

Fritz

## Authoring Tool for Visual Piling

Students:

- Emily Gagne, *Emily.Gagne001@umb.edu*
- Tung Duong, *Tung.Duong002@umb.edu*
- Kanthraj Shantharam, *K.Shantharam001@umb.edu*
- Josh Schultz, *josh.schultz001@umb.edu*

Instructor:

- Asst. Prof. Daniel Haehn, PhD, *daniel.haehn@umb.edu*

Teaching Assistants:

- Chengjie Zheng
- Tianyu Kang
- Jahnavi Leburu
- TAs contact: *staff@cs410.net*

Client:

- Fritz Lekschas, *lekschas@seas.harvard.edu*

Final Project Documentation v1.0 05/20/2022

# 1 Introduction

## 1.1 Purpose

The purpose of this software project is to provide a tool that allows analysts to use visual piling to analyzing visual data with minimal technical knowledge. The software will provide tools to allow a Visualization Designer to easily create the desired workflow, which then allows the Analyst to import the visual data and analyze it.

## 1.2 Scope

The tentative name for this product is *Authoring Tool for Visual Piling*. This software will allow a visualization designer to create an environment for an analyst to view and organize the visual data needing to be analyzed. The visualization designer will be able to pick from pre-made configuration options and will also be able to write supplemental code to enable custom use of a desired visualization library such as D3. The analyst will then be able to use the software with only the settings necessary for their data, allowing the analysis work to be performed more efficiently. The software will benefit analysts when looking at large amounts of small groups of visual data. The analyst will be able to quickly sort through the data with minimal distractions from extra settings and focus on the actual analysis work. An analysis team will be able to set this tool up with no programming knowledge.

## 1.3 Overview

The Authoring Tool for Visual Piling is an interactive tool which makes it easy to both visualise and customize large scale data sets. Whether someone wishes to simply sort a large amount of visual data by a specified parameter, or stacking images into specific groups. It allows designers to create custom templates in order to easily save their desired data visualization/rendering settings via a user-friendly sidebar. Designers can also preview a single instance of their data according to their current settings in real time, to provide a better experience in achieving the perfect settings. These settings can be imported by anyone, including designers or analysts, without any coding, saving lots of time and effort.

### 1.3.1 Existing Work

- MultiPiles: <https://aviz.fr/~bbach/multipiles/>

### 1.3.2 Limitations

Computer hardware (RAM, CPU or GPU) may be a foreseeable limitation in this project. However there is a low chance of this being a serious issue and it appears that all of the other tasks we have been assigned to implement are possible without foreseeable limits.

## 1.4 Definitions

- HTML: Hypertext Markup Language
- CSS: Cascading Style Sheets
- REPL: Read-Evaluate-Print Loop
- JSON: JavaScript Object Notation

## **2 Requirements**

- User Friendly Visualization Designer
- Exportable & Importable Templates
- Real Time Single Data Item Preview

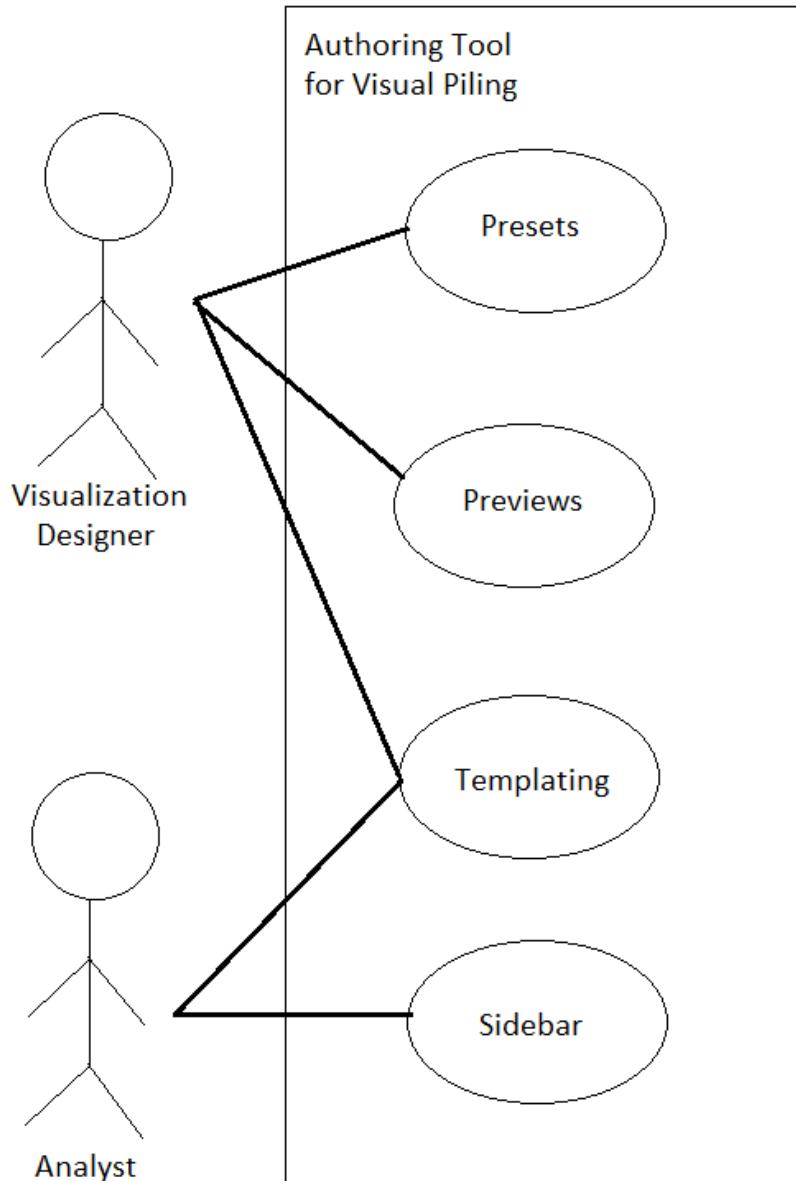
## **3 Specifications**

- Software must be able to implement direct manipulation of the look of the data by adding check boxes, sliders, radio buttons along with code editor.
- Software must be able to choose different templates while importing data from a third party service.
- Software must be able to render preview of single data and display it in new tab of output screen.

## **4 Design**

### **4.1 Use Cases**

- Visualization Designer: this user creates a workflow within the software that enables the desired data to be analyzed most effectively.
- Analyst: this user imports visual data and analyzes it by grouping data that is visually similar together into piles.



**Figure 1: Use Case Diagram**

The Visualization Designer uses Presets to determine how the data will appear. The Visualization Designer then uses Previews to see a sample of how each piece of data will appear. Once satisfied with the settings, the Visualization Designer then uses the Templating feature to export the code. The Analyst then uses the exported code via GitHub Gists to analyze the data. The Analyst can use settings via the Sidebar to make adjustments to how the data is displayed.

## 4.2 Architecture

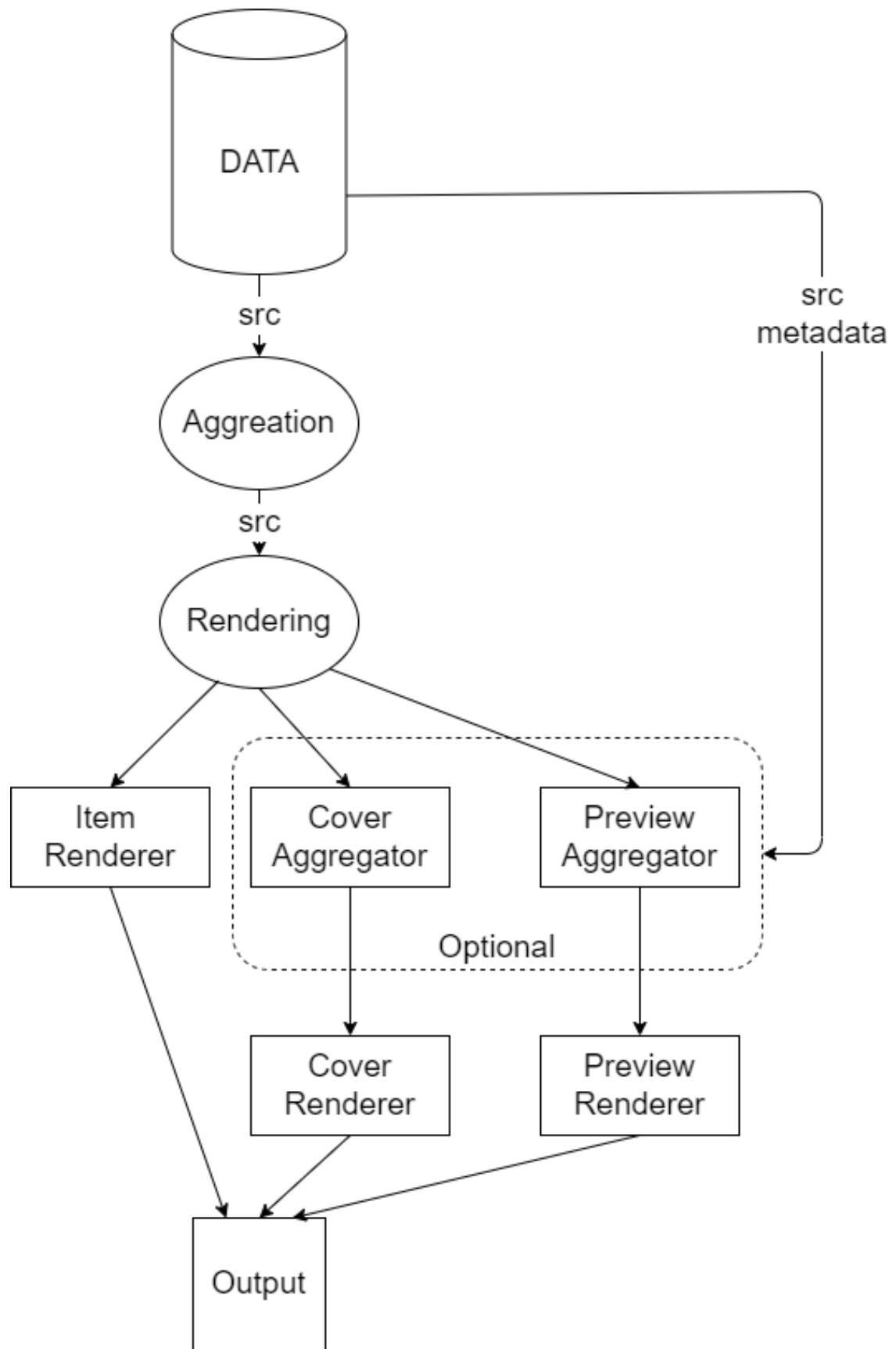


Figure 2: Architecture Diagram

### 4.3 Workflows

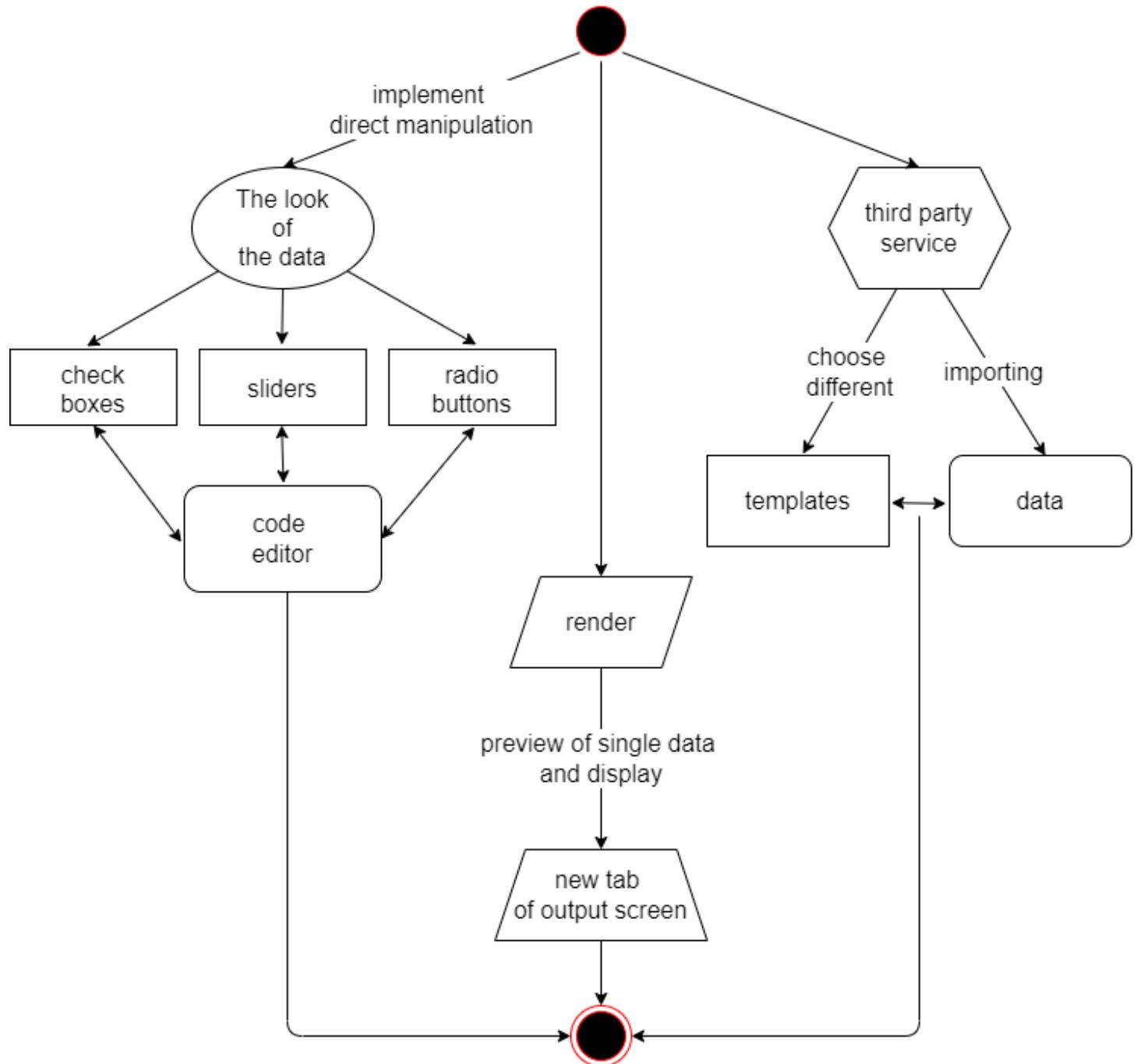


Figure 3: Workflows Diagram

#### 4.4 Technologies and Implementation Details

This project will be using the following technologies for frontend:

- CSS
- Svelte

The following technologies will be

used for backend:

- JavaScript
  - JSON
  - WebGL
- PixiJS
  - PilingJS

These technologies have been chosen by the client and are already used in the existing prototype software.

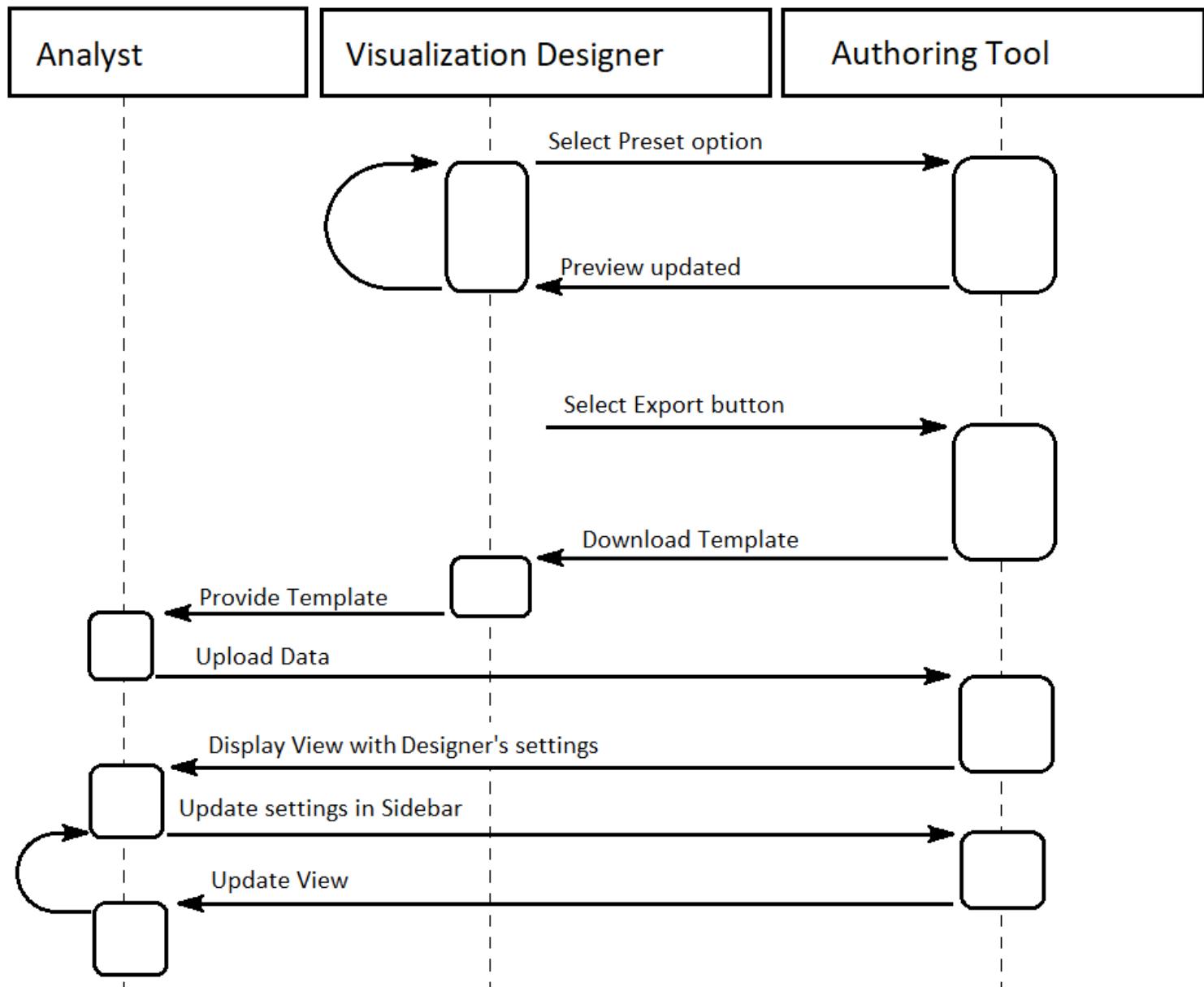
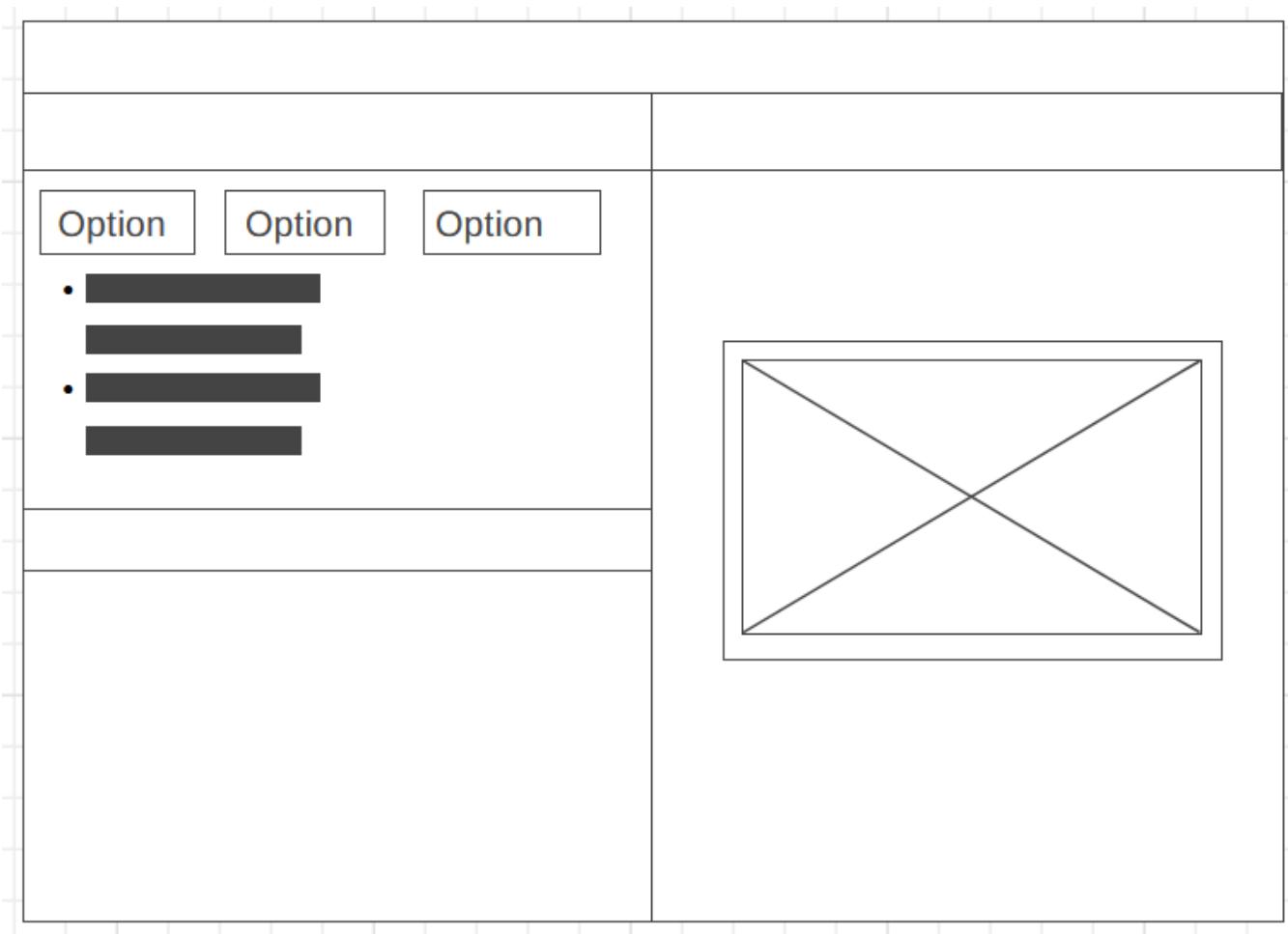


Figure 4: Technologies Diagram

## 4.5 User Interface

### Presets:



**Figure 5: UI Presets**

The Visual Designer will have some preset options available to them when designing the view that the analyst will see. These options will apply to the Renderer, Aggregator, Styles, Grouping, and Arrangement settings. The options will appear in a section above the code editor and to the left of the preview window. These are the features currently for Presets:

- Allow the author to shape the look of the data.
- This feature make the author easy to look what kind of data is present and make necessary decision based on it.
- Presets value is synced to code editor so author can have multiple ways to shape data.

UI Presets will go in the area at the top left, labeled "buttons here" as shown below.

**Piling.js Authoring**

App Data Renderers Aggregators Styles Group Arrange Sidebar +

Options

columns: 3    cellpadding: undefined    pileCellAlignment:  left  center  right    pileBorderSize: 1    pileVisibilityItems:  previewAlignment:  left  center  right

Import Data

Result Intermediate View

Custom Code

```

1  const style = {
2    columns: 3,
3    cellPadding: 16,
4    itemSizeRange: [1, 1],
5    pileCellAlignment: 'center',
6    pileBorderSize: 1,
7    pileVisibilityItems: true,
8    previewAlignment: 'left',
9    previewScaleToCover: ['auto', true],
10   };
11
12  export default style;

```

Console

CLEAR

Figure 6a: UI Presets 1

**Piling.js Authoring**

App Data Renderers Aggregators Styles Group Arrange Sidebar +

Options

columns: 5    cellpadding: undefined    pileCellAlignment:  left  center  right    pileBorderSize: 1    pileVisibilityItems:  previewAlignment:  left  center  right

Import Data

Result Intermediate View

Custom Code

```

1  const style = {
2    columns: 3,
3    cellPadding: 16,
4    itemSizeRange: [1, 1],
5    pileCellAlignment: 'center',
6    pileBorderSize: 1,
7    pileVisibilityItems: true,
8    previewAlignment: 'left',
9    previewScaleToCover: ['auto', true],
10   };
11
12  export default style;

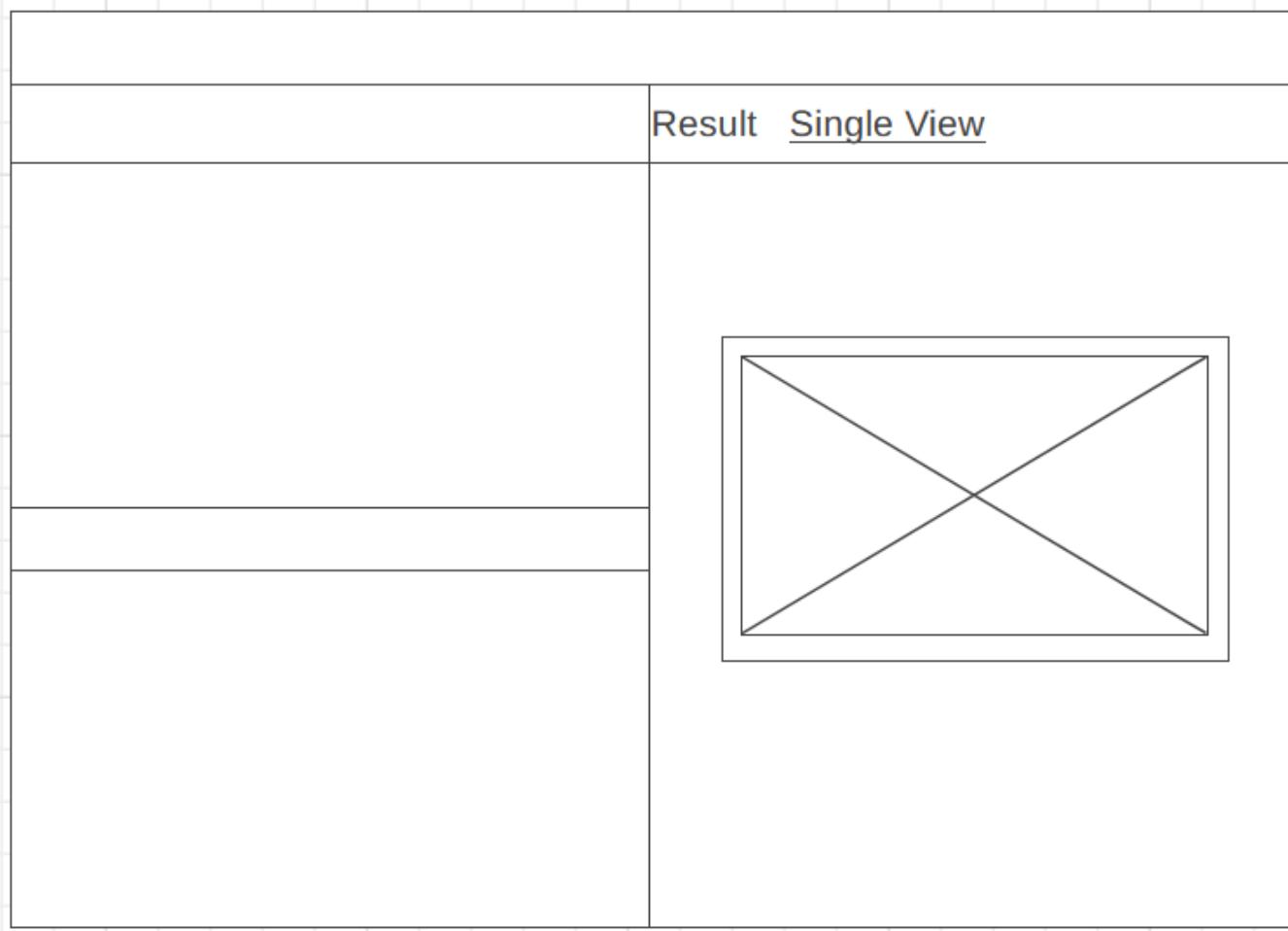
```

Console

CLEAR

Figure 6b: UI Presets 2

## Previews:



**Figure 7: UI Previews**

The goal: the Visual designer is able to see a preview of a group of the data. We will be implementing another view option that displays only a single data item, allowing the designer to focus on how a single item will be displayed. These are the features currently for Previews:

- Allows the author to select a specific data.
- This feature can help the author to focus on the data and work on it separately as needed.
- In addition, to the previous and next options for each data, it also allows the user to access a specific ID of the input data (the sequence number of a data in the raw data).

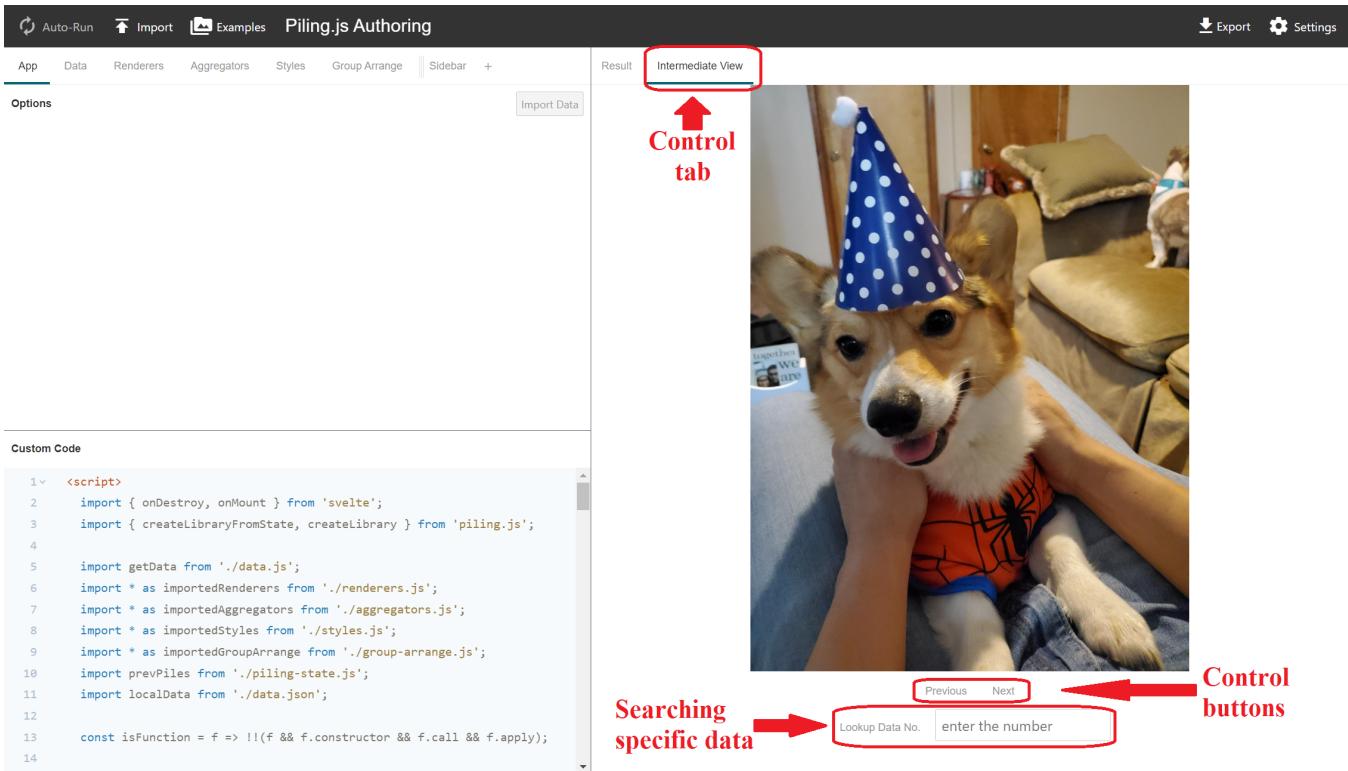


Figure 8a: UI Previews with control and searching feature

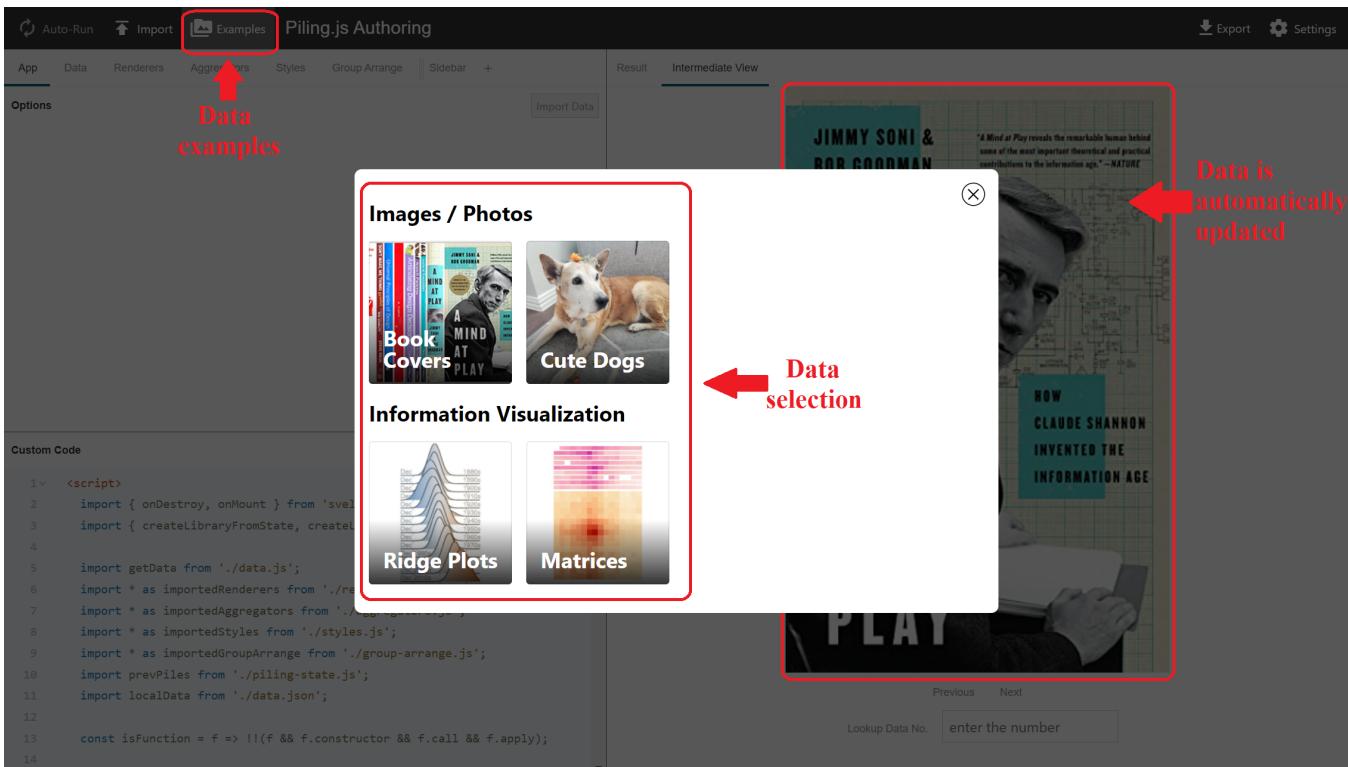
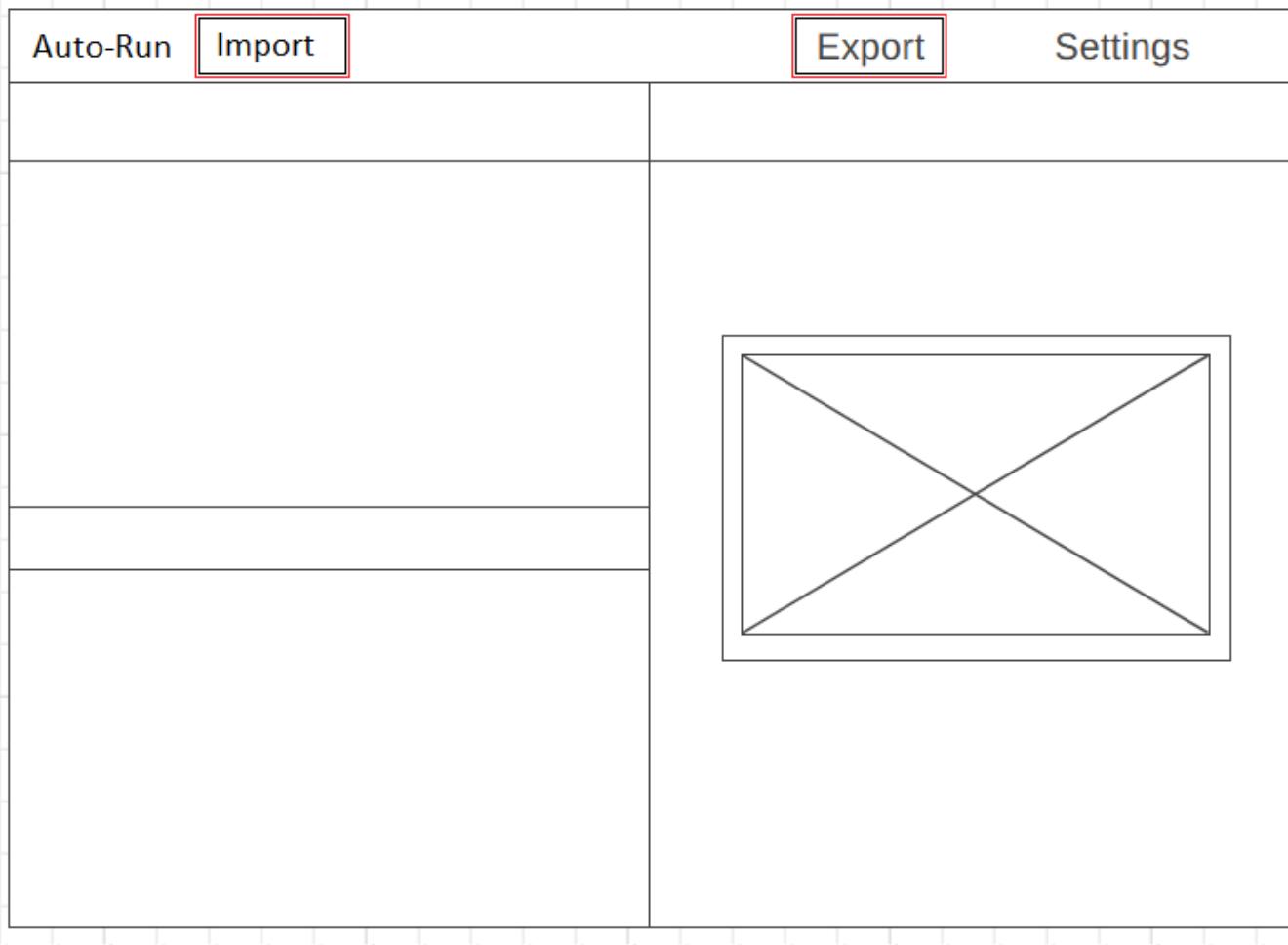


Figure 8b: UI Previews with data selection

## Templating:



**Figure 9: UI Templating**

The visual designer will be able to export a finished template using the "Export" button on the right of the main UI bar. This export can then be imported into an Analyst Tool to allow the Analyst to use the chosen settings in a clean environment.

Should the Analyst decide to change the exported settings, the downloaded file can be imported utilizing the "Import" button on the left of the main UI bar. This Import button will automatically distinguish between files that contain sample data and files that contain settings and handle them appropriately.

Piling.js Authoring

Auto-Run Import Examples Export Settings

App Data Renderers Aggregators Styles Result Intermediate View

Options Import Data

Custom Code

```
1 <script>
2   import { onDestroy, onMount }
3     from 'svelte';
4   import {
5     createLibraryFromState,
6     createLibrary } from 'piling.js';
7
8   import getData from
9   https://umb-cs410-fritz.github.io/ec446c96-d0eb-4e03-8321-420c0575ffe6
```

blob:<https://umb-cs410-fritz.github.io/ec446c96-d0eb-4e03-8321-420c0575ffe6>

Console CLEAR

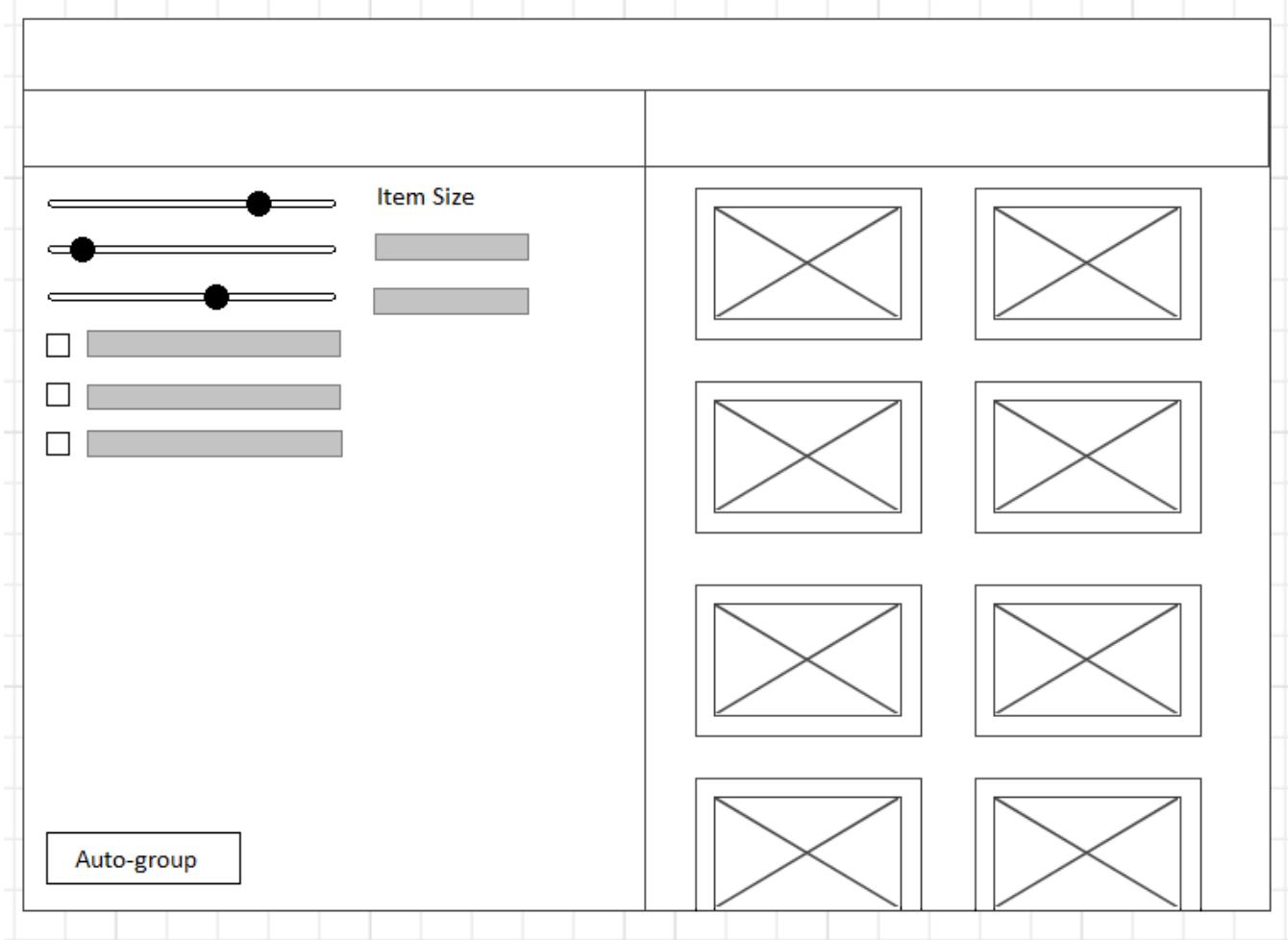
The screenshot shows the Piling.js Authoring interface. At the top, there are navigation buttons for Auto-Run, Import, Examples, Export (with a hand cursor icon), and Settings. Below the navigation is a horizontal menu bar with tabs: App, Data, Renderers, Aggregators, Styles, Result, and Intermediate View. The Result tab is selected. On the left, there's a sidebar with 'Options' and 'Import Data' buttons, and 'Custom Code' section containing Svelte code. To the right of the code is a grid of six dog photos. At the bottom, there's a 'Console' and 'CLEAR' button.

Figure 10: Export Button allows the user to download a JSON file that contains the chosen settings.



Figure 11: Clicking the Export Button downloads a "piling-settings.json" file.

## Sidebar:



**Figure 12: UI Sidebar**

These are the features currently for Sidebar:

- Allows the author to provide the analyst with specified options within the analyst sidebar, allowing the analyst to make quick and easy edits to the data.
- Prevents the analyst from making any edits that they shouldn't.
- Makes it easier for the analyst, saving room on the sidebar and making it clear which options will be useful.

Auto-Run Import Examples Piling.js Authoring Export Settings

App Data Renderers Aggregators Styles Group Arrange Sidebar + Import Data

**Options**

**Pile/Item**

Item Size

**Layout**

Cell Size  
 Cell Padding  
 Number of Columns  
 Row Height  
 Cell Alignment

**Arrangement**

**Group By**

Row  
 Column  
 Grid  
 Overlap  
 Distance

Result Intermediate View

Custom Code Console CLEAR

Figure 13a: UI Sidebar 1

Auto-Run Import Examples Piling.js Authoring Export Settings

App Data Renderers Aggregators Styles Group Arrange Sidebar + Import Data

**Options**

**Pile/Item**

Item Size

**Layout**

Cell Size  
 Cell Padding  
 Number of Columns  
 Row Height  
 Cell Alignment

**Arrangement**

**Group By**

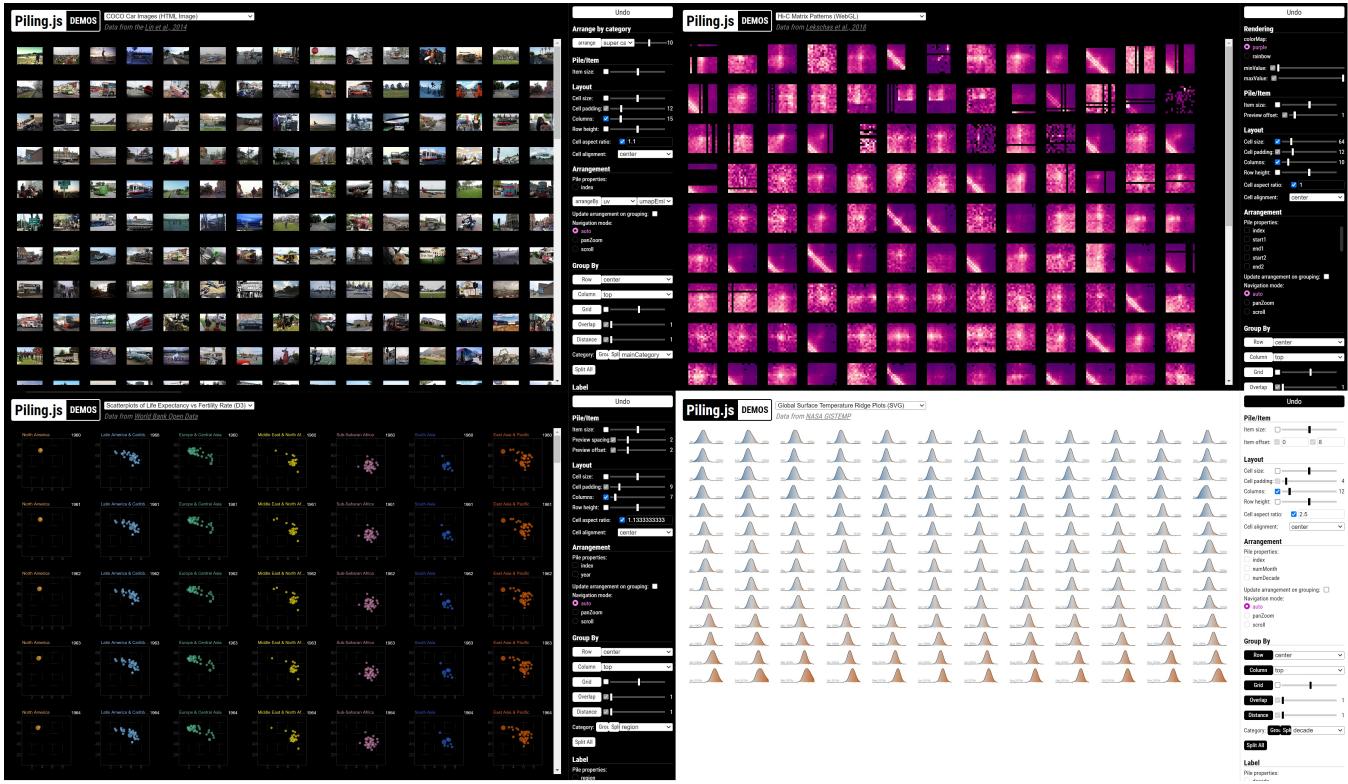
Row  
 Column  
 Grid  
 Overlap  
 Distance

Result Intermediate View

Custom Code Console CLEAR

Figure 13b: UI Sidebar 2

## Samples:



**Figure 14: Samples of what an analyst view may look like.**

The Analyst will have a Sidebar visible while analyzing the data. This will allow the Analyst to edit the current display through options chosen by the Visualization Designer, such as "item size", and apply any specified custom filters or grouping.

## 5 Timeframe and Milestones

- Client present: **2/7, done!**
- Team Selection: **2/18, done!**
- Project Proposal (Rough Draft): **2/28, done!**
- Projects
- Purpose
- Enhance the Features
- Revised Project Proposal: **3/25, done!**
- Code: Preview
- Code: Presets
- Code: Templating
- Code: Sidebar

- Testing Software
- Fix Bugs
- **Project Presentations: 5/06, done!**
- Finalize Documentation
- **Final Project Documentation: 5/20**

## 6 Supplemental Documents and Notes

- Project Proposal
  - [Proposal from Client](#)
  - [Fritz's Original Repository](#)
- Completed Project
  - [Authoring Tool Demo](#)
  - [Authoring Tool Repository](#)
- Supplemental Project
  - [Settings Import Demo](#)
  - [Settings Import Repository](#)