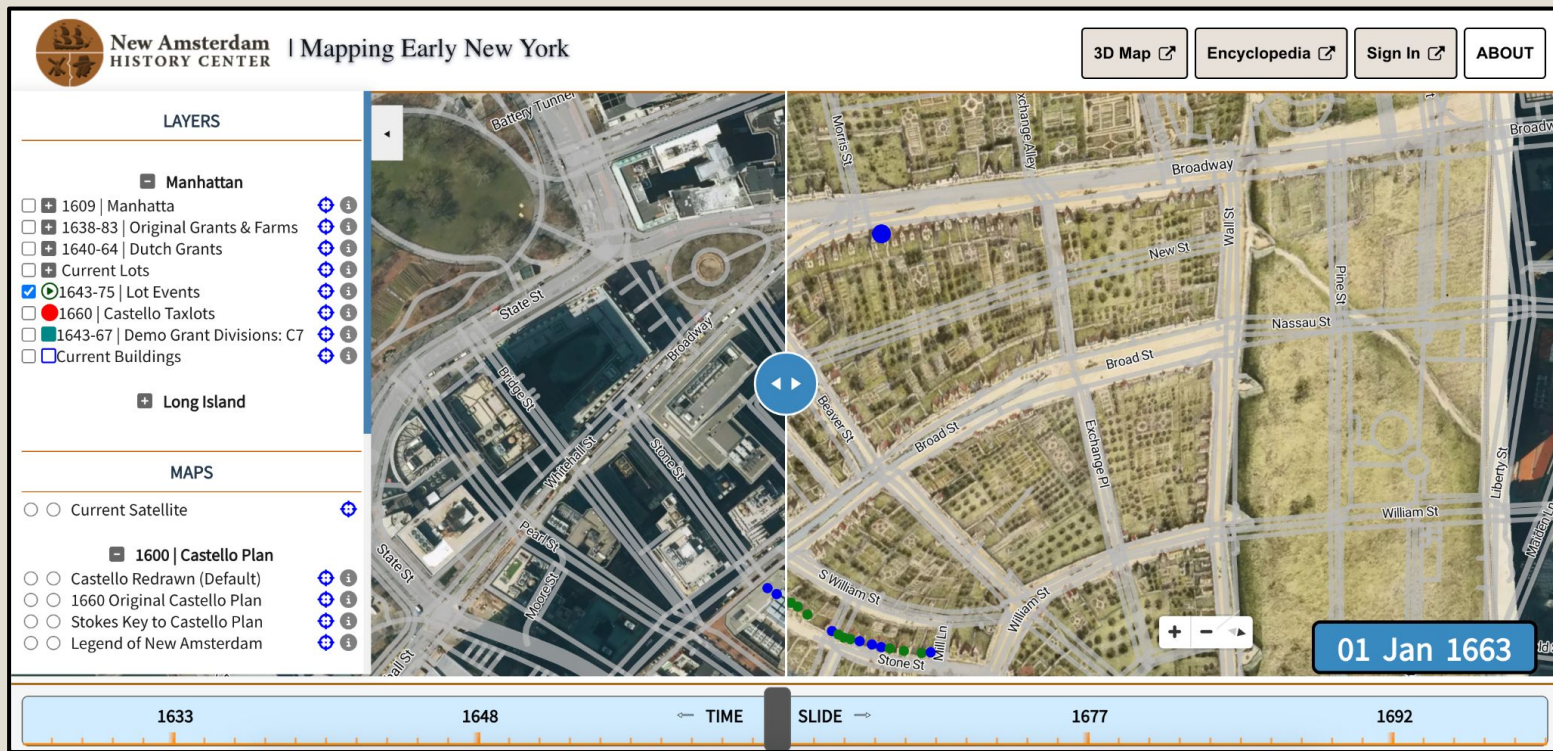


# WebMap

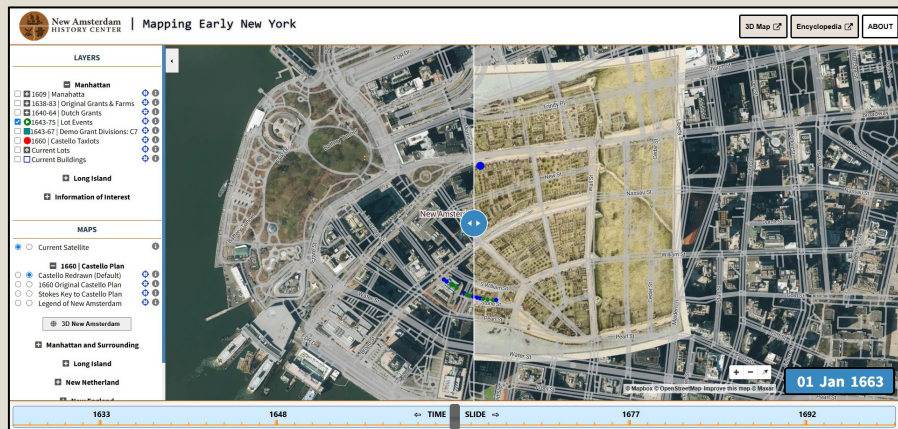


Aren Ashlock, Subham Bhattacharya, Michael Gelineau, Braeden Hegarty, Seth Watgen

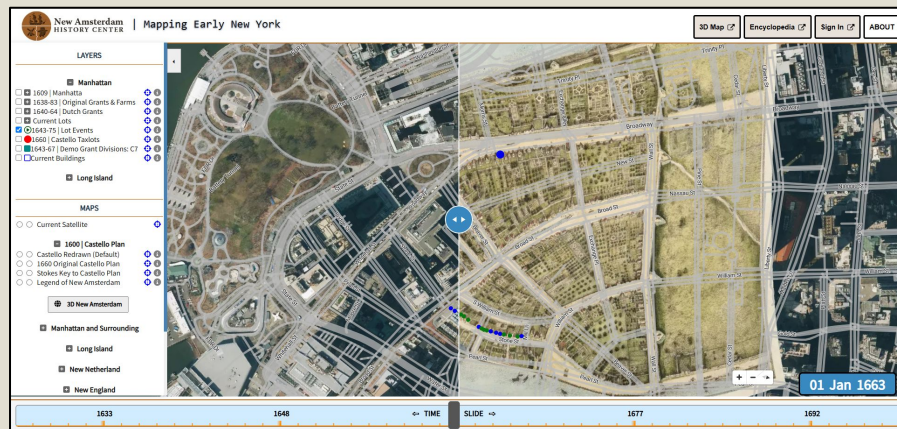
# Our Project

# 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

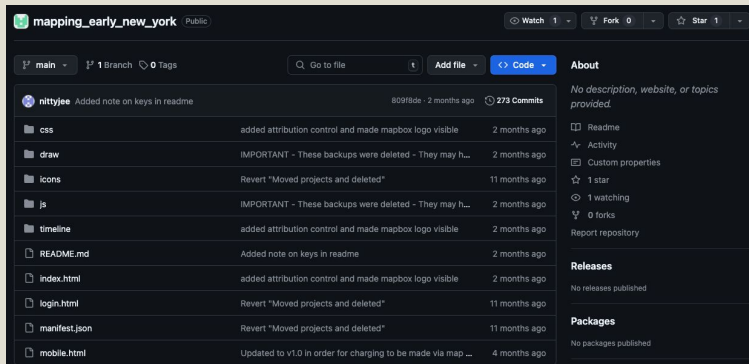
# Fixing the Bugs

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



```
app > northamerica > landowners > components > slider > slider-with-date-panel.component.tsx > @ SliderWithDatePanel
1 // Import various things from React
2 import React, { useState, useRef, useEffect } from "react";
3 // Import moment for date manipulation
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 CSS
8 import "@app/slider-timeline-date.css";
9
10 // Custom debounce definition
11 const debounce = (func: (...args: any[]) => void, delay: number) => {
12   let timeoutId: NodeJS.Timeout;
13   return (...args: any[]) => {
14     clearTimeout(timeoutId);
15     timeoutId = setTimeout(() => func(...args), delay);
16   };
17 };
18
19 // Define the type with one field -- callback, which is a function that takes a moment.Moment or null
20 type SliderWithDatePanelProps = {
21   callback: (date: moment.Moment | null) => void;
22 };
23
24 // ----- Main Function -----
25
26 // props is the parameter that is passed with a type of SliderWithDatePanelProps
27 const SliderWithDatePanel: React.FC<SliderWithDatePanelProps> = (props) => {
28
29   // ----- useState variables -----
30
31   // Variable for tracking the date set by the user -- default
32   const [currDate, setCurrDate] = useState<moment.Moment | null>(moment("1900-01-01", "YYYY-MM-DD"));
33   // Essentially a numerical value for where the slider button is
34   const [sliderValue, setSliderValue] = useState<number>(0);
35   // Whether the user is dragging the slider button or not
36   const [isDragging, setIsDragging] = useState<boolean>(false);
37   // The time slider shows "TIME SLIDE" until the user moves it
```



# Same Interactivity

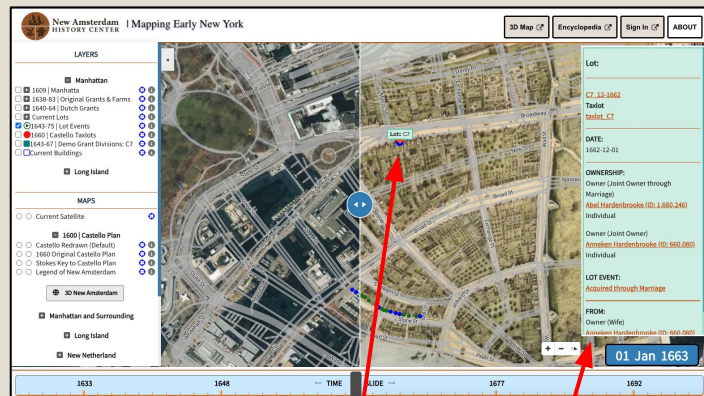
Resources to help resolve:

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

Challenges:

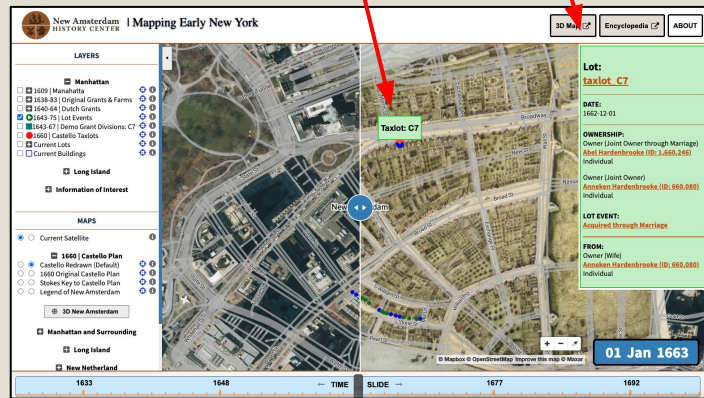
- Many layers of CSS building
- Timing, sizing, colors had to match
- Dynamic elements depending on window size

(Development site)



Click

Appear



(Original site)

# 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

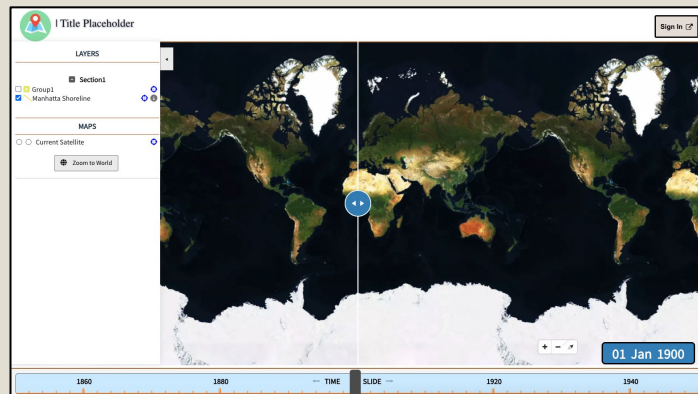
```
app
└─ page.tsx
```

```
app
├─ mappingNY
├─ page.tsx
├─ northamerica/landowners
└─ page.tsx
```



MENY

NAL

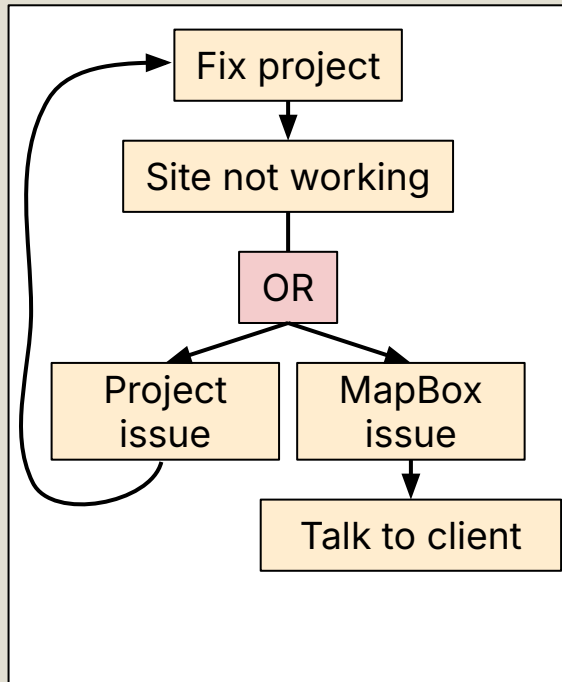




# Separate Instances

## 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



```
12:32:30.133 This happened because Prisma Client was generated for "darwin-arm64", but the actual deployment required "rhel-openssl-3.0.
12:32:30.133 Add "rhel-openssl-3.0.x" to 'binaryTargets' in the "schema.prisma" file and run 'prisma generate' after saving it:
12:32:30.133
12:32:30.133 generator client {
12:32:30.133   provider      = "prisma-client-js"
12:32:30.133   binaryTargets = ["native", "rhel-openssl-3.0.x"]
12:32:30.134 }
```

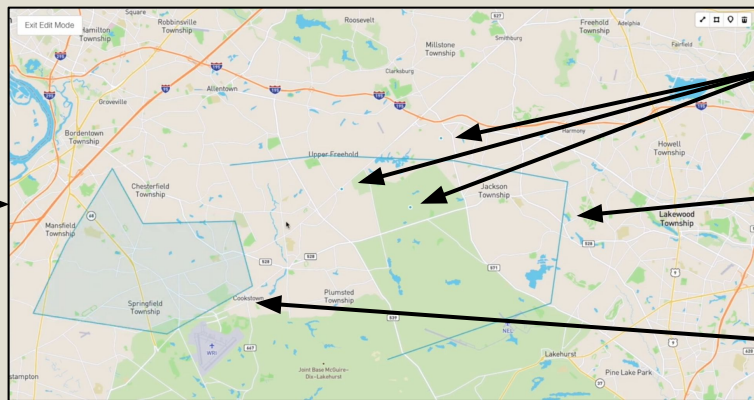
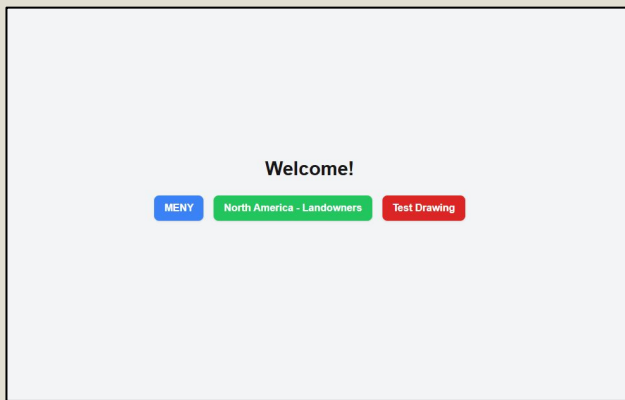
## Prisma Issues

# 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



Points

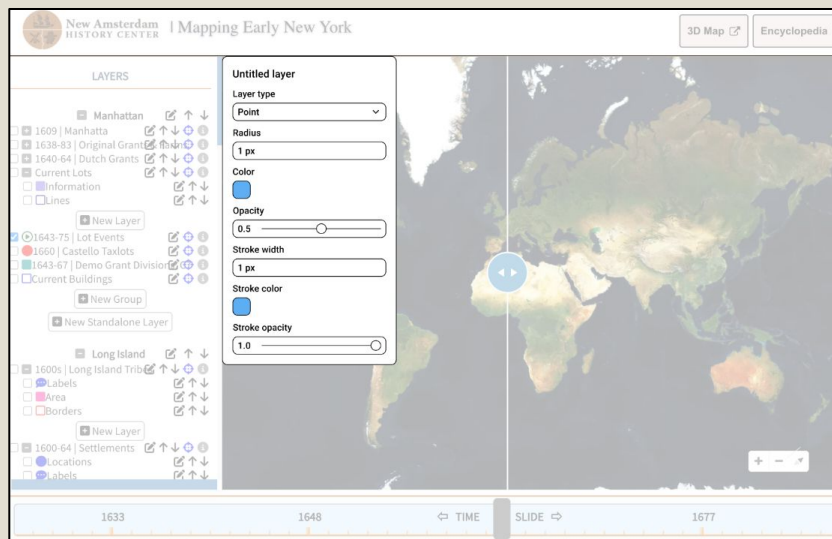
Lines

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

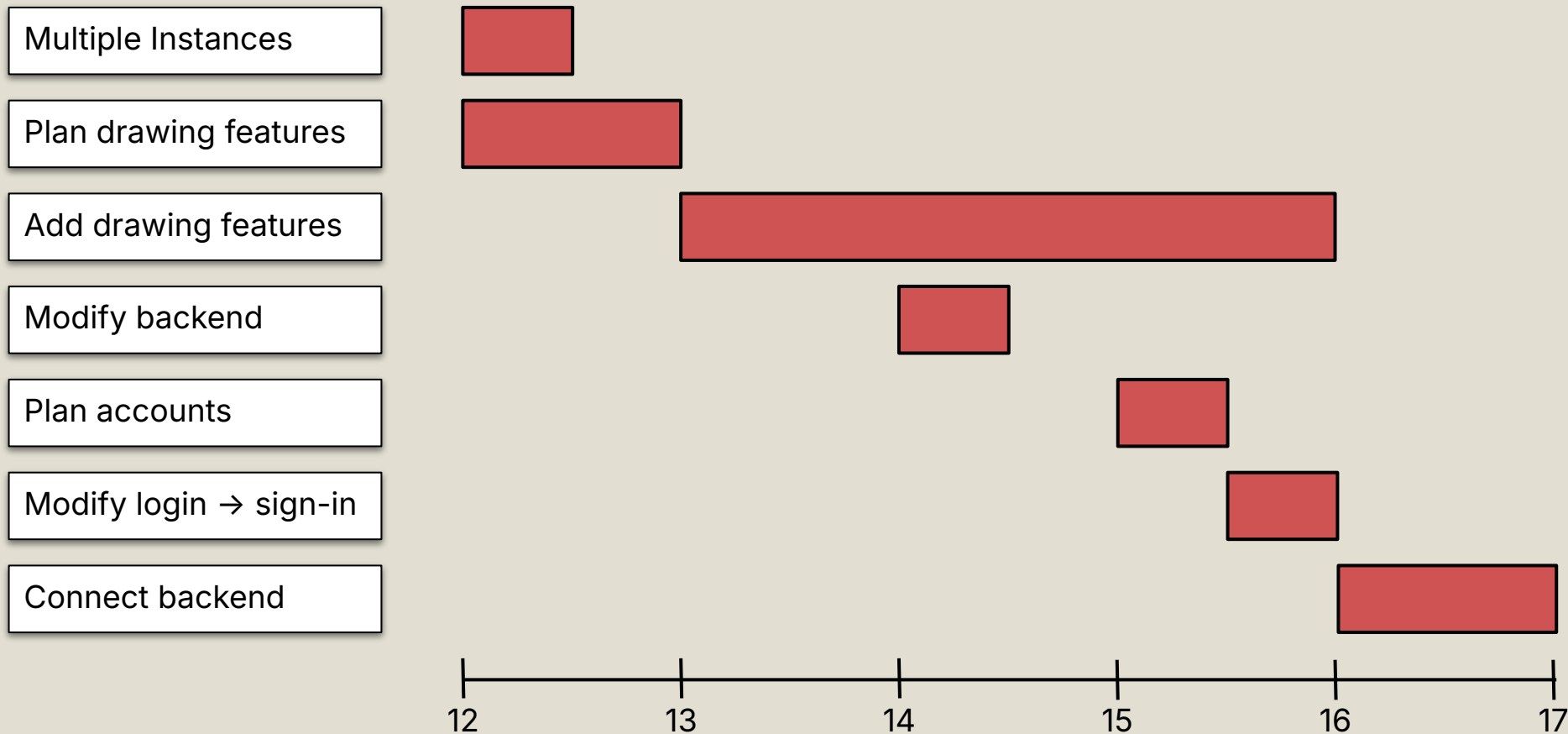
☒ Make this layer visible when the page loads

Pin color Pin opacity

Pin radius Pin blur

# Remaining Work Timeline

# Timeline





**Demo !**

# WebMap



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

[https://youtu.be/yx\\_9zTAlpuo](https://youtu.be/yx_9zTAlpuo)