the back-end in Rust that was able to scrape the US Department of State Travel Advisory site and export the advisory data into a JSON file with each country being an object.
- **Justin**: Combined the scraped JSON file with a JSON version of the SVG map coordinates to link a country's advisory level with its geographic coordinates. Handled calculating the map coordinates from the SVG then displaying the map visually in Swift.
- **John**: Implemented a general user interface (UI) which allowed for map navigation, country selection, and data presentation. Expanded upon the visuals of the map.

4.2 How the Roles Changed Over Time

Over time, the roles shifted with additional members of our team moving to the front-end due to the problems being faced. This shifting of responsibilities allowed for breakthroughs on the front-end which lead to completion of the project.

5 Project Management

5.1 Goal Completion Timing

Our group completed the majority of the planning and early development (back-end) goals on time, but we fell a bit behind when developing the front-end. We simply did not anticipate some of the challenges we would run into in the front-end such as connecting the SVG coordinate data with the scraped advisory levels. Due to falling a bit behind, we were unable to reach the full extent of our UI feature goals.

6 Reflection

6.1 Looking Back on the Project

Throughout this journey, there were things that went very well in addition to things that did not go as well. Here is a breakdown of how we felt throughout the course of this project.

- **Planning**: Overall, this was a smooth process once the travel advisory idea was established. The only discrepancy here was that there were a couple of different ideas about how we could potentially go about doing this. The first was partnering with Google maps to add an extension overlay to their existing map architecture, while the second was to just build our own app and map. We chose to go with the second option due to licensing and implementation questions.
- **Development**: The creation of the back-end was a very straightforward process using Rust, so it was not long until we started working on the front-end. The front-end presented a variety of problems such as finding/creating an SVG map, parsing SVG coordinates, and making the map interactable with the user. These problems took a while to resolve and caused most of the developmental stress.
- **Testing**: Testing was not a hard process. The majority of our testing went into ensuring that the vast majority of the countries had an advisory label attached to them. Additionally, during the testing phase we added simple visualization changes to make the product look more appealing.
- **Team Management**: Our group communicated effectively through an iMessage group chat when we needed to finalize project details or were approaching a deadline.

6.2 Was the Project a Success?

Given the objectives we set at the beginning of this class, we believe that our project was a success! We accomplished the primary goal of creating a basic app that visually displays country advisory warnings on individual and clickable countries on a map. There are many features that we could still implement in the future, but the basic product is complete.