HAW HAMBURG

BACHELOR THESIS
Michelle Anastasya Valentina

Extension of a Data Hub for Epidemiological Data with a Data Coverage and Analysis Component

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Michelle Anastasya Valentina

Extension of a Data Hub for Epidemiological Data with a Data Coverage and Analysis Component

Bachelor thesis submitted for examination in Bachelor's degree
in the study course *Bachelor of Science Angewandte Informatik*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Thomas Clemen
Supervisor: Prof. Dr. Marina Tropmann-Frick

Submitted on: 7 June 2025

Michelle Anastasya Valentina

Title of Thesis

Extension of a Data Hub for Epidemiological Data with a Data Coverage and Analysis Component

Keywords

Data Hub, dashboard development, data visualization, dataset management, data analysis, data tracking

Abstract

This thesis presents the development of a dashboard for the Data Hub, a Geographic Information System (GIS) designed for epidemiological research. The dashboard aims to support researchers in managing large datasets, as well as in visualizing and analyzing data to uncover patterns, relationships, and trends. The development process employed Python, JavaScript, HTML, and CSS, along with various libraries and frameworks, to build an interactive platform for data exploration and management.

Contents

List of Figures

List of Tables

1 Introduction

Data analytics and exploration have become important components of modern research, making it possible to collect, process and visualize vast amount of data. One important type of data is geographic data, which refers to information linked to specific locations on Earth (ISO, 2015). Since nearly all human activities occur in specific locations on or near the Earth's surface, geographic data is essential for understanding, managing, and responding to these activities (Longley, 2008).

To effectively work with geographic data, researchers rely on Geographic Information Systems (GIS) - a computer systems designed to store, manage, analyze, and visualize this type of data (Chang, 2019).

GIS has been implemented across various sectors such as public services, urban planning, and health care. Over the years, numerous projects have aimed to develop and apply GIS technologies for different purposes, including improving road safety (Poletaikin et al., 2024), modernizing and optimizing local governance (Del Pilar et al., 2024), managing socio-economic processes within universities (Gritsenko et al., 2020), and monitoring and analyzing human mobility (Gao et al., 2020).

The **Data Hub** is one of these projects. It was developed to help researchers in creating, harmonizing and visualizing data related to Global Health sector, especially in the area of epidemiology (*Data Snack*, 2024).

Epidemiology studies the disease distribution patterns within populations and the factors that affect this distribution. It is based on the principle that disease or illness does not occur randomly but comes from various characteristics - genetic or environmental - that either increase vulnerability or offer protection (Gordis, 2014).

Uncovering these patterns requires an extensive and meticulous process. Researchers have to analyze a wide range of datasets before they are able to see the patterns that cause a disease to occur. In this context, data visualization techniques become crucial in helping researchers analyze these datasets effectively (Chui et al., 2011).

1.1 Goals

The goal of this thesis is to develop a dashboard for the Data Hub. This dashboard aims to assist researchers in visualizing and analyzing datasets within the Data Hub by enabling them to identify patterns, relationships, and trends in epidemiological context.

The equally important goal of this dashboard is to help researchers maintain the expansive datasets within the Data Hub. Managing these datasets can be difficult, as they often originate from different sources and may sometimes be incomplete, making it demanding for researchers to track missing datasets. By developing a dashboard, the process of identifying incomplete datasets can be made effortless while also improving the efficiency of managing these datasets.

1.2 Structure

The thesis starts with the **Introduction**, which provides the background of the research topic, its significance, and the primary goals of this thesis. The **Material and Methods** section follows, detailing further background, related works as well as the techniques and methodologies used.

In the **Requirements and Use Cases** section, the specific needs of the thesis are identified, along with use cases that illustrate how the system should work in real-world scenarios. Next, the **Implementation** section focuses on the technical details of how the system was developed. The **Evaluation** section follows, where the performance, functionality, and usability of the system are assessed by conducting a questionnaire. Finally, the thesis wraps up with the **Discussion and Conclusion** section.

2 Materials and Methods

This chapter starts off with a theoretical overview of geographic data and dashboards, followed by a review of related works in the design and implementation of analytical dashboards. It then introduces Data Hub, detailing its key elements and structure. Finally, the chapter provides a brief overview of the tools and methodologies used, including Django and data normalization methods.

2.1 Geographic Data

Geographic data, also known as geospatial data, refers to data or information that is either directly or indirectly linked to a specific location on Earth (ISO, 2015).

Geographic data can be represented in two primary formats, namely; vector and raster data model. Vector data model represents spatial features, such as buildings, forests, streams, using points, lines, and polygons. Raster data model on the other hand uses a grid of cells to represent spatial features: point features are shown as individual cells, line features as connected cells, and polygon features as groups of adjacent cells (Figure 2.1) (Chang, 2019).

Displaying geographic data requires a whole different approach than displaying quantitative information, such as sales numbers. While quantitative information can easily represented by creating a graph or a table, geographic data requires a focus in illustrating physical and/or abstract characteristics. Physical features might include landforms, terrains, and bodies of water, whereas abstract features can contain political boundaries and similar attributes (Few, 2009).

When combining quantitative information and geographic data, two approaches are really popular and widely used: color intensity and size, where "size" refers to the use of shapes, that vary in size based on the information. Color intensity approach, also

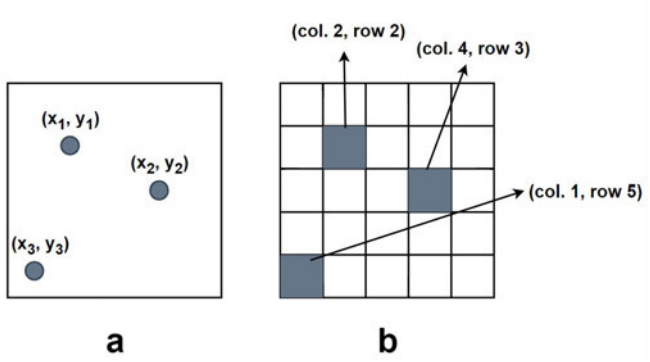


Figure 2.1: Point features represented in vector model (a) and in raster model (b) (Chang, 2019).

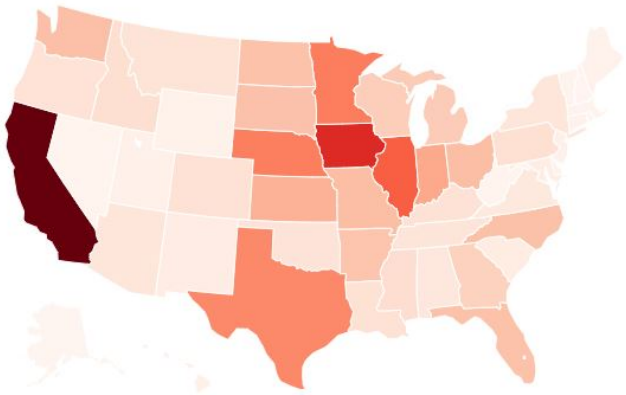


Figure 2.2: Color approach of displaying Quantitative Information in a Map - *Choropleth Map* (Plotly Technologies Inc., 2024)

known as choropleth map, encodes quantitative information into different intensities of color (Figure 2.2). The size approach utilizes shapes, typically circles, to encode the quantitative information into different sizes of the shape. The larger the value is, the bigger the shape becomes (Figure 2.3) (Few, 2009).

2.2 Data Visualization Techniques and Tools

Data visualization refers to the process of creating visual representations of data, such as charts or graphs, to make the information easier to interpret and understand (Srivastava, 2023).

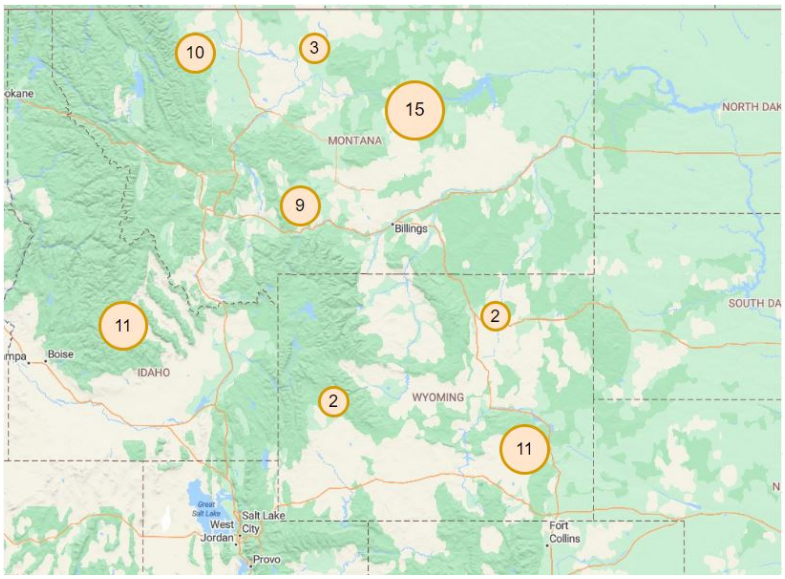


Figure 2.3: Size approach of displaying Quantitative Information in a Map

2.2.1 Data Visualization Techniques

According to Srivastava, several data visualization techniques are commonly used. A summary of these techniques is provided in the table below:

Visualization Type	**Purpose**	**Key Features**
Bar Chart	Compare data across categories	Rectangular bars with lengths representing data magnitude
Line Graph	Show trends over time	Points connected by lines; each point represents a time-based data value
Scatter Plot	Show relationships between two variables	Individual points represent pairs of values for two variables
Heat Map	Display value distribution in a matrix	Colored cells represent the significance of data values in a grid
Network Diagram	Visualize relationships between entities	Nodes represent entities, edges represent relationships; size and color can indicate attributes

Table 2.1: Summary of Data Visualization Types (Srivastava, 2023)

In epidemiology, line graphs and bar graphs are commonly used as means of data visualization to summarize and provide an overview of the data (Adewuyi et al., 2022). Given that the Data Hub hosts temporal data covering an extended time span, a line graph was selected as the most appropriate visualization type. With line graphs, it would be easier to observe the trends clearly over time.

2.2.2 Data Visualization Tools

Srivastava introduced three classification of data visualization tools:

Spreadsheets are among the most widely used and accessible tools for data visualization. They offer basic charting capabilities such as bar charts, line graphs, and scatter plots. Common examples include Microsoft Excel and Google Sheets.

Data Visualization Software refers to specialized tools designed for in-depth data analysis and visualization. These platforms support advanced features like interactive dashboards, heat maps, and network diagrams. Examples include Tableau, QlikView, and Power BI.

Programming Libraries offer a highly customizable and flexible approach to data visualization. While they allow for more tailored visualizations, they require a greater level of technical knowledge. Popular libraries include Matplotlib, ggplot2, and D3.js.

Since the application of data visualization in this thesis requires a highly customizable and flexible approach, programming libraries were chosen as the visualization tools. Specifically, libraries such as Plotly and Leaflet were used, as they allow for the creation of interactive and customized visualizations.

2.3 Dashboards

A dashboard visually represents information as a combination of text and most importantly graphics. They utilize graphical elements with the main purpose of communicating the information effectively as graphical elements can convey more meaning than text alone (Few, 2006).

Dashboards can be used for many different purposes, but Few classified dashboard according to its role into three categories:

- **Dashboard for Strategic Purposes**
  Used mainly in businesses, this type of dashboard focuses on key performance indicators and forecasts, offering quick insights to assist in decision making processes. For this, simple, static displays work best, with data snapshots taken on a daily, weekly, or monthly basis rather than in real-time.

- **Dashboard for Analytical Purposes**
  This type of dashboard needs a more detailed approach as context is highly relevant for analytical purpose. The dashboard should allow user interactions, enabling users to look into the details and explore the causes behind what the users see. This type is suitable for uncovering patterns and is also beneficial from snapshots of data instead of real-time data.

- **Dashboard for Operational Purposes**
  Unlike the other two types of dashboard, this one benefits from real-time data as it is mainly used in cases that require immediate and appropriate response. Therefore, simple but eye-catching design is recommended. Furthermore, the information is usually displayed with more detail to offer deeper insight.

The dashboard for this thesis falls mainly into the *Analytical Dashboard* category, as complex datasets are explored as well as details about these datasets are necessary.

2.4 Related Works

Over the years, numerous studies have focused on designing and implementing analytical dashboards. Some of them are discussed in this section (Figures 2.4 and 2.5).

De Ruvo (2024) implemented a dashboard named SPREAD, short for Spatiotemporal Pathogen Relationships and Epidemiological Analysis Dashboard, which is an innovative web-based tool designed to improve public health by addressing challenges in monitoring infectious disease outbreaks. It integrates genomic, geographical, and temporal data to provide detailed insights into disease transmission dynamics, helping identify pathogen relationships and spatial patterns. The authors utilized GrapeTree to manage genomic relationships, Leaflet in combination with OpenStreetMap for handling geographical data,

and ReportTree for detecting genetically similar samples. It was primarily developed with JavaScript, while its user interface was created using CSS3 and HTML5.

Lopes et al. (2022) developed MultiMaps, a tool that integrates data from various sources, enabling users to analyze and evaluate geospatial data, therefore simplifying the decision-making process. The goal of this tool was to improve decision-making in controlling epidemics, as well as infectious and neglected diseases, within urban environments in Brazil. MultiMaps lets users import data from various sources, process it geographically, and organize it into layers. It then lets users (via its user-friendly web interface) decide how to weight each layer to create detailed, informative choropleth maps that can help with understanding and responding to different phenomena, like epidemics. The tool was developed using the Django framework with data stored in PostgreSQL with the PostGIS extension, and the frontend was built with JavaScript, HTML, CSS, and Leaflet.

Curriero et al. (2021) implemented a dashboard, that uses geographic viewpoint to track tickborne diseases globally while also providing spatial data and contextual maps to represent a One Health approach. Curriero et al. decided to utilize R Shiny, an open-source web application framework for R, because of familiarity, its flexibility and also its advanced statistical functionalities. The dashboard offers three main features: an overview of Lyme disease data, a map exploration tool for tickborne diseases, and a storymap that provides context for the other two features.

Chen et al. (2025) developed a framework that combines an urban digital twin with a geospatial dashboard to visualize flood-related infrastructure vulnerabilities across various spatial and temporal scales. This project utilized Dash by Plotly as its main framework. The Urban Digital Twin (UDT) part was developed using Deck.gl and embedded into the dashboard as an HTML component. Plotly was used to generate interactive charts and graphs, while Dash Bootstrap Components were used to design the user interface. Additionally, the dashboard incorporated Shepherd.JS, a JavaScript library, to implement a guided tour that improve the user experience.

Farmanbar and Rong (2020) created a cloud-based city dashboard prototype - named Triangulum City Dashboard, that was designed to collect, store, and visualize large volumes of heterogeneous data. It enables users to explore this data, improve communication with urban service providers, and gain an overview of the city's current state. The dashboard offers various functionalities, such as a real-time overview of the city from sensors, parking management dashboard to help users find an appropriate parking spot that includes an integrated map, energy consumption management dashboard, etc. Farmanbar

and Rong utilized Python and JavaScript in combination with Flask, Plotly, and Leaflet to develop the dashboard.

2.5 Data Hub

Data Hub is a Geographic Information System (GIS) project involving two main contributors, namely; Juliane Boenecke and Jonathan Ströbele, who are based at Bernhard Nocht Institute for Tropical Medicine (BNITM) in Hamburg, Germany. The main goal of the Data Hub is to support the global health sector by working with stakeholders in areas such as research, data management, analysis, epidemiology, and public health (*Data Snack*, 2024).

The Data Hub itself is an open-source platform that can be hosted locally, that focuses on supporting the global health sector in managing data, including data creation, harmonization, and visualization. It combines diverse spatial and temporal data providing a foundation for advanced data analysis (*Data Snack*, 2024).

2.5.1 Main Elements

The two main elements of the Data Hub are:

Shape A shape in Data Hub refers to a geometry of a specific area that follows a hierarchical structure, where each shape can have a parent and multiple children. This hierarchy is typically organized according to administrative levels. For instance, a country has multiple states and each state can have multiple cities.

Data Layer A data layer in Data Hub represents the value of a specific parameter and maps it to an available shape and time.

2.5.2 Functionalities

In the current version of Data Hub, users can either import their own shapes and data layers or utilize the available preprocessed data in the form a data dump. Users can then browse these shapes and data layers to view details such as the shape's geometry on a

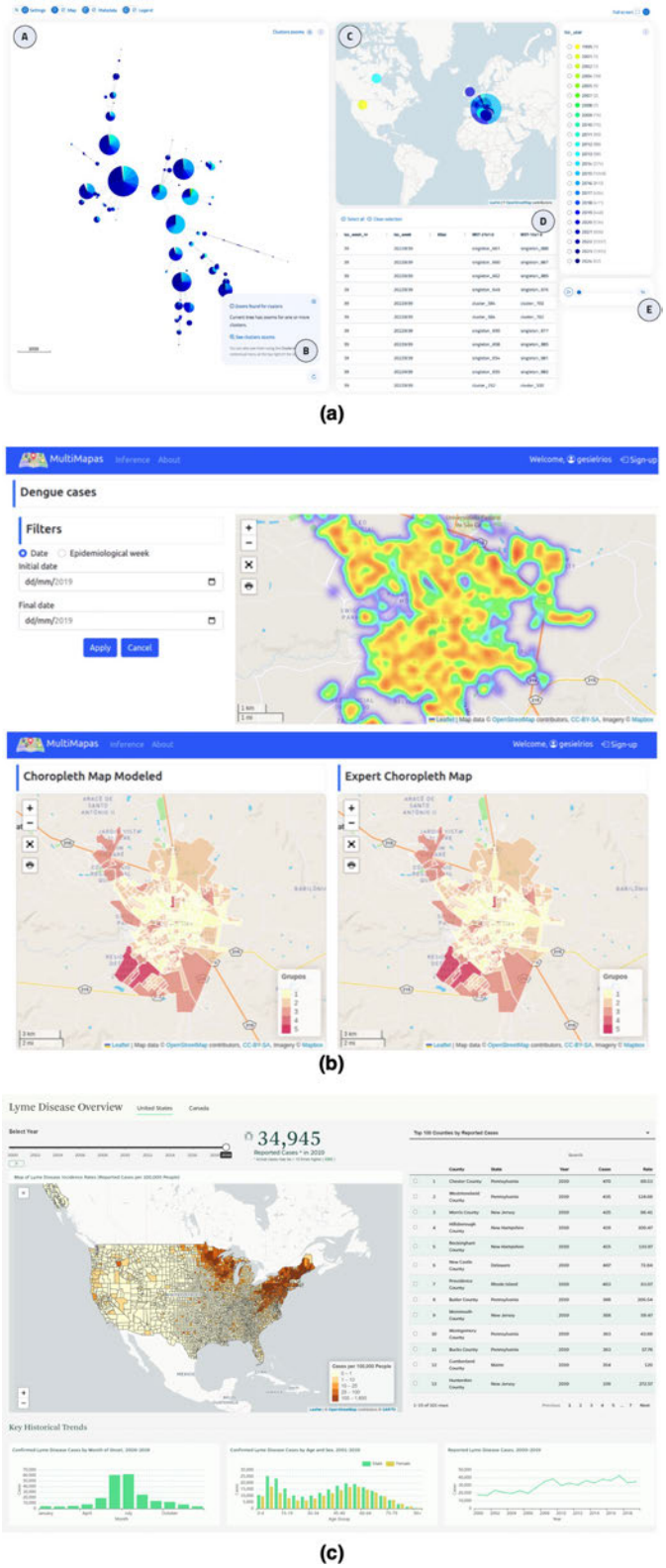


Figure 2.4: Dashboard Implementations Overview: (a) **Spatiotemporal Pathogen Relationships and Epidemiological Analysis Dashboard** (De Ruvo, 2024) (b) **MultiMaps** (Lopes et al., 2022) (c) **Lyme Map** (Curriero et al., 2021)

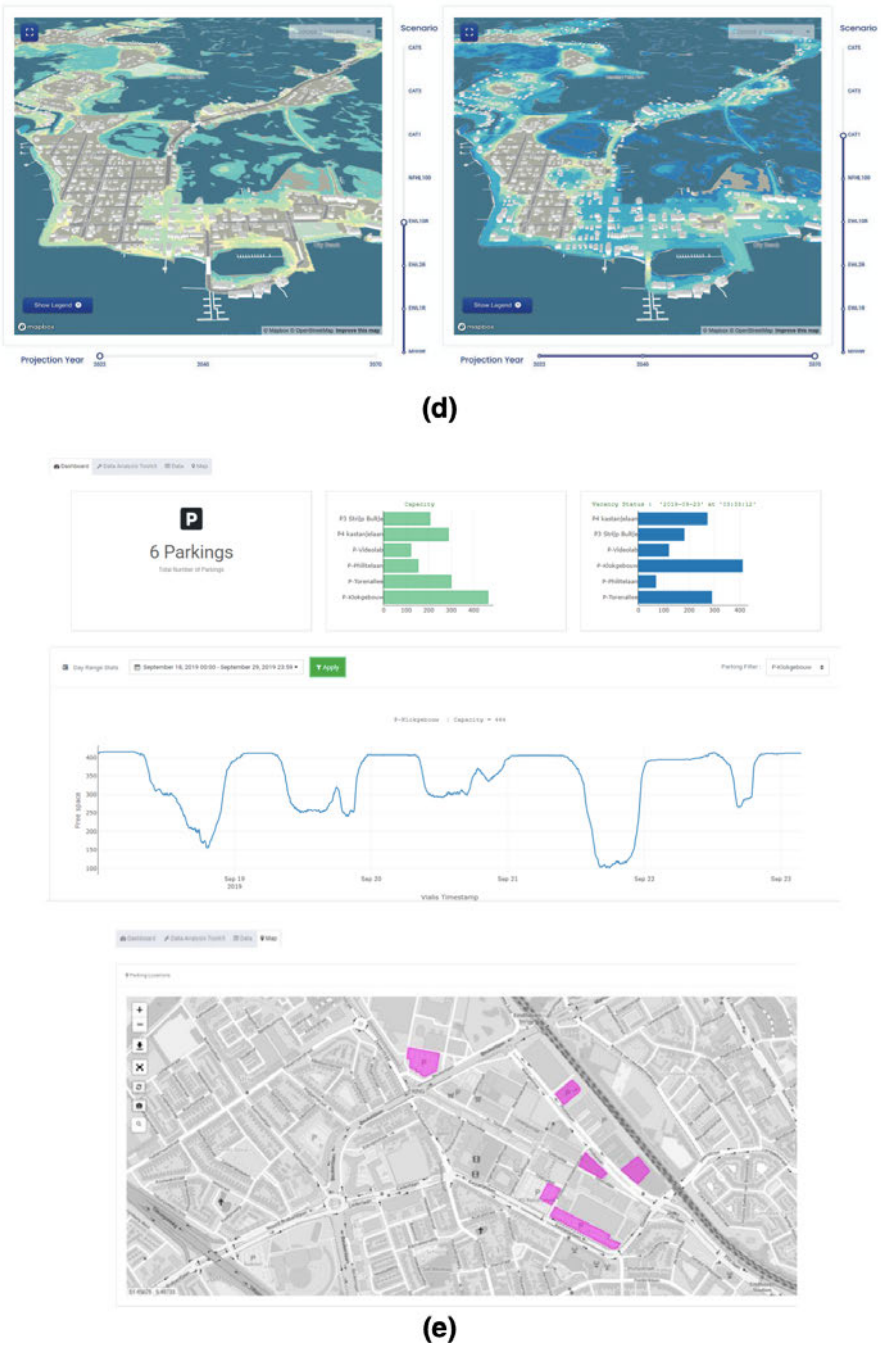


Figure 2.5: Dashboard Implementations Overview: (d) **Cloud-Based Geospatial Dashboard** (Chen et al., 2025) (e) **Triangulum City Dashboard** (Farmanbar and Rong, 2020)

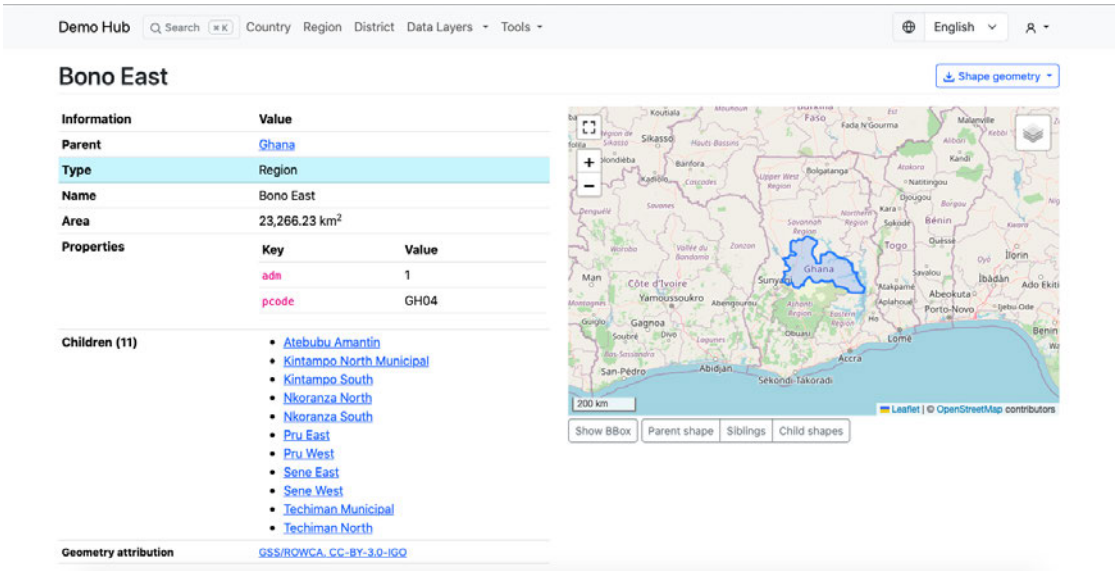


Figure 2.6: Shape Detail Page of the Data Hub (*Data Snack*, 2024)

map, its relationships to other shapes, and various metadata associated with the data layers (Figure 2.6).

Furthermore, users can visualize different datasets on a map, allowing them to explore these across both time and space. As an example, users can view how a data layer, such as population density, differs between various regions of a country in a specific year (Figure 2.7).

2.5.3 Structure

The Data Hub is built with Python using the Django framework, alongside PostGIS, a PostgreSQL-based database designed for managing geospatial data. It consists of two primary components: the Django project and the Database. The Django project currently contains three Django apps, each with its own unique functionalities (Figure 2.8).

2.5.4 Database

There are three main components stored in the database, namely (Figure 2.9):

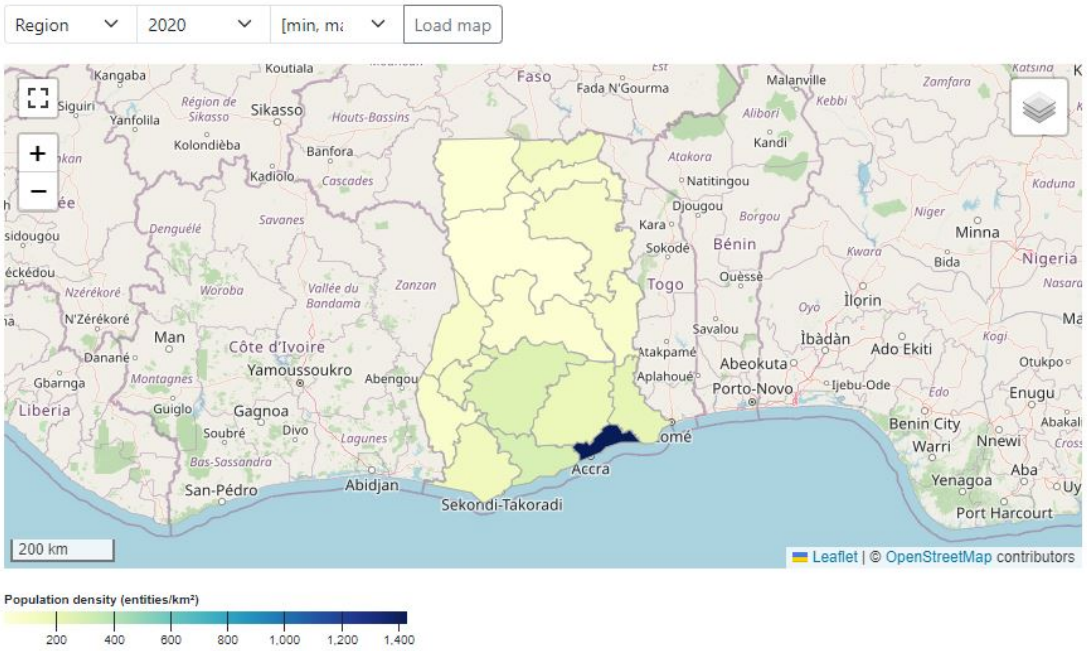


Figure 2.7: Population Density of Regions in Ghana in 2020 from Data Hub (*Data Snack*, 2024)

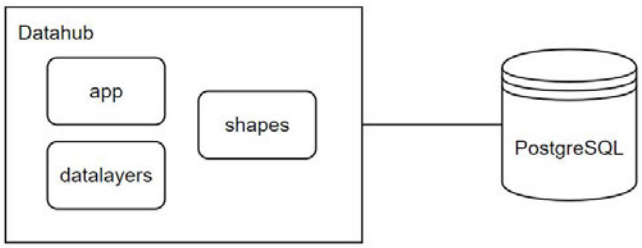


Figure 2.8: Data Hub Components - *simplified*

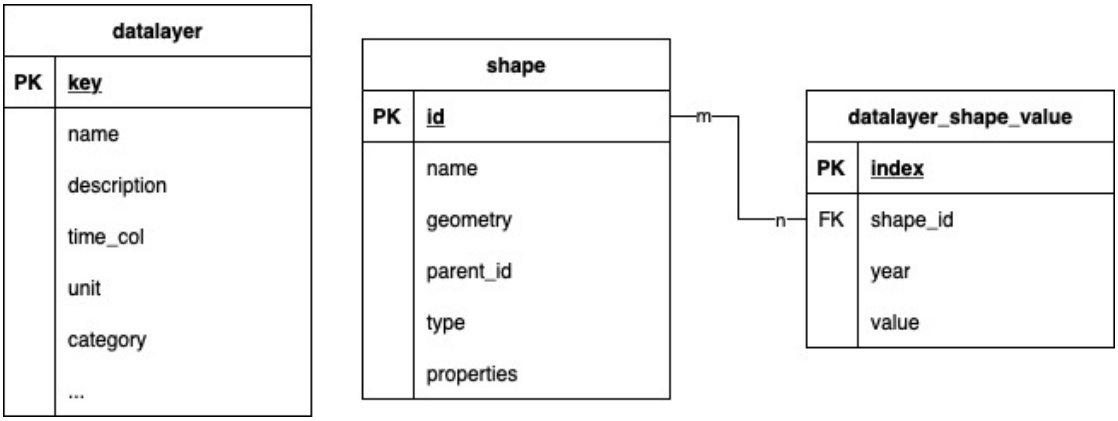


Figure 2.9: Data Hub Entity-Relationship Model

Shape is stored as a table listing all available shapes with unique IDs and their geometry data represented as polygons.

Data Layer is stored as a table listing all available data layers with unique keys and their metadata information.

Data Layer Value (Relationship between Shape and Data Layer) is stored into multiple tables, with the data layer keys serving as the table names. Each table has ID of shapes as foreign keys as well as the value and time.

2.5.5 Data

As previously stated, the Data Hub offers a readily available data dump, which includes geospatial shapes and public health data specific to Ghana. This dataset includes 277 preprocessed shapes categorized into three types: country, region, and district - sourced from the Humanitarian Data Exchange. Additionally, the data dump provides access to 37 preprocessed data layers relevant to public health in Ghana, such as population data, temperature data and topographic data - which were collected from various sources. To ensure efficiency, this thesis will take a full advantage of this data dump.

2.6 Django

The development of the Data Hub relies heavily on Django. Django is a free-to-use, Python-based web development framework that follows the MVC (Model-View-Controller) pattern. The *models* represent the tables and their relationships in the database. The *views* define the information that should be sent to the client, and the *controllers* handle the logic performed by the server. However, in Django, the roles of views and controllers are replaced by "views" and "templates," respectively (Dauzon et al., 2016).

2.6.1 Architecture

As illustrated in Figure 2.10, Forcier mentioned that the HTTP communication protocol is the closest to the user. Users can send requests to Django web applications and receive responses in their web clients through URLs. These clients may also run JavaScript with Ajax to handle server interactions in the background.

The database serves as the persistent storage, managed by models and the Django ORM. Communication with the database occurs through Python's DB-API and the database's client library in the form of an adapter.

Between the user and the database, Django serves as the core of the application. The views manage the process of creating, updating, and deleting data through the ORM, interacting with the database, and rendering the final user view using templates.

Incoming HTTP requests are passed from the web server to Django, beginning at the request middleware layer. Django then uses the URLconf to route the request to the correct view, which performs the main operations, utilizing models and templates as needed to generate the response. Finally, the response passes through another middleware layer for any final processing before being sent back to the web server and forwarded to the user (Forcier, 2009).

2.6.2 Why Django?

There are a few advantages that make Django a popular choice among developers. Firstly, Django is an open source framework, which makes it completely free as well as freely use- and modifiable. Secondly, Django is highly customizable, making it easy for developers

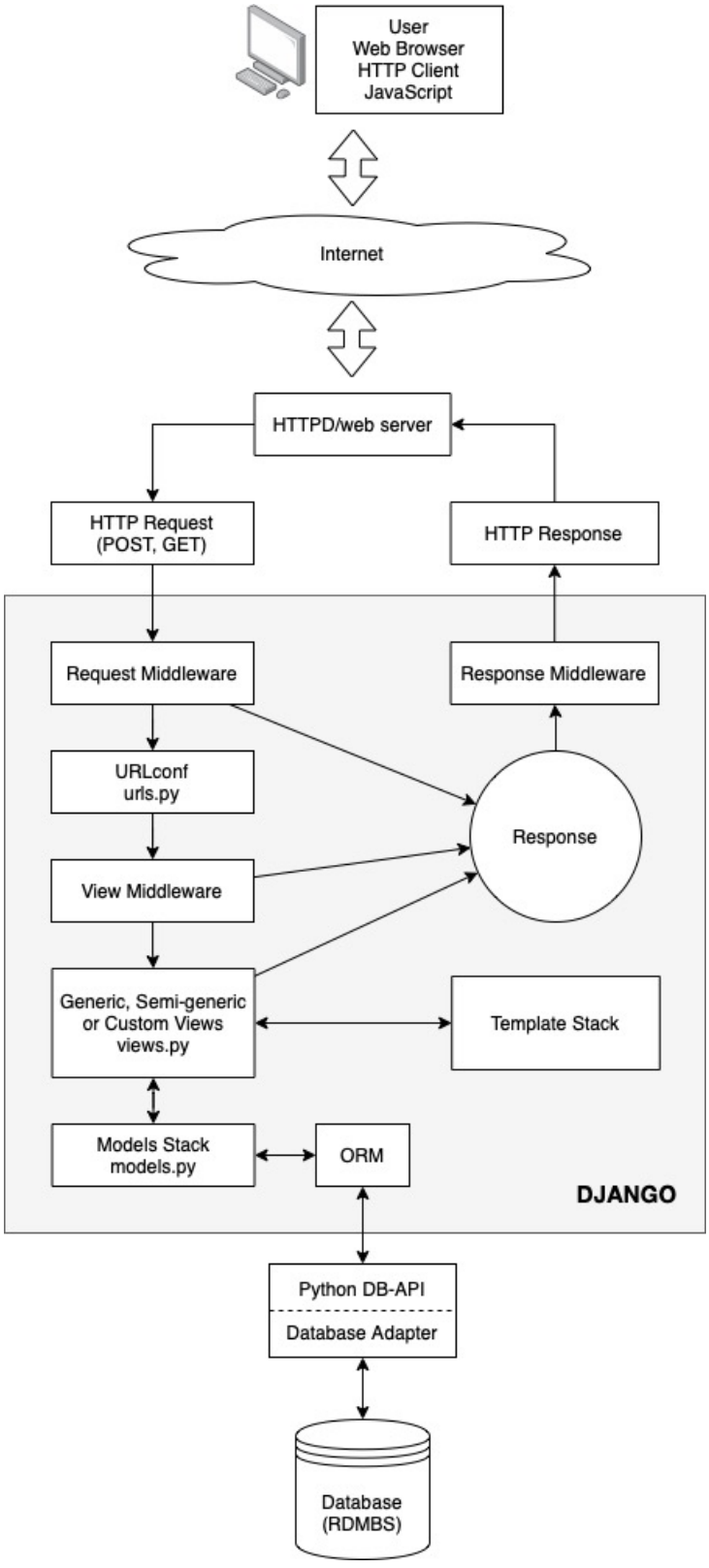


Figure 2.10: Django Overall Architecture (Forcier, 2009)

to adapt to it by creating modules, enabling them to tailor the framework to suit the needs of specific applications.

Furthermore, since it uses Python, developers can take advantage of Pythons's libraries and its readability. Lastly, Django was designed with a focus on clear code and solid architecture, following the "Don't Repeat Yourself" (DRY) principle, which holds on to simplicity and efficiency by eliminating redundant code (Dauzon et al., 2016).

Considering these advantages, as well as the need to maintain consistency and ensure compatibility with the existing Data Hub, which also uses Django, the decision was made to choose Django for this thesis. This option helps to create a more seamless and cohesive extension of the Data Hub.

2.7 Data Normalization

The Data Hub is designed to host a wide variety of datasets. But these datasets can have different value ranges and units of measurement. And as stated before, one of the goals of this thesis is to allow researchers compare these datasets and assist them in identifying patterns and relationships between the datasets. In order to enable meaningful comparisons and improve the accuracy of the analysis, it is necessary to apply data normalization techniques. Data normalization or data standardization is the process of transforming the data to fit within a common range (Han et al., 2012).

In their book, Han et al. introduced three data normalization methods:

Min-max normalization is performed by computing:

$$v_i' = \frac{v_i - \min_A}{\max_A - \min_A} (\text{new\_max}_A - \text{new\_min}_A) + \text{new\_min}_A$$

In this equation:

- v_i' is the normalized value of v_i, the original value of attribute A.
- $\min_A$ and $\max_A$ are the minimum and maximum values of the attribute A in the original data.
- new_min_A and new_max_A are the new desired minimum and maximum values for the normalized data range.

Min-max normalization is useful if the data distribution should be preserved.

z-score normalization is performed by computing:

$$v_i' = \frac{v_i - \overline{A}}{\sigma_A}$$

In this equation:

- v_i' is the normalized value of v_i, the original value of attribute A.
- $\overline{A}$ is the mean of attribute A
- σ_A is standard deviation of attribute A, which indicate how spread out a data distribution is (higher standard deviation means the data are spread across a wider range).

Z-score if useful if the minimum and maximum values are unknown.

Normalization by decimal scaling is performed by computing:

$$v_i' = \frac{v_i}{10^j}$$

In this equation:

- v_i' is the normalized value of v_i, the original value of attribute A.
- j is the smallest integer such that

$$\max\left(|v_i'|\right) < 1$$

For this thesis, **min-max normalization** with the range between 0 and 1 will be applied to standardize the datasets during comparisons.

Min-max normalization is preferred due to its efficiency, as the minimum and maximum values are easier to determine than the mean and standard deviation in z-score. Min-max normalization also preserves the original distribution of the data allowing the data to be consistent and as close as the original non-normalized data.

3 Requirements and Use Cases

This chapter offers an in-depth introduction to the dashboard's requirements, outlining the key objectives and functionalities needed. It also explores various use cases, illustrating how the dashboard should perform in different scenarios.

3.1 Requirements

3.1.1 Functional Requirements

The functional requirements were gathered from the two main contributors of the Data Hub and can be broken down into three primary functional requirements, along with one secondary requirement.

Functional requirements define the system's observable behaviors under certain conditions and the actions it will allow users to perform (Wiegers et al., 2013). Each functional requirement addresses a specific aspect of the system's behavior.

Primary Functional Requirements

1. **FR-1: Data Coverage Map**
    - Description: The Data Hub shall present data coverage and density that reflects the database.
    - Inputs:
        - Shape Type
    - Outputs:

  - Choropleth Map - displays the number of available data layers as colors for each shape based on the amount of available data layers.
  - Data Coverage Ranking - displays a ranking of shapes based on the selected shape type, ordered from the highest to the lowest number of available data layers.
  - Available and Missing Data Layers - displays all the available and missing data layers for selected shapes.
  - Binary Matrix - shows the data availability for each shape and data layer combination.

2. **FR-2: Temporal Slider Map**
   - Description: The Data Hub shall allow users to visualize the changes in the value of various data layers over time using a temporal slider. Additionally, the Data Hub shall allow users to compare the values of data layers across similar geographic shapes.
   - Inputs:
     - Data layer(s)
     - Shape type
   - Outputs:
     - Choropleth Map - displays the value of the selected data layer as colors for shapes within the selected shape type.
     - Value Trend Graphs - display of the value trends of the highest available administrative level shape and selected shapes.
3. **FR-3: Temporal Trends**
   - Description: The Data Hub shall allow users to compare the value of various data layers for a specified shape, to help visualize the relation between different data layers, potentially uncovering patterns
   - Inputs:
     - Shape

  - Data Layer

- Outputs:
  - Individual graphs displaying the individual value trends of selected data layers for a specified shape.
  - A combined normalized graph showing the value trends of all selected data layers for a specified shape.

Secondary Functional Requirements

1. **FR-11: Choropleth Map Colors and Transparency**
   - Description: The Data Hub shall allow users to modify the legend colors using predefined presets and adjust the transparency of choropleth maps.
   - Inputs:
     - Color preset
     - Transparency
   - Outputs:
     - Choropleth Map with adjusted colors and transparency.

3.1.2 Non-Functional Requirements

Non-functional requirements are the system's constraints, important quality attributes, and external interface requirements, such as user and hardware interfaces (Wiegers et al., 2013). Below are the non-functional requirements for this thesis.

1. **NFR-1: Performance**
   - Description: The Data Hub shall respond to user requests within 150 seconds (2,5 minutes).
   - Priority: High
2. **NFR-2: Usability**

- Description: The Data Hub shall allow users to interact with it easily without referring to any documentation. The user-interface shall be simple and straightforward.
- Priority: High

3.2 Use Cases

To help make the functional requirements easier to understand as well as to help clarify how the system will work, use cases are provided below. A use case outlines a series of interactions between a system and an external actor, such as a person, another software system, or a hardware device, producing an outcome that benefits the actor (Wiegers et al., 2013).

Each use case in this section covers the interaction between the future users and the system of the primary functional requirements outlined in the previous section with the same name (Section 3.1.1).

1. **FR-1: Data Coverage Map**
   - Description: Allow users to visualize the coverage and density of available data layers across available geographic shapes and evaluate data availability within the database.
   - Main Success Scenario:
     1) The user selects a shape type (e.g. country, region, district).
     2) The user selects relevant data layers.
     3) The user clicks on the load button.
     4) The system generates a choropleth map showing data density for each shape.
     5) The system displays a ranking of shapes, ordered from the highest to the lowest number of available data layers.
     6) The system displays a list of available and missing data layers for the selected shapes.

7) The system displays a binary matrix that represents the shapes as rows and data layers as columns.

8) The user reviews the data visualizations and rankings to analyze data coverage and density.

2. **FR-2: Temporal Slider Map**

- Description: Allow users to overlay different data layers across available geographic shapes and visualize how the values change over time.
- Main Success Scenario:

  1) The user selects a shape type (e.g., country, region, district).

  2) The user selects a data layer.

  3) The system generates a choropleth map reflecting the values of the selected data layer.

  4) The system generates a graph for the selected data layer.

  5) The system adds a trace showing the value trend of the highest available administrative level shape into the graph (e.g., country, global).

  6) The user reviews the data visualization and graphs to analyze the trends.

- Alternative Scenarios:

  – Add a new data layer

    1) The user selects a new data layer.

    2) The system generates a new choropleth map layer, reflecting the values of the newly selected data layer, overlaying the previous data layer.

    3) The system generates graphs for the selected data layer.

    4) The system adds a trace showing the value trend of the highest available administrative level shape into the graph (e.g., country, global).

    5) The system adds traces of the previously chosen shapes into the graph.

  – Add a trace to the graphs

1) The user clicks on a shape on the choropleth map.
2) The system adds a trace showing the value trends of the selected shape to all graphs.

– Drag the temporal slider

1) The user drags the temporal slider to a different year.
2) The system updates the choropleth map that reflects the values of the selected data layers for the chosen year.

– Remove a data layer

1) The user clicks on the remove layer button.
2) The system removes the selected data layer from the choropleth map.
3) The system removes the corresponding graph for the removed data layer.

3. **FR-3: Temporal Trends**

- Description: Allow users to compare various data layers and visualize how these create a pattern.
- Main Success Scenario:

1) The user selects a shape type (e.g. country, region, district).
2) The user selects a shape.
3) The user selects at least one data layer.
4) The user clicks on the load button.
5) The system generates a graph showing the trend of every chosen data layer.
6) The system generates a graph showing the normalized values of the chosen data layers.
7) The user reviews the graph to analyze the data and uncover patterns.

- Exception:

- If a data layer is missing, a message is shown to the user informing that the data layer is not available in the database.

4 Implementation

This chapter describes the implementation of the dashboard, covering its architecture, data pre-processing, as well as the directory and URL structures. It also presents sequence diagrams that illustrate the interactions between the user, backend, frontend, and database. Additionally, this chapter discusses the tools and frameworks used throughout the development process.

4.1 Architecture

For this thesis, a new Django app called 'dashboard' was added. This new component communicates with the existing components of the Data Hub as well as the database to achieve the stated goals (Figure 4.1).

4.2 Data Pre-processing

As stated in Section 2.5.4, the Data Hub stores data layer values in separate tables, with the data layer key serving as the table name (See Table 4.1). This led to an issue when

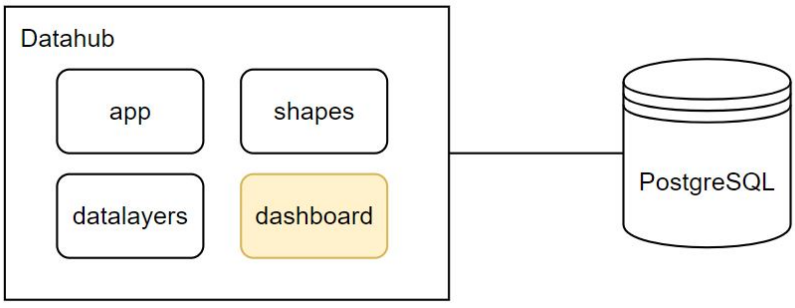


Figure 4.1: New Architecture of Data Hub after adding Dashboard - *simplfied*

iterating through multiple data layer tables, as a new query had to be executed for each table. The iteration process became very time-consuming that it eventually triggered a time-out.

To address this issue, a new table was introduced to collect all data layer values in one place. This table consists of 4 attributes, i.e. shape ID, data layer key, year and the value itself (Figure 4.2). An index on shape ID, data layer key, and year was created for each entry, significantly improving the query performance (Anchlia, 2024).

Since the time column (temporal resolution) of a data layer can be daily, monthly or yearly, it was necessary to convert non-yearly temporal resolutions, such as daily and monthly, into a yearly temporal resolution by calculating the average value for each year (Figure 4.3).

The data pre-processing procedure consists of the steps shown in Algorithm 1.

Algorithm 1 Data layer pre-processing procedure

```
for each data layer do
    if temporal_solution = YEAR then
        copy all entries
    else
        calculate AVG(value) per year
        insert entry with AVG(value) into table
    end if
end for
```

With this step, the duration needed to iterate through multiple data layers were reduced, thus improving the performance of the dashboard.

Table 4.1: World Population Value Table in the Database

worldpop_popd		
shape_id	year	value
1	2010	24723534
2	2010	478982
3	2010	4804411
...	...	...

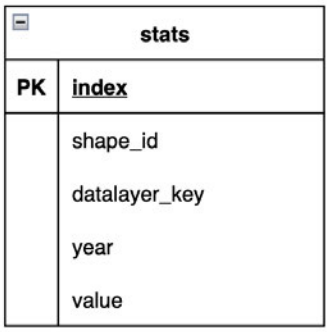


Figure 4.2: Table Structure for Data Pre-processing

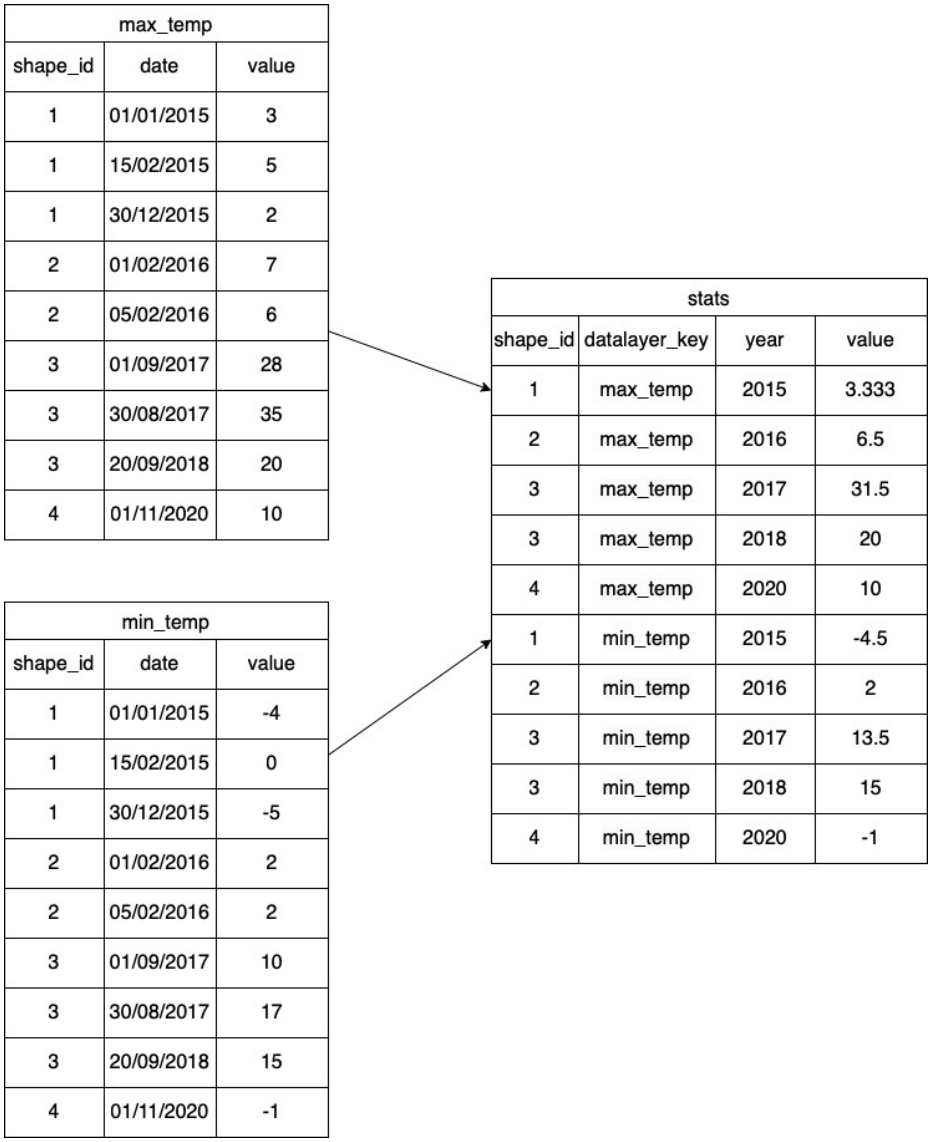
4.3 Implementation Details

The implementation of the dashboard is based on the three primary functional requirements defined in Section 3.1.1. Each implemented feature corresponds directly to one of these functional requirements, ensuring that the system behavior aligns with the intended use case with the same name. This section describes how each requirement is implemented through the system's architecture, URL routing, and communication between the components.

4.3.1 Directory Structure

The dashboard app was integrated into the Data Hub project, resulting in the creation of a new folder with the same name, which is visualized by the directory tree below:

```
.
└── datahub
    ├── app/
    ├── data/
    └── dashboard/
        ├── management/
        │   └── commands/
        │       └── preprocess_shape_dl_year.py
        ├── migrations/
        └── static/
            └── css/
                └── info_map.css
```



max_temp		
shape_id	date	value
1	01/01/2015	3
1	15/02/2015	5
1	30/12/2015	2
2	01/02/2016	7
2	05/02/2016	6
3	01/09/2017	28
3	30/08/2017	35
3	20/09/2018	20
4	01/11/2020	10

min_temp		
shape_id	date	value
1	01/01/2015	-4
1	15/02/2015	0
1	30/12/2015	-5
2	01/02/2016	2
2	05/02/2016	2
3	01/09/2017	10
3	30/08/2017	17
3	20/09/2018	15
4	01/11/2020	-1

stats			
shape_id	datalayer_key	year	value
1	max_temp	2015	3.333
2	max_temp	2016	6.5
3	max_temp	2017	31.5
3	max_temp	2018	20
4	max_temp	2020	10
1	min_temp	2015	-4.5
2	min_temp	2016	2
3	min_temp	2017	13.5
3	min_temp	2018	15
4	min_temp	2020	-1

Figure 4.3: Example of Data Preprocessing of Non-Yearly Temporal Solution Datasets

```
│   │   │   ├── slider.css
│   │   │   └── temporal_trend.css
│   │   └── js/
│   │       ├── customization_utils.js
│   │       ├── info_map.js
│   │       ├── info_map_api.js
│   │       ├── slider.js
│   │       ├── slider_api.js
│   │       ├── slider_utils.js
│   │       ├── temporal_trend.js
│   │       └── temporal_trend_api.js
│   ├── templates/
│   │   ├── home.html
│   │   ├── info_map.html
│   │   ├── slider.html
│   │   └── temporal_trend.html
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   └── views.py
└── ...
```

Within this newly created folder, several key components can be highlighted (Melé, 2024):

- management/: contains data-preprocessing command.
- migrations/: contains database migrations for table related to dashboard, to track model changes and synchronize database.
- static/: contains static files such as CSS and JavaScript.
- templates/: contains the HTML templates.
- models.py: contains data model for the data pre-processing.

- tests.py: contains tests for the functionalities.
- urls.py: contains URL routing for the new functionalities.
- views.py: contains functions that process user requests and send back responses.

4.3.2 URLs

The new functionalities integrated into the Data Hub are accessible via unique URLs. These new URLs correspond to the primary requirements mentioned in Section *3.1.1* and provide access to the new features for the users. The new URLs are mapped as follows:

- /**dashboard**/: Dashboard homepage - links to URLs with dashboard features.
- /**dashboard**/**info-map**/: corresponds to **FR-1: Data Coverage Map**.
- /**dashboard**/**slider**/: corresponds to **FR-2: Temporal Slider Map**.
- /**dashboard**/**temporal-trend**/: corresponds to **FR-3: Temporal Trends**.

4.3.3 Sequence Diagrams

The diagrams in this section illustrate the communication between the User, Frontend (JavaScript), Backend (Django), and the Database for the three primary functionalities *(FR-X)* outlined in Section 3.1.1, and show how each corresponding use case is implemented.

FR-1: Data Coverage Map

The sequence diagram shown in Figure 4.4 illustrates the implementation of use case **FR-1: Data Coverage Map**, where the user visualizes the coverage and density of available data layers across selected geographic shapes.

After selecting the shape type and relevant data layers, the user can click the "Load" button, which triggers an AJAX call that passes the selected shape type, year, and data layer information to the backend.

The backend processes this request by creating four lists of dictionaries to store relevant data: available data layers, missing data layers, geometries of the shapes, and shape

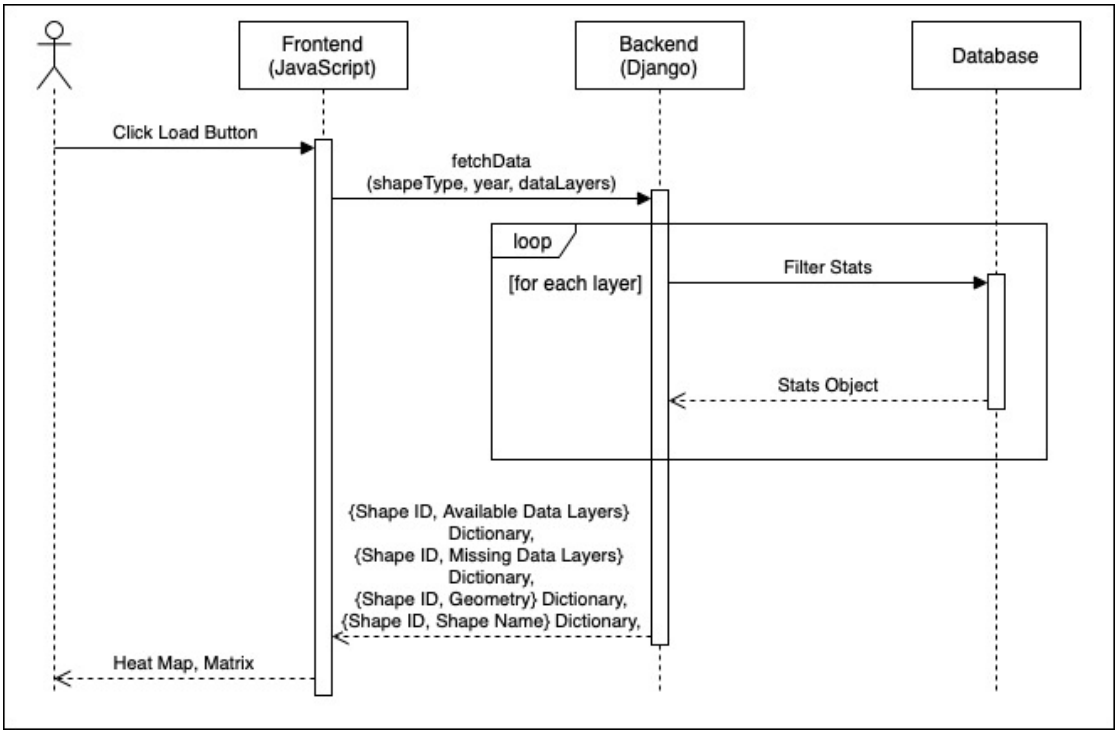


Figure 4.4: Sequence Diagram for Data Coverage Map

names. It then iterates through the data layers, querying the database for the availability of each data layer for the specified shape and year. Once the data is fetched, these dictionaries are returned to the frontend.

The frontend clears any existing map layers and updates them with the newly fetched data. Each geographical shape on the map is processed, and its data is visualized using color coding based on the availability of data layers, resulting in the creation of a choropleth map.

In addition to the map updates, a ranking list is generated and displayed, showing all shapes along with their associated data layers. The list is sorted by the number of available data layers, and users can expand entries to view more details. Furthermore, a matrix is created, providing a detailed breakdown of which data layers are available or missing for each shape. This process covers the *main success scenario* highlighted in the use case.

FR-2: Temporal Slider Map

The sequence diagram in Figure 4.5 illustrates the implementation of use case **FR-2: Temporal Slider Map**, which allows users to overlay multiple data layers across geographic shapes and visualize how their values change over time.

After selecting a shape type, the user can add one or more data layers. Each time a data layer is added, the frontend initiates an AJAX request that sends the data layer key to the backend. The backend responds by querying the available years for that data layer and returning them to the frontend. Based on this, the frontend either initializes a new temporal slider (if none exists) or updates the existing slider's range to reflect the new data.

Next, the frontend sends another AJAX request containing the data layer and shape type information. The backend responds with the minimum and maximum values for the data layer. Using this information, the frontend updates the choropleth map, adjusting the color of each shape based on its corresponding value. This same process is also triggered whenever the user interacts with the temporal slider to select a different year *(Alternative Scenario - "Adjust the temporal slider").*

In addition, the frontend generates a graph for the newly added data layer. It then sends another request to the backend to retrieve the historical values for the highest administrative level shape (e.g., country or global). The backend returns a list of year-value pairs, which the frontend processes to plot a trend line (trace) on the graph.

This sequence of events covers the full interaction described in the *main success scenario* of the use case, and also supports the *alternative scenario "Add a new data layer".*

The second diagram shown in Figure 4.6 illustrates the *alternative scenario "Add a trace to the graphs".* When a user clicks on a shape on the choropleth map, the frontend loops through all the previously selected data layers and sends an AJAX request containing the shape ID and data layer key to the backend. The backend queries the database for the selected shape and returns a list of value-year pairs. The frontend then processes this data and adds a new trace to the individual data layer graph, visualizing the selected shape's historical data.

Finally, another alternative scenario involves removing a data layer. When the user clicks the "Remove Layer" button, the frontend removes the corresponding choropleth map

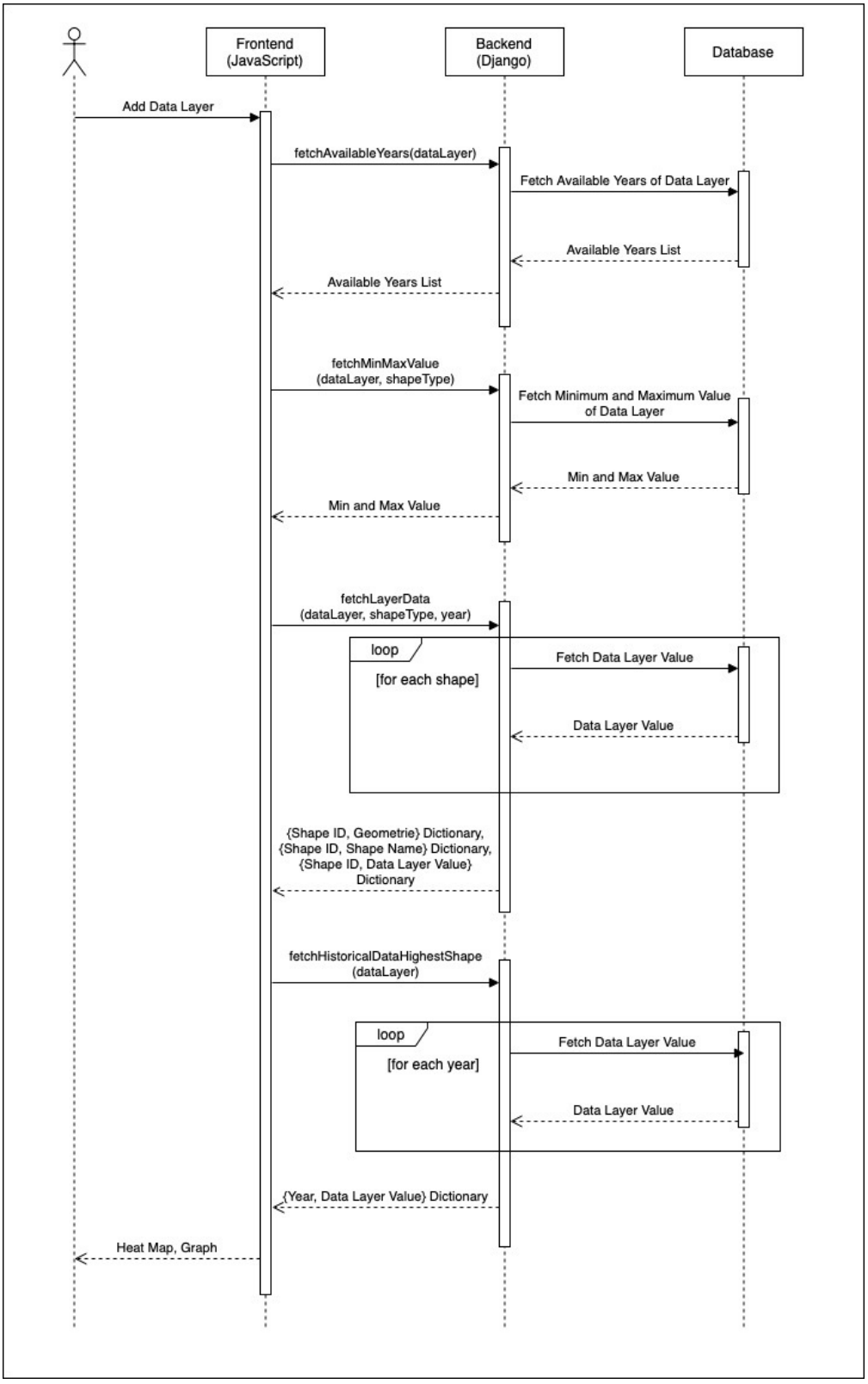


Figure 4.5: Sequence Diagram for Temporal Slider Map

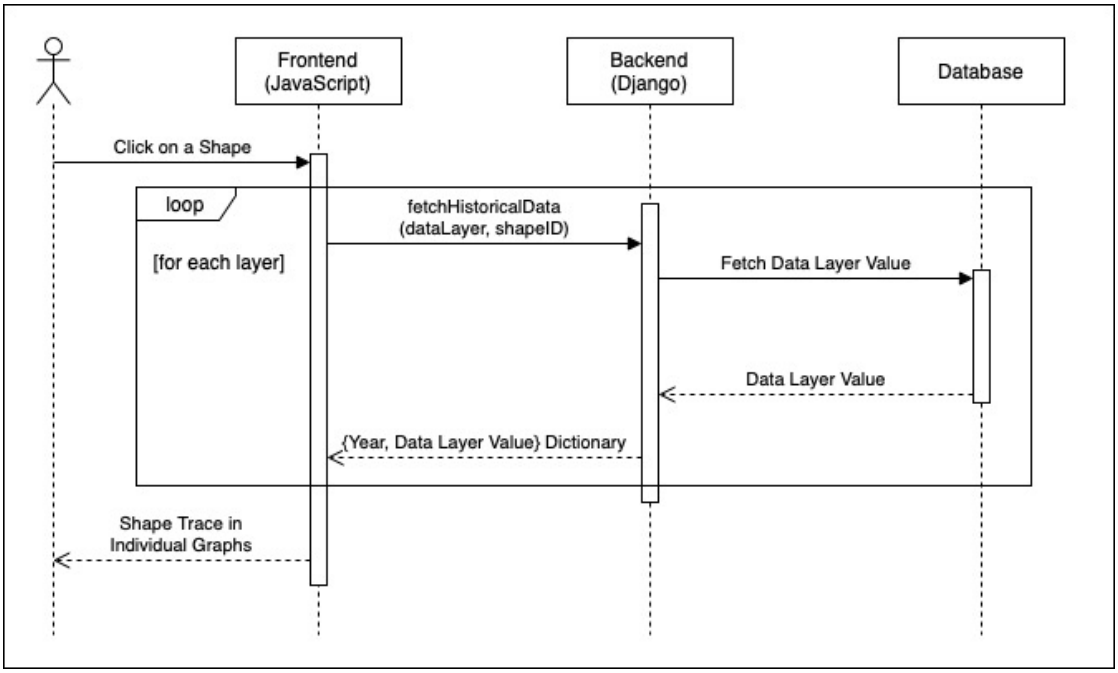


Figure 4.6: Sequence Diagram for Alternative Scenario in Temporal Slider Map — Add a Trace to the Graphs

layer and deletes the associated graph. This supports the use case *alternative scenario "Remove a data layer"*.

FR-3: Temporal Trend

This sequence diagram in Figure 4.7 shows the implementation of use case **FR-3: Temporal Trends**, which enables users to select a shape and multiple data layers to explore patterns and relationships in the data over time.

Users can select a shape and one or more data layers. Upon clicking the "Load" button, the frontend iterates through the selected data layers and sends an AJAX request containing the data layer key and shape ID to the backend. It also creates a graph container to hold all normalized traces.

The backend processes the request by querying the database for the values corresponding to each data layer for every available year and the specified shape. The results are stored in a list, where each entry contains a year-value pair. This list is then returned to the frontend, which uses it to generate a new graph. If the list is empty, the frontend

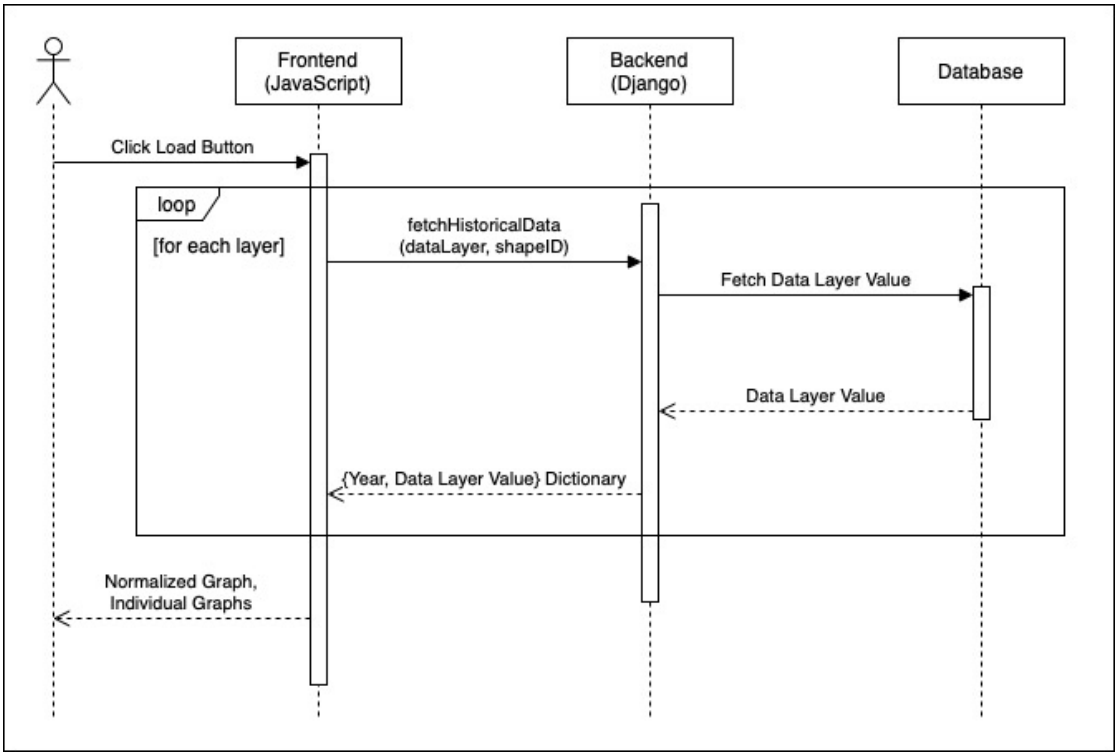


Figure 4.7: Sequence Diagram for Temporal Trend

instead displays an error message indicating that no data could be found. The graph's trace is constructed from the year-value pairs in the list. Additionally, the frontend normalizes the trace using the **min-max normalization** method and incorporates it into the normalized graph. This sequence of steps implements the *main success scenario* of the corresponding use case.

4.3.4 Tools and Frameworks

In Section 2.4, Curriero et al. employed R Shiny for the development of their dashboard. In contrast, De Ruvo and Lopes et al. developed the frontend of their dashboards using the combination JavaScript, CSS, HTML and Leaflet. Similarly, the development of this project utilized a variety of tools and frameworks, including JavaScript, CSS, HTML and Leaflet.

The tools and frameworks that were used in this project are outlined and described in more detail below:

1. **Languages**
    - **Python**: Used for backend development due to its readability and compatibility with the Data Hub.
    - **JavaScript**: Utilized for frontend development and create dynamic contents.
    - **HTML**: Used to construct the structural foundation of the web pages.
    - **CSS**: Applied for styling and layout to create responsive and visually appealing web pages.
2. **Backend Framework**
    - **Django**: The Django framework was chosen for backend development to maintain consistency with the original Data Hub's architecture. Utilizing Django ensured smooth integration with the existing apps while taking advantage of its features.
3. **Database**
    - **PostgreSQL**: The original Data Hub used PostgreSQL with the PostGIS extension to handle geospatial data. This setup was retained to ensure compatibility.
4. **Frontend Technologies**
    - **Leaflet**: An open-source JavaScript library for interactive maps that was utilized for creating choropleth maps.
    - **OpenStreetMap**: A mapping project embedded into Leaflet for geographic data.
    - **Plotly**: A graphing library used for interactive data visualizations, especially the trend graphs.
    - **Bootstrap**: A CSS framework for styling and creating user interfaces effectively.
    - **noUiSlider**: A JavaScript library that was utilized for creating interactive sliders.

- **jQuery**: A JavaScript library to simplify HTML document traversal, event handling, and Ajax interactions.

5. **Version Control**
   - **Git**: A version control system used to track changes and manage code versions.

4.4 Testing

To ensure the correct functionality of the methods within the backend, unit testing was implemented. Unit Testing, also known as "Module Testing", is a form of test that focuses on the smaller but foundational components of the programs instead of the entire program all at once (Myers, 2012).

The unit tests were executed using Django's built-in unit test, which relies on the Python standard library module, "unittest" (Django Project, 2025).

In the setup, a few essential data objects were created, which include:

- Two shape types with unique identifiers and positions
- Two shapes with specific type (from the previously created types) and geometry: one as a root shape and the other as a child shape that references the first
- Two data layers, each with a unique name and key

Additionally, the shapes were linked to the data layers by creating records that store data values for each shape across different years. This arrangement allows for testing various interactions between shapes, types, data layers, and their associated values over time.

5 Results

This chapter presents the results of the dashboard implementation, including an overview of its features and screenshots.

The dashboard is available at https://github.com/z1zzle/datahub. This dashboard has three primary features, each corresponding to one of the primary requirements (FR-X) outlined in Section 3.1.1, which will be highlighted in this chapter.

5.1 Data Pre-processing

To be able to utilize the dashboard, users need to perform the data pre-processing by running this Django custom command:

```
$ docker compose exec datahub
        python manage.py preprocess_shape_dl_year
```

This command will populate the table from which the dashboard retrieves its data, as previously stated in Section 4.2. This pre-processing operation is essential to fulfill the non-functional requirement NFR-2: Performance.

5.2 Feature 1 (FR-1): Data Coverage Map

Feature 1 implements the functional requirement **FR-1 Data Coverage Map**, enabling users to visualize the coverage and density of available data layers across selected geographic shapes in the Data Hub.

Users begin by selecting a shape type (e.g., country, region, district) and relevant data layers, which triggers the system to generate visualizations that reflect the database (Figure 5.1).

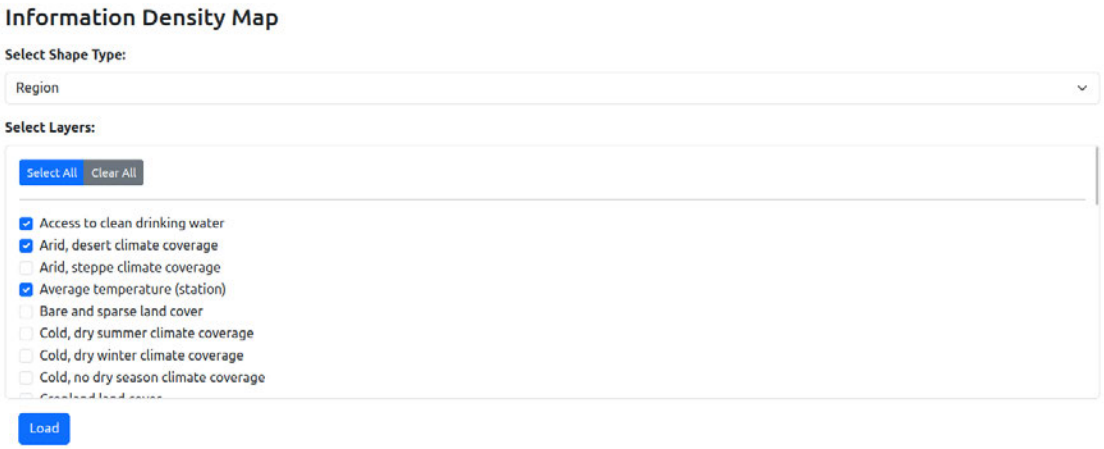


Figure 5.1: User Inputs for Feature 1 (FR-1): Data Coverage Map

By clicking the "Load" button, the system generates four outputs, as outlined in the functional requirement FR-1. As shown in Figure 5.2, these outputs are described in detail below:

Choropleth Map [1] that displays all geographic shapes within the selected shape type, with color coding that reflects the number of available data layers. The colors and the transparency of the choropleth map can be adjusted by the users as described in the secondary functional requirement FR-11: Choropleth Map Colors and Transparency.

Data Coverage Ranking [2] that lists the shapes ranked from the highest to lowest number of available data layers. The items in this ranking show the following elements to help users with the overview of the data coverage:

- Number of available data layers (shown in a blue badge)
- Number of missing data layers (shown in a red badge)
- Coverage percentage of available data layers (shown in a green badge)

Available and Missing Data Layers [3] provides detailed lists for each shape, showing both present and absent data layers in the database. This is shown when users click on the individual item from the ranking list or when users click on a shape on the choropleth map.

Binary Matrix [4] is shown under the map and ranking list. This matrix presents an overview where rows represent shapes, columns represent data layers, and cells indicate data availability.

With the slider, users can adjust the year which automatically updates the components mentioned above, based on the available data for the selected combination of layers, shapes, and year.

This feature follows the main success scenario described in the use case, where the frontend and backend work together to process the user's inputs and visualize the results. Researchers can interact with these visual components to analyze, track, and manage data availability within the Data Hub.

5.3 Feature 2 (FR-2): Temporal Slider Map

Feature 2 implements the functional requirement **FR-2 Temporal Slider Map**, allowing users to explore how the values of selected data layers change over time across geographic shapes.

Users start by selecting a shape type (e.g., country, region, district) and adding a data layer (Figure 5.3).

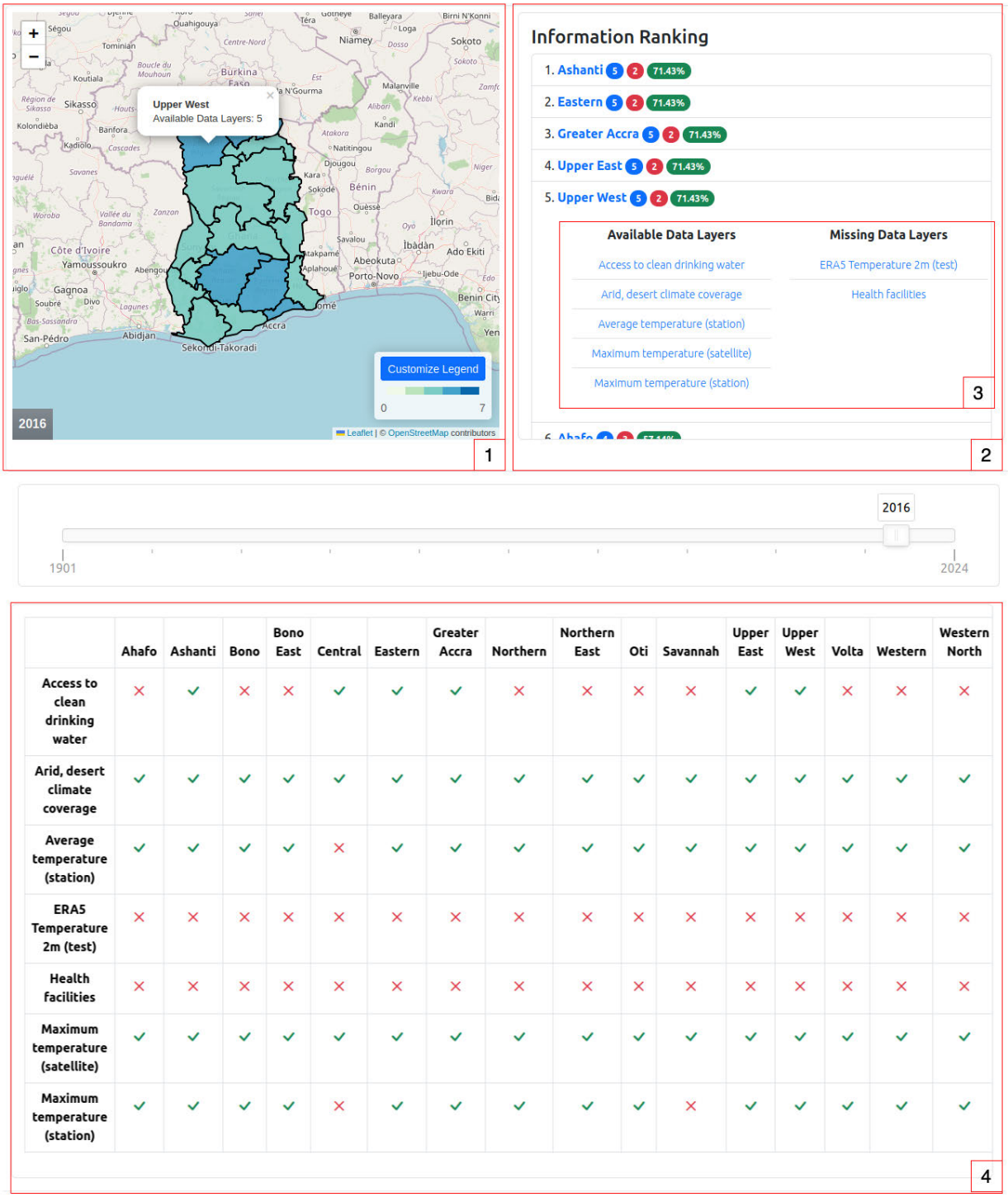
The system automatically generates the following components (Figure

When a new data layer is added, the system creates a new choropleth map layer, stacking it on top of the previously selected layer. It also generates a corresponding individual value trend graph, positioned below the graph of the preceding data layer (Figure 5.5). This process corresponds to the alternative scenario "Add a new data layer" described in use case FR-2.

Users can interact with the map by clicking on specific shapes to add their historical trends (traces) to the graphs, enabling deeper, shape-specific analysis (Figure 5.6). This fulfills the alternative scenario "Add a trace to graphs" from the same use case.

The slider allows users to adjust the year dynamically, updating the map to reflect changes in the data over time.

By clicking on the individual layer option, the users also have the option to customize the color preset and transparency for each data layer allowing them to tailor the visualization



	Ahafo	Ashanti	Bono	Bono East	Central	Eastern	Greater Accra	Northern	Northern East	Oti	Savannah	Upper East	Upper West	Volta	Western	Western North
Access to clean drinking water	✗	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
Arid, desert climate coverage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Average temperature (station)	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ERA5 Temperature 2m (test)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Health facilities	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Maximum temperature (satellite)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Maximum temperature (station)	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓

Figure 5.2: System Outputs Feature 1 (FR-1): Data Coverage Map - [1] Choropleth Map, [2] Data Coverage Ranking, [3] Available and Missing Data Layers, [4] Binary Matrix

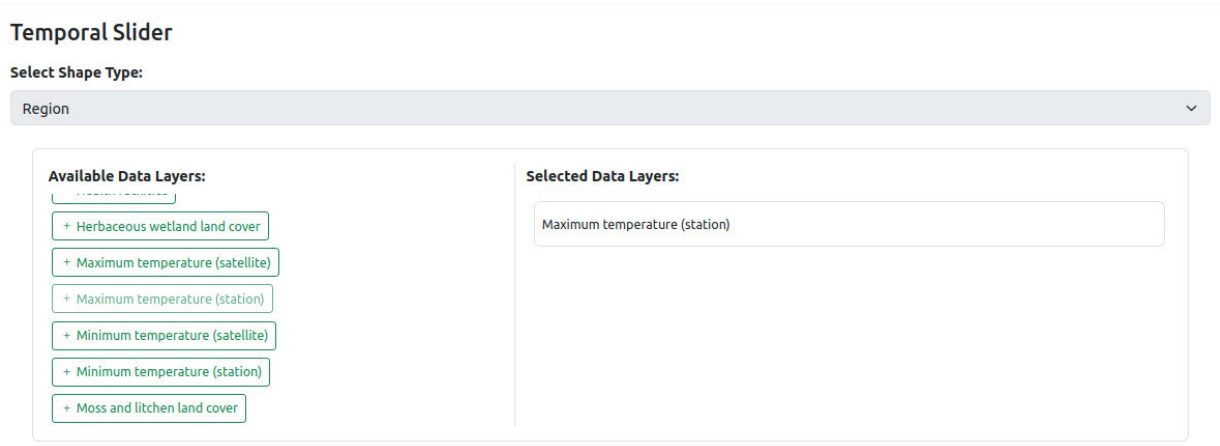


Figure 5.3: User Inputs for Feature 2 (FR-2): Temporal Slider Map

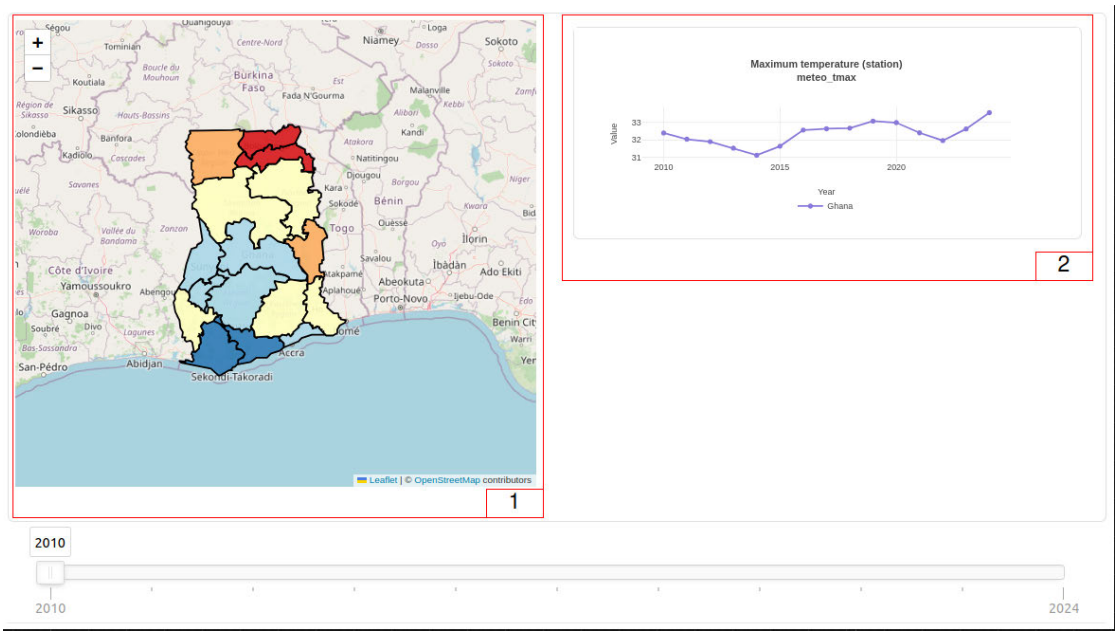


Figure 5.4: System Outputs for Feature 2 (FR-2): Temporal Slider Map

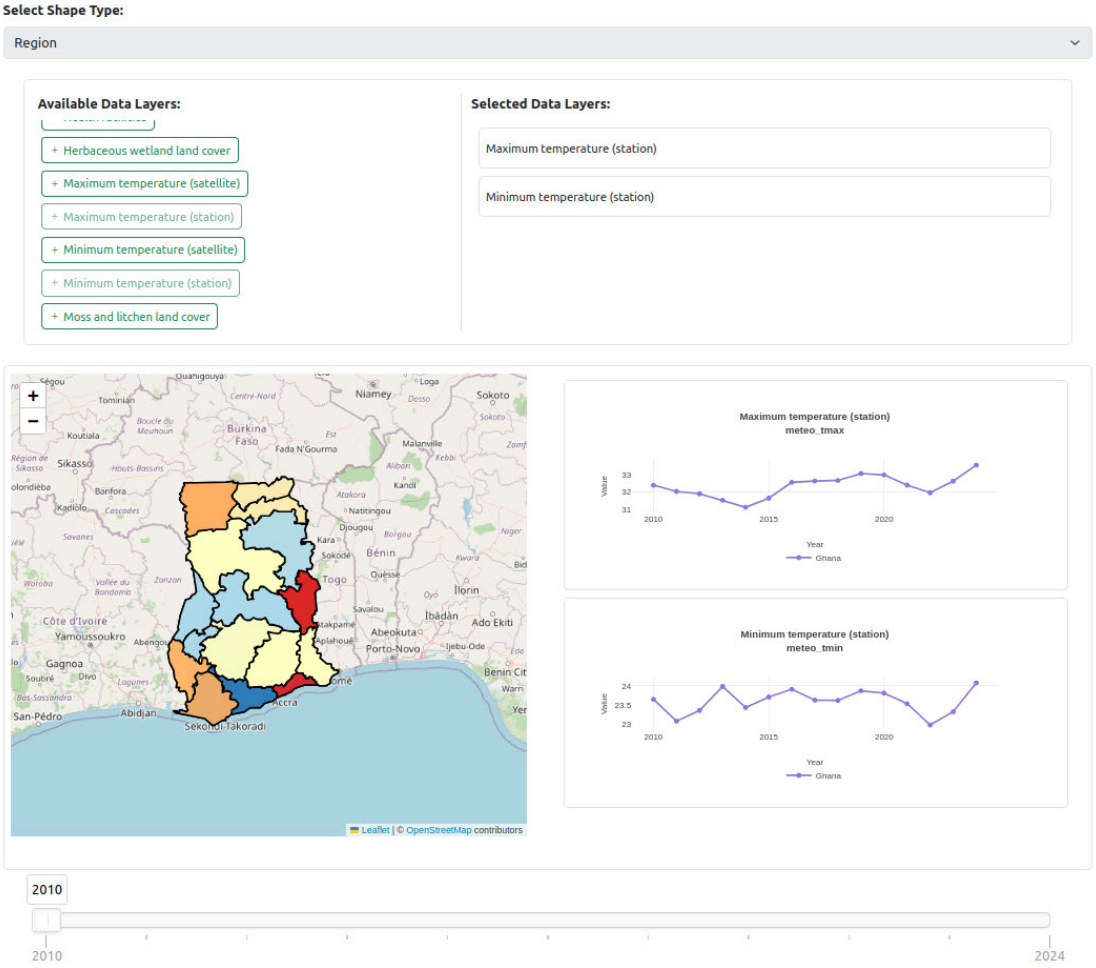


Figure 5.5: Feature 2 (FR-2): Temporal Slider Map - Add a new data layer (Alternative Scenario)

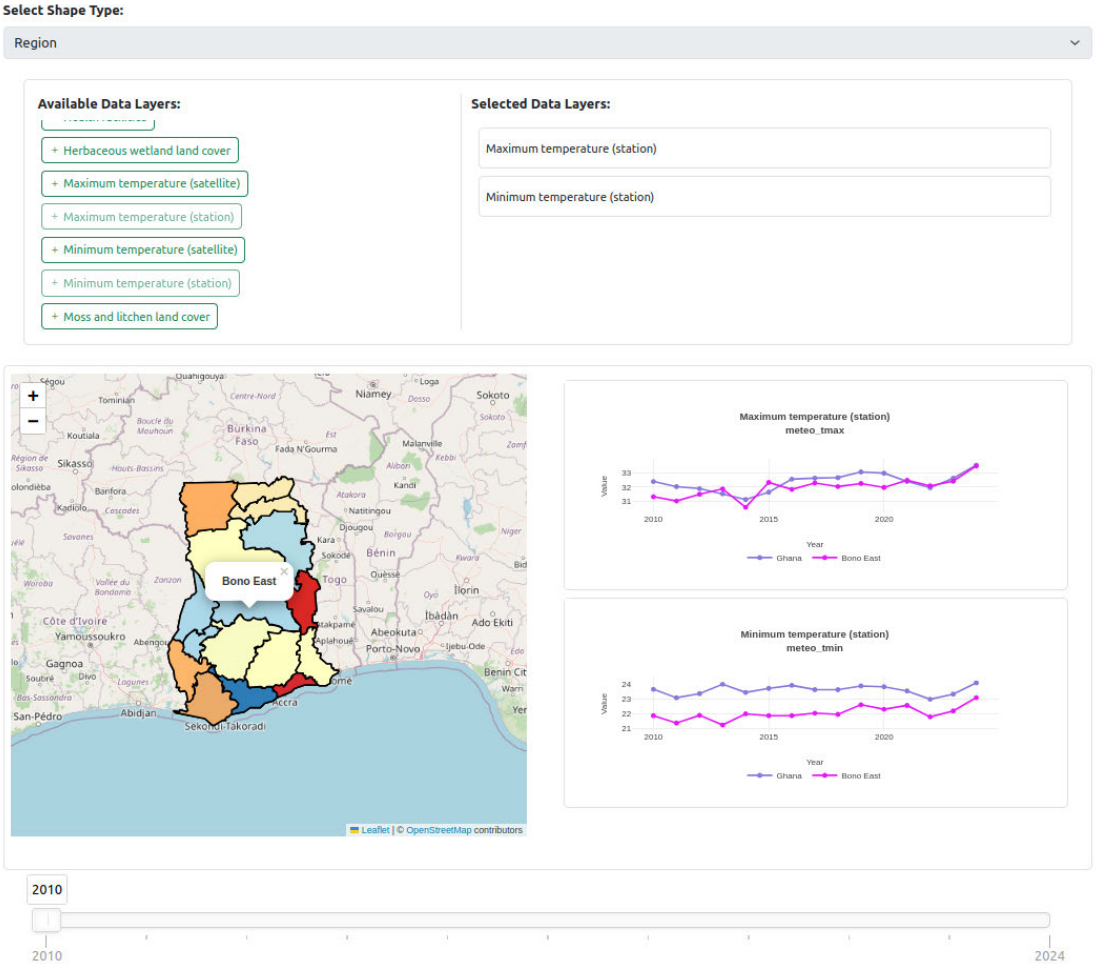


Figure 5.6: Feature 2 (FR-2): Temporal Slider Map - Add a trace to graphs (Alternative Scenario)

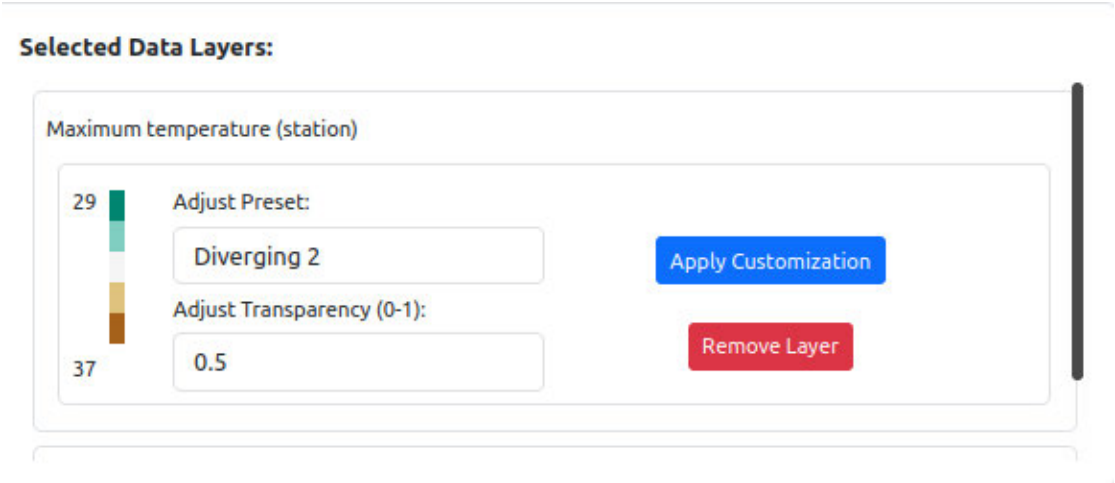


Figure 5.7: Feature 2 (FR-2): Temporal Slider Map - Choropleth Map Layer Color and Transparency Customization

to their specific needs. Transparency adjustments in this feature provide flexibility in visualizing overlapping data layers, making sure that important details from multiple layers remain visible without cluttering the map (Figure 5.7). For more information, refer to Section 5.5.

Furthermore, users also have the option to remove individual data layers, which automatically updates both the map and the associated graphs, keeping the visualization focused on relevant data, which was described in one of the alternative scenarios of use case FR-2.

This feature follows the main success scenario as well as the alternative scenarios highlighted in the use case FR-2: Temporal Slider Map.

5.4 Feature 3 (FR-3): Temporal Trends

Feature 3 implements the functional requirement **FR-3 Temporal Trend**, enabling users to compare the values of multiple data layers over time for a specific geographic shape, helping uncover relationships and potential patterns across datasets.

Users can select a shape type (e.g., country, region, district), choose a specific shape within the selected shape type, and add one or more data layers. They can continue adding as many data layers as they wish (Figure 5.8).

Figure 5.8: User Inputs for Feature 3 (FR-3): Temporal Trends

By clicking the "Load" button, the system retrieves the historical values for each selected data layer and generates a set of visualizations.

Specifically, the feature produces two types of graphs, as shown in Figure 5.9:

Combined normalized graph [1]: All selected data layers are plotted together using normalized values, calculated with the min-max normalization technique (scaled from 0 to 1). This normalization makes it easier to visually compare trends and detect patterns and relations, regardless of the differing value scales across datasets.

This is illustrated more clearly in Figure 5.10. Since population count and population density increase in parallel, their values follow the same trend. As a result, min-max normalization returns identical normalized values for both, causing them to overlap in the normalized graph.

Individual graphs [2]: Each selected data layer is plotted in its own graph, showing its raw value trends over time for the chosen shape. These individual graphs enable the users to perform further analysis.

Figure 5.9: System Outputs for Feature 3 (FR-3): Temporal Trends - [1] Combined normalized graph, [2] Individual graphs

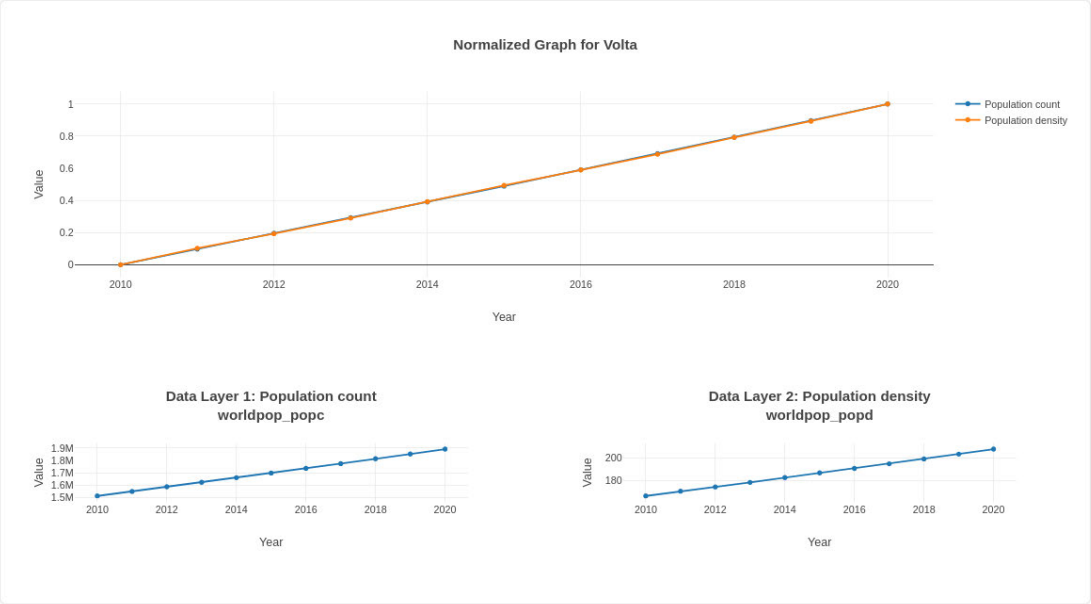


Figure 5.10: System Outputs for Feature 3 (FR-3): Temporal Trends - Normalized trends of population count and population density

This feature follows the main success scenario outlined in the use case FR-3: Temporal Trends.

If a selected data layer is missing for the specified shape or time period, the system displays an error message, informing that the data cannot be found in the database (Figure 5.11). This covers the exception scenario in the use case FR-3.

5.5 Additional Feature (FR-11): Choropleth Map Colors and Transparency

This additional feature implements the secondary functional requirement **FR-11: Choropleth Map Colors and Transparency**, allowing users to customize the choropleth map colors and layer transparency in Feature 1 (FR-1) and Feature 2 (FR-2).

The ability to customize choropleth map colors and layer transparency improve both usability and visual clarity. Custom color schemes allow users to improve contrast as well as highlight specific data ranges. Transparency settings enable the visualization of

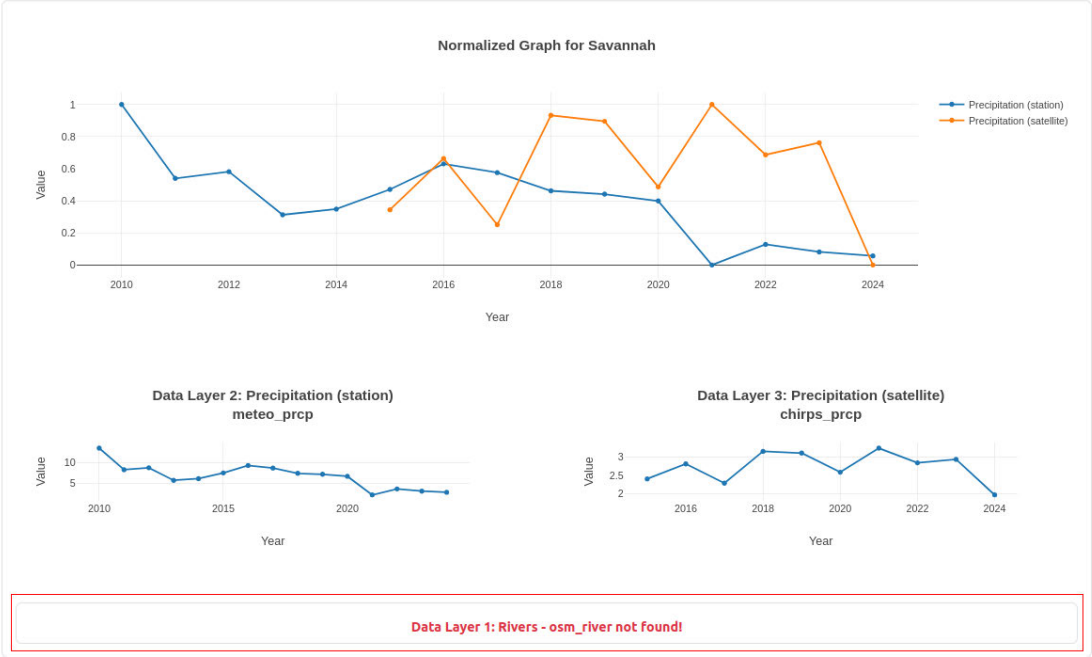


Figure 5.11: Exception Scenario for Feature 3 (FR-3): Temporal Trends

overlapping layers, making the map easier to interpret. This customization ensures the map remains clear, uncluttered and effective for different user preferences.

The dashboard offers five preset color palettes, as shown in Figure 5.12.

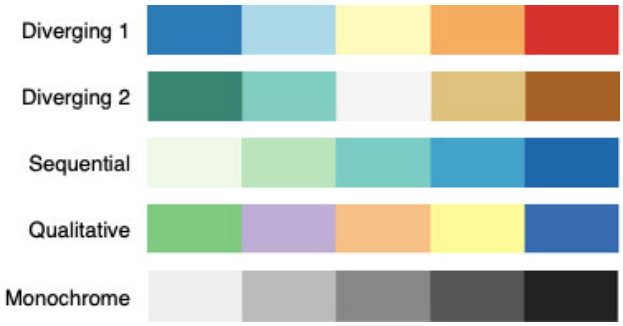


Figure 5.12: Five preset color palettes available for choropleth map customization

The color palette types are based on a work-in-progress book by Nowosad and Tennekes:

Figure 5.13: FR-11: Choropleth Map Colors and Transparency — (1) Transparency set to 0.1, (2) Transparency set to 0.5, (3) Transparency set to 1

Sequential palette type is used to display continuous data where order matters, with colors transitioning from low to high values, often through changes in lightness - commonly used for population density, GDP, etc.

Diverging palette is used for continuous data that centers around a neutral midpoint, with colors spreading toward two contrasting extremes - commonly used for temperatures, income variations, etc.

Qualitative palette is used to represent distinct categories or groups, where each color is equally important and the order doesn't matter - commonly used for regions, land cover, etc.

The transparency of the choropleth map layers can be adjusted within a range from 0 to 1 (Figure 5.13), with a default value of 0.9.

6 Evaluation

This chapter presents an evaluation of the dashboard implementation. To achieve this, a survey was created using online forms and distributed to the main contributors on the Data Hub to collect the necessary feedback.

The survey aimed to gather feedback on various aspects of the dashboard, including usability, performance, functionality, the usefulness of its features, and whether the features fulfilled the intended functional requirements stated in Section 3.1.1. The participants were asked to assess the dashboard's features, usability and performance. Additionally, open-ended questions were included to collect suggestions for potential improvements.

The responses provided valuable insights into the strengths and limitations of the dashboard, allowing for future updates to make it easier to use for researchers.

6.1 Participants

The participants in this survey were the two main contributors working on the Data Hub project. Both participants have extensive experience with the datasets as well as dashboards. Their feedback was especially valuable, as they are the primary users and maintainers of the Data Hub.

6.2 Questions

6.2.1 General Questions

This section aims to paint a picture of the participant's experience with the dashboard and datasets to to contextualize their feedback on the system's overall usability, performance, and features.

1. How often do you work with datasets?
2. How often do you work with dashboards?
3. How familiar are you with dataset management and comparison tools?

6.2.2 Functionalities

This section evaluates the three main features of the dashboard that are based on the functional requirements gathered as well as the use cases presented in Chapter 3. Some of the questions might also refer to the defined non-functional requirements, such as NFR-2: Usability.

Feature 1 (FR-1): Data Coverage Map

1. How easy was it to determine which data layers were available or missing for a given shape and year?
2. How easy was it to interpret the map? *For example, did you clearly understand what the colors (or color intensities) represent?*
3. Did the ranking prove any helpful insights?
4. Did the table provide a helpful overview of data layer availability?
5. Is this feature useful for managing datasets?
6. On a scale of 1-10, how was this feature overall? *(Performance, Usability, Functionality, etc.)*
7. Any remark or comment about this feature?

Feature 2 (FR-2): Temporal Slider Map

1. How easy was it to add or overlay a data layer on the map?
2. How easy was it to interpret the map with the overlapping layers?
3. Did adjusting the transparency improve map readability?
4. Did this feature help you with visualizing how the data changes overtime?
5. Did the ability to change map layer colors help with map readability?
6. Did the individual graphs on the right side assist with data analysis?
7. Did you find this feature helpful for data analysis?
8. On a scale of 1-10, how was this feature overall? *(Performance, Usability, Functionality, etc.)*
9. Any remark or comment about this feature?

Feature 3 (FR-3): Temporal Trends

1. How easy was it to choose a shape and add the data layers?
2. Did this feature help you with identifying patterns between the datasets?
3. Did the normalized values in the graph help reveal relationships between datasets?
4. Did the individual raw data graphs assist with data analysis?
5. On a scale of 1-10, how was this feature overall? *(Performance, Usability, Functionality, etc.)*
6. Any remark or comment about this feature?

6.2.3 Performance

This section aims to assess the dashboard's responsiveness and overall performance (refer to NFR-1: Performance)

1. How would you rate the dashboard's speed and responsiveness?

2. Did you experience any delays or lag while exploring the dashboard?
3. How satisfied are you with the overall performance of the dashboard?

6.2.4 Usability

This section assess the dashboard's navigation and layout (refer to NFR-2: Usability).

1. How easy was it to navigate and use the dashboard?
2. Did you find the dashboard's layout and design intuitive?

6.2.5 Overall Satisfaction and Suggestions

This section focuses on overall satisfaction and gathers suggestions for potential improvements.

1. How is your overall impression of the dashboard?
2. On a scale of 1-10, how satisfied are you with the dashboard?
3. What features or functionalities of the dashboard did you find most helpful or appealing?
4. Were there any aspects of the dashboard that you found difficult to use or think could be improved?
5. Which features/changes do you think could be added in the future to improve the dashboard?
6. Any additional comments or remarks?

6.3 Results

6.3.1 General Questions

One respondent works with datasets on a daily basis, while the other does so weekly. Both of them interact with dashboards on a weekly basis and are somewhat familiar with

dataset management and comparison tools. This result indicates that both respondents are experienced with both dataset management/comparison tools and dashboards.

6.3.2 Functionalities

Feature 1 (FR-1): Data Coverage Map

The evaluation of Feature 1, aligned with the functional requirement **FR-1** and its main success scenario, was generally positive, with a few areas identified for improvement.

- Both participants found it somewhat easy to determine which data layers were available or missing for a given shape and year.
- One participant rated the map interpretation (colors and intensities) as extremely easy, while the other found it somewhat easy, suggesting the visual design is mostly effective but could benefit from slight clarification.
- Regarding the usefulness of the ranking, feedback was mixed: one participant answered "yes," while the other said "maybe," indicating that the ranking might not be fully intuitive to all users.
- Both participants found the binary matrix (table) helpful as an overview of data layer availability.
- One participant found the feature extremely useful for managing datasets, while the other rated it as somewhat useful, highlighting its strong use for dataset management.
- Participants rated this feature 7 and 8 out of 10, giving an average rating of **7.5/10**.

Strengths:

- Highly useful visualizations.
- Customizability (map colors and transparency) of the tool.
- Clear and informative data presentation through tables and maps.

Weaknesses / Suggestions:

- Linking shapes directly to their detail pages could be useful.

- Adding brief descriptions for individual feature elements could enhance usability.
- Including an overview focusing on the temporal data coverage would be useful.
- Introducing a system that recommends the optimal data density (for both time and space) was suggested to enhance the user experience.

Overall, participants considered **Feature 1 (FR-1): Data Coverage Map** a valuable feature that effectively supports the visualization and analysis of data availability, fulfilling its functional requirements, with room for usability improvement.

Feature 2 (FR-2): Temporal Slider Map

The evaluation of Feature 2, aligned with the functional requirement **FR-2** and its main and alternative scenarios, was more mixed compared to the previous feature, with notable areas for improvement.

- One participant found it extremely easy to add or overlay a data layer into the map, while the other rated it as somewhat easy.
- Interpreting the map with overlapping layers was rated somewhat uneasy by one participant and neutral by the other, suggesting that map readability could be improved when multiple layers are combined.
- Adjusting the transparency slightly improved map readability for both participants.
- The ability to change map layer colors significantly improved map readability for both participants.
- The individual graphs on the right side were considered extremely useful by one participant and somewhat useful by the other for assisting with data analysis.
- In terms of overall helpfulness for data analysis, one participant found the feature somewhat useful, while the other was neutral.
- Both participants gave an overall rating of 6 out of 10, resulting in an average rating of **6/10**.

Strengths:

- Helpful for visualizing the temporal development of data layers.

- The ability to display different shapes individually within the timeline was appreciated.
- Changing layer colors significantly enhanced readability.
- Individual graphs provided valuable assistance for data analysis.

Weaknesses / Suggestions:

- The spatial representation on the map was sometimes misleading and provided less additional insight.
- A general overview of the temporal availability of data layers would be beneficial.
- The form cannot be reset without refreshing the page.
- After forgetting to select a shape type, the popup correctly appears, but the dropdown becomes disabled, preventing selection.

Overall, this feature was found to be somewhat helpful in visualizing changes over time, although it is apparent that overlaying data layers on top of each other is not be the most ideal solution for visualizing multiple data layers at the same time. Improvements in map readability, form handling, and temporal data overview would significantly enhance the user experience.

Feature 3 (FR-3): Temporal Trends

The evaluation of Feature 3, aligned with the functional requirement **FR-3** and its main success and exception scenario, was very positive, with participants highlighting its usefulness while also suggesting some improvements for further clarity and advanced analysis.

- Choosing a shape and adding data layers was rated as extremely easy by both participants.
- Identifying patterns between datasets using this feature was found to be extremely helpful.
- The normalized values in the graph were considered somewhat helpful for revealing relationships between datasets.

- The individual raw data graphs were rated as extremely helpful by one participant and somewhat helpful by the other.
- Participants rated this feature 7 and 9 out of 10, resulting in an average rating of **8/10**.

Strengths:

- Extremely helpful for visually exploring data trends over time.
- Easy and intuitive to select shapes and add data layers.
- Strong support for analyzing raw data through individual graphs.

Weaknesses / Suggestions:

- Additional guidance could clarify that normalized values indicate potential associations, not direct relationships.
- When comparing multiple data layers with the same units, normalization could be adjusted to consider mutual maxima and minima for better comparability.
- Integrating an AI assistant to suggest potential association patterns between datasets could enhance the feature's analytical capabilities.

Overall, the feature was very well received, offering strong usability and valuable support for data exploration, with opportunities for advanced improvements in data interpretation and analysis assistance.

6.3.3 Usability

Both participants found the dashboard generally easy to navigate, with one rating it as somewhat easy and the other as extremely easy. The layout and design were considered somewhat intuitive by both participants. Since both users work with dashboards regularly, it is possible that their familiarity made the navigation easier for them. New users with no prior dashboard experience might find it less intuitive at first.

6.3.4 Performance

The participants rated the dashboard's speed and responsiveness very highly, giving it an average rating of **9.5/10**. Both reported that they experienced no delays or lag while exploring the dashboard, describing the interaction as smooth throughout. Overall satisfaction with the dashboard's performance was also positive, with an average rating of **8.5/10**, indicating a strong performance with minor room for further optimization.

6.3.5 Overall Satisfaction and Suggestions

- The overall impression of the dashboard was quite positive, with the average rating of **7.5/10**.
- Satisfaction with the dashboard was also rated **7.5/10**, indicating a good general experience with some room for further improvements.
- The most helpful or appealing features were:
  - Feature 1 (FR-1): Data Coverage Map - with 2 votes
  - Feature 3 (FR-3): Temporal Trends - with 1 vote
- Areas identified for improvement included:
  - Removing data layers and changing form states could be made more intuitive.
  - Feature 2 (Temporal Slider) could be refined to better support temporal data comparisons.
  - Adding short descriptions for visual outputs would help guide users in interpretation.
- Suggested future improvements:
  - Including brief explanatory descriptions alongside visual outputs.
  - Enabling crosslinking between different parts of the dashboard and detail pages on the Data Hub (e.g., linking to shape or data layer detail pages).
- Additional remarks:

– One participant commented that the dashboard was an inspiring tool for exploring differences in data.

7 Discussion and Conclusion

This chapter presents a brief discussion of the evaluation results followed by the conclusion of the thesis.

7.1 Discussion

The user evaluation revealed that the dashboard generally fulfilled its requirements highlighted in Chapter 3.

Feature 3 (Temporal Trends) received the highest average rating of 8/10. Participants found it useful for identifying patterns between datasets. They also found both raw and normalized graphs helpful in supporting further data analysis.

Feature 1 (Data Coverage Map) got an average rating of 7.5/10, helping users visualize data availability over space and time through customizable maps and clear table overview.

Feature 2 (Temporal Slider Map) received the lowest rating of 6/10. While it contributed to visualizing time-based patterns, participants found it hard to interpret overlapping data layers and track changes over time. These issues show that better ways to layer or filter the data on the map are needed. Also, problems with the form made the dashboard harder to use.

Across all features, participants consistently found it helpful to change map colors and transparency, showing that visual control is important in complex dashboards.

Although the evaluation provided valuable insights, some limitations should be noted. Only two participants took part in the survey. This small number limits how well the results can be applied to a broader user group. The participants also regularly work with dashboards and datasets. Their experience may have led to more positive usability ratings, while less experienced users might find the dashboard more challenging.

7.2 Conclusion

The objective of this thesis was to create a dashboard for a Geographic Information System (GIS) focused on epidemiological data, called the Data Hub. The goal of the dashboard was to assist users of the Data Hub in managing the vast number of datasets within the Data Hub and to provide tools for visualizing the data. This would help users analyze the information and potentially uncover relationships, trends, and patterns among the datasets.

To implement the dashboard, three primary and one secondary functional requirements (FRs), along with two non-functional requirements (NFRs), were identified. These requirements were collected from the two main contributors of the Data Hub.

The functional requirements were realized by implementing three different key features:

- **Feature 1 (FR-1): Data Coverage Map**
  This feature provides a visual representation of the Data Hub database, allowing users to view data coverage and density across available data layers, based on the geographic shapes stored in the Data Hub.
- **Feature 2 (FR-2): Temporal Slider Map**
  This feature enables users to explore how values in selected data layers change over time, displayed across geographic regions within the Data Hub.
- **Feature 3 (FR-3): Temporal Trends**
  This feature allows users to compare values from multiple data layers over time for a specific geographic area, aiding in the identification of relationships and potential patterns between datasets.
- **Additional Feature 11 (FR-11): Choropleth Map Colors and Transparency**
  This feature allows users to customize the choropleth map colors and layer transparency in Feature 1 (FR-1) and Feature 2 (FR-2).

The implementation process relied heavily on Django as the backend framework and utilized various frontend frameworks and libraries, such as Plotly for trend graphs, Leaflet with OpenStreetMap integration for choropleth maps, and Bootstrap for styling, etc.

The evaluation highlighted both strengths and limitations in the dashboard implementation. Features 1 and 3 were found to be useful in assisting users with managing and

visualizing datasets. In contrast, Feature 2 was less valuable, indicating that overlapping data layers may be not so intuitive and difficult to interpret. However, the additional feature to adjust map color and transparency was considered valuable, as it improved the readability of the map.

Future improvements should focus on addressing the weaknesses identified during evaluation (see Chapter 6) and implementing user feedback. Adding brief explanations or tooltips for features, inputs, and outputs would also enhance usability, especially for new users.

In conclusion, the implemented dashboard significantly improves the management and visualization of datasets within the Data Hub. It simplifies the process of analyzing multiple data layers concurrently, therefore helping researchers in identifying patterns and trends in epidemiological contexts. Nevertheless, further enhancements based on the evaluation will be essential for improving the dashboard's usability.

Bibliography

Adewuyi, A., Adeoye, V., Okon, D. A., Faderin, E., Idowu, O. A. and Akindahunsi, T. (2022), 'Data visualization in diseases epidemiology', *GSC Advanced Research and Reviews* **9**(3), 151–163.
https://doi.org/10.30574/gscarr.2021.9.3.0231

Anchlia, A. (2024), 'Enhancing query performance through relational database indexing', *International Journal of Computer Trends and Technology* **72**(8), 130–133.
https://doi.org/10.14445/22312803/IJCTT-V72I8P119

Chang, K.-T. (2019), *Introduction to geographic information systems*, ninth edition. edn, McGraw-Hill Education, New York.

Chen, C., Han, Y., Galinski, A., Calle, C., Carney, J., Ye, X. and Van Westen, C. (2025), 'Integrating urban digital twin with cloud-based geospatial dashboard for coastal resilience planning: A case study in florida', *Journal of Planning Education and Research* .
https://doi.org/10.1177/0739456X251316185

Chui, K. K. H., Wenger, J. B., Cohen, S. A. and Naumova, E. N. (2011), 'Visual analytics for epidemiologists: Understanding the interactions between age, time, and disease with multi-panel graphs', *PLoS ONE* **6**(2), e14683.
https://doi.org/10.1371/journal.pone.0014683

Curriero, F. C., Wychgram, C., Rebman, A. W., Corrigan, A. E., Kvit, A., Shields, T. and Aucott, J. N. (2021), 'The lyme and tickborne disease dashboard: A map-based resource to promote public health awareness and research collaboration', *PLOS ONE* **16**(12), e0260122.
https://doi.org/10.1371/journal.pone.0260122

Data Snack (2024). Available at: https://datasnack.org/. Accessed: 2024-12-30.

Dauzon, S., Bendoraitis, A. and Ravindran, A. (2016), *Django: web development with python: from an idea to a prototype–a complete guide for web development with the Django framework*, Packt Publishing.

De Ruvo, A. (2024), 'Spread: Spatiotemporal pathogen relationships and epidemiological analysis dashboard', *Veterinaria Italiana* p. 1.
https://doi.org/10.12834/VetIt.3476.23846.1

Del Pilar, C. G. E., Nale, S. G. R., Jinio, A. L., Abando, D. S., Blanco, M. C. R. and Sison, A. A. R. C. (2024), B-gis: Modernizing local governance through a web-based barangay geographic information system for barangay 99, tondo, manila, *in* '2024 International Conference on Intelligent Cybernetics Technology Applications (ICICyTA)', pp. 161–166.
https://doi.org/10.1109/ICICYTA64807.2024.10912845

Django Project (2025), 'Writing and running tests | django documentation'. Accessed: 2025-03-06.

Farmanbar, M. and Rong, C. (2020), 'Triangulum city dashboard: An interactive data analytic platform for visualizing smart city performance', *Processes* **8**(2), 250.
https://doi.org/10.3390/pr8020250

Few, S. (2006), *Information dashboard design: the effective visual communication of data*, O'Reilly Associates, Sebastopol, CA.

Few, S. (2009), 'Introduction to geographical data visualization', *Visual Business Intelligence Newsletter* **2**.

Forcier, J. (2009), *Python Web development with Django*, Developer's library, Addison-Wesley, Upper Saddle River, N.J.

Gao, S., Rao, J., Kang, Y., Liang, Y. and Kruse, J. (2020), 'Mapping county-level mobility pattern changes in the united states in response to covid-19', *SIGSPATIAL Special* **12**(1), 16–26.
10.1145/3404820.3404824

Gordis, L. (2014), *Epidemiology*, 5. ed edn, Elsevier Saunders, Philadelphia.

Gritsenko, Y. B., Senchenko, P. V. and Kalentiev, K. A. (2020), Development of a geographic information system for managing university campus, *in* '2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)', pp. 1–6.
https://doi.org/10.1109/AICT50176.2020.9368698

Han, J., Kamber, M. and Pei, J. (2012), *Data mining: concepts and techniques*, Morgan Kaufmann series in data management systems, 3rd ed edn, Elsevier/Morgan Kaufmann, Amsterdam Boston.

ISO (2015), *ISO 19109:2015 Geographic information — Rules for application schema*, Standard, International Organization for Standardization.

Longley, P. A. (2008), *Geographical information systems and science*, 2. ed., reprinted edn, Wiley, Chichester.

Lopes, G. R., Delbem, A. C. B., Da Silva, R. F., Júnior, C. B., De Mattos, S. H. V. L., Scatolini, D., Ghiglieno, F. and Saraiva, A. M. (2022), Multimaps: a tool for decision-making support in the analyzes of multiple epidemics, *in* 'Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology', ACM, Seattle Washington, p. 22–25.
https://doi.org/10.1145/3557995.3566119

Melé, A. (2024), *Django 5 By Example: Build powerful and reliable Python web applications from scratch*, 1 edn, Packt Publishing Limited, Birmingham.

Myers, G. J. (2012), *The art of software testing*, 3rd ed edn, J. Wiley Sons, Hoboken, N.J.

Nowosad, M. and Tennekes, J. (2021), *Chapter 6 Visual variables | Elegant and informative maps with tmap*. Accessed: 2025-04-10. Available at: https://r-tmap.github.io/tmap-book/

Plotly Technologies Inc. (2024), 'Choropleth maps in python'. Available at: https://plotly.com/python/choropleth-maps/. Accessed: 2024-12-30.

Poletaikin, A. N., Monastyrskaya, T. I., Skvortsova, E. B. and Englezi, I. P. (2024), Geographic information system of recording and analyzing geographic data on traffic

control facilities, *in* '2024 IEEE 3rd International Conference on Problems of Informatics, Electronics and Radio Engineering (PIERE)', pp. 590–594.
https://doi.org/10.1109/PIERE62470.2024.10804974

Srivastava, D. (2023), 'An introduction to data visualization tools and techniques in various domains', *International Journal of Computer Trends and Technology* **71**(4), 125–130.
https://doi.org/10.14445/22312803/IJCTT-V71I4P116

Wiegers, K. E., Beatty, J. and Wiegers, K. E. (2013), *Software requirements*, Best practices, 3. ed. [fully updated and expanded] edn, Microsoft Press, Redmond, Wash.

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort | Datum | Unterschrift im Original