CSC431

Real Time Tracking and Analysis of COVID Hotspots

System Architecture Specification

Team #13

Demir Demirsoy Berk Mankaliye Arjun Misra

Version History

Version	Date	Author(s)	Change Comments
3.0	May 5th, 2021	Demir Demirsoy, Berk Mankaliye, Arjun Misra	Final Draft
2.0	April 13th, 2021	Demir Demirsoy, Berk Mankaliye, Arjun Misra	2nd Draft
1.0	March 30, 2021	Demir Demirsoy, Berk Mankaliye, Arjun Misra	1st Draft

Contents

Table of Figures

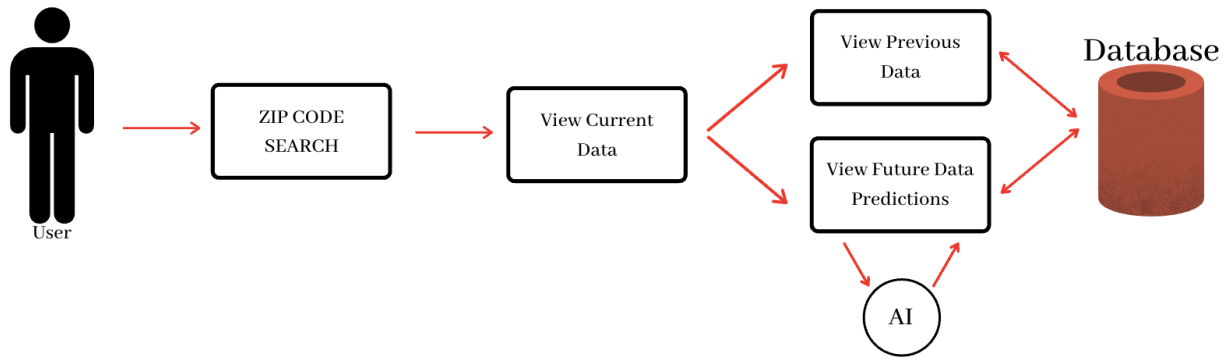
1 System Analysis

1.1 System Overview

Our system will revolve around two parts: the DataHandler and the CasePredictor. DataHandler is responsible for retrieving and sorting the requested COVID related information (past/present/future) from our storage which is interconnected to our external database. The CasePredictor is an artificial intelligence tool, which uses machine learning algorithms (such as the logistic-regression analysis and k-nearest neighbor mathematical models) to predict future COVID data - metrics such as cases, hospitalizations, deaths, and vaccinations - in a specific location. Our pipeline works as follows: User performs a ZIP code search. Next, the web application displays the map with concurrent COVID-19 information. From here, the user has the ability to request viewing of the past COVID data for this given location. Furthermore, the user can choose to view future case predictions generated by our CasePredictor algorithm.The three-tier architecture will be used for this project. The initial client-web browser tier allows the server to handle requests. The database-centric tier queries the database to obtain relevant COVID-19 data. The intermediary application logic/business layer serves to handle the displaying and managing the obtained data from the storage and/or handle the computations necessary to predict future data.

1.2 System Diagram

Fig.1 Search System Diagram



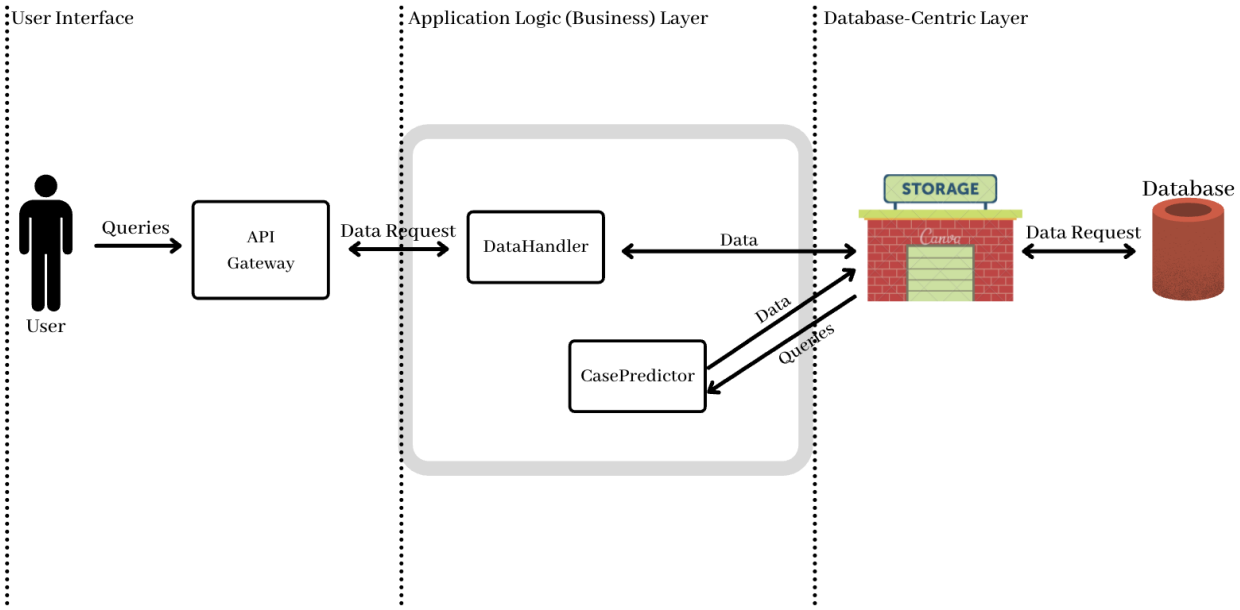
1.3 Actor Identification

There is only one human actor in the case of our webpage. As long as the user meets the requirements such as having cellular / wireless and having a suitable platform to view the data. Additionally, the system has two non-human actors; the API which serves to make data from databases provide vital up-to date COVID-19 information as well as the AI system (CasePredictor) that functions in the background to generate potential cases for the given zip code.

1.4 Design Rationale

1.4.1 Architectural Style

Fig.2 Architectural Style Diagram



Our architectural style exemplifies the 3-tier architecture model. As seen in Figure 2, the presentation layer primarily is essentially the User Interface, and it primarily consists of the API Gateway, which enables users to query for COVID data pertaining to a specific location and time frame. The API Gateway relays this query to the Application Logic Layer and, when the Application Logic Layer returns the requested data, the API Gateway displays it in a format specified by the user. As ease of access is important, the User Interface should be easy to use: simple, self-explanatory, and free of visual clutter. The second layer is the Application Logic (or Business) Layer, which resides on the server and mediates the interaction between the user and the database. It contains a Data Handler, which facilitates the categorization and processing of user requests and delivers these queries to the Database-Centric Layer before instructing the presentation layer to inform the user that the request is being processed. When the Database-Centric Layer returns the appropriate data, the Data Handler relays it back to the API Gateway in the User Interface Layer. The Application Logic Layer also contains a Case Predictor, which is an artificial intelligence tool utilizing machine learning algorithms such as k-Nearest Neighbor and Logistic Regression Analysis. Logistic Regression Analysis allows us to utilize current and past COVID data to predict future COVID data using the logistic regression

model, whereas k-Nearest Neighbor allows us to classify regions as high, medium, or low risk COVID areas, information key to the graphical displays which overlay maps. The Case Predictor exchanges data with the Data Handler when the user queries future COVID data. The Database-Centric layer is the third layer of the architectural plan. The Database-Centric Layer contains Storage, which is a temporary unit that functions like a cache in that it stores current data and data from the past 2 months. As these types of data are the types most often quested, they are readily accessible. However, if future predictions are necessary, the Storage queries the Case Predictor, which runs the necessary algorithms and returns the requested data to the Storage, which returns it to the Data Handler. In turn, the Data Handler returns it to the API Gateway, where the User Interface appropriately displays it.

1.4.2 Design Pattern(s)

Our architecture lends itself to the implementation of a number of design patterns. The decorator design pattern is used when considering the feature that allows users to view past COVID data or predicted future COVID data. This feature extends the functionality of the API call to view the current data, but allows users to also toggle between the current data and past data or predicted future data. Therefore, it extends the "View Data" API's functionality without modifying it. The observer design pattern is utilized in the data-centric layer to monitor for changes in COVID data or predictions. When changes occur, it calls database APIs and informs them of the changes.

1.4.3 Framework

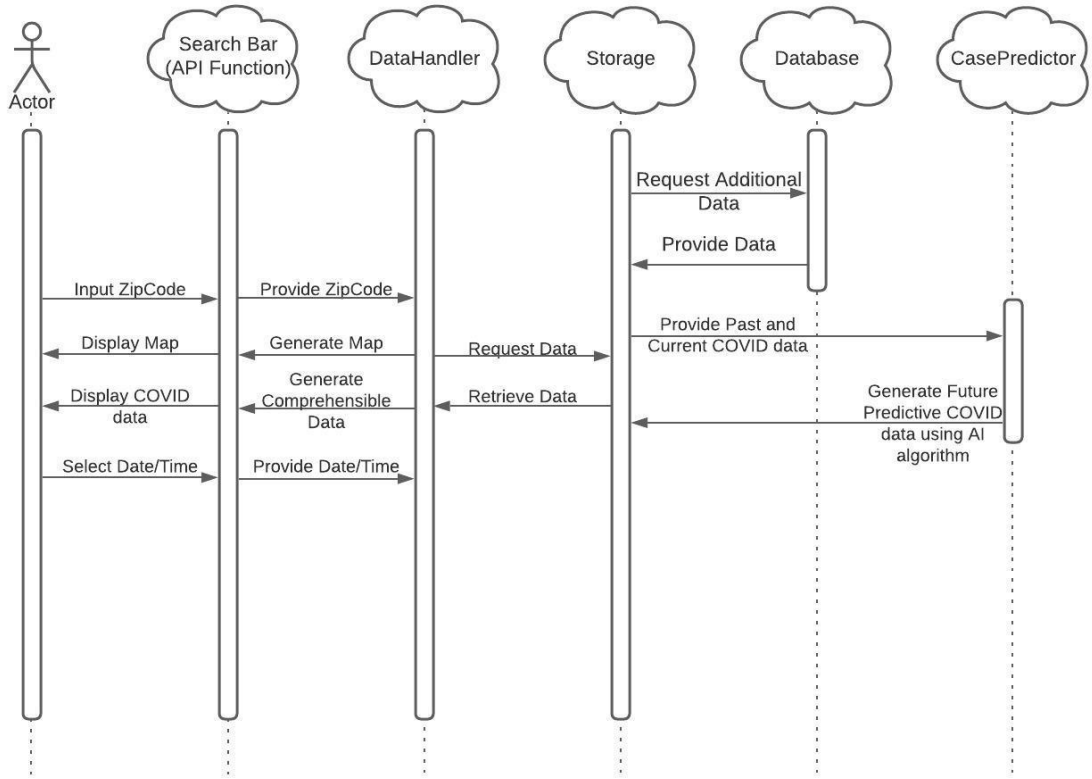
Our webpage will be relying on Vue.js JavaScript that will power our SQL server framework. SQL will facilitate our communication with databases that store and provide our COVID-19 data. Additionally, React.js will help provide the JavaScript library access to enhance user experience.

Links:

-**https://vuejs.org/**
-**https://www.mysql.com/**
-**https://reactjs.org**/

2 Functional Design

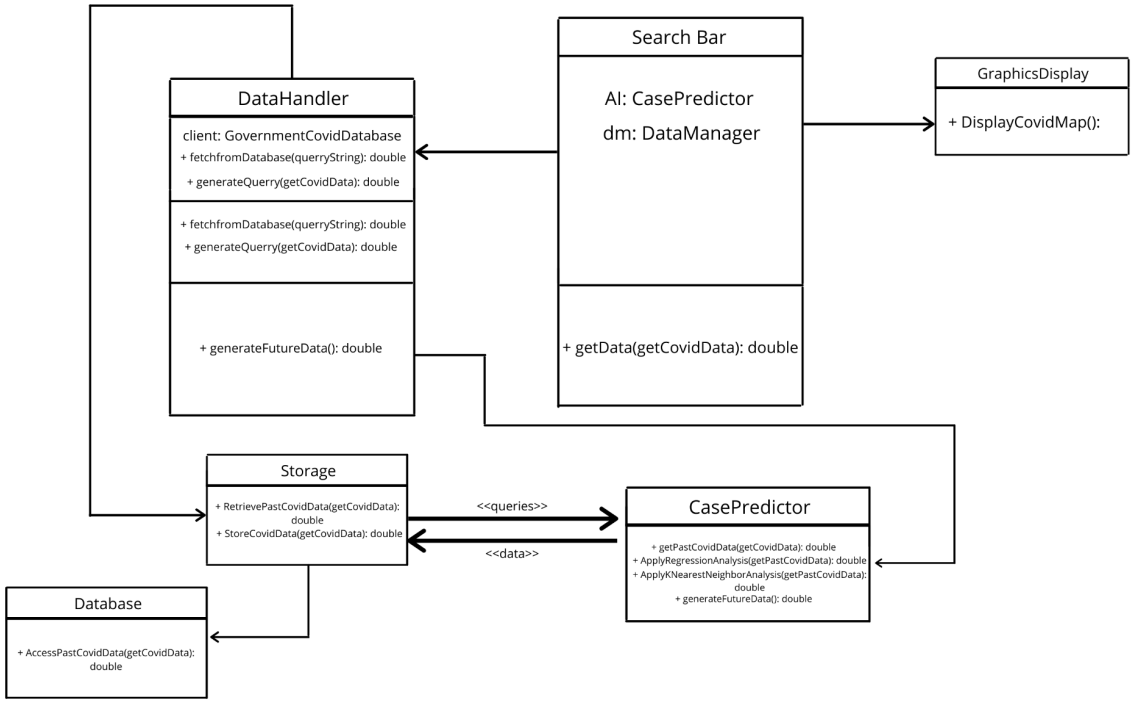
Fig.3 Functional Design Diagram



- The User first inputs their desired ZIP code, from which then the API takes over and communicates with the DataHandler.
- DataHandler communicates with storage and indirectly with the database to acquire the necessary data to provide the User with a map display and COVID-related data.
- If the User wishes to view current and past data, the functioning of the system remains the same. However, if future predictions are requested, then the CasePredictor requests current and past information from the storage and generates a computational analysis of potential cases in the requested area.
- Regardless of what the user desires, the system always returns a map and its associated COVID data to the user at the end of a request.

3 Structural Design

Fig 4. UML Structural Design Diagram



- The Search Bar component of the Class Diagram refers to the user interface where the user has access to make modifications within the system (a.k.a changing ZIP code, changing date/time of case viewings)
- DataHandler is split into three sections, where each corresponds to the present, past and future component of our web application.
- Regardless of which function of DataHandler is requested, the next step in the system is communication between itself and Storage which either stores the necessary data, or if not it requests it from the external database.
- CasePredictor requires that the storage provides both current and past data in order to accurately generate predictive future COVID-related data.