Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2024

Team Name : Covid Catchers
Product Name : FindMyCovid

Dev Naganoolil
Muhammad Zaharudin
Muhammad Zuhaimi
Aryan Mainkar

Revision History

Revision	Date	Author(s)	Description
0.1	3.26.2024	DN	document creation
1.0	3.26.2024	DN, MZ, MZ, AM	complete draft

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

The global impact of the COVID-19 pandemic has underscored the critical need for real-time, accurate, and accessible data to understand its spread and impact. Our project aims to address this need by developing a comprehensive website that provides real-time mapping of COVID-19 cases worldwide. This platform will serve as a valuable resource for individuals, healthcare professionals, policymakers, and researchers seeking up-to-date information on the pandemic's status across geographical regions.

The core concept of our project revolves around the integration of two vital components: real-time COVID-19 data retrieval and visualization through an interactive map interface. Leveraging APIs for both COVID-19 data and mapping services, our platform will dynamically gather the latest statistics and display them visually on an intuitive map interface. Users will have the ability to explore COVID-19 case trends, distribution, and related data with ease, fostering a deeper understanding of the pandemic's impact on a global and local scale.

The scope of our project encompasses the development of a user-friendly website accessible across various devices. This platform will utilize APIs to fetch real-time COVID-19 data, ensuring accuracy and timeliness in information presentation. The map interface will provide interactive features, allowing users to zoom in/out, filter data by region, view detailed statistics, and track historical trends. The primary focus will be on delivering a seamless user experience, providing informative visualizations, and ensuring the reliability of the displayed information.

2 SYSTEM OVERVIEW

The following diagram outlines the logical subsystems embedded within the layered architecture of the system. Each layer is composed of distinct subsystems that collectively facilitate the flow and processing of data essential for the representation of COVID-19 information on the mapping interface. The interactions/interfaces between these subsystems delineate the flow of data, ensuring the accurate visualization and display of COVID-19-related data across geographical regions.

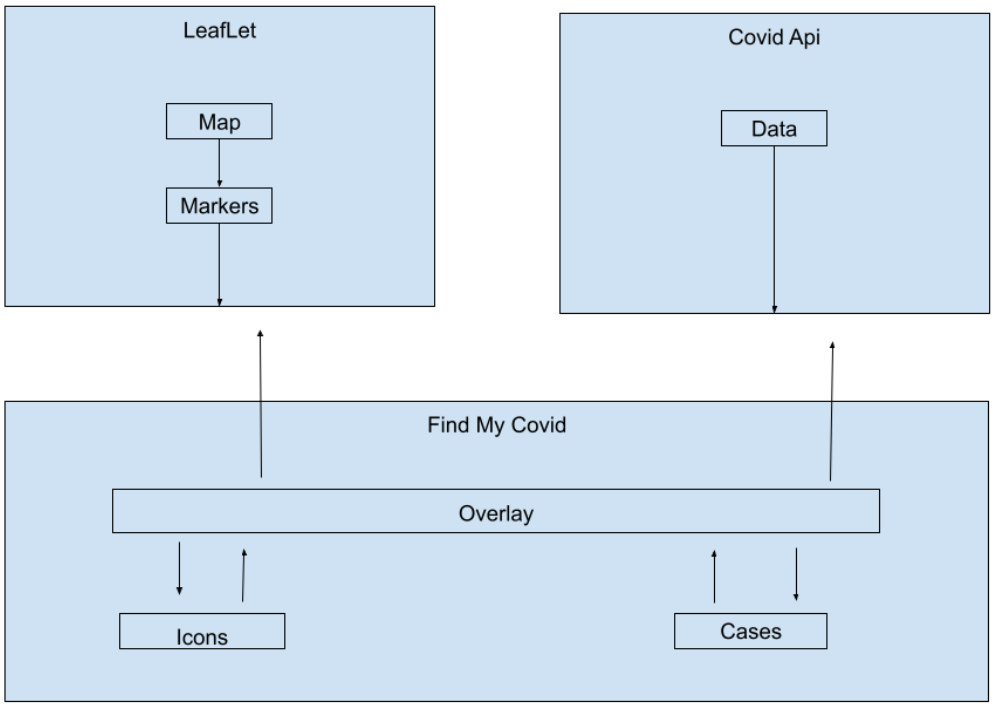

Figure 1: System Architecture

3 X Layer Subsystems

3.1 Layer Hardware

The Leaflet API primarily operates within web browsers and does not require any specific hardware components at the layer level.

3.2 Layer Operating System

The Leaflet API is platform-independent and can run on any operating system that supports modern web browsers, such as Windows, macOS, Linux, iOS, and Android.

3.3 Layer Software Dependencies

The Leaflet API relies on the following software dependencies:

- Web browser: Leaflet is a JavaScript library designed to work within web browsers. Therefore, any dependencies required by the specific web browser being used (such as Chrome, Firefox, Safari, etc.) are indirectly required by the Leaflet API.

3.4 Map Subsystem

The Map Subsystem is responsible for managing the display of COVID-19 data on a map of the United States using Leaflet.js library. It interacts with the Data Markers Subsystem to display relevant data.

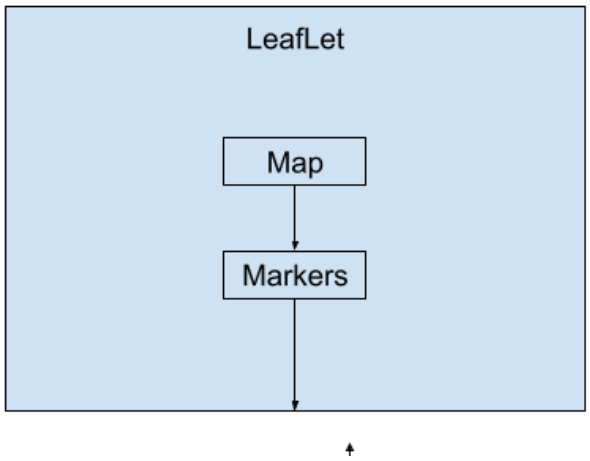

Figure 2: Leaflet API Subsystem Overview

3.4.1 Subsystem Hardware

No specific hardware components are required for the Map Subsystem beyond the devices that support web browsers.

3.4.2 Subsystem Operating System

No specific operating system is required for the Map Subsystem beyond those supported by the web browser.

3.4.3 Subsystem Software Dependencies

- Leaflet.js library: Provides functionalities for map rendering, interaction, and customization within the web browser environment.

3.4.4 Subsystem Programming Languages

- JavaScript: Used for implementing the Map Subsystem's functionalities.

3.4.5 Subsystem Data Structures

The Map Subsystem utilizes various data structures internally to represent geographical features, layers, markers, and other map elements. These data structures are managed by the Leaflet.js library and optimized for efficient rendering and interaction.

3.4.6 Subsystem Data Processing

The Map Subsystem employs algorithms for rendering vector-based maps, handling user interactions such as panning and zooming, and integrating external COVID-19 data for display on the map. Common algorithms include tile-based rendering and geospatial calculations for efficient map visualization.

3.5 Data Markers Subsystem

The Data Markers Subsystem generates markers on the map based on provided COVID-19 data, allowing users to filter markers and view relevant information.

3.5.1 Subsystem Hardware

No specific hardware components are required for the Data Markers Subsystem beyond the devices that support web browsers.

3.5.2 Subsystem Operating System

No specific operating system is required for the Data Markers Subsystem beyond those supported by the web browser.

3.5.3 Subsystem Software Dependencies

- Leaflet.js library: Used for integrating data markers onto the map and handling interactions with them.

3.5.4 Subsystem Programming Languages

- JavaScript: Used for implementing the Data Markers Subsystem's functionalities.

3.5.5 Subsystem Data Structures

The Data Markers Subsystem utilizes data structures to represent COVID-19 data points on the map, allowing for efficient storage and retrieval of information associated with each marker.

3.5.6 Subsystem Data Processing

The Data Markers Subsystem processes COVID-19 data to generate markers with relevant information, implements filtering functionalities based on user-defined criteria, and ensures accuracy in the representation of data points on the map.

4 Y Layer Subsystems

4.1 Layer Hardware

No specific hardware components are required for the COVID API Layer beyond the devices that support web browsers and internet connectivity.

4.2 Layer Operating System

The COVID API Layer is platform-independent and can run on any operating system that supports modern web browsers, such as Windows, macOS, Linux, iOS, and Android.

4.3 Layer Software Dependencies

The COVID API relies on the following software dependencies:

- Web browser: COVID-19 Tracking from RapidAPI is an API designed to work within web browsers. Therefore, any dependencies required by the specific web browser being used (such as Chrome, Firefox, Safari, etc.) are indirectly required by the API.

4.4 COVID Data Subsystem

The 'Data' subsystem within this layer functions as the core unit, facilitating the retrieval and processing of crucial COVID-19 statistics necessary for the system's mapping interface. Its pivotal role lies in ensuring the availability of accurate and up-to-date information, essential for informing users about the global pandemic landscape.

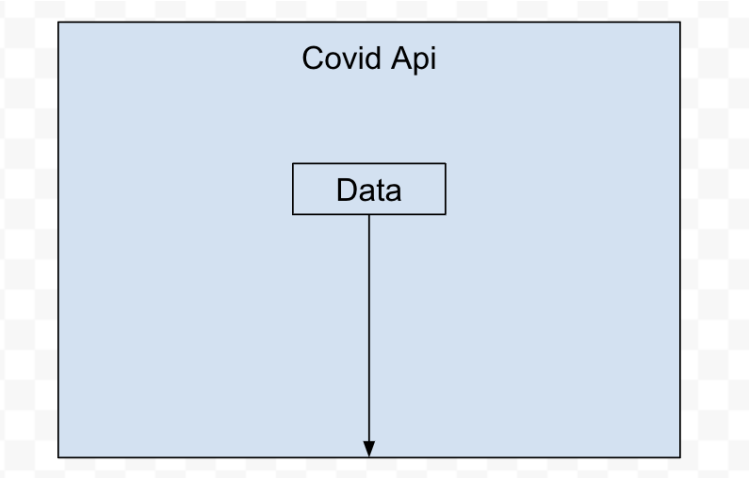

Figure 3: COVID data subsystem in the COVID API Layer

4.4.1 Subsystem Hardware

No specific hardware components are required for the COVID Data Subsystem beyond the devices that support web browsing and internet connectivity.

4.4.2 Subsystem Operating System

The COVID Data Subsystem can operate on any operating system compatible with web browsers and internet connectivity, including Windows, macOS, Linux, iOS, and Android.

4.4.3 Subsystem Software Dependencies

- Web browser: Required for accessing and interacting with external COVID-19 data APIs.
- JavaScript libraries or frameworks for handling asynchronous HTTP requests and data processing, such as Axios or Fetch API.

4.4.4 Subsystem Programming Languages

The COVID Data Subsystem primarily utilizes JavaScript for implementing functionalities related to data retrieval, processing, and integration within the web browser environment.

4.4.5 Subsystem Data Structures

The COVID Data Subsystem may utilize various data structures, such as JSON objects, arrays, and JavaScript classes, to represent and manipulate COVID-19 data retrieved from external APIs. Data structures may vary based on the format and structure of the received data.

4.4.6 Subsystem Data Processing

The COVID Data Subsystem employs algorithms and processing strategies for:

- Data Retrieval: Implementing mechanisms for accessing designated external COVID-19 data APIs, handling authentication, and making HTTP requests to fetch real-time statistics.
- Data Formatting: Validating, cleansing, and standardizing incoming data to ensure consistency and compatibility with the mapping interface.
- Data Storage and Management: Maintaining a structured repository to store and manage retrieved COVID-19 data efficiently.
- Data Aggregation for Visualization: Aggregating and categorizing COVID-19 statistics based on geographical regions for seamless integration into the mapping interface. This may involve calculating averages, percentages, or other aggregate metrics to provide meaningful representations of the pandemic landscape.

5 Z Layer Subsystems

5.1 Layer Hardware

The Overlay layer does not require specific hardware components. Instead, the hardware support is provided by the devices accessing the system through web browsers, such as personal computers, laptops, tablets, and smartphones. These devices serve as the platform for executing the client-side components of the system.

5.2 Layer Operating System

The system's client-side components, such as the user interface and interactive features, are implemented using web technologies such as HTML, CSS, and JavaScript, which are supported by all major operating systems, including Windows, macOS, Linux, Android, and iOS. This ensures that users can access the system seamlessly regardless of the operating system running on their devices.

5.3 Layer Software Dependencies

The layer's software dependencies include web frameworks like React.js for user interface development, mapping libraries such as Leaflet.js for map rendering. AJAX libraries like Axios facilitate seamless data retrieval, while CSS frameworks like Bootstrap aid in styling the interface. Additionally, utility libraries like lo dash assist in handling common tasks, and testing frameworks ensure the system's reliability and quality through comprehensive testing.

5.4 Front-End Icons Subsystem

The Icons Overlay Subsystem is responsible for navigating through the website application, showcasing the COVID-19 data through graphs and diagrams, and enabling interaction between the end-user and the front-end UIs.

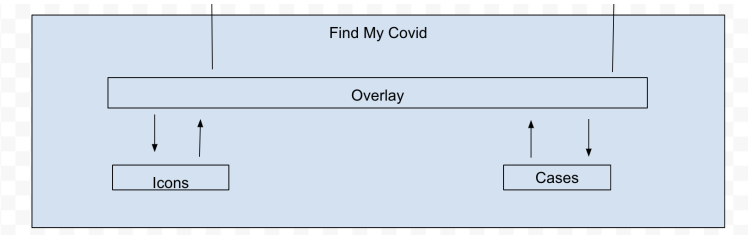

Figure 4: Front-End Overlay Subsystem Overview

5.4.1 Subsystem Hardware

As the Icons subsystem operates within a web browser environment, it does not directly involve any dedicated hardware components at the subsystem level. At the subsystem level, there are no specific hardware components involved, and the functionality is primarily dependent on the capabilities of the user's device and web browser.

5.4.2 Subsystem Operating System

The Icons Subsystem does not have specific operating system requirements at the subsystem level since it operates within a web browser environment. Users can access the system from various operating systems, including Windows, macOS, Linux, Android, and iOS, as long as they have a compatible web browser.

5.4.3 Subsystem Software Dependencies

The Icons Subsystem relies on several software dependencies to function effectively within the web browser environment. These dependencies include mapping libraries like Leaflet.js or Google Maps API for rendering interactive maps and data visualization libraries such as D3.js for presenting COVID-19 data. Additionally, CSS frameworks like Bootstrap and REACT may be used for styling the interface.

5.4.4 Subsystem Programming Languages

The Icons Subsystem is typically implemented using web development languages such as HTML, CSS, and JavaScript. HTML is used for structuring the map interface, CSS for styling the components, and JavaScript for adding interactivity and integrating data sources.

5.4.5 Subsystem Data Structures

The Icons Subsystem utilizes various graphs and diagrams to organize and display COVID-19 data effectively on the front-end interface. For example, we utilize the React JS framework for various UI components to visualize COVID-19 data on the web page.

5.4.6 Subsystem Data Processing

The Icons Subsystem does not require to process data, but to showcase that data on interactive graphs and diagrams. The data processing is the responsibility of the Cases subsystem.

5.5 COVID Cases Subsystem

The Cases Overlay subsystem is responsible of showcasing the data analysis of COVID-19 data.

5.5.1 Subsystem Hardware

As the Cases subsystem operates within a web browser environment, it does not directly involve any dedicated hardware components at the subsystem level. At the subsystem level, there are no specific hardware components involved, and the functionality is primarily dependent on the capabilities of the user's device and web browser.

5.5.2 Subsystem Operating System

Similar to the Icons subsystem, it does not have specific operating system requirements at the subsystem level since it operates within a web browser environment.

5.5.3 Subsystem Software Dependencies

The Cases subsystem depends on the COVID-19 Layer API for obtaining real-time COVID-19 data. It also depends on the Overlay System to be able to showcase the data.

5.5.4 Subsystem Programming Languages

Similar to the Icons subsystem, the Cases subsystem implemented using web development languages. Primarily, it utilizes the JavaScript language to showcase its data.

5.5.5 Subsystem Data Structures

The Cases Subsystem utilizes various data structures to organize and display COVID-19 data effectively on the front-end interface.

5.5.6 Subsystem Data Processing

The Cases Subsystem employs algorithms and processing strategies to handle the retrieval and visualization of COVID-19 data. While standard algorithms like data parsing and filtering may be used, specific processing strategies may be implemented to optimize performance and enhance user experience. For instance, asynchronous data fetching techniques are commonly employed to ensure seamless

updates and minimize latency when retrieving and displaying real-time COVID-19 statistics on the map. Additionally, algorithms for marker clustering and dynamic data rendering may be utilized to improve scalability and responsiveness, especially when dealing with large datasets.

6 Appendix A

REFERENCES