CroCoMap
Implementation Documentation

Public Repository: *https://github.com/nubcaker/CroCoMap*

A.R. Breurkes, D.A.L. van Tetering, S.M.M. Rosas, T.D. Westerborg
{a.r.breurkes, d.a.l.vantetering, s.m.m.rosas, t.d.westerborg}@student.tudelft.nl

June 14, 2020

1 Introduction

This document holds the documentation and justification for implementation choices for CroCoMap: Crowdsourced Corona Map. The following sections will cover the architecture, components, state and utilities. In each section, the choices behind the implementation are covered if not self-evident.

2 Architecture

CroCoMap is a stand-alone web application, built using the Vuejs framework[1]. This prototype has no back end system to process and store the gathered data, but rather reads JSON files to set up tasks and writes JSON files to store the results. In any possible future deployment, it is highly recommended to incorporate a back end system into the process. The back end system was not implemented for this prototype due to time constraints. The persistence of the system's state will be covered later in section 3.

Like any Vuejs application, CroCoMap loads the application's main componenets from `/src/main.js`, with as default component `/src/App.vue`. Here, the Google API key for Maps and Street View is loaded from the environment variable declaration, as well as the router, store and material design framework[2]. Finally, the available routes are defined and the application is instantiated.

For more information on the specifics of Vuejs, please resort to their documentation[3]. As for the imported libraries, resort to their respective documentation.

3 State

The state of CroCoMap is maintained in a so called "store" using Vuex[4]. The store and its methods are implemented in `/src/store/index.js`. Through the store, the system's state can be saved and variables can be communicated between components. Each of the variables has its own getter and setter (mutations). The only difference in behavior happens when changing the value of `verifyVotes`, during which a value is pushed into the array, rather than replacing the current value.

The store also contains `actions`, which are methods that perform a certain action and then commit their changes to the state, notifying the application of the occurred change.

4 Utilities

An extra utility CroCoMap relies on is `raycast.js`, written by Sihang Qiu[5]. The find task relies on this utility to calculate the position of the marker accurately based on the click. The file can be found in `/src/utils/raycast.js`.

5 Components - Implementation Justification

This sections will cover the individually implemented components of CroCoMap. The components are divided into two categories, pages and elements, and will be covered in that order. Other than for the state in section 3, this section only covers important implementation choices. For near line-by-line documentation, resort to the files.

One part of the implementation applies to all pages: the downloading of the JSON file. This prototype of CroCoMap does not process or store the data and results using a back end system. To work with this data and not let is reside in memory, but rather in storage, this prototype works with JSON files. Result files are automatically downloaded using the worker's browser's download function. It uses this function because web applications cannot store files elsewhere than the download folder provided by their user.

[1] https://vuejs.org/
[2] https://vuematerial.io/
[3] https://vuejs.org/v2/guide/
[4] https://vuex.vuejs.org/
[5] https://github.com/qiusihang

5.1 Pages

CroCoMap contains three pages, one for every task phase: Find, Fix and Verify. The worker can navigate to these these pages by opening the drawer using the menu button in the top-left corner of the screen. In the top-right corner of the screen, the worker can see their username and karma score. No back end is included due to time constraints and users can cannot log in, hence the latter mentioned functionality contains only placeholder information. All pages reside under `/src/components/pages/`, which will be used in this section to declare relative paths from.

5.1.1 Find

The find task page is implemented in `./Find.vue`. This page consists of two tabs, one with the map view and one with the street view. To correctly display the map and the street view, the find page imports the Google map via a library[6]. This library handles setting up the maps and street view component.

On the find page, the entire map is first covered with a red overlay to indicate the off-limits area to the workers. After the worker has confirmed their preferred location, CroCoMap will divide the map into 144 equal rectangles based on the bounds of the map relative to the view width and height. Only 5 will be randomly chosen for the worker to annotate in. Both these numbers can be adjusted easily, these value are just used to showcase the prototype.

On the street view tab, the instructions and buttons are placed at the top of the screen to avoid blocking any objects that might have to be annotated. These objects are namely most likely to be near the ground and not far on or above the horizon.

When annotating, a circle overlay is shown to indicate where the worker will achieve the highest accuracy when placing a marker.

When taking snapshots, the find page loads the street view image from the Static Street View API, and writes it to an HTML canvas. It does so, because there is no way to take screenshots in web browsers for security reasons. This is a workaround that works fairly well, but does require an extra API to be used.

A future recommendation is to include a mini-map when on the street view tab. This is not included in this prototype because a street view panorama cannot be linked to more than one map object. The first tab already included this map object, which we did not want to remove for ease of use. A way around would be to resize the map and place it as an overlay on the street view tab.

5.1.2 Fix

The fix task page is implemented in `./Fix.vue`. This page contains only a street view component with one marker per annotation. The markers are loaded one at a time while the user fixes or accepts their corresponding snapshots. The instructions and buttons are placed at the same locations as on the find page for the same reasons.

One could argue that fixing the panorama in place would be good for this task to restrict the worker from walking around the street view freely. However, Google's API can show some quirks. As seen in the demo[7], positions provided by the API do not always correctly reflect the position of the street view panorama. Therefore, we allow workers to move and look around freely, so they can overcome any difficulty and complete their task.

5.1.3 Verify

The verify task page is implemented in `./Verify.vue`. This page shows six cards, of which one with the street view panorama for context and five with snapshots taken by workers. This number can be increased easily in future implementations.

The street view does not contain the marker like in the find and verify tasks, because the panorama is small and the marker might show up too large, blocking the view and negating the purpose of showing the street view for context.

This page does not contain instructions and has the menu button in the bottom-left of the page for two reasons. First to avoid obstructing the worker from viewing potentially vital information on the screen, and second, there are no instructions because the verify requires no complex tasks, as opposed to the other tasks where workers have to take snapshots (for example).

The "non-risky" and "uncertain" buttons are located at the bottom of the page, since only one of these have to be created. If an annotation is deemed non-risky, all snapshots for that object are invalid after all.

[6] https://www.npmjs.com/package/vue2-google-maps
[7] https://www.youtube.com/watch?v=iJY4w2S06PA

5.2 Elements

CroCoMap uses three separate smaller elements to offer its services. These elements reside under `/src/components/elements/`, which will be used in this section to declare relative paths from.

5.2.1 Tutorial

The tutorial is implemented as an element and resides in `./Tutorial.vue`. The tutorial has a welcome page, a page for each task and a concluding page. On the task pages, the worker is provided a textual and visual explanation of the task. After completing the tutorial, the worker can start working on tasks. They can always return to the tutorial using the menu button at each task page.

5.2.2 Street View Card

The street view card is displayed on the verify page only and serves the purpose of providing context to the shown snapshots. Its implementation resides in `./StreetViewCard.vue`. No marker is shown because this card is rather small. A marker could obstruct the view, negating the purpose of the card.

5.2.3 Verify Card

The verify card contains the a snapshot and vote button for a corresponding annotation. It resides in `./VerifyCard.vue`.