**API Integration and Technology Stack Utilization for Interactivity**

While developing BriSentinel, our team efficiently utilized various technologies to ensure a compelling and interactive user experience. The fundamental layer of our project was laid down using vanilla HTML, CSS, and JavaScript. By leveraging these technologies, we integrated different functionalities as follows:

**Model-View-Controller (MVC) Architecture**

The MVC architecture was primarily used in our 'Rank Page', where we maintained a list of suburbs and their corresponding descriptions. On this page, when a suburb from the list is clicked by our user, the 'addEventListener' function was used to detect this action. The suburb name, which served as our primary key, was used to modify the browser's href.

Upon this change in href, showcasing the MVC architecture, the Controller was triggered first to monitor the href change. Subsequently, the Model fetched the corresponding suburb data, and the View rendered the specific suburb's information.

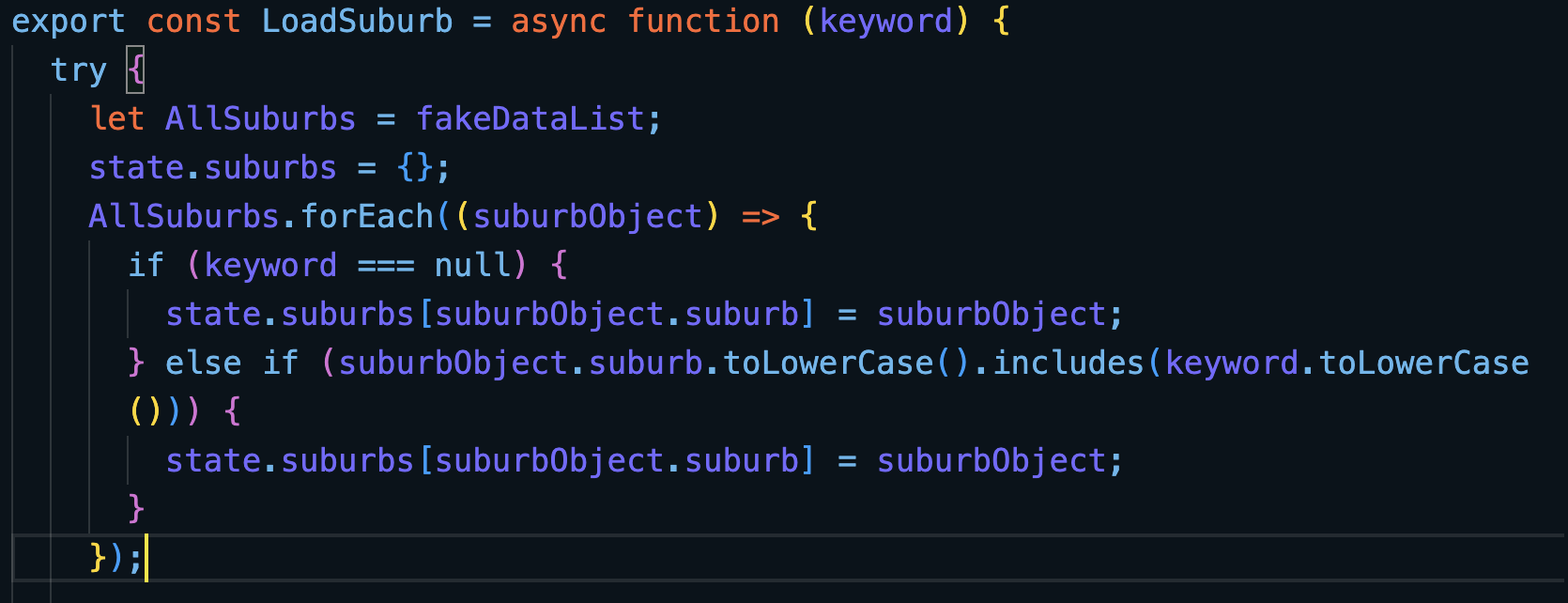


Figure 1 Part of Code Screenshot in Model Part to Grab Matching Data

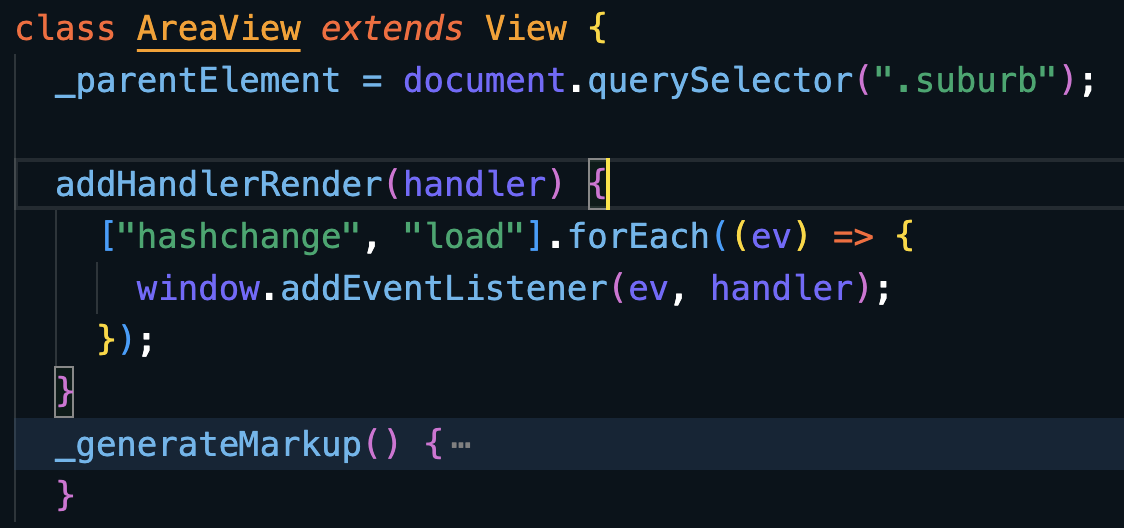


Figure 2 Part of Code Screenshot in View Part to Render Suburb Information

A screenshot of a computer

Description automatically generated

Figure 3 Part of Code Screenshot in Controller Part to Handle the Sort Status

This workflow ensured that our users could receive real-time feedback when interacting with various areas of our web page. It allowed us to manage multiple functionality states with 'addEventListener', such as sorting and searching, enhancing the maintainability and scalability of our site.

**Client-side Data Storage**

During interactions on our *Properties* and *Profile* pages, users can save specific properties for easy retrieval in their profiles. This feature was implemented using HTML's Local Storage functionality. We stored the selected properties' IDs on the client side, using them as primary keys.



Figure 4 Part of Code Screenshot in Profile Page to Get Data from Local Storage

When users navigate to their profiles, we store the property IDs from Local Storage and then use these IDs to fetch the corresponding properties, which we display using a carousel-style layout as the following picture

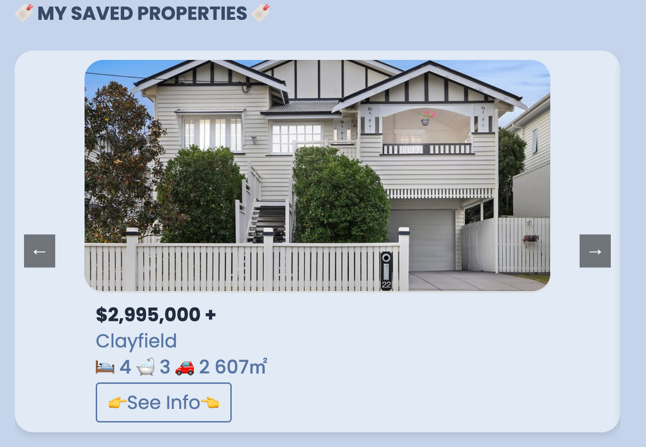


Figure 5 Carousel-style of Saved-Property Function

This functionality ensures even in the absence of a login/logout page, this approach allows users to save information of interest. This not only elevates the interactivity of our web page but also satisfies users' need to save property data, thus enhancing the overall user experience.

**Object-oriented Programming**

In our map page, despite processing backend data, the Leaflet library could not efficiently handle three different data types and render them separately on the map. To overcome this, we leveraged object-oriented programming (OOP) principles.

We transformed different types of disaster data into specific objects and passed these into the Leaflet rendering methods. This approach not only assures the maintainability of our website but also supports the integration of more diverse disaster data in the future. This capability satisfies our target users' specific requirements of filtering different types of disasters, thus enhancing the adaptability and usefulness of our site.



Figure 6 Object-oriented Programming Style for Handling Different Types of Data

1. **API Integration**

Our website mainly leverages the Leaflet and Google Charts APIs to boost interactivity.

* **Leaflet Library**

Initially, we utilized the Leaflet library as it can allow our users to better specify different suburbs' locations in Brisbane. However, as we progressed with user testing after prototype development, we discovered that the suburb markers provided by Leaflet were not distinctive enough for users to specify. As a solution, we retrieved geographical boundary data (longitude and latitude) for each suburb from the crime cases dataset. These data points were used to draw polygon boundaries as a separate layer on the Leaflet map, thus enhancing the distinctness of different suburbs. This use of the Leaflet library helped users interact more accurately with specific areas and familiarize themselves with Brisbane's regions.

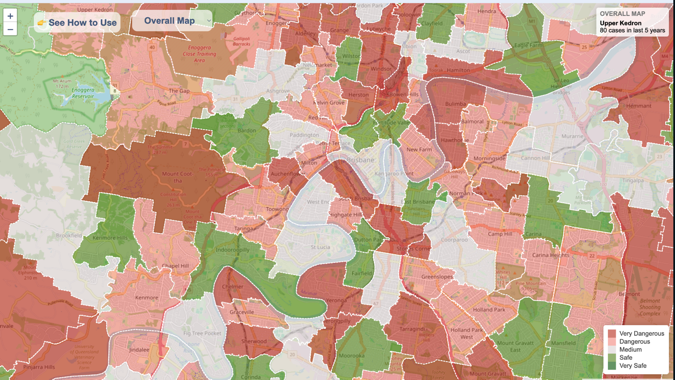
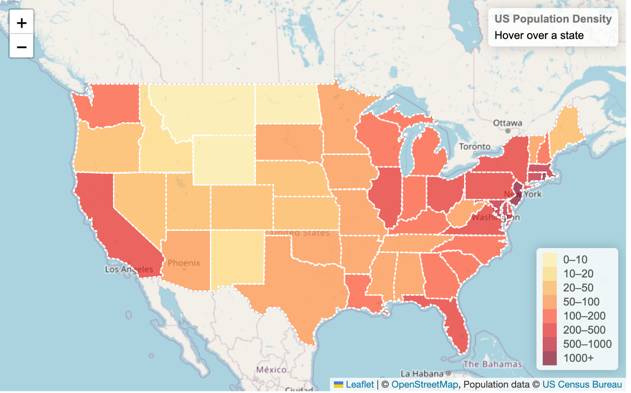


Figure 7 Map Sample of Leaflet Website (Left) & Our Use of Leaflet Library (Right)

* **Google Charts**

We integrated the Google Charts API to present bushfire, flood, and crime case data interactively across different areas. Google Charts not only ensures users can click on each icon for data interaction but also offers a variety of presentation styles like donut charts, line charts, and bar charts. These visualizations help users compare different areas and the average values for all areas in Brisbane from multiple perspectives. Consequently, users gain more comprehensive insights, enabling them to make more informed decisions when deciding suburbs they wish to move to.

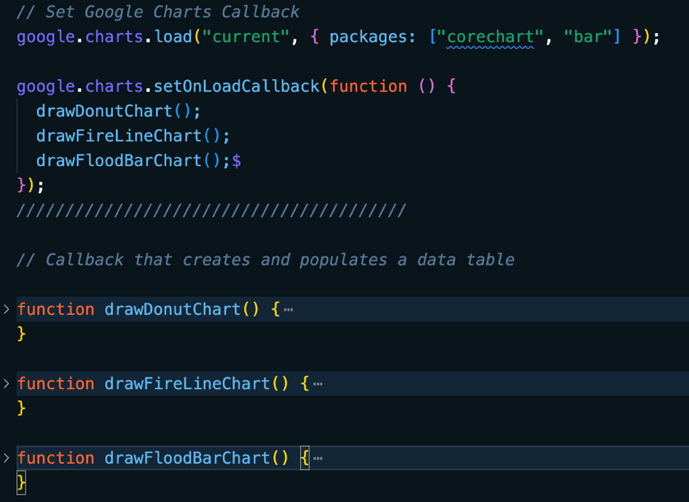


Figure 8 Part of Code Screenshot of Using Google Chart API

A screenshot of a data analysis

Description automatically generated

Figure 8 Sample of Google Chart API (Top-left) & Our Use of Different Chart (the Rest)

In conclusion, our choice to integrate those technologies into BriSentinel have proven to be crucial to allow us to create a more interactive, and robust website. The emphasis we placed on above technologies demonstrates our group's approach to adjust and enhance our technological utilization based on the evolving needs of our users. This shows our belief in user-focused programming, where the user experience and needs are the driving force behind our development decisions.

## Utilized Datasets and Its Interactivity

### SLQ Datasets

We have chosen three datasets from the SLQ database. They are **Crime Map**, **Flood Awareness Creek** and **Wild Fire History** respectively. To better utilize these data and correctly display information from these data, we have used *Postman* as our dev tool to figure out the the request and response format while calling the provided APIs.

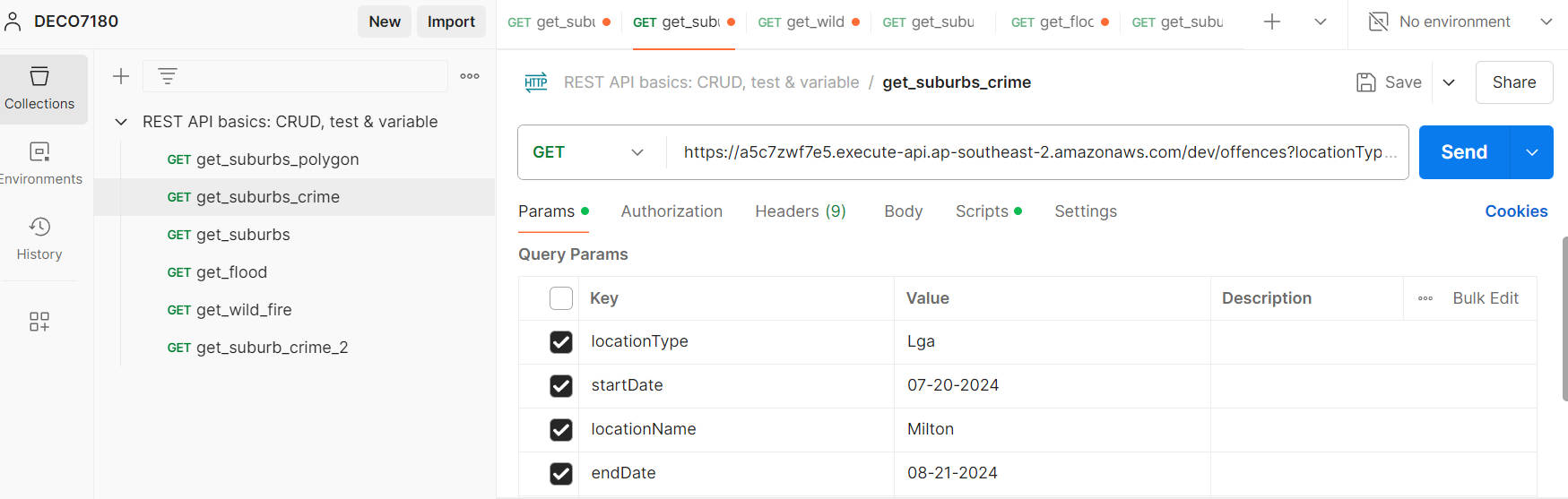


Figure 1Postman Interface Including all APIs used

#### Crime Map

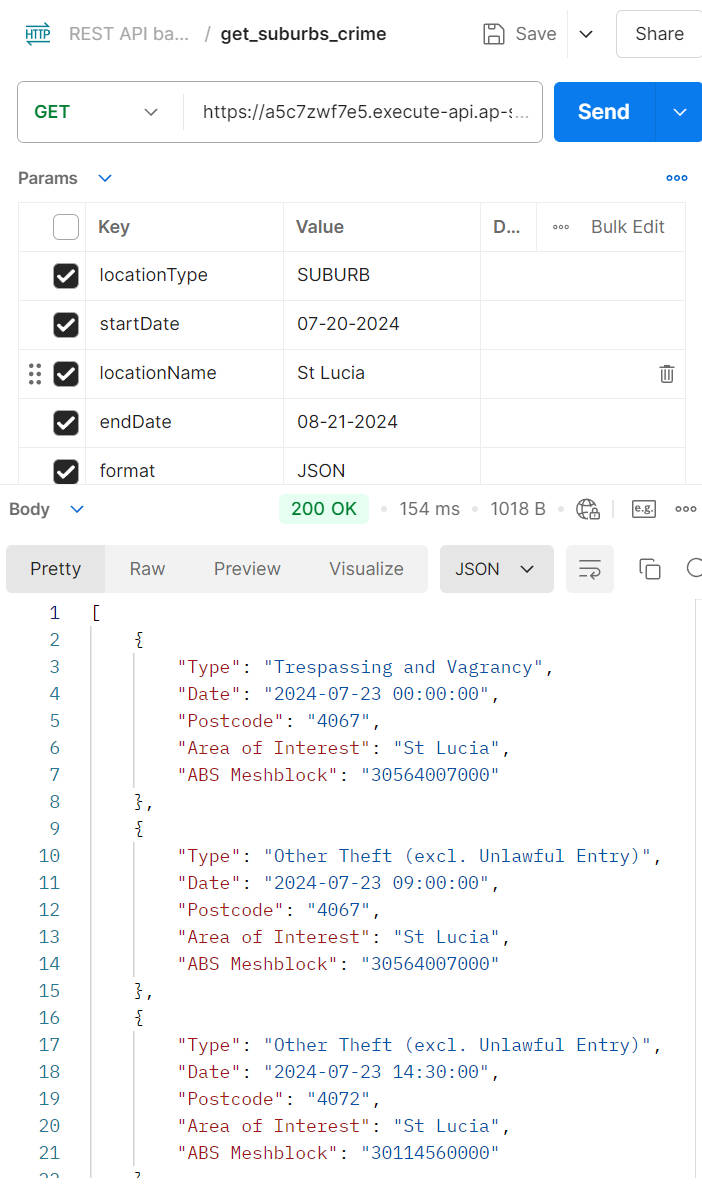
Crime Map is the most complicated dataset among the three.

##### Purpose

We have used it for two main purposes.

1. Crime case retrieval of each suburb as its name suggests.
2. Geographical information retrieval on each suburb mainly for polygon boundaries data, which is essential for the drawing of Leaflet on our *Risk Map* page later.

Below are the two examples on how we have used Crime Map. The first one is to get crime cases of St Lucia and the second one is to get the geodata of St Lucia.

A screenshot of a computer

Description automatically generated

Figure 2&3 Crime Map to Get Crime Cases & Geographical Data of St Lucia

##### Data Process

We have to further process data from Crime Map.

1. **Geographical Data**: It has two features.
   1. It is static data and is unlikely to change in the foreseeable future.
   2. It is heavily loaded if all suburb information is integrated together due to the complicated boundary data.

* Considering these features, we have saved all geographical data into a JSON file, which we will not modify again once we finish the process. By doing so, we can guarantee the *Risk Map* can get the needed data quickly.

1. **Crime Data**: It works for one purpose.

* We will first call API to fetch all crime data in the last five years of each suburb. Next, we will group the retrieved data by the crime type and get the total count of each crime. Then we will store it into a JSON file so that later *Chart* page will utilize it to display detailed data on each crime of one suburb in a doughnut chart.

#### Flood Awareness Creek

Flood Awareness Creek offers data on the flood information of Brisbane.

##### Purpose

We have used it for getting information on how a suburb has been affected by flood in the last five years. Such data will be displayed on both *Risk Map* and *Chart* pages. Below is how the data looks for one call.

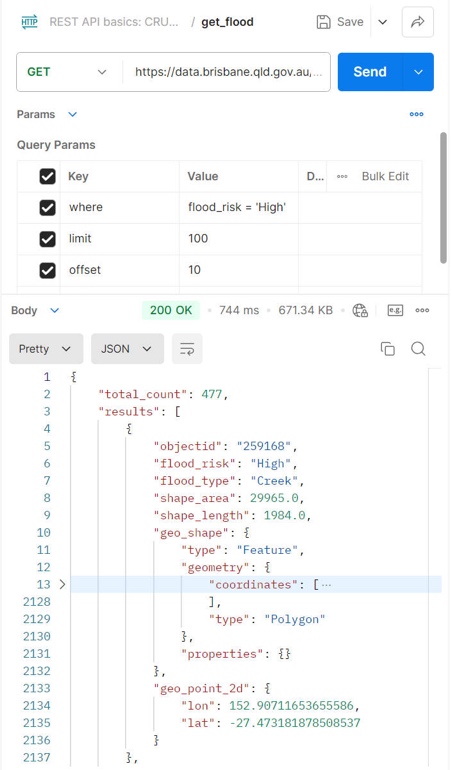


Figure 4 Flood Awareness Creek to Get Data of High Flood Risk

The data has categorized flood threat level into four levels, which are 'High', 'Medium', 'Low' and 'Very Low'. The order of magnitude for 'Low' data amount is much higher than the one for 'High' level, which imposed some challenges for data processing to be explained in the next part.

##### Data Process

We have faced two big challenges while processing the data returned.

1. **Limit of 100**: For each call only a maximum of 100 rows can be fetched. So to get all the needed data, we have to do repetitive calling around 1000 times. This would be a big performance issue if data were retrieved on the fly, hence we decided to store data into a JSON file as a cache for more quicker response.
2. **Determine Flood Threat on Each Suburb**: The data retrieved does not provide direct information related to each suburb. So we need to implement a function to determine whether a suburb was affected by the flood. Fortunately, *Leaflet* provided an API to calculate the distance and we are able to compare the distance between flood centre and the suburb centre with the flood range to check if the suburb is affected.



Figure 5 The Example API from Leaflet to Calculate Distance

#### Wild Fire History

Wild Fire History offers data on the wild fire history in Brisbane.

##### Purpose

We have used it to retrieve information on how a suburb has been affected by wild fire. It is quite similar to Flood Awareness Creek on the format of data. But it has a much smaller amount of data and is much easier to process with the experience of Flood Awareness Creek. Below is how the data looks in one call.

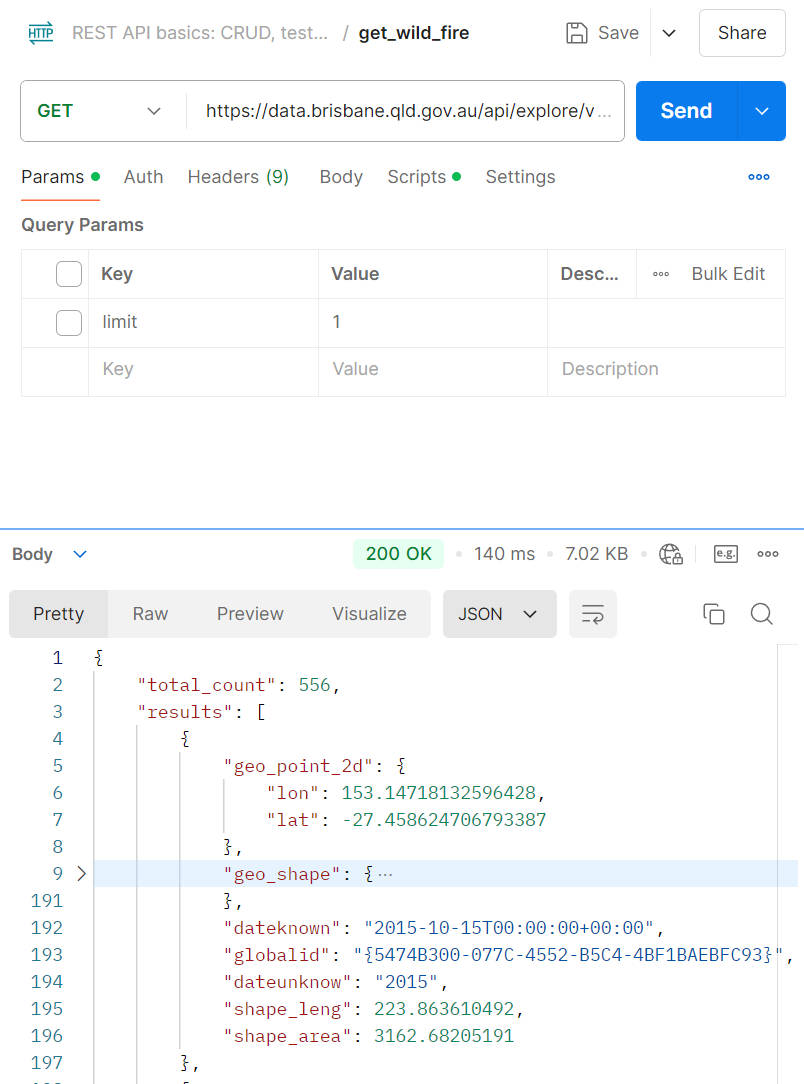


Figure 6 Wild Fire History to Get All Data in Brisbane

##### Data Process

Like Flood Awareness Creek, we have to use *Leaflet* API to check if a suburb was affected by wildfire. It had the same process as explained in the previous part.

### Other Datasets

Except for the required datasets from SLQ, we have also used other datasets for simulation purpose.

#### [RealEstate](https://www.realestate.com.au/)

RealEstate is the website we have referred to for two purposes.

1. Retrieve data for average housing and rental price of each suburb for the *Rank* page.
2. List simulated properties for the *properties* page.

#### [Wikipedia](https://www.wikipedia.org/)

Wikipedia is the website we have used to get necessary information like suburb images and description for the *Rank* page.

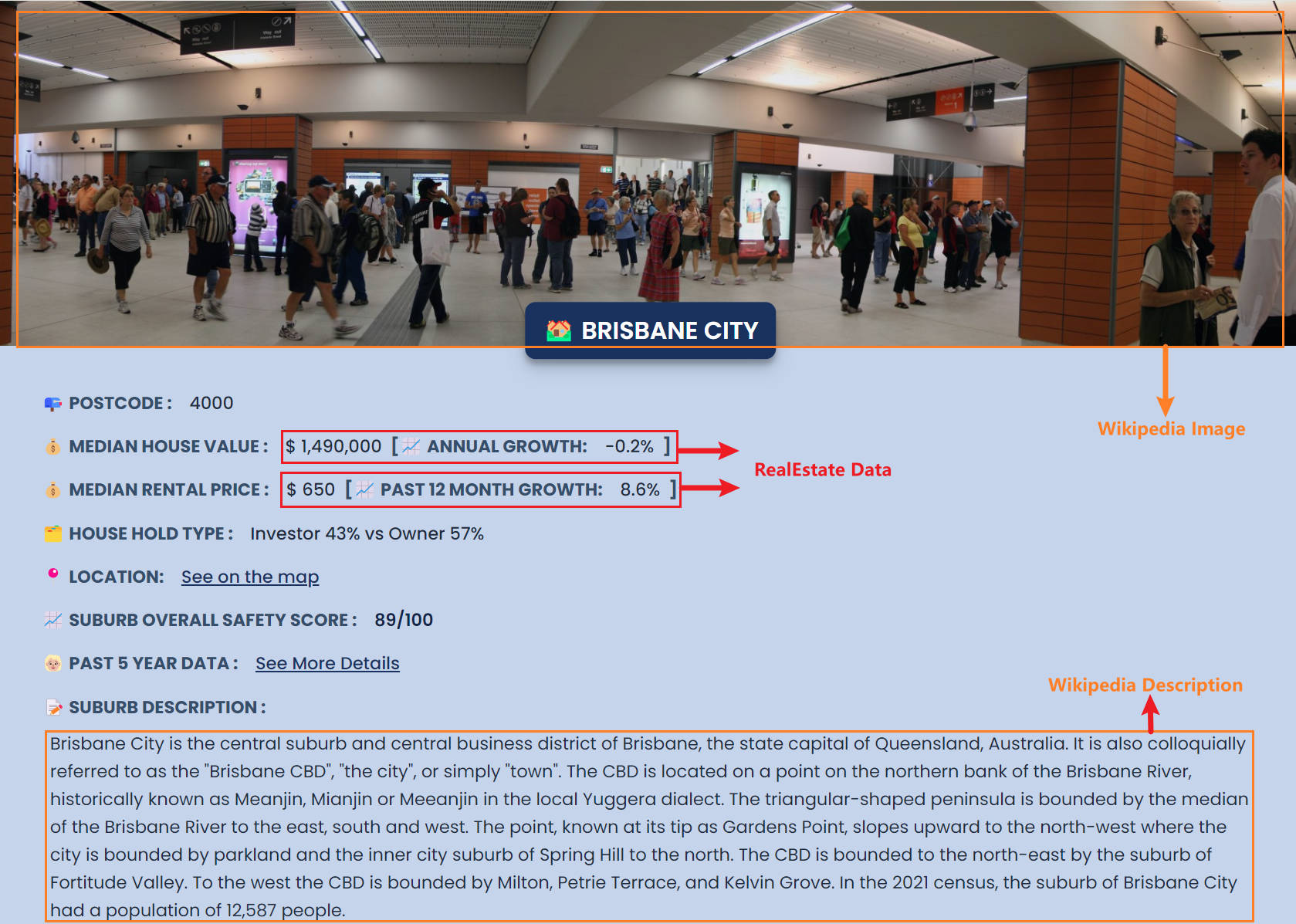


Figure 7 Usage Example of Other Datasets on Brisbane Court