

# **WebMap (ugrad\_SD\_7)**

**Team:** Aren Ashlock, Braeden Hegarty, Michael Gelineau, Seth Watgen, Subham Bhattacharya

**Client:** Nitin Gadia

**Course:** COM S 4020C

**Instructor:** Simanta Mitra

**TAs:** Mohammed Rahman, Yonas Sium

# Requirements

We began the semester with a few different requirements or criteria goals from our client. There were some major issues that needed to be addressed following the completion of development from the previous semester's group. Other requirements that were requested included new features such as layers outside of groups, and the ability to split the website into multiple instances, each of which could be specifically tailored to a certain need.

We began by working on requirement number one. This took the brunt of our time developing, with all developers working through a list of notable bugs. Originally, we had hoped to get these completed within 2-3 weeks, but some of them proved to be rather messy. Overall, it took from week 2 (around when development actually began) up until spring break (week 8) to complete an acceptable number of the bugs. We say acceptable because we did not end up getting through the entire list, focusing on higher priority bugs, and fixing regression bugs as they arose with further development.

Once we had the website to a point that our client found it suitable, he requested we change our focus to helping him learn about the structure, and specifically, splitting the instances of the website and implementing documented steps for how he could do so as well. There was significant research focus and development done to take the existing structure of the website and host it on a separate index page, with access to a new database connection. Through completing this, we had to figure out how to split the Prisma schemas and how to deal with the API routes being shared among the components on different pages.

At the same time, the development was being done to complete the above, there was an effort put into adding the ability for users to draw layers. One of the requirements from our client was that the users of the website have existing GIS functionality from multiple other software, in one hosted site. The previous group had already implemented the ability to link layer tilesets from existing GIS data, and he wanted us to add the ability to draw and save our own tilesets, which could display for any authenticated user of the website.

Finally, after everything above had been completed, we were asked to form detailed documentation for the website, including the old code which was handed off without any documents, and for the new features and abilities we researched and implemented. We have since formed a library of documents laying out specific steps to recreate any of the platform hosted development we completed such as splitting instances, a video detailing all website features and including a code-walk of where each of those features or routes are, and also increased inline documentation for future developers, so they can have easier logical references to the logic in methods as they are working on them.

# Design

The Webmap project is architected as a modern, modular web application built using Next.js with TypeScript and integrated with Mapbox GL JS for mapping, Prisma for database access, and MongoDB as the database backend. Our goal was to create a scalable, extensible platform that allows for dynamic map interaction, drawing tools, and instance-based deployment—all without sacrificing performance or user experience.

## Application Architecture

At the core, the application is structured as a monorepo-style Next.js app, with dynamic routing and server-side API routes for interacting with the database. The frontend and backend share the same codebase, and the app is deployed using Vercel, which simplifies builds and supports scalable hosting.

Key technologies include:

- **Next.js + TypeScript**: Offers server-side rendering and static site generation for optimal performance.
- **Mapbox GL JS**: Handles all map rendering, interactivity, and layer control.
- **Prisma**: Interfaces with MongoDB to manage persistent data for maps, layers, and drawn features.
- **Tailwind CSS**: Used throughout the application for responsive, utility-first styling.

## Layer and Drawing System

The layer system is designed to accommodate both predefined GIS tile layers and user-drawn features. Tile layers are configured using a combination of JSON schema and API-managed documents. Users can toggle layers on and off, zoom to an extent, and interact with map features like lot boundaries and historical overlays.

We extended this design by implementing a custom drawing tool using [`@mapbox/mapbox-gl-draw`](#), enabling users to:

- Draw polygons, lines, and markers.
- Edit or delete drawn features.
- Save drawings to persistent storage with metadata like source labels and layer names.
- View and toggle these new layers after submission and refresh.

This functionality was integrated cleanly into the existing map UI and supports both group-based and standalone layers.

## API Design

The API is built using Next.js API routes and Prisma's ORM abstraction. Core endpoints manage:

- **User-authenticated layer creation** and saving of drawings.
- **Layer retrieval and toggling** via RESTful GET and POST endpoints.
- **Instance separation**, allowing each subdomain to point to a different Prisma schema and MongoDB collection.

This modular API design allows clients to configure multiple maps from a single codebase, while still supporting full CRUD operations on layer data.

## Instance Separation

To support different historical or geographical datasets, we implemented an instance-splitting architecture. Each instance:

- Runs from a unique entry point/index page.
- Connects to a dedicated MongoDB collection and Prisma schema.
- Can be configured independently using `.env` files and schema-specific deployments.

This flexibility empowers the client to generate new maps for other regions (e.g., North America vs. Manhattan) without engineering support.

## Work Done

### Aren:

New technologies - Next.js, TypeScript, Prisma, MapBox API, Vercel, Package.json scripting.

Bug Fixes - Changed icons from links to globes, Hardcoded the "Zoom to World" button, Helped figure out layers not loading, Optimized data loading to reduce crashing of the site, Worked on the hash parameters to get them to update and auto zoom to a specified level, Reduced the lag of the timeline slider, Fixed the styling of the info modals, Made it so certain sections opened by default.

Separate Instances - Restructured the project to support multiple instances on the same domain, Made an instructions document, Modified the prisma schema, Recorded a video showing how to make a blank instance, Assisted the client with their first attempt to make a blank instance.

### Braeden:

New technologies - Next.js, TypeScript, Prisma, MapBox API, GeoJSON

Bug Fixes - CSS Styling, layers not appearing, Screen crash, Layers not having zoom info, Zoom settings not being accessible for standalone layers

New Features - Standalone layer implementation, backend work for drawing functionality, implementing editability of drawn layers with existing api routes and code

**Michael:**

New technologies - Next.js, TypeScript, Prisma, MongoDB, MapBox API.

Bug Fixes - Enable by default function, Lot events displayed by default, Lot Event render issue, Layers not appearing, White screen crash, Zoom buttons for layers not appearing in groups, Enable by default not displaying as standalone layer.

New Features - CSS changes and functionality identical to original site (nahc-mapping), Initial Drawing research, New layer dialog, Drawing features on new layer front-end development.

**Seth:** Worked on various bugs and assisted some with the drawing features.

**Subham:**

New technologies - Next.js, TypeScript, Prisma, MapBox API, MongoDB.

Bug Fixes - Slider hold feature where I changed how the data loads, info box scrolling (made it like a list instead of a single box), made data load faster in the northamerica/landowner page (modified the api calls before data was loading at an extremely slow speed it was taking few minutes for data to load, I changed it so it loads in a few seconds), info box appearance, CSS styling adjustments.

New Features - CSS changes and added functionality to the MENY site, added markers and polygons to the initial drawing page as shown during demo 2. Also worked on some documentation.

## Results

The initial expectation for this project was to fix any bugs left over from the previous group assigned to this site, and then add in drawing features for the New York Map. Our group accomplished this and much more.

The initial approach we took to handling bug fixes was an “all hands on deck” approach, in which all devs would grab a bug to work on and, when completed, grab a new one or assist a developer with their bug. In order to manage this, we used an issue tracking software called JIRA, which allowed us to assign point values, determining how long we expect this bug will take to be completed. Using this method, we were able to fix recurring crashes, implement standalone layers, fix Lot Events appearing by default, as well as many more.

After this initial bug fix development phase, our client wanted us to shift to integrating additional maps and subdomains into the site. This required lots of research into our project in determining which components were needed to replicate this map and allow new data to be loaded onto a

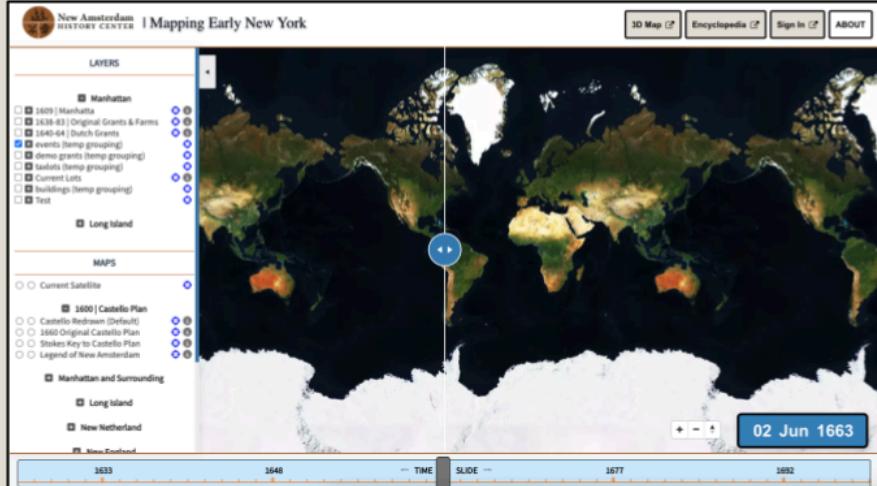
blank map for our client. We ended up creating lots of documentation for implementing these new sub-domains, as well as a video showcasing us creating a new sub-domain. Our client was then able to create new sub-domains without the help of developers.

Finally, we set out to implement the drawing features into the New York portion of the site. Using either the standalone layer or the new layer within a group, a user is able to draw polygons, lines, and markers and edit/delete them as they please. After choosing to submit, they will be prompted with a few more editing options for these drawings, and finally submit after adding in a label, sourceId, and source label. These new layers can then be seen after a refresh.

The initial goal for this project was to add drawing features to the site, however, we managed to accomplish this and much more. From reinforcing the logic behind the site, preventing crashes and layers from failing to load; to creating new sub-domains as well as documentation to describe how our client can repeat this process; finally to adding new drawing features to create new layers, we were able to make significant progress on this site and create an easier transition to any new developers that might take on this project.

# Appendix

## Demo 1 Slides



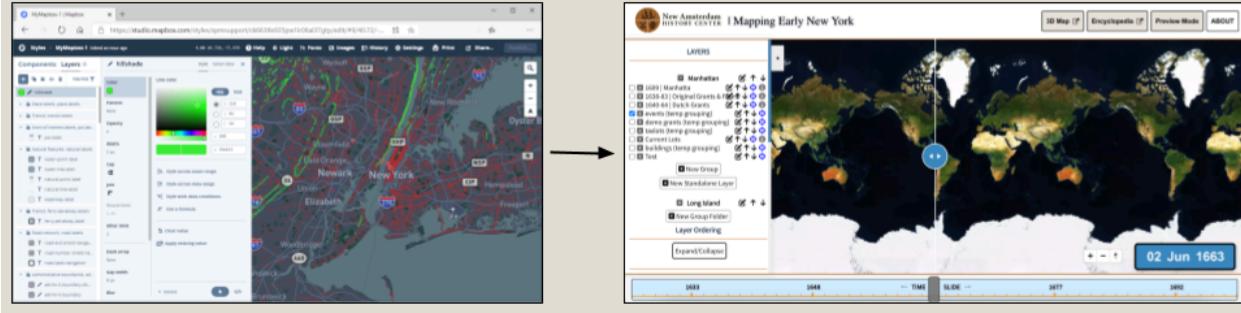
The screenshot shows a historical map interface titled "New Amsterdam HISTORY CENTER | Mapping Early New York". The main area displays a world map with a blue dot indicating the location of New York City. A sidebar on the left, titled "LAYSERS", contains a list of historical layers: Manhattan (selected), 1609 | Manhattan, 1638-83 | Original Grants & Farms, 1640 | Manhattan Grants, events (hemp grouping), demo grants (hemp grouping), taskslots (hemp grouping), Current Lots, buildings (hemp grouping), and Test. Below this is a section for "MAPS" with options for "Current Satellite" (selected) and "Manhattan and Surrounding". At the bottom, a timeline slider shows dates from 1633 to 1682, with "02 Jun 1663" highlighted. Navigation buttons for "3D Map", "Encyclopedia", "Sign In", and "ABOUT" are at the top right.

Aren Ashlock, Subham Bhattacharya, Michael Gelineau, Braeden Hegarty, Seth Watgen

Problem Being Addressed

# What is Our Project Solving?

- There aren't many options for creating interactive maps
  - Google My Maps, MapCreator.io
- MapStructor emphasizes customization
- Make map creation simpler



## Basic Features

- Choose an area to start building
- Import or draw map layers
- Share map with others and collaborate

# **Development Challenges and Choices**

## **Challenges**

- Discovering wants of client
- Catching up to previous group
- Picking up an existing project
  - Understanding structure, code, and functionality
  - Utilizing their code to fix bugs quick
- Working with MapBox GIS
- Communicating limitations/capabilities of project with the client

## **Choices**

- Spend time learning about the site and its functionality
  - Both new and old sites
  - Watching videos and talking with the client
- Begin working on bugs and assisting other developers
  - When done with one, pick up another
  - "All hands on deck" approach to identifying issues
- Keep an honest yet driven schedule with our client

## **Software Development Practices and Tools**

## Programming language and framework

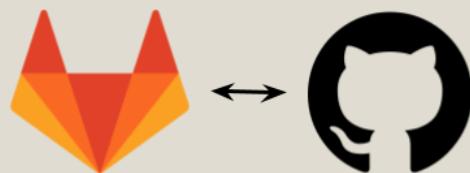


## Database and APIs



## Practices and Tools

- Agile development
- Require a minimum of 1 peer review before every merge
- Mirroring GitLab to GitHub
- JIRA Ticket Software
- MapBox API



# Current Progress and Challenges

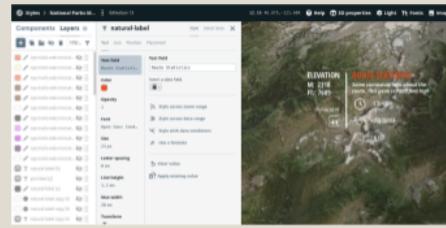
## Current Progress

- Understand the code better
  - New MENY
  - Basic Map Editor
- Initial bugs are mostly fixed
  - Keep discovering more bugs

#	Priority	Progress	Area	Status	W. Due Date	Completion Rate	Notes
1	Medium	In progress	Backend	Pending review		0%	After we implement this one, we will have to implement the same for the other two areas.
2	Medium	In progress	Backend	Pending review		0%	Balance the backlog item count between the three areas. We will start with the first one because it is the easiest.
3	Low	In progress	Frontend	Pending review		0%	Change the message to "no power in house".
4	Low	In progress	Frontend	Pending review		0%	Checkboxes are not checked off when there is no data appearing.
5	Medium	In progress	Backend	Pending review		0%	We are able to add and edit user input switch group only for events.
6	Low	In progress	Frontend	Pending review		0%	Zoom button should not work when it is grouped or for optional.
7	Low	In progress	Frontend	Pending review		0%	Info bubble model switches to red when it is selected.
8	Low	In progress	Frontend	Pending review		0%	Info bubble model switches to green when it is selected.
9	Low	In progress	Frontend	Pending review		0%	Lat/Long coordinates, pitch, bearing does not change.
10	Low	In progress	Frontend	Pending review		0%	Lat/Long coordinates, pitch, bearing does not change when moving slider.
11	High	In progress	Frontend	Pending review		0%	
12	High	In progress	Frontend	Pending review	Tue, Feb 13	0%	Changing position of link before (checkboxes are accepted).
13	High	In progress	Frontend	Pending review	Thu, Feb 15	0%	Link button changes from "link" to "links".
14	High	In progress	Frontend	Pending review	Thu, Feb 15	0%	Info type panels do not push down when new ones are added.
15	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	When clicking more than one type simultaneously, only one info type panel is shown.
16	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	"Type" field does not update when info of related locations.
17	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	"Type" field does not update when info of related locations.
18	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	Lat/Long Shrink & Stretch info window disappears (Q3, remaining bug).
19	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	Make info box completely identical to MENY.
20	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	Find location to move under power remaining just to see how.
21	Medium	In progress	Frontend	Pending review	Thu, Feb 15	0%	"Type" selection boxes info current or both info panel and separate one not appearing.

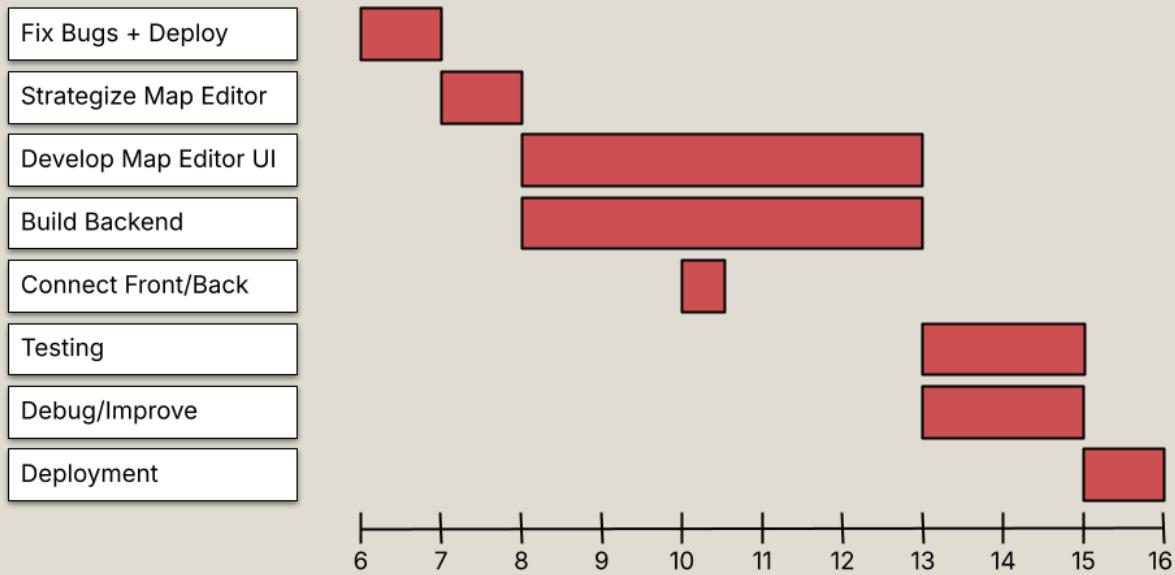
# Challenges

- Learning a new language (TypeScript)
- Lack of documentation in code
- Understanding MapBox GIS
- Discovering more bugs impedes the development timeline
- Application crashes when launched with VPN



## Remaining Work Timeline

# Timeline



Demo!

## Demo 2 Slides

# WebMap

New Amsterdam HISTORY CENTER | Mapping Early New York

LAYERS

- Manhattan
  - 1609 | Manhatta
  - 1638-83 | Original Grants & Farms
  - 1640-64 | Dutch Grants
  - Current Lots
  - 1643-75 | Lot Events
  - 1660 | Castello Taxlots
  - 1643-67 | Demo Grant Divisions: C7
  - Current Buildings
- Long Island

MAPS

- Current Satellite
- 1600 | Castello Plan
  - Castello Redrawn (Default)
  - 1660 Original Castello Plan
  - Stokes Key to Castello Plan
  - Legend of New Amsterdam

TIME SLIDE

01 Jan 1663

1633 1648 1660 1677 1692

Aren Ashlock, Subham Bhattacharya, Michael Gelineau, Braeden Hegarty, Seth Watgen

# Our Project

# Recap

- Map Creation Site ([maptructor.org](http://maptructor.org)) - Mapping New York
- Goal #1 - Build a stronger foundation for site
  - Fix bugs
  - Improve UI
  - Complete features
- Goal #2 - Replace old implementation of the site



(Original site)



(Development site)

## Current Goals

- Fix remaining bugs
- Make cosmetics/interactions identical



Up until 1.5 weeks ago

- 
- Separate instances for different maps
  - Create feature to draw maps
  - Add accounts
  - Allow users to select a map



Past 1.5 weeks and on

# Design and Implementation

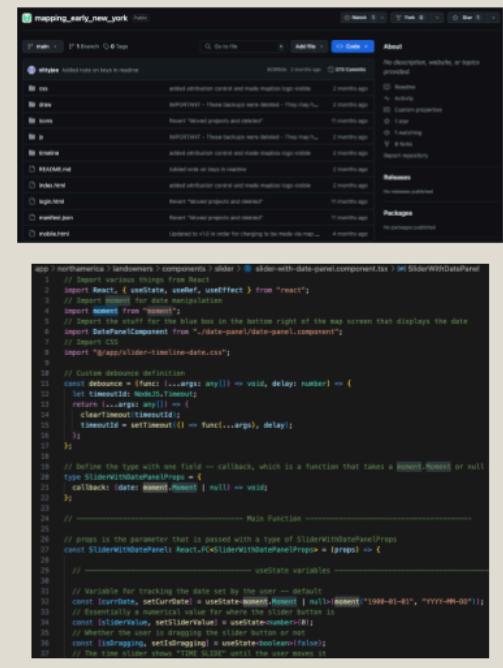
## Fixing the Bugs

How we solved them:

- Work together to learn interactions
- Comment the old code (and refactor)
- Check the console log for "breakpoints"

Challenges:

- Understanding the API
- Past design choices make them more difficult



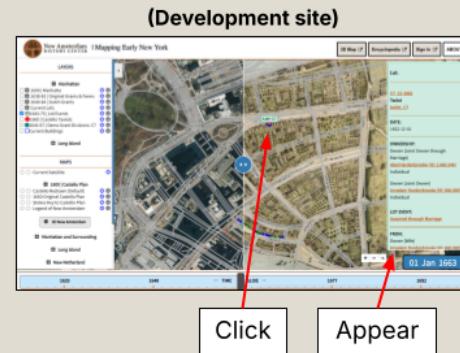
The screenshot shows a GitHub repository named 'mapping\_early\_new\_york' with a commit history and a code editor. The commit history lists several commits from various authors, mostly 'miketaylor', with dates ranging from 2 months ago to 4 months ago. The code editor displays a file named 'SliderWithDatePanelComponent.tsx' with approximately 30 lines of TypeScript code. The code includes imports for React, useState, useEffect, moment, and a date-panel component. It defines a function 'debounce' and a type 'SliderWithDatePanelProps'. The main logic involves setting a timeout ID and calling a callback function with the current date when the user stops dragging the slider.

```
app / northamerica / handovers / components / slider / > SliderWithDatePanelComponent.tsx [SliderWithDatePanel]
1 // Import various things from React
2 import React, { useState, useReducer, useEffect } from "react";
3 import { useReducer, useState } from "react";
4 import moment from "moment";
5 // Import the stuff for the blue box in the bottom right of the map screen that displays the date
6 import DatePanelComponent from "../date-panel/date-panel.component";
7 // Import the slider
8 import { "app/slider-timeline-date.vss" };
9
10 // Custom debounce definition
11 const debounce = (func: (...args: any[]) => void, delay: number) => {
12   let timeoutId: NodeJS.Timeout;
13   return (...args: any[]) => {
14     if (timeoutId) {
15       clearTimeout(timeoutId);
16     }
17     timeoutId = setTimeout(() => func(...args), delay);
18   };
19 }
20
21 // Define the type with one field -- callback, which is a function that takes a null or not null
22 type SliderWithDatePanelProps = {
23   callback?: (date: null | moment) => void;
24 };
25
26 // Main Function
27
28 // props is the parameter that is passed with a type of SliderWithDatePanelProps
29 const SliderWithDatePanel: React.FC<SliderWithDatePanelProps> = (props) => {
30   // useState variables
31
32   // Variable for tracking the date set by the user -- default
33   const [currentDate, setCurrentDate] = useState<moment>();
34   const [isDragging, setIsDragging] = useState<boolean>(false);
35   const [isSliderValue, setIsSliderValue] = useState<number>(0);
36
37   // Whether the user is dragging the slider button or not
38   const [isDragging, setIsDragging] = useState<boolean>(false);
39
40   // If the user changes their mind about the date
41   const [isDragging, setIsDragging] = useState<boolean>(false);
```

# Same Interactivity

Resources to help resolve:

- Tab between sites to notice differences
- Using the old site's code as reference



Challenges:

- Many layers of CSS building
- Timing, sizing, colors had to match
- Dynamic elements depending on window size

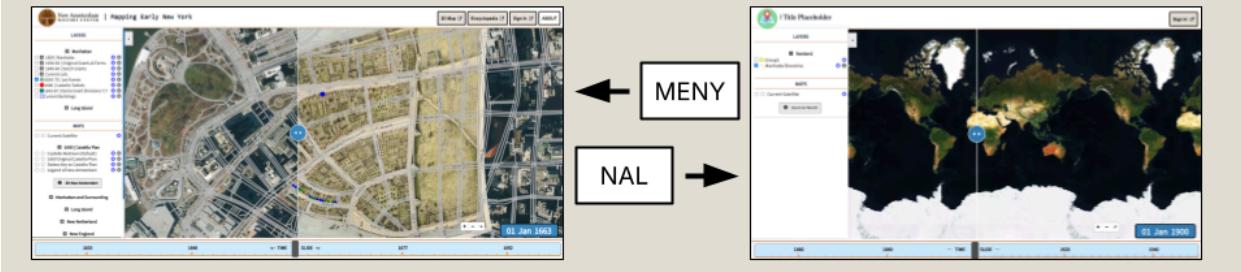


# Separate Instances

Client: Wanted different domain paths

Solution: Restructure the code

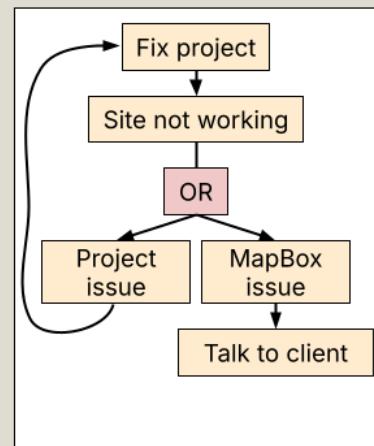
- Determine what code needed to be separate
- Build new backend (and generate prisma)
- Document process for future implementations



# Separate Instances

Challenges:

- Next.js API fetches are specific
- Incorporating multiple environment variables
- Building different prisma schema
- Ensuring project is functional on Vercel
- Testing locally vs deployment
- Understanding if issues are MapBox or code
- Lots of little changes needed



```
12:32:58.133 This happened because Prisma Client was generated for "darwin-arm64", but the actual deployment required "x64-openssl-3.0.x".
12:32:58.133 Add "x64-openssl-3.0.x" to binaryTargets in the "schema.prisma" file and run prisma generate after saving it:
12:32:58.133
12:32:58.133 generator client {
12:32:58.133   provider      = "prisma-client-js"
12:32:58.133   binaryTargets = ["native", "x64-openssl-3.0.x"]
12:32:58.134 }
```

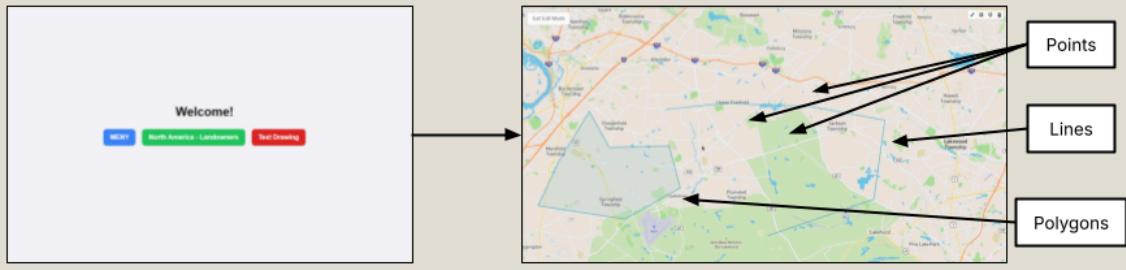
Prisma Issues

# Drawing

Client: Ability to draw maps using tools

What we have learned so far:

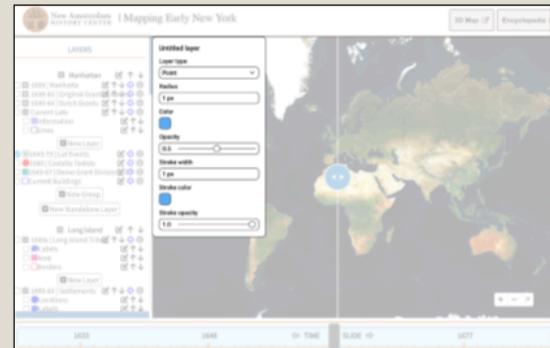
- Development page for research/experimenting with new drawing features
- Ability to add markers, lines, polygons



# Drawing

Early thoughts:

- Create a new collection
  - Just stores drawing information
  - GeoJSON
- Connect user accounts to drawing
  - Already implemented, just expand
  - Include registration
  - Only let creators see data they make



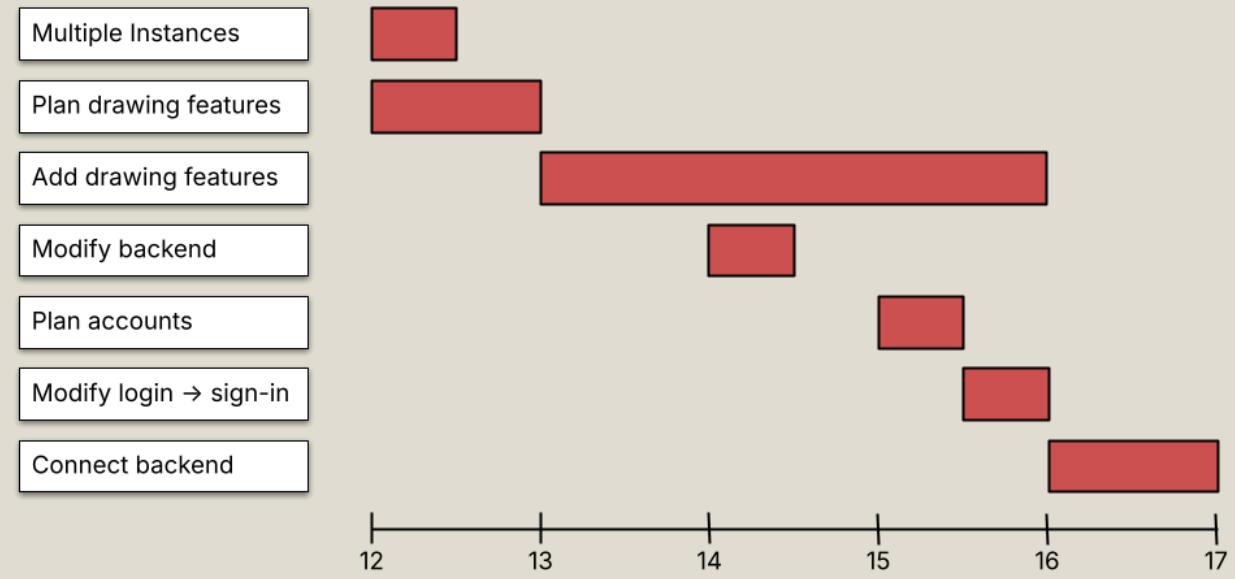
Layer name			
Label	Latitude	Longitude	Owner (1)
Lot C7	40.70	87.012	Mark Roberts
Lot H4	76.014	60.708	Steve Richard

Make this layer visible when the page loads

Pin color: #38A170 | Pin opacity: 100%  
Pin radius: 2px | Pin blur: 0

## Remaining Work Timeline

# Timeline



# Demo!

## WebMap

Aren Ashlock, Subham Bhattacharya, Michael Gelineau, Braedon Hegarty, Seth Watjen

<https://youtu.be/P3KrNbE0vMc>

## Demo 3 Slides

# WebMap

Aren Ashlock, Subham Bhattacharya, Michael Gelineau, Braeden Hegarty, Seth Watgen

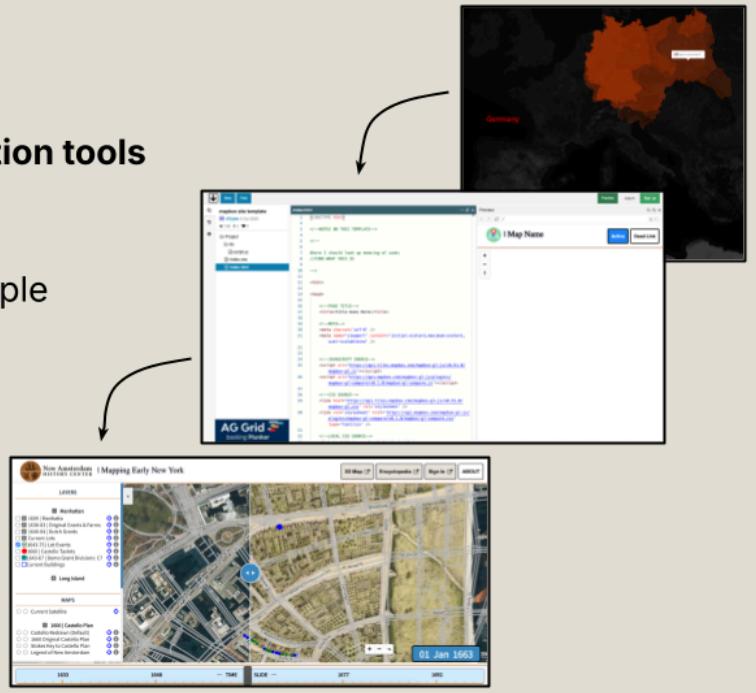
# Problem Being Addressed

# Problem

## Lack of existing map creation tools

Current process:

1. Create map using multiple software
2. Start a website
3. Upload data via code
4. Make data look good

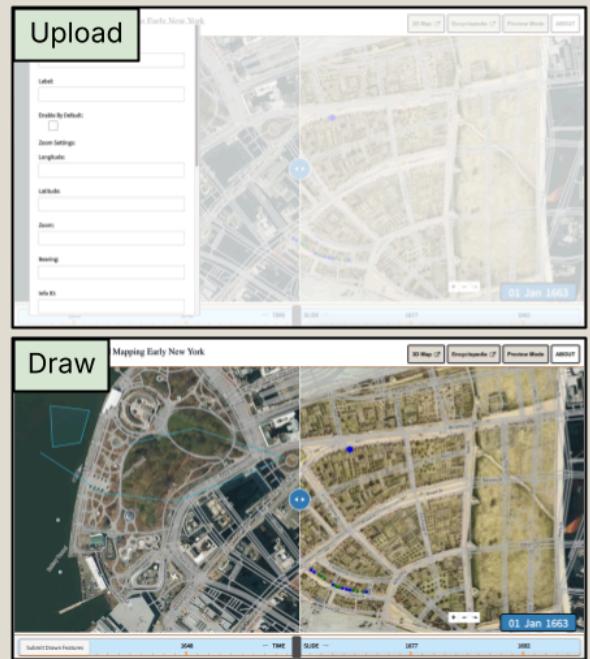


# Our Solution

## Streamline the Process

Goal:

1. Have our existing site
2. Users can add data
3. Users can draw directly on map
4. Share/work on maps with others



# Design Challenges/Choices

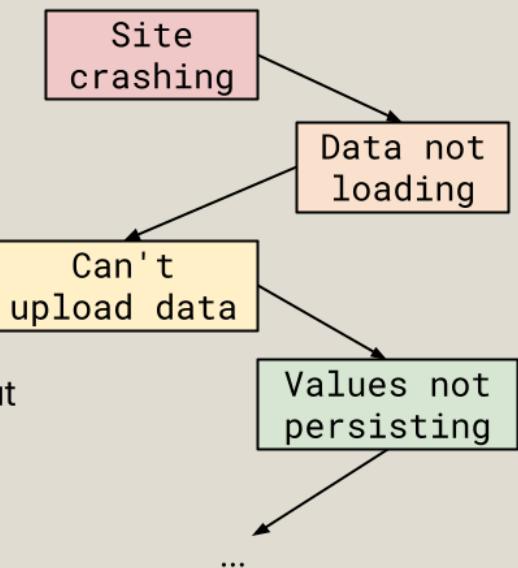
## Challenge - Buggy Code

### Issues:

- Existing bugs affected development
- Solving certain bugs would reveal others

### Solution:

- "All hands on deck" approach
- Consistent feedback from client about functionality



# Challenge - MapBox API

## Issues:

- Unsure what properties of the API mean
- Documentation is difficult to understand

The screenshot shows the MapBox API Reference for the Map object. It includes sections for Properties and options, Methods and controls, Interaction handlers, Events, and event types, and Animation and styling. A detailed description of the Map object is provided, along with examples and troubleshooting tips.

## Solution:

- Refer back to old code to understand the API
- Incorporate code written by client's colleagues

The screenshot shows a GitHub commit history for the 'mapping\_early\_new\_york' repository. It lists several commits from different authors, each adding specific logic or features to the project. A callout box labeled 'Existing code' points to this screenshot.

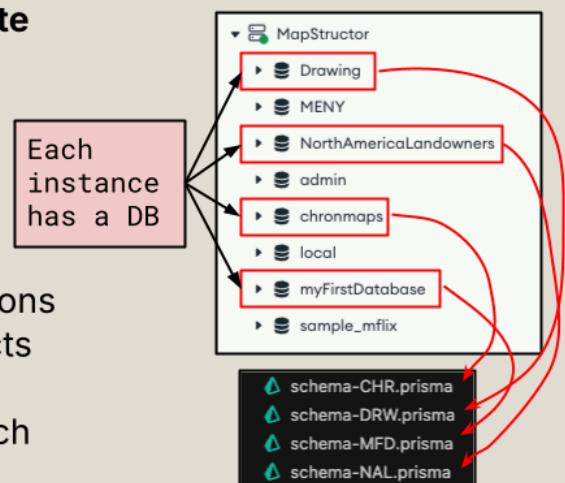
# Choice - Multiple DBs

## Choice: Separate instances = separate databases

- Environment variables include database connection URIs

## Solution: Split the prisma schema

- Allows multiple database connections
- Flexibility to change schema objects per page
- Modify build script to generate each time



# Choice - Draw Feature

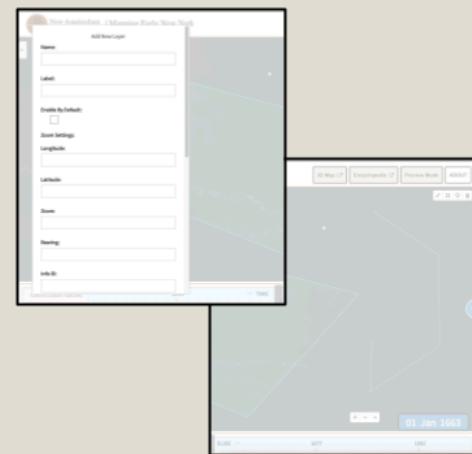
## Choice: How to implement and get data

- Data type is GeoJSON rather than vector

```
{ "type": "FeatureCollection", "features": [{ "type": "Feature", "geometry": { "type": "Point", "coordinates": [-93.6164, 41.5896] }, "properties": { "name": "Des Moines, Iowa" } } ] }
```

## Solution: Utilize existing UI

- Minimizes the collections in database
- Keep data similar
- Info box is already functional
- Able to tie drawing info to existing properties



# Software Development Practices and Tools

## Programming language and framework



## Database and APIs



## Tools

Jira:

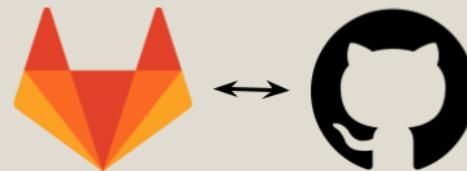
- To-Do → In Progress → Ready to Merge → Done
- Point estimates

GitLab → GitHub:

- Mirror repositories
- Client had a public repository beforehand

The Jira screenshot shows a Kanban board with three columns: To Do, In Progress, and Ready for Merge. Each column contains several tasks represented by cards with titles and descriptions. The 'In Progress' column has the most cards, indicating active development work.

Column	Task Description	Status
To Do	BUG: Be able to add stand alone event to device group (eg Lot events)	In Progress
In Progress	BUG: Lot Events Info panel does not change when moving older	In Progress
In Progress	BUG: Make site look identical to Mappy	In Progress
In Progress	BUG: Hide lower panel and right info bar when map is clicked	In Progress
In Progress	Create mockups of the page layout for drawing	In Progress
In Progress	Update the login page for users who have never logged in (that is possible and update the form)	In Progress
In Progress	Modify the timeline older so it doesn't update until the mouse is let go	In Progress
In Progress	Display GeoJSON Data from DB	In Progress
In Progress	BUG: globalwebmapppper MapBox data won't load	In Progress
In Progress	Add drawing features when creating new layer	In Progress
Ready for Merge	Create pop-up when adding layers to a map that are drawing and current default	Ready for Merge
Ready for Merge	DISCOVERY: Mess around with coding markers to a map. No final product needed, just for research.	Ready for Merge
Ready for Merge	Add Layers to Groups in NAL	Ready for Merge
Ready for Merge	BUG: Eventually default does not display now on a stand-alone layer	Ready for Merge
Ready for Merge	BIG: Zoom levels for layer's line thicknesses do not persist	Ready for Merge
Done	---	Done
Done	---	Done
Done	---	Done



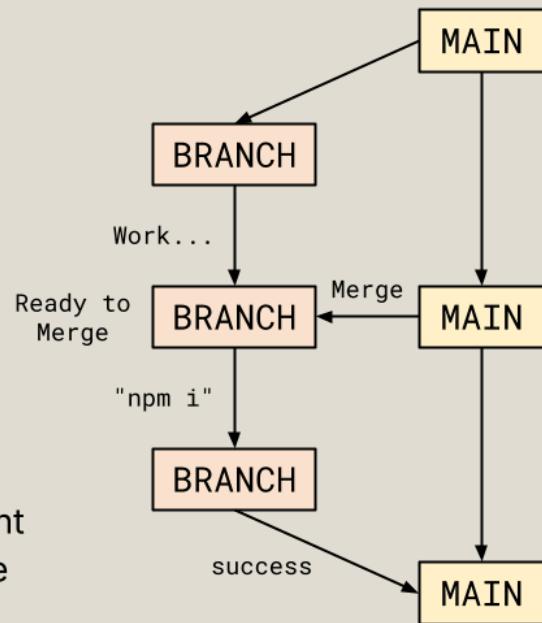
# Practices

SCRUM:

- Daily standups
- Demo = sprint

Git:

- MASTER branch as archive
- Branch per feature/bug
- Merge main into branch first
- Run "npm i" to ensure deployment
- Minimum of 1 reviewer per merge



All We Did

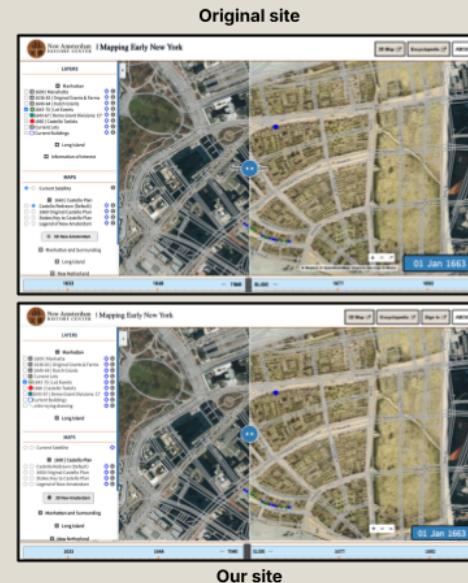
# Bugs/Work From The Last Group

Bugs:

- Site doesn't crash
- Data loading/manipulation is functional and more reliable
- Styling and interactions are more similar

Standalone Layers:

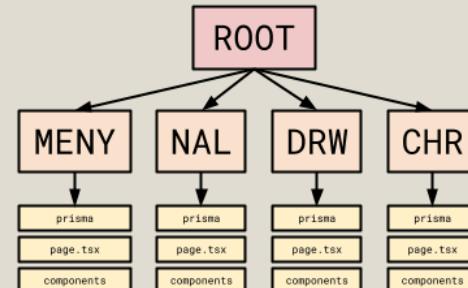
- New feature left from last group
- Layers can exist outside of groups



# Making The Project Cloneable

Project Restructuring:

- Determined what files/folders need separated
- Modified prisma loading and build scripts



Documentation:

- Steps to follow
- Video demonstration
- Assisting client with their attempt



# Drawing Functionality (Pt 1)

- Initial experimentation
  - Separate page dedicated to learning how to implement drawing features
- Transition to main site
  - Adjust current map instance so it can be shared across React components
    - Shared MapContext file allows us to share the Mapbox instances
    - Wrap HTML element of the page.tsx file with <MapContext.Provider> tag to allow any component nested in this tag to access the map
  - Access MapContext through layer components to allow drawings
  - Turn drawings into GeoJSON data for submission

# Drawing Functionality (Pt 2)

- Experimenting with MapBox Adding GeoJSON as Source
  - Discovered existing API methods worked for JSON in correct format
- Research GeoJSON format to ensure proper data read
  - GeoJSON Files have numerous possible data types and structures
  - Built out data format files and passed them to the frontend devs once I was able to display from the database
- Integrate GeoJSON style layers into our existing edit form
  - Used existing API routes
  - Once the layer is drawn, the type and GeoJSON data are auto-set in form
  - Code parses the layer 'link' as a JSON object to display the layer on the frontend

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "coordinates": [
          [
            [
              -93.84032280113189,
              41.742399076776388
            ],
            [
              -93.43429904334857,
              41.47455894431778
            ]
          ],
          "type": "LineString"
        ]
      }
    }
  ]
}

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "coordinates": [
          [
            [
              -93.77764561192302,
              41.54149293366018
            ],
            [
              -93.4427495751854,
              41.54149293366018
            ],
            [
              -93.4427495751854,
              41.69524668931314
            ],
            [
              -93.77764561192302,
              41.69524668931314
            ],
            [
              -93.77764561192302,
              41.54149293366018
            ]
          ],
          "type": "Polygon"
        ]
      }
    }
  ]
}
```

## **What We Are Most Proud Of...**

1. Fixed some VERY MAJOR issues with the website

-----

2. Added the drawing features as a starting point

## **Challenges Encountered**

# Challenges Encountered

Lack of handoff from last group:

- Hard to understand what they did
- Bugs were more difficult → took longer
- Features were left unimplemented (standalone layers)

Issue	Priority	Progress	Area	Description	Comments
1. Initial UI design, wireframes, and user flow	High	In Progress	Frontend	Initial UI design, wireframes, and user flow	UI design and wireframes are mostly complete. Some minor refinements are still needed.
2. Frontend code to handle user input and validate form data	Medium	In Progress	Frontend	Frontend code to handle user input and validate form data	Frontend code is being developed. Some bugs are found and need to be fixed.
3. Backend API to store user data and handle authentication	Medium	In Progress	Backend	Backend API to store user data and handle authentication	Backend API is being developed. Some database integration issues are found.
4. Database schema design and normalization	Medium	In Progress	Backend	Database schema design and normalization	Database schema is being designed. Some normalization issues are found.
5. Integration of payment gateway (e.g. Stripe)	Medium	Pending Review	Frontend	Integration of payment gateway (e.g. Stripe)	Payment gateway integration is pending review. Some security concerns are found.
6. User role management and permissions	Medium	Pending Review	Backend	User role management and permissions	User role management and permissions are pending review. Some access control issues are found.
7. Performance optimization of the system	Low	Pending Review	System	Performance optimization of the system	Performance optimization is pending review. Some scalability issues are found.
8. Documentation and README file	Low	Pending Review	System	Documentation and README file	Documentation and README file are pending review. Some clarity issues are found.
9. Configuration of AWS Lambda functions	Low	Pending Review	System	Configuration of AWS Lambda functions	AWS Lambda configuration is pending review. Some deployment issues are found.
10. CI/CD pipeline setup	Low	Pending Review	System	CI/CD pipeline setup	CI/CD pipeline setup is pending review. Some integration issues are found.
11. Unit tests and integration tests	Low	Pending Review	System	Unit tests and integration tests	Unit tests and integration tests are pending review. Some coverage issues are found.
12. Deployment scripts and automation tools	Low	Pending Review	System	Deployment scripts and automation tools	Deployment scripts and automation tools are pending review. Some complexity issues are found.
13. Monitoring and logging setup	Low	Pending Review	System	Monitoring and logging setup	Monitoring and logging setup is pending review. Some visibility issues are found.
14. Security audit and review	Low	Pending Review	System	Security audit and review	Security audit and review is pending review. Some vulnerabilities are found.
15. Performance monitoring and analysis	Low	Pending Review	System	Performance monitoring and analysis	Performance monitoring and analysis is pending review. Some trends are found.
16. User feedback and bug reports	Low	Pending Review	System	User feedback and bug reports	User feedback and bug reports are pending review. Some user requirements are found.
17. Feature requests and backlog	Low	Pending Review	System	Feature requests and backlog	Feature requests and backlog are pending review. Some new features are proposed.
18. Configuration of AWS Lambda functions	Low	Pending Review	System	Configuration of AWS Lambda functions	AWS Lambda configuration is pending review. Some deployment issues are found.
19. CI/CD pipeline setup	Low	Pending Review	System	CI/CD pipeline setup	CI/CD pipeline setup is pending review. Some integration issues are found.
20. Deployment scripts and automation tools	Low	Pending Review	System	Deployment scripts and automation tools	Deployment scripts and automation tools are pending review. Some complexity issues are found.
21. Monitoring and logging setup	Low	Pending Review	System	Monitoring and logging setup	Monitoring and logging setup is pending review. Some visibility issues are found.
22. Security audit and review	Low	Pending Review	System	Security audit and review	Security audit and review is pending review. Some vulnerabilities are found.
23. Performance monitoring and analysis	Low	Pending Review	System	Performance monitoring and analysis	Performance monitoring and analysis is pending review. Some trends are found.
24. User feedback and bug reports	Low	Pending Review	System	User feedback and bug reports	User feedback and bug reports are pending review. Some user requirements are found.
25. Feature requests and backlog	Low	Pending Review	System	Feature requests and backlog	Feature requests and backlog are pending review. Some new features are proposed.

Technical unfamiliarity:

- Next.js project structure
- Mapbox (API & map creating)



# Contributions

**Aren:** Bugs - Site crashing, Layers not loading, Hash parameters (url, initial zoom), Timeline slider lag, Page styling (buttons, modals, sections open by default); Project Restructuring - Whole thing (restructure, documentation)

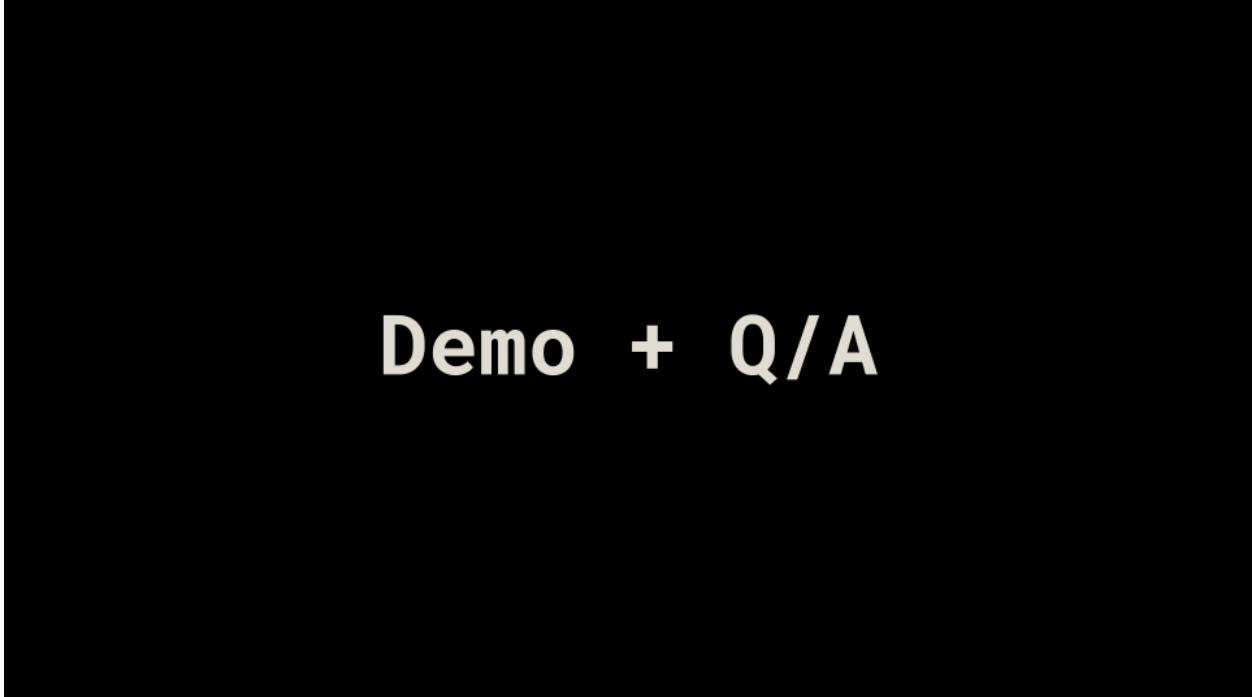
**Braeden:** Bugs - Standalone layer implementation, CSS adjustments (page styling), Documentation, Layers not appearing; Drawing Features - Research backend adjustments, GeoJSON formatting, integrate drawn layers into existing API methods

**Michael:** Bugs - Enable by default function, Lot events displayed by default, Lot Event render issue, Layers not appearing, White screen crash, Zoom buttons for layers not appearing in groups, Enable by default not displaying as standalone layer

New Features - CSS changes and functionality identical to original site (nahc-mapping), Initial Drawing research New layer dialog, Drawing features on new layer front end development.

**Seth:** Bugs - various tasks; Drawing Features - various tasks

**Subham:** Bugs - Slider hold feature, info box scroll, made data load faster in the northamerica/landowner page, info box appearance, CSS adjustments, Documentation; Drawing Features - Research, Added markers and polygons.



**Demo + Q/A**