Visualising Maternal Health: Analysing Maternity Services Through Interactive Data Visualisation

**Miles Simmonds 100571823**

Contents

[Introduction 4](#_Toc197611268)

[Literature Review 5](#_Toc197611269)

[Maternal Healthcare Overview 5](#_Toc197611270)

[The Role of Data Visualisation in Healthcare 6](#_Toc197611271)

[Visualisation Techniques in Healthcare Data Analytics 7](#_Toc197611272)

[Challenges in Healthcare Data Visualisation 8](#_Toc197611273)

[Research Gaps and Justification for Study 9](#_Toc197611274)

[Methodology 10](#_Toc197611275)

[Goals/Tasks of the Dashboard 10](#_Toc197611276)

[1.Temporal Analysis of Maternal Health Trends 10](#_Toc197611277)

[2. Geographic Disparities and Regional Comparisons 10](#_Toc197611278)

[3. Demographic and Antenatal Care Analysis 11](#_Toc197611279)

[4. Communication and Reporting 11](#_Toc197611280)

[Data Collection 11](#_Toc197611281)

[Data Preprocessing and Cleaning 11](#_Toc197611282)

[System Architecture 12](#_Toc197611283)

[Dashboard Design 14](#_Toc197611284)

[Implementation 15](#_Toc197611285)

[Front-End Development (React) 15](#_Toc197611286)

[Navigation 15](#_Toc197611287)

[Visualisation Components 15](#_Toc197611288)

[Chart Visibility Control 20](#_Toc197611289)

[Dashboard Logic 21](#_Toc197611290)

[User Interactivity: Grouping and Time Filters 22](#_Toc197611291)

[Chart visibility control 25](#_Toc197611292)

[Back-End Development (Flask) 27](#_Toc197611293)

[Database Implementation (MySQL) 29](#_Toc197611294)

[Deployment Strategy 30](#_Toc197611295)

[Version Control and Collaboration 32](#_Toc197611296)

[Evaluation 33](#_Toc197611297)

[Questionnaire Design 33](#_Toc197611298)

[Quantitative Results 34](#_Toc197611299)

[Qualitative Analysis 37](#_Toc197611300)

[Strengths 37](#_Toc197611301)

[Weaknesses 38](#_Toc197611302)

[Effectiveness of Interactivity vs Static Visualisation 39](#_Toc197611303)

[Alignment with Task Design 39](#_Toc197611304)

[Iterative Development and Response to Feedback 40](#_Toc197611305)

[Conclusion 41](#_Toc197611306)

[References 42](#_Toc197611307)

[Table of Figures 44](#_Toc197611308)

# Introduction

Over the last 20 years, increased digitisation of record keeping has allowed for an unprecedented volume of patient data, ready to unlock valuable insights into maternal health outcomes. There are extensive datasets regarding maternity, including details of maternal mortality rates, prenatal care accessibility, and postnatal complications, but the challenge lies with interpreting and visualising the data accurately. Traditional methods of data presentation often use static reports, failing to provide easy to read, actionable insights which limits their usefulness. To address this challenge, data visualisation has emerged as a powerful tool that has the potential to take the raw healthcare data and transform it into interpretable, intuitive and interactive formats.

Healthcare visualisations play a crucial role in the identification of any trends, disparities or general areas of potential intervention. By leveraging these visualisations with web-based tools, the data can be represented in a more interactive manner, being more publicly accessible than the data visualisations found in static reports. Interactive dashboards allow users to explore data trends across various datapoints and manipulate them in real time to further understand the dataset(s).

Despite efforts to improve maternal healthcare, there are still disparities that persist across various different geographic regions, socioeconomic groups and ethnic backgrounds. While large datasets are available their wide array of datapoints can become too complex for the average person to interpret, hindering the ability to identify and high risk areas, assess policy effectiveness and allocate resources more efficiently.

A major challenge is that static reports and spreadsheets are usually not well suited to identifying trends and correlations, especially in healthcare data. The primary aim of this dissertation is to create interactive visualisations that display maternal health trend outcomes across the UK. By creating these interactive visualisations, we aim to identify key patterns across various regions and demographics using a publicly available dataset as the basis for the analysis, Evaluating the effectiveness of those interactions in improving data comprehension in comparison to more traditional static reports. Culminating in an exploration into how policy makers and healthcare professionals can use these analytics, to better enhance their decision making in maternal health services.

This dissertation aims to contribute to the massive field of maternal healthcare whilst also demonstrating how interactive visualisation tools can provide a deeper insight into maternal health trends as well as offering a practical web application for the Maternity services, hoping to aid their identification of and shortfalls in their policies and workflow to improve patient outcomes with a data-driven approach.

In an era where data driven decision making is becoming increasingly vital and widespread, providing intuitive and interactive data visualisations offer greater transparency, accessibility and usability for healthcare professionals and policymakers, enhancing maternal healthcare insights which can ultimately contribute to better decision making and patient outcomes by demonstrating modern data visualisation techniques.

# Literature Review

Healthcare systems are becoming increasingly more data rich, due to the more widespread adoption of electronic health records (EHR’s)*.* This has meant that effective data visualisation has become a critical tool to gain insight into complex health information. Visual analytics, described as the combination of computational analysis and interactive visual interfaces, have been identified as the key to enabling “data driven learning health systems” (Rhyne, Gotz and Borland, 2016) allowing for the development of more advanced visualisation techniques and technologies. Transforming the raw medical data into actionable insights, improving clinical decision-making and population health management *(Gotz and Borland, 2016).* Healthcare data visualisation has a long history of producing actionable insights that improve the health of patients, a prime example in being a cholera map from 1854, illustrating how accurate visualisation can reveal patterns unseen in textual analysis. (Benbba, 2021)



Figure John Snow cholera map from 1854 (Wikipedia Contributors, 2025)

Given the high stakes regarding healthcare, clarity and efficiency in data communication are essential. Data visualisation leverages on human’s pattern recognition ability, which enables faster and more informed decision making. This highlights trends, outliers and any relationships in health data that can be more obscure within traditional statistic-based reporting. (Hehman and Xie, 2021). This literature review examines the use of data visualisation in healthcare, emphasizing it’s use in maternal healthcare. Maternal health encompasses pregnancy, childbirth and the postnatal period, a critical area due to its direct impact to morality. In 2020, 800 women a day died from pregnant related, preventable causes – underlining the need to for better data tools to improve maternal outcomes (World Health Organization, 2023).

### Maternal Healthcare Overview

Global maternity morality remains a critical concern, nearly 287,000 women died from pregnancy related causes in 2020 alone, and though this figure is over a 30% reduction from the year 2000, it still unacceptably high. Maternal deaths are concentrated in situations where resources are more limited, with most maternal deaths being preventable with timely adequate care. Access to skilled health professionals and their insight is imperative during pregnancy and childbirth, with the World Health Organisation noting that proper care before during and after birth can save the lives of both women and their newborns. There have been global initiatives like the Sustainable Development Goals help highlight the importance of maternal health, with their targets to reduce global MMR below 70 per 100,000 by 2030, reflecting international commitment to improving maternal health outcomes (World Health Organization, 2023).

Maternal healthcare is provided through a series of essential stages, antenatal care - regular checkups during pregnancy, intrapartum care – the skilled attendance and emergency support during labour and delivery, and postnatal care – follow ups in the weeks after birth. Consistent access to these services are linked to improved health outcomes. To assess the effectiveness of maternal healthcare, health systems are used to monitor and identify key indicators affecting maternal health outcomes.

Even in high-income countries such as the UK, disparities exist. By analysing maternal health indicators stakeholders can identify problem areas (e.g. elevated morality rates based by region and implement targeted improvements. It is here where robust data analysis and visualisation become crucial, as it enables the transformation of raw maternal health data into accessible, actionable insights for policymakers and healthcare providers.

### The Role of Data Visualisation in Healthcare

Data visualisation plays a vital role in making complex healthcare datasets understandable and actionable. Modern healthcare generates enormous volumes of data from electronic health records to population health statistics. Advances visualisation techniques are essential for transforming this raw data into meaningful, actionable information. Visual analytics are defined as “the science of analytical reasoning facilitated by visual interfaces” (Chishtie et al., 2022) combining interactive visualisations with data analysis algorithms to support human insight. In healthcare, this means leveraging computers to process large datasets whilst using visual interfaces to present the results in intuitive ways that clinicians, researchers and policy makers can interpret. Effective visualisation bridges the gap between large data and decision-making, it was noted in 2016 that despite the scale and complexity of health data, visualisation tools have the potential to convert data into “actionable insights” that improve patient care, as well as enabling better population health management and advance medical research (Rhyne, Gotz and Borland, 2016).

A major advantage of visualisation is the leverage of human pattern recognition ability. The human eye and brain excel at spotting trends and anomalies when data is presented visually. In healthcare, important signals can be missed if data remains in text or spreadsheets. Plotted data often reveals what raw statistics might hide, this is illustrated by Anscombe’s quartet, in which four datasets with identical summary statistics, ended up looking very different when graphed.

A group of graphs with dots

AI-generated content may be incorrect.In practice, well designed charts and dashboards can instantly highlight a spike in maternal complications or a decline in antenatal care coverage, prompting investigation and eventually action. By contrast, the issues found in charts and dashboards may not be found until much later (or not at all) if relying solely on static tables and dense reports. Visualisation can also encode more information in a single view than traditional formats, For instance, timeline displays/line graphs displaying a patients maternity history (showing blood pressure readings, clinic visits, interventions over time etc.) allows doctors to grasp the trajectory of care at a glance, something that would otherwise require the reading of pages notes to detect the same issues without the visual aid (Rhyne, Gotz and Borland, 2016).

(Fitzmaurice, 2017)

Figure Anscombe's Quartet (Fitzmaurice, 2017)

Interactive visualisation expands this capability further by allowing users to engage with data dynamically. Traditional static graphs usually show relationships between two variables, but many health outcomes are multi-factorial, and the visualisations should reflect this. Interactive tools enable analysts to filter, zoom and drill down into data, examining multiple variables or subgroups simultaneously. For example, an Antenatal Care professional could use an interactive platform to adjust filters for age, region or socioeconomic status and immediately see how each factor influences health outcomes via updated charts/maps. This interactivity supports sense-making, which users can test hypotheses on the fly and discover the complex associations to outcomes that would be hard to discern otherwise (Chishtie et al., 2022). This ability for exploratory analysis is a crucial aspect of healthcare research and quality improvement. By engaging directly with the visualised data, healthcare professionals can observe trends and test hypothesis leading to new discoveries or more informed decisions.

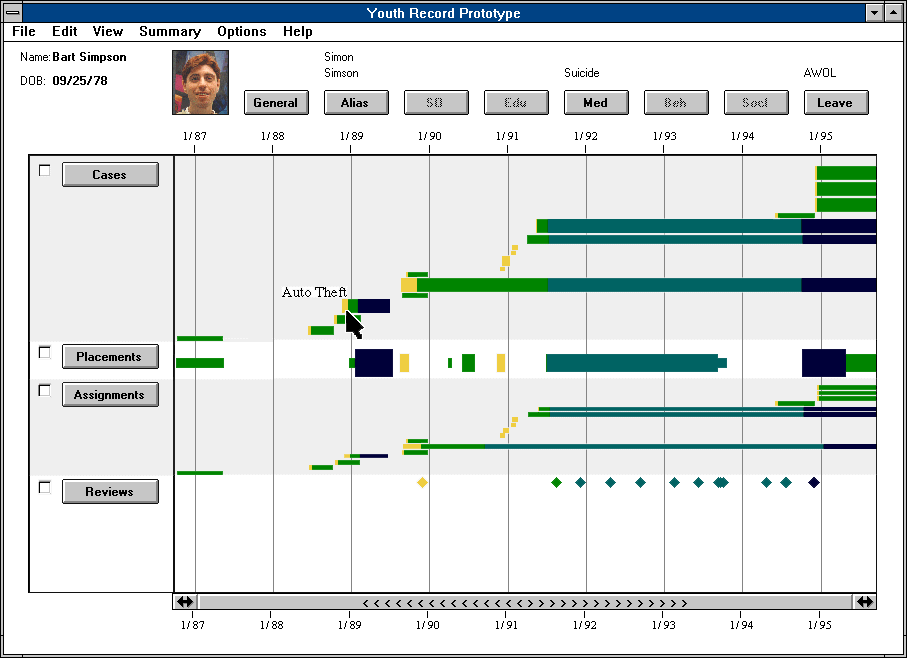


Figure Screenshot of a lifelines dashboard (Plaisant et al. (1996)

Furthermore, well designed visualisations can enhance communication and collaborative decision making. In clinical settings, complex patient information is best displayed visually. Research prototypes like lifelines and similar tools have demonstrated how timeline displays enable quicker and more accurate interpretations of longitudinal data (West, Borland and Hammond, 2014). At the public health level, visual dashboards are used to present indicators in decision-makers in an accessible format. Health departments employ dashboards to monitor important metrics like immunisation rates, maternal morality and disease outbreaks in real time, using simple visual ques such as colour coding in order to direct attention as most needed, condensing complex data into an at-a-glance format. Visualisation supports better awareness and quicker responses.

Importantly, data visualisation in healthcare is not just about creating charts but about integrating into the decision workflow. A meaningful interactive visualisation must be tailored to the needs of its audience, this being a surgeon reviewing patient outcomes or a policy official reviewing national health statistics. Studies do however emphasise the necessity of aligning visual tools with the users’ tasks and context to truly make an impact (Rhyne, Gotz and Borland, 2016). Other fields such as genomics and engineering have already shown that advanced visualisations improve understanding of complex data, and the healthcare sector is increasingly recognising this potential (Younas et al., 2016). In summary, interactive data visualisation serves as a powerful tool in healthcare by improving data transparency and enabling data driven improvements in patient outcomes and healthcare performance

### Visualisation Techniques in Healthcare Data Analytics

There is a variety of visualisation techniques that are employed to analyse and communicate healthcare data, here are some notable examples are:

**Time-series/timeline visualisations:**

Much health analysis requires an understanding of changes over time. Simple line and bar charts show trends (e.g. maternal mortality rates by year(s)). More specialized timeline visualisations plot events on a temporal axis. The Lifelines system pioneered interactive timelines for individual patient records, displaying a patient’s medical history in a single view (West, Borland and Hammond, 2014). Extensions such as LifeFlow aggregate multiple patient timelines to reveal common sequences and patterns into large cohorts. These time-based techniques help users detect temporal trends and outliers e.g. identifying at what point in pregnancy certain complications tend to occur (Domova and Shiva Sander-Tavallaey, 2019).

**Dashboards and multi-faceted displays:**

Dashboards integrate multiple visual components into one interface for a holistic view of performance. In healthcare, dashboards often combine several charts (line, pie, etc.) to provide an overview of a dataset at a glance. Typically, dashboards support interactivity such as filtering to tailor the view to the individual user’s needs. A maternal dashboard for example might allow a user to select a region and instantly see all indicators (mortality rates, antenatal visits, etc.) update for that selection (Callaghan-Koru et al., 2024). A single dashboard can therefore be used to show a map of maternal morality by province alongside a timeline of monthly care uptake and a bar chart of outcomes by age, all linked so selecting a particular grouping highlights the relevant data in all other views – this is the power of dashboards and multi-faceted displays.

**Geographical visualisations:**

Maps are essential for health planning, Choropleth maps display healthcare indicators across regions, such as maternal morality ratios by district or region, highlighting geographic hotspots for poor outcomes at a glance. Geographic information systems (GIS) further enable overlaying of multiple layers to explore spatial correlations. Geospatial visualisations help identify regional disparities and guid resource allocation to areas of greatest need, by highlighting said areas visually (Besançon et al., 2020).

**Hierarchical and Network visualisations:**

Some healthcare data is hierarchical – such as diagnosis classification or relational – disease co-occurrence. To analyse hierarchies, techniques like tree maps and icicle charts depict data as nested diagrams, and node link graphs illustrate networks of connections. In electronic health record research advanced layouts (such as radial diagrams) can be used to uncover patterns in complex patient datasets (West, Borland and Hammond, 2014). These methods help to reveal cluster structures as well as outlies in connections that might be missed in raw tables.

### Challenges in Healthcare Data Visualisation

**Data scale and complexity:**

Healthcare datasets are often very large and highly complex, containing a multitude of variable and longitudinal entries. Visualizing “big data” in healthcare requires techniques that can handle volume and complexity without overwhelming the user. Combining heterogeneous data sources is also a difficult task as missing or incomplete data is commonplace (Wahyuningsih, 2020).

**User and context diversity:**

Healthcare data visualisations have vastly varying users/use cases, each with different goals. A visualisation that would be deemed useful for a single patient used in a bedside clinical decision would not be suitable for a public health analyst examining population trends. Designing tools that fit into diverse workflows is challenging and the solutions but be tailored to be context specific in order to be effective (Agrawal et al., 2015).

**Data quality and standardization:**

Visual outputs can only be as reliable the data supplied to them. If the supplied data is inaccurate, inconsistent, delayed, etc. then the visualisation can mislead those viewing them from creating accurate insights. In maternal health dashboards, data accuracy and timeliness have been noted as significant limitations that hinder their effective use (Olufisayo Olakotan, Jennifer and Pillay, 2024). Ensuring high quality data inputs to visualisations are an ongoing challenge and essential to overcome to enable accurate actions/insights.

**Usability and training:**

Advanced analytics tools can be complex; however, healthcare professionals typically receive very little training regarding data visualisation, Complex interfaces increase the training time required and this has been seen to deter busy professionals (Hehman and Xie, 2021). If a data visualisation tool, interactive or not, Is not intuitive or efficient, users may misrepresent the data or ignore the visualisation entirely. Ease of use and clear design are therefore critical for their uptake.

**Validation and trust:**

In the domain of healthcare, visual analytics must be rigorously validated. Clinicians need confidence that the visualisations are viewing accurately reflects reality; emerging patterns found in these visualisations are meaningless unless they are sound and clinically meaningful Rhyne, Gotz and Borland (2016). The healthcare community has historically been cautious about embracing new visualisation techniques – therefore building trust through evidence of effectiveness is crucial for new tools to be accepted.

### Research Gaps and Justification for Study

This literature review highlights several gaps in current knowledge and practise regarding data visualisation in maternal healthcare. Whilst visual analytics is a continually emerging field, there is a shortage on research on interactive visualisation in real world healthcare settings. West et al. (2014) found that healthcare providers had not yet taken full advantage of advanced visualisation techniques to exploit the wealth of electronic data available. The effectiveness of current dashboards in improving maternal outcomes remains largely unproven, leading to new approaches and more thorough evaluations becoming needed (Olufisayo Olakotan, Jennifer and Pillay, 2024). This justifies the focus of study on designing and evaluating an interactive maternal health data visualisation dashboard.

Maternal health is a critical area in which better data tools can translate into lifesaving improvements, yet it remains under served by advanced analytics. This project will build upon insights from prior visual analytic work and apply them to maternal healthcare – responding to calls for exploring and refining new visualisation techniques in healthcare.

# Methodology

This project is being developed to provide a meaningful and interactive way of exploring maternal health data in the UK. To achieve this, an interactive dashboard has been designed around a series of real-world tasks, that if completed would allow healthcare professionals to create actionable change and gain an insight regarding the data they have been provided. By clearly defining each phase, from data acquisition to evaluation. This methodology ensures the accuracy, consistency, and effectiveness of the final visual outputs. The goal is to ensure the tasks shaped every part of the system and that each component not only displayed data effectively but also enabled users to discover insights relevant to maternal health outcomes.

The development process followed a staged workflow that involved acquiring relevant maternal health data, cleaning and preprocessing the dataset to ensure quality, designing and implementing interactive visualisations, and finally evaluating their impact and usability through feedback from users.

## Goals/Tasks of the Dashboard

The following tasks define the practical purposes behind each visualisation in the dashboard, therefore defining the capabilities of the dashboard as a whole. These analytical objectives are grounded in real world maternal health challenges and ensure that the interactive visualisations provide insight and actionable intelligence. Each task can be fulfilled by engaging with multiple charts, promoting multidimensional exploration of the data.

### 1.Temporal Analysis of Maternal Health Trends

***1.1 Temporal Analysis of Maternal Health Trends***

Identifying positive and negative long-term trends helps midwives and policymakers assess the effectiveness of past initiatives and decide whether current strategies should be maintained, adjusted, or continued, as well as determine if trends are being affected over specific time periods. Time-series line charts can display a selected group's changes over a set time period. When combined with bar charts and scatter plots, it allows users to observe how indicators evolve and compare the progress of groups and/or outcomes over time.

#### 1.2 Profile specific population subsets and track their experience over time

Understanding which groups face worse outcomes supports efforts to reduce health inequalities, improve culturally competent care, and fulfil obligations for ethical care and equity under government maternal health policies. The filters allow the user to isolate one group at a time or directly compare several at once. The time-series view can further reveal whether disparities have widened or narrowed over time.

### 2. Geographic Disparities and Regional Comparisons

#### 2.1 Compare outcomes across regions and detect geographic disparities

This helps target under-resourced areas or regions with poor outcomes, allowing maternal healthcare leaders and policymakers to allocate funding, training, or staff more equitably based on need. Choropleth maps offer a visual comparison of complication rates, birth volumes, etc., while bar charts and sunburst diagrams reveal what types of outcomes contribute to these regional differences. Allowing users to interact with the visualisations and apply regional filters, they can narrow in on specific regions and then use other visualisations to explore the reasons certain areas perform better or worse than others.

#### 2.2 Explore causes behind regional differences

After identifying regional differences, users can utilise other charts on the dashboard to drill down and find potential causes for these disparities, examining which outcomes contribute to negative or positive performance in certain areas. The interactive design facilitates detailed exploration of regional trends.

### 3. Demographic and Antenatal Care Analysis

3.1 Assess disparities in care and outcomes between demographic groups  
Understanding which groups face worse outcomes supports efforts to reduce health inequalities, improve culturally competent care, and fulfil obligations for equity under government maternal health policies. Stacked bar charts and sunburst visualisations show the proportions of various outcomes across demographic categories such as ethnicity or age. The filters allow users to isolate and compare groups directly, enhancing detailed demographic analysis.

3.2 Explore the relationship between antenatal care and birth outcomes  
Exploring these relationships helps validate (or challenge) assumptions about cause and effect, informing interventions aimed at improving maternal outcomes through better engagement in antenatal care. Using scatter plots alongside regional maps and time-series trends, users can visually assess whether areas with more antenatal visits experience fewer complications. Adjusting filters by region, ethnicity, or year helps uncover deeper patterns or exceptions. For example, a user might find that certain areas had low visit rates but still achieved positive outcomes, prompting further investigation.

### 4. Communication and Reporting

4.1 Summarise and communicate overall performance in maternal care  
Having a tool that summarises a decade of data in a visual, digestible format supports clear communication with both expert and non-expert audiences, including healthcare teams, commissioners, and the public, making the data more actionable. The dashboard is designed to provide a macro-to-micro-overview of maternity services, allowing users to switch seamlessly from high-level trends to detailed breakdowns. By combining various visualisations (line charts for trends, maps for locations, and bar/sunburst charts for group outcomes), a user can present a complete summary of service performance and disparities in a visually engaging manner.

## Data Collection

The visualisations created for this project were based on a publicly available dataset titled *Independent Midwives UK Research Database, 2002–2012*, sourced through the UK Data Service. This dataset consists of anonymised health records related to antenatal, intrapartum, and postpartum care and was originally compiled by Independent Midwives UK with the goal of capturing real-world clinical data across multiple stages of maternity care. With a decade’s worth of data and a wide array of variables, which range from maternal weight and ethnicity to antenatal visits and both maternal and neonatal complications; the dataset provided a strong foundation for producing comprehensive, long-term visual analysis.

The data collected consists of mixed formats, as the questionnaire used in which the data was used to collect was changed during the decade long study. This has meant that the data has been adapted as to portray the most accurate visualisations, which sometimes requires excluding datapoints that some would feel useful, however the data is too sparse to meaningfully add towards the visualisation(s). Despite this, the broad scope and temporal range of the dataset made it particularly well suited to the development of dynamic visualisations that could explore patterns, inequalities, and outcomes across different UK demographic and geographic dimensions.

All data was anonymised prior to acquisition, ensuring ethical compliance and the protection of patient confidentiality. This also allowed the visualisations to reflect real-world trends without risking the disclosure of any personally identifiable information

### Data Preprocessing and Cleaning

Before the data could be visualised, it underwent cleaning and preprocessing to ensure its usability and its accuracy. The data contained a mix of numerical, categorical and textual variables which all required tailored approaches to preprocessing. Given the complexity and wide range of datapoints within the dataset provided, the initial steps revolved around data cleaning, transformation and structuring to ensure the dataset could be accurately visualised.

A significant issue that needed to be addressed is the missing values in many areas of the dataset; these were addressed by using both imputation and deletion methods. Numerical variables were imputed using mean/median values to fill missing values whereas categorical variables such as ethnicity were reassigned to the most frequent variable. Any duplicates were removed to prevent an over-representation of particular data leading to a more reliable dataset. Where appropriate data will also be normalised to prevent large disparities when creating the data visualisations.

The biggest challenge encountered was the presence of mixed datatypes within the same columns, this was most present where integers and strings were used to represent the same data. Data standardisation was used to ensure that all variables followed the correct, uniform format, preventing errors when importing the data into the database.

## System Architecture

The project follows a three-tier web architecture consisting of a React frontend, a Flask backend, and a MySQL database. The interactive visualisation is provided as part of a user-interactive dashboard designed for exploring complex maternal health data across time, geography, and demographic groups.

The dashboard interface was designed with user readability in mind. Legends, axes, labels, and filters were carefully positioned to reduce visual clutter while preserving data clarity. Responsive layout techniques ensure that the dashboard is fully functional on desktops devices, promoting accessibility for a diverse audience (Hehman and Xie, 2021).

To support different analytical tasks, the dashboard includes a range of visualisation types. Time-series graphs powered by Chart.js reveal trends in service uptake and outcomes over time, supporting temporal analysis. D3.js was utilised for geospatial visualisations such as choropleth maps, which reveal regional disparities and allow interaction with data on a local level (Safwane, 2021). Stacked bar charts and sunburst diagrams allow for demographic and hierarchical exploration of outcomes.

Interactivity is further enhanced through dynamic filtering options, allowing users to tailor views based on variables such as ethnicity, antenatal visit counts, smoking habits, and more. Hover effects and interactive tooltips—implemented via Chart.js—provide detailed contextual information without overwhelming the user. This enables nuanced analysis while ensuring the system remains intuitive and visually digestible.

Each chart within the dashboard was implemented as an individual React component, with asynchronous data fetching handled via useEffect hooks. When a user interacts with the filters, a new SQL query is generated and passed to each component, triggering a fetch request to the Flask backend. The returned JSON data is then transformed into the correct structure for each charting library—in most cases, Chart.js for bar, line, and scatter charts, and D3.js for sunburst and choropleth visualisations.

Select charts support toggle buttons which allow users to show or hide all groupings to avoid visual clutter when comparing multiple groups and Legends are interactive to add/remove individual groups from view to help with analysis. The map and sunburst charts use D3’s dynamic rendering and transitions to present more complex or layered insights. All charts are linked to the same filtering logic, meaning updates happen across the dashboard in sync.

Filters are global, not scoped to a single chart, allowing the user to apply criteria once and then view those results through multiple visual perspectives. This ensures a smooth, consistent experience as the user explores the data. Care has been taken to ensure charts update responsively and without unnecessary API calls by structuring components around prop-based triggers and controlling render logic based on query changes, the dashboard remains responsive even with multiple filters active.

The React frontend was chosen for its component-based architecture and efficient rendering of dynamic data visualisations (Kothapalli, 2021). Each visualisation and UI control is encapsulated as a React component, which promotes code modularity, reuse, and maintainability. React also integrates seamlessly with popular charting libraries such as D3.js and Chart.js, allowing the dashboard to present maternal health trends in multiple formats—time-series graphs, bar charts, choropleth maps, and more—within a single-page webpage application. These capabilities are essential to support the dashboard’s dynamic interactive filtering system, enabling real-time updates and responsiveness as users explore different views of the dataset.

The Flask backend serves as a lightweight and flexible Python web framework that acts as an intermediary between the frontend and the database. Flask exposes RESTful API endpoints that the React frontend can call via asynchronous fetch requests to retrieve data. For instance, when a user changes a filter (e.g. selecting a specific region or ethnic group), React sends this input to Flask, which then queries the appropriate data from the database and returns it in JSON format. This structure ensures clean separation of concerns and a smooth user experience, allowing for scalable data retrieval logic and adaptable backend operations.

The MySQL database stores the pre-processed maternal health data, which spans a decade and includes various dimensions such as outcomes, patient demographics, and geographical regions. MySQL was selected for its robust querying capabilities and scalability, making it well-suited to support complex filtering, grouping, and aggregation operations required by the dashboard (S, V and V, 2023). For example, users can filter the data by year, region, or demographic category and receive immediate chart updates without performance degradation. The system remains responsive and accurate even under multiple filter layers, maintaining smooth interaction.

Figure System architecture diagram

React Frontend

FlaksAPI

Mysql Database

API Calls - JSON

SQL Queries

User (Web Browser)

HTTP Request

This modular architecture—React for interactive display, Flask for backend logic, and MySQL for scalable data storage—enables users to complete critical analytical tasks such as identifying long-term trends, comparing demographic groups, and detecting regional disparities. The separation of layers also allows individual components to be updated, maintained, or scaled independently (e.g., expanding database capacity or updating chart libraries) without affecting the overall system.

Together, this architecture and implementation strategy support the core goal of the dashboard: to empower users to engage with and interpret maternal health data in a way that is accessible, meaningful, and actionable.

### Dashboard Design

Navbar Component

Footer Component

TITLE

Select Box 1

Select Box 2

Select Box 3

Select Box 4

Chart Toggle

LineChart Component

ScatterChart Component

Stacked Bar Chart Component

Bar Chart Component

Sunburst Chart Component

choropleth map Chart Component (1)

choropleth map Chart Component (2)

Figure Dashboard design diagram

# Implementation

## Front-End Development (React)

The React frontend architecture leverages JavaScript and JSX, alongside key dependencies such as react-router-dom for client-side navigation, react-chartjs-2 for rendering interactive charts, and D3.js for complex, interactive geographic and hierarchical visualisations.

A screenshot of a graph

AI-generated content may be incorrect.

Figure Screenshot of the full deployed dashboard

### Navigation

Implemented globally via a modular Navbar.js component rendered at the top of every page, leveraging React Router’s <Link> for seamless, client-side routing between pages within the single page application without full page reloads, thereby enhancing user experience and application responsiveness. The interface is built around reusable, modular components designed to respond to dynamic SQL queries based on user interactions.

import React from 'react';

import { Link } from 'react-router-dom';

import './Navbar.css';

function Navbar() {

    return (

        <nav className="navbar">

            <div className="navbar-container">

                <Link to="/" className="navbar-logo">Maternity Insights</Link>

                <ul className="navbar-menu">

                    <li><Link to="/">Home</Link></li>

                    <li><Link to="/Dashboard">Dashboard</Link></li>

                    <li><Link to="/about">About</Link></li>

                </ul>

            </div>

        </nav>

    );

}

export default Navbar;

Figure Code snippet showing the Navbar Component



Figure Screenshot of Navbar at the top of the dashboard, with navigation links for Home, Dashboard, and About.

## Visualisation Components

Each visualisation (ScatterChartVisualization.js, BarChart.js, SunburstChartVisualization.js, MapChart.js etc.) is encapsulated as a React component. These components independently handle data fetching via asynchronous calls to the Flask backend.

The chart components are designed with modularity as the focus, with default declaration from the component returning the chart as well as it taking in a query to enable individuality. The lifecycle flow for the visualisation components are as follows:

* **Import and Setup**: The Chart.js modules are registered, and the component is declared.
* **Data Fetch**: A useEffect hook triggers a POST request to the Flask backend using the query provided by the dashboard.
* **Data Mapping**: The returned JSON response is mapped into Chart.js/D3.js-compatible format.
* **Chart Rendering**: A bar chart is rendered with dynamic axes, labels, and interactive tooltips.

For example, the BarChart.js component is structured as follows:

First, we import React various hooks and states to maintain the state of the component, followed by importing the appropriate Chart.js modules for the chart in question, (in this case we import the Bar chart) and register the elements of that chart we have imported.

import React, { useEffect, useState } from "react";

import { Bar } from "react-chartjs-2";

import {

  Chart as ChartJS,

  CategoryScale,

  LinearScale,

  BarElement,

  Tooltip,

  Legend,

} from "chart.js";

import './BarChart.css';

// Register Chart.js components

ChartJS.register(CategoryScale, LinearScale, BarElement, Tooltip, Legend);

Figure Code snippet showing the import and setup of the BarChart Component

As per React, we then register the component, making it intake a query prop, (provided via user interactivity on the dashboard), react will then trigger a fetch to the FlaksRepo sending the query as JSON and then receives the structured JSON results ready for charting. This method enables for reactive querying and decouples data access from the frontend logic.

  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await fetch("https://flaksrepo.onrender.com/api/query", {

          method: "POST",

          headers: { "Content-Type": "application/json" },

          body: JSON.stringify({ query }), // Use the passed query

        });

        if (!response.ok) {

          throw new Error(`HTTP error! Status: ${response.status}`);

        }

        const result = await response.json();

Figure Code snippet showing the BarChart component logic for fetching a query

A screenshot of a computer

AI-generated content may be incorrect.

Figure DevTools screenshot showing the returned query results from the API

Once the JSON response is received from the FlaskAPI, the data is mapped to the format required by Chart.js. In the below example script, the bar chart transforms the raw SQL response in to separate labels and datasets, in which each bar represents a different demographics antenatal visit frequency.

        if (result.results && result.results.length > 0) {

          // Extract labels and data from the query results

          const labels = result.results.map((item) => item.GroupKey);

          const percentages = result.results.map((item) => item.PercentageIMVisits);

          const totals = result.results.map((item) => item.TotalIMVisits);

Figure Code Snippet showing the mapping logic of the returned API results from the requested query

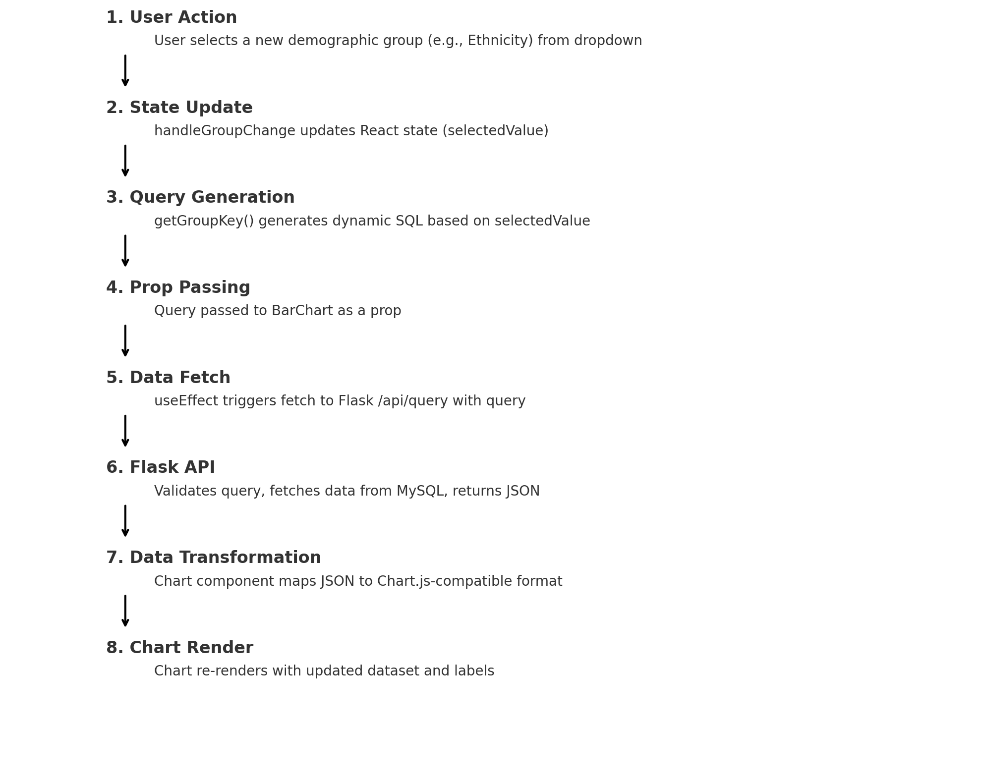


Figure Data flow from user interaction to chart rendering

This transformed data is passed into a Chart.js configuration object that defines visual styling, axis scales, and tooltip behaviour. The inclusion of detailed tooltips improves interpretability by showing both the percentage and absolute values on hover

          const datasets = [

            {

              label: "Percentage of IM Visits",

              data: percentages,

              backgroundColor: "rgba(75, 192, 192, 0.7)",

              borderColor: "rgba(75, 192, 192, 1)",

              borderWidth: 1,

              totals, // Store the total numerical values for tooltips

            },

          ];

Figure Code snippet showing dataset configuration used to display antenatal visit percentages. This setup includes a secondary value (totals) for enhanced tooltip readability, enabling display of both percentage and absolute counts.

Finally, the component will then return a div containing the formatted chart, containing tooltips that the user can hover over to view data in more detail.

  return (

    <div className="chart">

      <Bar

        data={chartData}

        options={{

          responsive: true,

          plugins: {

            legend: {

              position: "top",

            },

            tooltip: {

              titleFont: { size: 16 },

              bodyFont: { size: 16 },

              footerFont: { size: 14 },

            callbacks: {

              label: (context) => {

                const index = context.dataIndex;

                const dataset = context.chart.data.datasets[context.datasetIndex];

                const percentage = dataset.data[index]; // Percentage from the query

                const total = dataset.totals[index]; // Total IM visits from the query

                return `${percentage}% / ${total} IM Visits`; // Format percentage to 1 decimal place

              },

            },

            },

          },

          scales: {

            x: {

              title: {

                display: true,

                text: "",

              },

            },

            y: {

              beginAtZero: true,

              title: {

                display: true,

                text: "Percentage of IM Visits (%) / Total IM Visits",

              },

              ticks: {

                callback: (value) => `${value}%`, // Display percentages on the y-axis

              },

            },

          },

        }}

      />

    </div>

  );

};

Figure Code Snippet showing the BarChart t rendering configuration with a custom tooltip logic that displays both percentage and raw counts per data point

Finally, each visualisation is rendered declaratively using JavaScript. A div container with a class of chart holds the chart component, along with any dynamic titles and explanatory text that update in response to user-selected filters:

          <div className="chart">

            <label htmlFor="ChartTitle">Total Antenatal Care visits by {selectedLabel}:</label>

            <BarChart query={BarChartQuery} />

            <p>This chart shows the total number of antenatal care visits by {selectedLabel}. The chart is stacked to show the average number of visits, total number of visits, and the number of babies with birth complications, birth defects, resuscitation needed, baby deaths, and neonatal unit transfers.</p>

          </div>

Figure Code example of conditional chart rendering and context-aware descriptions. Chart content updates based on user-selected filters such as grouping category and date range.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Screenshot of a fully rendered chart.js Barchart

This pattern is consistent across all chart types and allows the system to display complex, real-time data in a visually digestible format.

## Chart Visibility Control

To further improve user interactivity over the visualisation interface, a "Toggle All" feature is implemented within select chart components that tend to become overwhelming, allowing users to show or hide all visible datasets with a single click. This is particularly useful when comparing many groups simultaneously, as it enables users to declutter the chart or focus on individual datasets without needing to toggle them manually one by one.

The functionality is achieved using a simple useState hook that tracks whether all datasets are currently visible. A boolean flag, showAll, determines the current state, and a handleToggleAll function inverts this value when the user clicks the button:

 const [showAll, setShowAll] = useState(true); // Toggle flag

Figure Code snippet of the ToggleAll useState hook

  const handleToggleAll = () => {

    if (!chartData) return;

    const updatedDatasets = chartData.datasets.map(ds => ({

      ...ds,

      hidden: showAll,

    }));

    setChartData({ ...chartData, datasets: updatedDatasets });

    setShowAll(!showAll);

  };

Figure Code snippet showing the toggleall button logic

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Figure Comparison of chart appearance before and after using “Toggle All”

In addition to custom interactivity such as the “Toggle All” button, several visualisations also benefit from Chart.js’s built-in interactive legends. These allow users to click on individual legend items to selectively hide or reveal specific data series.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Display of Chart.js integrated legend toggle feature

## Dashboard Logic

The central dashboard component is the central container for the entire visualisation interface, responsible for coordinating the frontend as well as visual outputs. The dashboard is responsible for loading all chart components as well as dynamically building the SQL queries based on user input.

All desired charts are imported to the dashboard; by importing each chart separately we ensure each chart type remains encapsulated, they all accept custom query props and fetch/render the data accordingly. The dashboard aggregates the charts into a unified interface whilst maintaining their ability to manage their own data lifecycle.

import React, { useState } from "react";

// Importing chart components and utilities

import ScatterChartVisualization from "../../components/ScatterChartVisualization/ScatterChartVisualization";

import SunburstChartVisualization from "../../components/SunburstChartVisualization/SunburstChartVisualization";

import StackedBarChartVisualization from "../../components/StackedBarChart/StackedBarChart";

import BarChart from "../../components/BarChart/BarChart";

import SelectBox from "../../components/Dashboard1Select/Select";

import LineChart from "../../components/LineChart/LineChart";

import MapChart from "../../components/MapChart/MapChart";

import MapChart2 from "../../components/MapChart2/MapChart2";

import Modal from "../../components/Modal/Modal";

import ChartVisibilityDropdown from "../../components/ChartVisibilityDropdown/ChartVisibilityDropdown";

import './Dashboard1.css'; // Importing CSS for styling

Figure Code snippet showing import statement for the dashboard, enabling the use of the required chart components.

Core react states are set up to allow for the user driven interactivity of the dashboard; by setting the states it causes all components to synchronize their rendering. useState hooks are declared to store and manage user input and the application state. selectedValue: stores the primary demographic variable used for grouping (e.g. ethnicity or region).

startYear and endYear determine the temporal range of the analysis. secondaryGroup allows an optional secondary demographic grouping, this adds more drill down capability and chartVisibility is a state object that tracks which charts should be visible based on user toggles. For each state variable, an accompanying handler function (such as handleGroupChange or handleStartYearChange) is defined to update the state when user input changes.

This approach ensures that all aspects of the dashboard—grouping variables, time filters, and visibility—are centrally managed and reactively updated, contributing to a responsive user experience.

const Dashboard = () => {

  // State variables for managing user selections and chart visibility

  const [selectedValue, setSelectedValue] = useState("v5WomEthnicOriginCode");

  const [startYear, setStartYear] = useState(2006);

  const [endYear, setEndYear] = useState(2012);

  const [modalContent, setModalContent] = useState(null);

Figure code snippet showing the dashboard state variables, as well as their default use state to use then the page is initially loaded

  // Handler for primary group selection

  const handleGroupChange = (event) => {

    setSelectedValue(event.target.value);

  };

Figure Code snippet displaying handling logic for the primary group selection, the logic is consitent for other value selections within the application

## User Interactivity: Grouping and Time Filters

Users have the option to interact with the dashboard through dropdown selectors in the SelectBox.js component, by changing the selected value in the select box the that grouping will now be applied to all visualisations. The select box does not allow multiple selections as this would break the query,

const SelectBox = ({ options, value, onChange, isMulti = false }) => {

  return (

    <div>

      <select

        id="selectBox"

        multiple={isMulti}

        value={value}

        onChange={(event) => {

          if (isMulti) {

            const selected = Array.from(event.target.selectedOptions, (opt) => opt.value);

            onChange(selected);

          } else {

            onChange(event);

          }

        }}

      >

        {options.map((option, index) => (

          <option key={index} value={option.value}>

            {option.label}

          </option>

        ))}

      </select>

    </div>

  );

};

export default SelectBox;

Figure Code snippet of SelectBox component for user-driven filtering

When the user selects a new primary grouping selection (Ethnic Grouping, Area Population size etc.) the handleGroupChange function updates the selected value state.

  // Handler for primary group selection

  const handleGroupChange = (event) => {

    setSelectedValue(event.target.value);

  };

Figure Code snippet JavaScript handler function logic for updating the selected demographic grouping

The selected value is then applied to the query via a variable GroupKey, there is an added condition for if the user decides they want to add a second grouping, else the method returns the sole value.

  const getGroupKey = () => {

    if (secondaryGroup && secondaryGroup !== selectedValue) {

      return `CONCAT(m.${selectedValue}, '-', m.${secondaryGroup})`;

    }

    return `m.${selectedValue}`;

  };

Figure Logic for constructing the GroupKey used in SQL queries. If a secondary grouping is selected, it concatenates both fields

That value is then fed into the queries allowing them to change based on user input, all queries are developed this was to ensure that they are all showing the same data in various formats. An example query used in the LineChart, shows this in operation:

  const LineChartQuery = `

  SELECT

    ${getGroupKey()} AS GroupKey,

    m.YearOfBirth,

    COUNT(b.BabyKey) AS TotalBabies

  FROM

    motherv5 m

  JOIN

    babyv5 b ON m.MainKey = b.MainKey

  WHERE

    m.YearOfBirth BETWEEN ${startYear} AND ${endYear}

    AND m.${selectedValue} IS NOT NULL

    AND m.${selectedValue} <> 'NULL'

    ${secondaryGroup ? `AND m.${secondaryGroup} IS NOT NULL AND m.${secondaryGroup} <> 'NULL'` : ""}

    AND b.BabyKey IS NOT NULL

  GROUP BY

    GroupKey, m.YearOfBirth

  ORDER BY

    m.YearOfBirth, GroupKey;

`;

Figure Code snippet displaying dynamically generated SQL query used for the LineChart component intergrating the user selected filters.

Each chart receives the query as a prop, When the user selects new filters—such as setting Ethnic Group and Year Range = 2008–2012. React triggers a state change using the useState hook. These values are then used to generate a dynamic SQL query string. For example, the following query may be constructed internally in the dashboard and sent to the API:

A screenshot of a computer

AI-generated content may be incorrect.

Figure DevTools screenshot of the constructed query sent to the API, based on the user input via the interactive dashboard

The explanation of the graph will change its details based on the user selection.

     {/\* Line chart \*/}

          {chartVisibility.lineChart && (

          <div className="chart">

            <label htmlFor="ChartTitle">Total babies born by {selectedLabel} over time:</label>

            <LineChart query={LineChartQuery} />

            <p>This line chart shows the total number of babies born over time for each {selectedLabel} group. The chart provides insights into the trends and patterns of births over the selected years.</p>

          </div>

        )}

Figure Code snippet of conditional rendering of the LineChart component. The chart title and description dynamically update based on user-selected filters

The below image shows the users ability to select the grouping based on the select box component.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Screenshot of the user viewing a chart based on ethnic grouping and set timescale

As we can see, compared to the previous image the chart content and descriptions have changed based on the user’s new selection. This architecture design allows for every chart to be tightly coupled to the current user defined filters which enables a fully interactive data exploration experience,

A screenshot of a computer

AI-generated content may be incorrect.

Figure Image displaying the change in chart data, Titles and descriptions based on user selection.

### Chart visibility control

To enhance usability and minimise cognitive overload, the dashboard incorporates a visibility management feature that allows users to selectively toggle charts on or off. This functionality is implemented through a centralised state object called “chartVisibility”, which tracks the visibility status of each visualisation component:

  const [chartVisibility, setChartVisibility] = useState({

    lineChart: true,

    scatterChart: true,

    stackedBarChart: true,

    barChart: true,

    mapChart2: true,

    mapChart1: true,

    sunburstChart: true,

  });

Figure Code snippet of react useState hook managing chart visibility. The chartVisibility object tracks which visualisation components are currently displayed on the dashboard

This state is passed as props to a component named “ChartVisibilityDropdown”, which renders a dropdown menu containing toggleable checkboxes corresponding to each chart. The user can interact with these checkboxes to hide or reveal individual charts

        <div className="item4 "><ChartVisibilityDropdown

        chartVisibility={chartVisibility}

        setChartVisibility={setChartVisibility} />

        </div>

Figure Code snippet of the ChartVisibilityDropdown component rendered on the dashboard

Each chart component on the dashboard is conditionally rendered based on its associated flag in the chartVisibility object. For example, the bar chart is only displayed if chartVisibility.barChart is set to true. The button is then implemented on the dashboard:

        <div className="item4 "><ChartVisibilityDropdown

        chartVisibility={chartVisibility}

        setChartVisibility={setChartVisibility} />

        </div>

Figure Code snippet of the ChartVisibilityDropdown component rendered on the dashboard

{/\* Bar chart \*/}

        {chartVisibility.barChart && (

          <div className="chart">

            <label htmlFor="ChartTitle">Total Antenatal Care visits by {selectedLabel}:</label>

            <BarChart query={BarChartQuery} />

            <p>This chart shows the total number of antenatal care visits by {selectedLabel}. The chart is stacked to show the average number of visits, total number of visits, and the number of babies with birth complications, birth defects, resuscitation needed, baby deaths, and neonatal unit transfers.</p>

          </div>

        )}

Figure Code snippet of conditional rendering for the BarChart component

This feature gives users granular control over the visual workspace, allowing them to remove unnecessary or irrelevant charts and focus only on the insights that matter to their current analysis.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Screenshot of ChartVisibilityDropdown in use

A screenshot of a computer

AI-generated content may be incorrect.

Figure Chart visibility example: Reduced chart selection

This seamless pipeline ensures that every user interaction results in a consistent and responsive experience, with all charts and labels dynamically updating in unison.

### Back-End Development (Flask)

The backend of this project is developed using Flask, a lightweight Python web framework that enables the construction of robust RESTful APIs. Its primary function is to act as a middleware layer between the React frontend and the MySQL database, processing client requests, executing SQL queries, and returning the results in a structured format. To ensure performance and maintainability, the backend code is structured around clear API endpoints. The core endpoint, /api/query accepts POST requests containing a SQL SELECT query string. The endpoint is safeguarded against SQL injection by enforcing that only read-only queries (i.e. those starting with SELECT) are permitted.

A screenshot of a computer

AI-generated content may be incorrect.

Figure API request activity captured in Microsoft Edge DevTools.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Example API request payload, showing dynamically generated SQL query sent from the React frontend to Flask.

Key API endpoints include:

* **/api/query:** Accepts POST requests containing SQL SELECT queries.

# Main Query Route

@application.route('/api/query', methods=['POST', 'OPTIONS'])

def run\_query():

        # Handle preflight CORS request

        return 'OK', 200  # Return 200 OK response for OPTIONS method

    try:

        # Get the query from the request body

        data = request.json

        query = data.get('query')

        # Only allow SELECT queries for security purposes

        if not query.strip().upper().startswith('SELECT'):

            return jsonify({"error": "Only SELECT queries are allowed"}), 400  # Return error for non-SELECT queries

        # Connect to the database

        conn = pymysql.connect(\*\*db\_config)

        cursor = conn.cursor()

        # Execute the query

        cursor.execute(query)

        results = cursor.fetchall()

        # Get column names from the query result

        column\_names = [desc[0] for desc in cursor.description]

        # Format results as a list of dictionaries

        formatted\_results = [dict(zip(column\_names, row)) for row in results]

        # Clean up and close the connection

        cursor.close()

        conn.close()

        # Return results as JSON

        return jsonify({"results": formatted\_results})

    except Exception as e:

        # Handle and return errors

        return jsonify({"error": str(e)}), 500

Figure Code snippet of Flask /api/query route for dynamic SQL execution

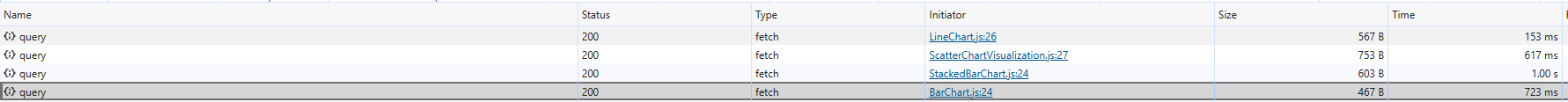


Figure Microsoft Edge DevTools Screenshot displaying multiple /query requests triggered by chart components

* **/api/mother and /api/baby:** Dedicated endpoints designed for retrieving entire datasets from respective MySQL tables (motherv5 and babyv5). Data from these endpoints is provided in JSON format, directly consumable by the React frontend visualisation components.

# Mother Route to fetch all data from the 'mother' table

@application.route('/api/motherv5', methods=['GET'])

def get\_mother\_data():

    """Fetch all data from the 'mother' table."""

    try:

        # Connect to the database

        connection = mysql.connector.connect(\*\*db\_config)

        cursor = connection.cursor(dictionary=True)

        cursor.execute("SELECT \* FROM motherv5")  # Query to fetch all rows from 'motherv5' table

Figure Code snippet of Flask /api/motherv5 route for retrieving full dataset

Database connectivity employs context-managed connections to optimise performance, ensuring resource efficiency and backend responsiveness. This backend structure ensures clean separation of concerns, where the API layer handles business logic and security, while the frontend remains focused on interactivity and visualisation.

### Database Implementation (MySQL)

MySQL, hosted securely on AWS RDS, stores and structures maternal health data spanning over a decade. The database employs relational design principles, maintaining referential integrity and supporting complex query operations essential for visualisation interactivity. The use of a relational database was chosen due to the structured nature of the data and the need for reliable joins and aggregations across different dimensions, such as time, region, and demographic groupings.

* **Tables:** Primary tables include motherv5 and babyv5, structured specifically to support robust querying and efficient aggregation operations.
* **Optimisation:** Schema design is optimised for the types of queries dynamically generated by user interactions on the dashboard, ensuring fast query responses.
* **Data Integrity:** Strict access controls and sanitisation procedures are implemented, safeguarding data integrity and reliability for accurate visualisations.

**Mother**

(PK) MotherID

**Baby**

(PK)BabyID

(FK) MotherID

Figure Database Schema

A screenshot of a computer

AI-generated content may be incorrect.

Figure Database ERD Screenshot from MySQL Workbench

### Deployment Strategy

The full-stack application is deployed using AWS Amplify for hosting the React frontend, providing a scalable, secure, and streamlined deployment process directly integrated with GitHub, and can be accessed with the following link: https://staging.d2id22gmrvore6.amplifyapp.com/

A screenshot of a computer

AI-generated content may be incorrect.

Figure Screenshot of AWS Amplify hosting panel for the frontend

The MySQL database is hosted on Amazon RDS, offering reliable, managed database services with automated backups and security features.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Screenshot of AWS RDS Console hosting the maternity database

The Flask backend is deployed on Render via FlaskREPO, providing a robust, managed hosting solution that integrates seamlessly with GitHub for continuous deployment, ensuring high availability, security, and ease of maintenance, the flaskAPI does however take 30-60 seconds to spin up as per the account tier being used for deployment.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Flask backend hosted on Render, showing deployment logs and status

### Version Control and Collaboration

A screenshot of a computer

AI-generated content may be incorrect.The project utilises GitHub for version control and tracking of development changes. Feature branches are created for implementing new functionalities, bug fixes, or updates, which are merged into the main branch after review through pull requests. GitHub Issues and Projects further support tracking of milestones, tasks, and progress, enhancing transparency and collaborative development.

Figure Screenshot of a list of GitHub commits for the project

# Evaluation

The goal of the evaluation was to determine how effective the developed maternal health dashboard was at enabling users to perform meaningful analytical tasks. The success of the system is determined by not just it’s technical robustness but also on its ability to support diverse user needs and clearly communicate data trends by making a complex dataset more accessible through bring interactivity to data visualisation.

Participants were recruited for user testing, representing a mix of midwives, technical users and members of the public. This balanced mix allows for an evaluation based on multiple perspectives, from specialist insights of midwives to the general accessibility viewed by those without domain expertise. All participants had at least a basic computer literacy.

Participants were asked to complete a set of core analytical tasks using the dashboard and then rate how well the dashboard supported completing each task on a scale of 0-10 (not at all- to – extremely well). Each questionnaire task corresponds to one of the key user objectives described in the Methodology. After each task in the questionnaire, there is an opportunity for any comments about the overall value of the interactivity, specifically revolving around whether the interactivity of the dashboard makes it easier to understand the information compared to if there was just a static chart.

## Questionnaire Design

A screenshot of a computer

AI-generated content may be incorrect.Participants were asked to complete a set of ten core analytical tasks using the dashboard, then rate how well the dashboard supported each task on a scale from 0 (not at all) to 10 (extremely well). Each task corresponded to one of the key user objectives described in the Methodology chapter. These tasks covered objectives such as identifying trends over time, comparing regional outcomes, analysing disparities among demographic groups, exploring relationships between antenatal care and outcomes, and summarising overall performance. By structuring the evaluation around these tasks, we directly tested whether the dashboard fulfilled its intended purposes.

After completing the tasks, participants answered an additional question about the overall value of interactivity; specifically whether using the interactive visualisations made it easier to understand the information compared to viewing the same data in static form (e.g. printed graphs or tables). Participants were also invited to provide free-form comments to explain their scores, highlight any usability issues, and suggest improvements. This combination of quantitative ratings and qualitative feedback provided a comprehensive view of the dashboard’s effectiveness.

Figure Questionnaire given to feedback participants.

## Quantitative Results

All participants were able to successfully complete the assigned tasks using the dashboard. Table 1 summarises the average scores given for each core task (0–10 scale). Overall, the results were positive: the mean score across all ten tasks was 8.3 out of 10, indicating that users generally found the dashboard very effective for their analytical needs. Scores for individual tasks ranged from 6.7 for the most challenging task up to 9.3 out of 10. Most notably when asked to compare the interactive dashboard’s effectiveness in comparison to their experience using static data presentations, participants gave very high ratings, reflecting a strong preference for interactivity when using visualisation tools. The following is a breakdown of each task’s outcome.

Figure Task performance average scores from user feedback

#### Task 1 – Identify trends over time

Participants found that the time series line charts made it easy to identify longitudinal trends in maternal health data, giving strong feedback on the clarity of the line chart for displaying trends. Noting the interactivity a few users raised the need for a tutorial on how to use the dashboard before starting would have improved their confidence in using the dashboard to adjust time periods, although after a short explanation those issues were resolved. Overall the users were able to use the interactive visualisations to detect both positive and negative trends across the decades worth of data with little difficulty, appreciating how quickly the interactive timeline conveyed changes over time. This task had an average rating of 8.6/10.

#### Task 2- Understand how indicators evolve across charts

This task involved using multiple charts together such as observing the line, bar and scatter charts to see how different indicators change in tandem when adjusting the groupings and time series of the dashboard. Participants overwhelmingly praised the design of the dashboards interactivity in this regard, commenting that’s the varying chart types helped better understanding if the indicators and their evolution. The very high average score of 9.3/10 indicates that the users felt the dashboard effectively enabled multi-dimensional analysis due to the interactivity of the dashboard with one user saying that provided a “comprehensive overview, allowing easy cross-referencing between different data views”. Showing the interactive dashboard made complex maternal health data easier to understand for users.

#### Task 3 - Compare outcomes across regions and detect geographic disparities

By using the choropleth charts on the dashboard, users were able to compare outcomes across regions to spot any geographic disparities in the maternal health dataset. The map was highly rated for providing a visual overview of the metrics in each area; particularly allowing those with high maternal knowledge to quickly determine which regions were performing worse and why. Some participants suggested some changes to the choropleth charts would like a more present key for the graphs to indicate what the data infers for less computer literate users. Another suggestion was to have the tooltips/hover labels to be more persistent to easier compare the data to other charts, however even with these lack of features, users were all still able to successfully complete the task, and rated it an average score of 8.2/10, implying the interactive dashboard allowed users to effectively highlight geographic disparities.

#### Task 4 – Explore causes behind the regional differences using the dashboard

After identifying these regional differences users were asked to use the other charts on the dashboard to drill down and find potential causes for those disparities, examining which outcomes contributed to those negative outcomes. Several uses commented that it was intuitive to use the dashboard to drill down to find this data, with one saying, “extremely easy to find and focus on outliers or specific segments.” Regarding observing the sunburst chart and the choropleth charts after applying filters to the data. One participant with less technical background gave a lower score for this task (mentioning that initially the multi-layered sunburst visualisation felt complex), but even he was able to interpret it after a brief orientation. Once familiar with the concentric chart, users could identify which specific outcome category (e.g. postpartum complications, antenatal visit frequency, etc.) was disproportionately high in a given region. The high average score indicates that, overall, the dashboard succeeded in helping users uncover the factors behind regional differences, with only a minor learning curve noted by a couple of individuals, users gave this task an average score of 8.5/10.

#### Task 5 – Compare Demographic groups

Achieving an average score of 8/10, this task required users to compare outcomes across different demographic groups using the various interactive tools the dashboard offers and observe how disparities between those groups develop over time. Users were successful in isolating and comparing groups using the interactive nature of the charts, a user saying the interactive group filters “allowed for clear highlights and removal of groups, enabling fully customisation”. Some users noted usability issues that arose when groups being compared returned increasing datapoints, as clicking away unwanted datapoints one at a time was called a “tedious” process. Another user admitted it was *“*initially confusing*”* to figure out how to manage the multiple group selections until she became familiar with the controls. These comments highlight a need to simplify the group selection interface. Despite those concerns, when properly filtered, the dashboard did reveal clear disparities between demographic groups, and users recognised the value of being able to compare demographic groups. The feedback from this task directly pointed to some improvements in the UI, even though participants generally succeeded in making the comparisons they wanted.

#### Task 6 – Track disparities between groups over time

Participants combined the filtering tools with the graphs to observe how disparities between groups changed over different time periods selected by the user. Most users indicated that once they used the tools to isolate the group of their interest, the charts made it easy to infer if the gap between those groups was widening or narrowing over time. One participant noted that the line graph “showed the trend visibility well when using the filters”. The good score average of 8.3/10 for this task suggests that the longitudinal tracking of disparities was well supported by the dashboard, and it seems based on user comments that the task’s success depended on the interactive features of the dashboard working smoothly, and improving upon these features would further enhance user experience.

#### Task 7 – Explore the relationship between antenatal care and birth outcomes

Achieving an average score by users as 6.7/10, this task was most challenging for users. Users were asked to use the dashboard to examine possible correlations between antenatal care and birth outcomes based on the various groupings available to select in the data. Those with a background in maternal healthcare found this less of a challenge, as they were aware of what grouping were most likely to show the findings they were looking for, however those with less knowledge were not as successful. All users found that groupings with higher birth complication rates tended to trend having lower antenatal care visits, however the common issue was found that the charts “became crowded quickly and hard to interpret.”, with other users indicating that the lack of clarity regarding the axis on the charts caused a lack of clarity. On the positive side, once the chart was explained or when users focused on a simpler subset of data, they could use it to explore the hypothesis that “better antenatal care leads to better outcomes.” However, the consensus was that this chart could be refined for clarity. The relatively modest score highlights that, unlike other parts of the dashboard, the scatter plot and related correlation exploration need improvements (e.g. clearer labels or on-chart guidance) to be fully effective for all users, though it is important to note that the interactivity of the charts is what allowed for more clarity of the data for the users.

#### Task 8 - Use filters to discover exceptions or patterns freely

Scoring 8/10 on average, this test allowed users to use the interactivity of the dashboard to make discoveries about the dataset. The task tested the dashboard’s ability to support open ended exploration, by allowing the users to apply filter to the data, select grouping and drill down into this data using the various charts. Most participants rated this task very highly commenting on how they found the interactivity of the dashboard “extremely powerful” for uncovering hidden insights. They praised how applying filters – such as selecting a particular region, time frame, or demographic subgroup, which instantly updated all relevant charts, enabling them to spot correlations that would have been difficult to see in a static dataset. Based on feedback, interactive filtering and free exploration emerged as some of the dashboards strongest features.

#### Task 9 - Profile a specific population group and track their outcomes

As another top performing scoring task achieving an average of 9/10. Participants used the dashboard to isolate on group at a time and examining their outcomes across variables in detail. he dashboard’s filtering by demographic, along with the ability to highlight or isolate one segment on charts, was designed to support this kind of isolated group tracking. Although this task scored extremely well, one user mentioned that it would be even better if the dashboard initially showed just the selected group and hid others automatically, to avoid any confusion. Nonetheless, given that most people naturally filtered down to a single group for this task, clutter was rarely a problem here. The high score indicates that the interactivity of the dashboard excelled at letting users drill down into particular groupings and focus on their data, which was a primary goal of the project.

#### Task 10 - Summarise overall performance in maternal care

This task evaluates how well the interactive dashboard provided a high level summary of maternal health outcomes and services for the dataset, allowing the user to see the overall picture of the dataset visually. Participants reviewed the dashboard as a whole, examining any patterns and improvements over time and major differences between regions or demographic groups. The interactive nature of the dashboard meant that users could quickly switch between different indicators to include in their summary. For example, one midwife participant described how she could start by showing a national trend (using the line chart for something like maternal mortality rates over time), then click on a region on the map that looked like an outlier to immediately drill into that region’s specifics, and even filter by a demographic factor if needed – all within the same interface. She found that this ability to seamlessly transition from a high-level view to detailed data greatly aided in telling the story of the dataset. The average score of 8.0/10 suggests that users were generally satisfied with the dashboard’s capability to communicate the “macro” picture of maternity services performance. They praised how multiple charts in combination provided a macro-to-micro perspective. There were a few minor suggestions to improve this summarisation experience: for instance, participants recommended adding clear titles or brief captions to each major chart when presenting to non-experts, to ensure that someone unfamiliar with the dashboard could quickly grasp what each visualisation represents.

Finally, users were asked to comment on the interactive nature of the dashboard in comparison to their previous experience with static graphs and/or reports. when comparing the interactive dashboard to static graphs or reports, participants were overwhelmingly positive. In response to the additional question “Is the interactive dashboard easier to understand and more insightful than viewing the same data in static form?”, every participant indicated a strong preference for the interactive dashboard. The average rating for this question was about 9 out of 10, with all six users giving a high score, this strong consensus highlights the value added by interactivity.

In summary, the quantitative results indicate that the dashboard met its design objectives to a high degree. Most analytical tasks received strong scores, with only a few specific areas such as the group selection interface, showing room for improvement. The interactive dashboard was found to be far more helpful than static data presentations, reinforcing the advantage of interactivity for data exploration.

Figure Participants ratings for the final question

## Qualitative Analysis

Using the participants comments, we can find a deeper understanding of why certain features worked well or where users encountered some issues with the dashboard. Overall feedback was very positive, the participants consistently remarked that the dashboard was easy to use and its interactive features significantly improved their understanding of the data. Some of which indicating that the interactive nature of the dashboard helped them to better engage with the data, compared to more traditional non-interactive methods. The following sections will be a summary of the key strengths and weaknesses highlighted by users.

### Strengths

#### Impact of interactivity over static formats

All Users strongly agreed that the interactive nature of the dashboard made it significantly easier to use and more insightful than traditional static charts or reports. This was a recurring theme in their comments, quantitatively underscored by the 9/10 average on the related question at the end of the questionnaire. Users said that being able to actively engage with the data by clicking on elements, filtering down to specific subsets, and hovering for details improved both their engagement with the content and their comprehension of it. Instead of being forced to connect information from multiple static graphs or scan through pages of tables, the interactive dashboard let them bring the information they cared about to the forefront with a few user based changes for example ability to personalise the view such as focusing on a particular region or demographic group of concern kept users more engaged, because each person could investigate questions relevant to them. This qualitative feedback reinforces the quantitative finding that interactivity adds substantial value for users. Participants often found themselves exploring beyond the initial questions – the interactive features invited them to test “what if” scenarios, try different filters, and satisfy their curiosity in ways a static report simply would not allow.

#### Filtering and drill down capabilities

A point of praise that consistently was mentioned by participants was the dashboards’ ability to flexibly filter data using the provided interface, making all charts instantly update based on the users change. Users valued the ability to apply global groupings and then further filter those groupings to see how those selections reflected across each chart simultaneously. This real time filtering capability was highlighted as being extremely helpful for uncovering patterns that would otherwise remain hidden in the data. Participants emphasised that the insights they found were revealed because of the ability to filter and drill down into data provided by the dashboard interactivity, particularly the dynamic filtering. Users felt confident that they could probe the data from multiple angles with ease, which in turn encouraged them to ask more questions and dig deeper than they would have with static charts.

#### Combination of charts and data layering

Participants highlighted their appreciation for the dashboards use of multiple charts and their coordination, each representing different facets of the data. The link between the line chart (for trends over time), bar/stacked bar charts (for categorical breakdowns), and the scatter plot (for correlations) was frequently mentioned as a strength, as user could observe a trend in the line chart and immediately see its breakdown by category in the bar chart below. This combination of views often helped users confirm or explore hypotheses such as seeing a rising trend in the line chart and then checking the scatter plot distribution to understand which sub-groups contributed to that rise. It provides a more well-rounded understanding than any single chart could in isolation.

#### Clarity of regional differences

The choropleth map was very well received as a tool for highlighting geographic disparities in maternal health outcomes. Midwives in particular commented on the effectiveness of the charts giving them an immediate intuitive sense of regional performance differences. By glancing at the colour intensities on the map, users could quickly pinpoint outlier regions as region shaded darker might indicate higher complication rates or lower service coverage. Users could then click that region to filter or highlight it and use the other charts to delve into why that region might be performing differently. Participants across backgrounds found that the visual impact of seeing the whole country’s data at once helped them grasp disparities without needing to wade through spreadsheets or lengthy reports in more traditional static methods.

### Weaknesses

Despite the generally positive feedback, participants did identify a few usability issues and areas where the dashboard could be improved. Many of these related to handling of complex interactions or edge cases in the interface.

#### Visual clutter when too many categories are shown

When all categories or groups were displayed simultaneously, certain charts could become overwhelming. This was especially noted for the sunburst chart and the stacked bar chart when every segment (all age groups, all ethnicities, etc.) was shown at once. In such cases, the visualisation could appear cluttered and difficult to interpret, defeating the purpose of providing clarity. While users appreciated that the dashboard allowed them to see “everything” if desired, they felt the default view should perhaps be simplified to prevent information overload. Participants recommended that the dashboard start in a cleaner state such as showing a single representative group by default. The design philosophy moving forward is to ensure that the dashboard presents a clear starting point and that any deep dive into multiple categories is done deliberately by the user, rather than by accident.

#### Demographic group selection interface

The most common critique was related to how multiple demographic groups are handled in the charts. As noted in the task-specific feedback, managing the visibility of many groups at once had a tendency to become overwhelming very quickly for users. In the initial design, comparing specific groups meant manually toggling each category on or off. Several participants found this tedious when they wanted to declutter a chart or focus on just one group quickly. A common request was for a one-click *“toggle all”* button to show or hide all groups at once. This would allow users to reset the group selection with one click, making it much easier to start from a clean slate or to isolate a single category. Implementing these changes will prevent user frustration when trying to manage groupings on charts.

#### Tooltip design on maps

Another issue raised by multiple participants concerned the tooltips on the choropleth map. By design, when a user hovers over a region on the map, a small tooltip appears with the region name and its data values. Some users found that these tooltips disappeared too quickly or were hard to read, especially if they wanted to compare two regions side by side. They suggested making the tooltip information more persistent or otherwise easier to inspect. Additionally, highlighting the region’s boundary more clearly on hover (such as changing its border colour or shading) was recommended to draw attention to exactly which region is being examined, particularly when several regions have similar colours. These changes would allow users to more comfortably examine and compare regional values without losing context or rushing to capture the info before the tooltip disappears

## Effectiveness of Interactivity vs Static Visualisation

The final question of the evaluation had the participants reflect on the value of the interactivity of the dashboard. The participants were asked “Was the interactive dashboard easier to understand and more insightful than viewing the same data in static form?” and the response was overwhelmingly positive. With an average rating of 9/10, all participants indicated a strong preference for the interactive dashboard over the static representations of the data.

Participants noted that static reports often felt overwhelming, with the data crossing multiple spreadsheet pages or charts displaying overwhelming amounts of data points, or on the contrary too few. This causes the reader to require more effort to interpret the data as well as more background knowledge on the subject matter to gain insight into the dataset. In contrast, the dashboard’s interactive nature allowed immediate filtering and comparison, so users could effortlessly answer their own questions, finding insight and trends. During discussion a participant gave the example of “If a question came to mind, like ‘Has outcome X improved in my region?’, I could instantly apply a filter or click on the region and get the answer, rather than flipping through pages of a report to find that information.” This immediacy of dynamically updating is a huge advantage of the interactive dashboard, with the highlights and tooltips making the dashboard more engaging and patterns more noticeable. A midwife contrasted their previous experience of viewing static reports compared to using the interactive dashboard stating “A static graph doesn’t let me drill into why something is happening; I’d need to look at multiple separate charts for more context. This dashboard gives me all of that in one place.” This quote encapsulates the sentiment shared by many users: the interactive dashboard integrates everything needed for analysis in a single platform, whereas static outputs are fragmented and passive.

Overall, this feedback is strong evidence that interactivity truly enhanced users’ experience and ability to understand the data, providing them with insight and noticing trends in which to action. Participants felt they could explore the data in ways that would be impossible with static graphs – especially for exploratory tasks where the relevant questions evolve as one looks at the data. The ability to iteratively ask and answer questions in real time by interacting with the visualisations was identified as a major advantage. In summary, users unanimously found the dashboard to be a clearer, more engaging, and more efficient tool for understanding maternal health trends than traditional static data presentations.

## Alignment with Task Design

In their feedback, users explicitly or implicitly touched on each of the major objectives. They used the time-series line charts (often alongside bar charts) to successfully identify longitudinal trends in the data, noting patterns over the years. They leveraged the choropleth map and supporting charts (sunburst and bar charts) to compare regions and uncover geographic disparities, and to examine demographic inequalities by filtering or selecting groups on the stacked bar charts. They utilized the scatter plot and filters to investigate relationships between antenatal visit counts and outcome measures. It was found that the interface allowed an easy switch from a high-level overview looking at aggregate data to focusing on a specific region or group, which was essential for profiling subsets of the population and summarising performance at different levels. In other words, each real-world scenario that the dashboard was intended to support proved to be achievable in practice during the user tests. This alignment between the design goals and the actual user experience is a strong validation of the dashboard’s design choices. the evaluation provides evidence that the dashboard enables meaningful and accurate exploration of the maternal health dataset in ways that map directly to the users’ real world questions and objectives.

## Iterative Development and Response to Feedback

The user feedback obtained from this evaluation did not only evaluate the final product but also formative as it guided final refinements to the dashboard in an iterative, design approach. Based on the participants’ comments about areas for improvement, several changes have been implemented/planned post-evaluation to better align the tool with user needs. Key improvements include:

#### “Toggle all” button for group visibility

A toggle-all control was added to the demographic selection panels on relevant charts. This allows users to quickly show or hide all groups with one click, making it much easier to declutter the view or focus on a single category at a time. This directly addresses the frustration some users had when dealing with multiple group toggles individually. After adding this feature, a user wanting to isolate one demographic group can simply “hide all” and then select the one of interest, streamlining the interaction.

#### Improved labelling and annotations

Across the dashboard, chart titles, axis labels, and legends are being refined for greater clarity by avoiding technical jargon where possible. Additionally, a short descriptive text or tooltip is being added below charts to guide first time users in how to read them. These changes aim to make the visualisations more self-explanatory, especially for non-expert users, and address the moments of confusion noted by participants. This should reduce the learning curve for new users.

These iterative improvements demonstrate the project’s commitment to a user-centred design, by observing actual users and listening to their suggestions, the dashboard evolved to become more usable and effective. Each adjustment corresponds to an issue identified during the evaluation, showing a direct line from user feedback to design change.

# Conclusion

In conclusion, the evaluation confirmed that the dashboard is a usable, effective, and valuable tool for exploring maternity service data. Participants from both technical and non-technical backgrounds were able to navigate the interface with relative ease and accomplish a variety of analytical tasks. The interactive features like as dynamic filtering, coordinated multiple charts, and real time updates, enabled users to gain insights that would have been difficult to obtain from static reports alone. In comparative terms, all users found the dashboard far more helpful than traditional static data presentations, underlining the advantage of interactivity for data exploration.

Critically, the dashboard allowed users to explore various disparities such as, between regions or demographic groups, validate assumptions and uncover exceptions within the data. The Dashboard encouraged users to develop new, data-driven questions as they interacted with the visualisations which is an essential part of successful exploratory analysis. A participant might start by noticing a regional trend, then question what demographic factors underline it, leading them to filter by that demographic and discover another pattern. This kind of hypothesis generation and exploration is facilitated by the dashboards responsive interactive design. Users’ ability to iteratively drill down and pivot their analysis in real time indicates that the dashboard supports open-ended inquiry, not just static queries.

The positive feedback and task performance outcomes have validated the design decisions made during development. The choice of chart types and other features such as the inclusion of interactive filters, and the integration of a database backend to handle live queries, all contributed to the successful user experience observed. Ultimately, the evaluation results suggest that the interactive dashboard aligns well with end-user needs and achieves its primary goal: it makes complex maternal health data transparent, understandable, and actionable for a range of stakeholders.

However, the evaluation also revealed areas for improvement. Some participants initially struggled to interpret more complex visualisations such as the sunburst chart or choropleth map, suggesting that further refinement of legends, onboarding tooltips, or brief tutorials could enhance accessibility. From a technical perspective, the decision to couple a React frontend with a Flask backend and MySQL database proved effective for real-time interaction. The use of parameterised queries and state-driven components created a modular, responsive interface that scaled across various user tasks. Nonetheless, some of the dashboard’s SQL queries—particularly those involving multiple joins and groupings—could become performance bottlenecks at scale. While current response times remained acceptable during testing, future iterations may benefit from backend optimisations or query caching strategies.

By supporting exploratory analysis in an intuitive way, the dashboard can help healthcare professionals and the public alike to derive meaningful insights from maternity service data, potentially informing better decision-making and highlighting areas for intervention. This alignment with user needs and the strong performance on real-world tasks indicate that the project has met its objectives, providing an effective interactive visual analytics tool for maternal health data. Additionally, although the system performed well under typical use, scalability under larger datasets or multi-user environments was not formally tested and remains a future consideration.

# References

Agrawal, R., Kadadi, A., Dai, X. and Andres, F. (2015). Challenges and opportunities with big data visualization. *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems - MEDES ’15*. doi:https://doi.org/10.1145/2857218.2857256.

Benbba, S. (2021) *Comparison of D3.js and Chart.js as visualisation tools*. Bachelor’s Thesis. Tampere University. Available at: https://urn.fi/URN:NBN:fi:tuni-202105243301

Besançon, L., Cooper, M., Ynnerman, A. and Vernier, F. (2020). *An Evaluation of Visualization Methods for Population Statistics Based on Choropleth Maps*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2005.00324.

Callaghan-Koru, J.A., Newman Chargois, P., Tiwari, T., Brown, C.C., Greenfield, W. and Koru, G. (2024). Public Maternal Health Dashboards in the United States: Descriptive Assessment. *Journal of Medical Internet Research*, 26, p.e56804. doi:https://doi.org/10.2196/56804.

Chishtie, J., Bielska, I.A., Barrera, A., Marchand, J.-S., Imran, M., Tirmizi, S.F.A., Turcotte, L.A., Munce, S., Shepherd, J., Senthinathan, A., Cepoiu-Martin, M., Irvine, M., Babineau, J., Abudiab, S., Bjelica, M., Collins, C., Craven, B.C., Guilcher, S., Jeji, T. and Naraei, P. (2022). Interactive Visualization Applications in Population Health and Health Services Research: Systematic Scoping Review. *Journal of Medical Internet Research*, 24(2), p.e27534. doi:https://doi.org/10.2196/27534.

Domova, V. and Shiva Sander-Tavallaey (2019). Visualization for Quality Healthcare: Patient Flow Exploration. doi:https://doi.org/10.1109/bigdata47090.2019.9006351.

Fitzmaurice, G. (2017). *Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing*. [online] Autodesk Research. Available at: https://www.research.autodesk.com/publications/same-stats-different-graphs/.

Hehman, E. and Xie, S. (2021). ASSOCIATION FOR PSYCHOLOGICAL SCIENCE Doing Better Data Visualization. 4(4), pp.1–18.

Kothapalli, M. (2021). The Evolution of Component-Based Architecture in Front-End Development. *Article in the Journal of Scientific and Engineering Research*. doi:https://doi.org/10.5281/zenodo.12772844.

Olufisayo Olakotan, Jennifer and Pillay, T. (2024). Challenges and opportunities in perinatal public health: the utility of perinatal health inequality dashboards in addressing disparities in maternal and neonatal outcomes. *BMC Pregnancy and Childbirth*, 24(1). doi:https://doi.org/10.1186/s12884-024-07056-z.

Plaisant, C., Milash, B., Rose, A., Widoff, S. and Shneiderman, B. (1996). LifeLines: visualizing personal histories. *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI ’96*. doi:https://doi.org/10.1145/238386.238493.

Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D.E. and Shneiderman, B. (1998). LifeLines: Using Visualization to Enhance Navigation and Analysis of Patient Records. *Elsevier eBooks*, pp.308–312. doi:https://doi.org/10.1016/b978-155860915-0/50038-x.

Rhyne, T.-M., Gotz, D. and Borland, D. (2016). *Data-Driven Healthcare: Challenges and Opportunities for Interactive Visualization*.

S, S., V, M. and V, L. (2023). Overview: PHP and MySQL Features for Creating Modern Web Projects. *openarchive.nure.ua*, [online] 7(1). Available at: https://openarchive.nure.ua/entities/publication/d32fc946-4bb2-4ad9-bce3-aac5f376b4eb.

Safwane, B. (2021). *COMPARISON OF D3.JS AND CHART.JS AS VISUALISATION TOOLS*.

Wahyuningsih, T. (2020). Problems, Challenges, and Opportunities Visualization on Big Data. *Journal of Applied Data Sciences*, 1(1), pp.20–28. doi:https://doi.org/10.47738/jads.v1i1.8.

West, V.L., Borland, D. and Hammond, W.E. (2014). Innovative information visualization of electronic health record data: a systematic review. *Journal of the American Medical Informatics Association*, [online] 22(2). doi:https://doi.org/10.1136/amiajnl-2014-002955.

Wikipedia Contributors (2025). *1854 Broad Street cholera outbreak*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/1854\_Broad\_Street\_cholera\_outbreak#/media/File:Snow-cholera-map-1.jpg.

World Health Organization (2023). *Trends in maternal mortality 2000 to 2020: estimates by WHO, UNICEF, UNFPA, World Bank Group and UNDESA/Population Division*. [online] www.who.int. Available at: https://www.who.int/publications/i/item/9789240068759.

Younas, A., Malik, None Khalil-ur-Rehman and Shahid, R. (2016). A detailed study on temporal data visualization techniques in electronic health records. [online] pp.638–643. doi:https://doi.org/10.1109/intech.2016.7845074.

# Table of Figures

[Figure 1 John Snow cholera map from 1854 (Wikipedia Contributors, 2025) 5](#_Toc197612710)

[Figure 2 Anscombe's Quartet (Fitzmaurice, 2017) 6](#_Toc197612711)

[Figure 3 Screenshot of a lifelines dashboard (Plaisant et al. (1996) 7](#_Toc197612712)

[Figure 4 System architecture diagram 13](#_Toc197612713)

[Figure 5 Dashboard design diagram 14](#_Toc197612714)

[Figure 6 Screenshot of the full deployed dashboard 15](#_Toc197612715)

[Figure 7 Code snippet showing the Navbar Component 16](#_Toc197612716)

[Figure 8 Screenshot of Navbar at the top of the dashboard, with navigation links for Home, Dashboard, and About. 16](#_Toc197612717)

[Figure 9 Code snippet showing the import and setup of the BarChart Component 17](#_Toc197612718)

[Figure 10 Code snippet showing the BarChart component logic for fetching a query 17](#_Toc197612719)

[Figure 11 DevTools screenshot showing the returned query results from the API 17](#_Toc197612720)

[Figure 12 Code Snippet showing the mapping logic of the returned API results from the requested query 17](#_Toc197612721)

[Figure 13 Data flow from user interaction to chart rendering 18](#_Toc197612722)

[Figure 14 Code snippet showing dataset configuration used to display antenatal visit percentages. This setup includes a secondary value (totals) for enhanced tooltip readability, enabling display of both percentage and absolute counts. 18](#_Toc197612723)

[Figure 15 Code Snippet showing the BarChart t rendering configuration with a custom tooltip logic that displays both percentage and raw counts per data point 19](#_Toc197612724)

[Figure 16 Code example of conditional chart rendering and context-aware descriptions. Chart content updates based on user-selected filters such as grouping category and date range. 20](#_Toc197612725)

[Figure 17 Screenshot of a fully rendered chart.js Barchart 20](#_Toc197612726)

[Figure 18 Code snippet of the ToggleAll useState hook 20](#_Toc197612727)

[Figure 19 Code snippet showing the toggleall button logic 21](#_Toc197612728)

[Figure 20 Comparison of chart appearance before and after using “Toggle All” 21](#_Toc197612729)

[Figure 21 Display of Chart.js integrated legend toggle feature 21](#_Toc197612730)

[Figure 22 Code snippet showing import statement for the dashboard, enabling the use of the required chart components. 22](#_Toc197612731)

[Figure 23 code snippet showing the dashboard state variables, as well as their default use state to use then the page is initially loaded 22](#_Toc197612732)

[Figure 24 Code snippet displaying handling logic for the primary group selection, the logic is consitent for other value selections within the application 22](#_Toc197612733)

[Figure 25 Code snippet of SelectBox component for user-driven filtering 23](#_Toc197612734)

[Figure 26 Code snippet JavaScript handler function logic for updating the selected demographic grouping 23](#_Toc197612735)

[Figure 27 Logic for constructing the GroupKey used in SQL queries. If a secondary grouping is selected, it concatenates both fields 24](#_Toc197612736)

[Figure 28 Code snippet displaying dynamically generated SQL query used for the LineChart component intergrating the user selected filters. 24](#_Toc197612737)

[Figure 29 DevTools screenshot of the constructed query sent to the API, based on the user input via the interactive dashboard 24](#_Toc197612738)

[Figure 30 Code snippet of conditional rendering of the LineChart component. The chart title and description dynamically update based on user-selected filters 25](#_Toc197612739)

[Figure 31 Screenshot of the user viewing a chart based on ethnic grouping and set timescale 25](#_Toc197612740)

[Figure 32 Image displaying the change in chart data, Titles and descriptions based on user selection. 26](#_Toc197612741)

[Figure 33 Code snippet of react useState hook managing chart visibility. The chartVisibility object tracks which visualisation components are currently displayed on the dashboard 26](#_Toc197612742)

[Figure 34 Code snippet of the ChartVisibilityDropdown component rendered on the dashboard 26](#_Toc197612743)

[Figure 35 Code snippet of the ChartVisibilityDropdown component rendered on the dashboard 27](#_Toc197612744)

[Figure 36 Code snippet of conditional rendering for the BarChart component 27](#_Toc197612745)

[Figure 37 Screenshot of ChartVisibilityDropdown in use 27](#_Toc197612746)

[Figure 38 Chart visibility example: Reduced chart selection 28](#_Toc197612747)

[Figure 39 API request activity captured in Microsoft Edge DevTools. 28](#_Toc197612748)

[Figure 40 Example API request payload, showing dynamically generated SQL query sent from the React frontend to Flask. 28](#_Toc197612749)

[Figure 41 Code snippet of Flask /api/query route for dynamic SQL execution 29](#_Toc197612750)

[Figure 42 Microsoft Edge DevTools Screenshot displaying multiple /query requests triggered by chart components 29](#_Toc197612751)

[Figure 43 Code snippet of Flask /api/motherv5 route for retrieving full dataset 30](#_Toc197612752)

[Figure 44 Database Schema 30](#_Toc197612753)

[Figure 45 Database ERD Screenshot from MySQL Workbench 31](#_Toc197612754)

[Figure 46 Screenshot of AWS Amplify hosting panel for the frontend 31](#_Toc197612755)

[Figure 47 Screenshot of AWS RDS Console hosting the maternity database 32](#_Toc197612756)

[Figure 48 Flask backend hosted on Render, showing deployment logs and status 32](#_Toc197612757)

[Figure 49 Screenshot of a list of GitHub commits for the project 33](#_Toc197612758)

[Figure 50 Questionnaire given to feedback participants. 34](#_Toc197612759)

[Figure 51 Task performance average scores from user feedback 35](#_Toc197612760)

[Figure 52 Participants ratings for the final question 38](#_Toc197612761)