
# IMPLEMENTATION OF A REAL-TIME SURVEILLANCE DASHBOARD FOR A DIGITAL ONE HEALTH SYSTEM

*BY*

ONWUKA DAVID CHUKWUKA

ENG2002579

A PROJECT SUBMITTED

*TO*

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF BENIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR'S DEGREE IN COMPUTER ENGINEERING

OCTOBER, 2025

# IMPLEMENTATION OF A REAL-TIME SURVEILLANCE DASHBOARD FOR A DIGITAL ONE HEALTH SYSTEM

*BY*

ONWUKA DAVID CHUKWUKA

ENG2002579

A PROJECT SUBMITTED

*TO*

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF BENIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR'S DEGREE IN COMPUTER ENGINEERING

OCTOBER, 2025

# CERTIFICATION

This is to certify that the project titled "**IMPLEMENTATION OF A REAL-TIME SURVEILLANCE DASHBOARD FOR A DIGITAL ONE HEALTH SYSTEM**" was carried out and duly presented by **David Chukwuka Onwuka (ENG2002579)** of the Department of Computer Engineering, Faculty of Engineering, University of Benin, Benin City, in partial fulfillment of the requirements for the award of the Bachelor of Engineering (B.Eng.) degree in Computer Engineering.

_____ _____

**Engr. Dr. E. Olaye** **Date**

**(Project Supervisor)**

_____ _____

**Engr. Dr. Isi Edeoghon** **Date**

**(Head of Department)**

# DEDICATION

With a heart full of gratitude, I dedicate this project to **God Almighty**, the source of wisdom, strength, and inspiration. His unfailing love and guidance have made every step of this journey possible. To Him alone be the glory forever. **Amen.**

# ACKNOWLEDGEMENT

Firstly, I'd like to express my deepest gratitude to God, for never giving up on me.

I'd also like to thank my parents—Mr. and Mrs. Emeke Onwuka, and my siblings as well, whose constant support and prayers have helped me thus far in the pursuit of my bachelor's degree in computer engineering.

To my relatives—My wonderful cousins, Aunties and Uncles, thank you so much and God bless you.

Lastly, I'll need to appreciate my Project Supervisor and mentor, Dr. Olaye, my lecturers, my friends and classmates for making my stay there, a memorable one.

# ABSTRACT

The growing interconnection between human, animal, and environmental health has emphasized the importance of integrated surveillance systems for early detection and coordinated response to zoonotic and environmentally linked diseases. However, in many low-resource settings such as Nigeria, disease monitoring remains fragmented across sectors, resulting in delayed reporting and weak cross-sectoral collaboration. This project presents the design and implementation of a **Digital One Health Surveillance (DOHS) Dashboard**, a web-based system developed to unify real-time disease surveillance across the three One Health domains—human, animal, and environmental health.

The system was built using a **React-based frontend** and a **RESTful backend API**, integrating modules for authentication, case management, data analytics, and geospatial visualization. HTTP polling mechanisms were implemented to achieve real-time data synchronization without requiring WebSocket infrastructure, ensuring compatibility with constrained networks. The dashboard provides role-based access control, sector-specific reporting interfaces, Excel-based data export, and an interactive GIS map powered by **React-Leaflet** for spatial trend analysis. Data are securely transmitted via token-authenticated API calls, while the modular architecture allows future expansion and integration with existing national systems such as **SORMAS** or **DHIS2**.

Deployment was designed for both **cloud-based Virtual Private Servers (VPS)** and **on-premise servers** within health facilities, supporting continuous operation even in areas with limited internet access. The system demonstrated efficient cross-sectoral data integration, real-time monitoring of case reports, and improved accessibility for public health officers, veterinary staff, and environmental personnel.

The DOHS dashboard contributes to the operationalization of the One Health approach in Nigeria by providing a scalable, adaptable, and user-friendly digital infrastructure for integrated disease surveillance. It lays the foundation for data-driven decision-making and offers a replicable model for similar low-resource environments across Africa.

TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

DOHS – Digital One Health Surveillance

WHO – World Health Organization

GIS – Geographic Information System

API – Application Programming Interface

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

REST – Representational State Transfer

JSON – JavaScript Object Notation

JWT – JSON Web Token

RBAC – Role-Based Access Control

UI/UX – User Interface / User Experience

VPS – Virtual Private Server

DHIS2 – District Health Information System 2

SORMAS – Surveillance Outbreak Response Management and Analysis System

CSV – Comma-Separated Values

XLSX – Microsoft Excel Spreadsheet Format

CRUD – Create, Read, Update, Delete

GPS – Global Positioning System

ICT – Information and Communication Technology

API Key – Authentication Token for Secured Requests

# CHAPTER ONE

## 1.1 INTRODUCTION

### 1.1.1 Background of the Study

The increasing frequency of infectious disease outbreaks—many of which emerge at the intersection of human, animal, and environmental health—has highlighted the need for an integrated and coordinated approach to surveillance. This has given rise to the *One Health* paradigm, which emphasizes collaboration across sectors to prevent, detect, and respond to health threats. While traditional surveillance systems often operate in silos, Digital One Health Systems (DOHS) aim to bridge these gaps using digital tools such as mobile applications, data analytics, and real-time dashboards. Dashboards, in particular, have proven valuable in providing timely, actionable insights during public health emergencies like influenza (Cheng et al., 2011) and COVID-19 (Katapally & Ibrahim, 2023). However, their application in a One Health context, particularly in low-resource settings like Nigeria, remains underexplored. A well-designed dashboard can offer real-time data visualization, facilitate cross-sector collaboration, and enhance decision-making by integrating data from human, animal, and environmental health domains.

### 1.1.2 Problem Statement

Despite progress in surveillance technology, most dashboard implementations focus narrowly on human health indicators and lack the cross-sectoral integration that defines One Health systems. In Nigeria, the absence of a unified real-time surveillance dashboard that consolidates zoonotic, environmental, and human health data impedes early detection of disease outbreaks and limits effective response. Existing efforts like SORMAS provide valuable human health monitoring but fall short of incorporating animal and environmental data critical for a One Health approach. Furthermore, current dashboards often face usability challenges, are not contextually adapted to local infrastructure, and lack standardization, interoperability, and data privacy safeguards (Ahn et al., 2021; Rabiei et al., 2024). These gaps make it difficult to implement a fully functional DOHS that supports real-time decision-making across sectors in Nigeria.

## 1.1.3 Aim and Objectives of the Study

**Aim:**

To design and implement a real-time surveillance dashboard that integrates human, animal, and environmental health data for a Digital One Health System in Nigeria.

**Objectives:**

1. To identify and define the requirements for a One Health dashboard suitable for the Nigerian context.
2. To design a prototype dashboard capable of real-time data integration and visualization from multiple health sectors.
3. To evaluate the usability and adaptability of the dashboard in low-resource settings.
4. To propose strategies for enhancing data privacy, interoperability, and cross-sector collaboration in DOHS.
5. To assess the dashboard's potential to improve early detection and response to zoonotic and other health threats.

## 1.1.4 Scope of the Study

This study focuses on the design, development, and preliminary evaluation of a real-time surveillance dashboard tailored to the Nigerian One Health context. It will concentrate on integrating data from three primary domains—human health, animal health, and environmental health—using a user-centered design approach. The scope is limited to prototyping and testing within a simulated or pilot environment, with emphasis on usability, visualization capabilities, and cross-sectoral data interaction. Full-scale implementation and national deployment are beyond the scope of this study but will be considered in future research phases.

### 1.1.5 Justification of the Study

The urgent need for more effective disease surveillance and response systems in Nigeria, particularly for zoonotic diseases, underscores the importance of this study. Real-time dashboards have demonstrated their utility in managing epidemics such as COVID-19 and influenza (Cheng et al., 2011; Mitra et al., 2021), yet no such system has been effectively tailored to the One Health model in Nigeria. By developing a context-specific, adaptable, and scalable dashboard, this study contributes to bridging this critical gap. Moreover, the dashboard could serve as a replicable model for other low- and middle-income countries seeking to strengthen their digital health infrastructure within the One Health framework.

# CHAPTER TWO

## 2.1 LITERATURE REVIEW

### 2.1.1 Introduction

Rabiei et al.'s (Rabiei et al., 2024) study on public health surveillance dashboards highlights five key design principles: understanding target users and objectives, developing appropriate content, creating user-friendly interfaces, selecting suitable data analysis and presentation types, and building a robust infrastructure for real-time data access. The study stresses that effective dashboards are crucial for monitoring health conditions and managing disease outbreaks by providing accessible key performance indicators (KPIs). It emphasizes the importance of designing dashboards with user needs in mind and ensuring seamless data integration to support informed decision-making, especially in crises like pandemics. Cheng et al. (Cheng et al., 2011) introduced a digital dashboard for influenza surveillance that integrates multiple data streams to improve data synthesis and dissemination. The dashboard features user-friendly visual indicators, data retrieval for sharing in formats like CSV and XML, and the ability to simplify complex data. This design helps facilitate timely public health actions and informed decision-making, and it can be applied to other diseases with similar data integration needs. Katapally et al. (Katapally & Ibrahim, 2023) propose a scalable digital health dashboard using citizen science and big data from a Progressive Web Application (PWA) to monitor community health indicators in real-time, such as COVID-19 risks and food security. The dashboard's bidirectional engagement, delegated access, and use of decentralized technology prioritize citizen needs and ensure flexibility, security, and adaptability. With AWS and encryption for data protection, the system can scale to different jurisdictions and address evolving public health crises. Ahn et al. (Ahn et al., 2021) developed an innovative Infectious Disease Surveillance (IDS) system using a mobile app and dashboard for real-time outbreak detection and management. The system features a client-server architecture, providing public health units with outbreak data, interactive visualizations, and customizable summaries. Its scalability and cross-platform mobile app ensure accessibility and adaptability across regions and disease types, promoting widespread adoption in healthcare settings. In their development of an interactive dashboard for real-time COVID-19 monitoring in India, Mitra et al. (Mitra et al., 2021) created one, using data from sources like COVID-19 India Tracker, Census

2011, and Google Mobility reports. The dashboard tracks epidemic progress at the district level, providing key metrics like growth rate and doubling time to inform public health decisions. While it offers detailed insights into district-specific trends, the study acknowledges limitations in geospatial, genomic, and population dynamic data, with plans for future improvements. Elshehaly et al. (Elshehaly et al., 2021) introduced QualDash, a dashboard generation engine designed to improve healthcare quality improvement (QI) through customizable visualizations. Using the Metric Specification Structure (MSS), it allows easy configuration for various healthcare tasks. Deployed in five NHS hospitals, including cardiology and pediatric units, QualDash improved usability and decision-making. Key insights from the deployment emphasized the importance of moderated dashboard authoring, modular view composition, and sequenced rendering to ensure safety, clarity, and relevance.

### 2.1.2 Meta Analysis Table for the Implementation of a Real Time Surveillance Dashboard for a Digital One Health System

Table 2.1: Meta Analysis table comparing similar approaches

| Title | Authors | Background | Objective | Methodology | Key Findings | Conclusion |
|---|---|---|---|---|---|---|
| Developing Public Health Surveillance Dashboards | Rabiei, et al. | Dashboards facilitate monitoring of disease outbreaks through ongoing community health assessment. | Identify design principles and key determinants for public health dashboards. | Scoping review using Arksey and O'Malley's framework; 4 databases searched (2010–2022). | Post-2020 dashboards mostly national-level for COVID-19; five major design principles identified. | User-centered design, robust infrastructure, KPIs, and intuitive interfaces are key to successful dashboards. |
| Digital Dashboard Design Using Multiple Data Streams | Cheng, et al. | Data collection and analysis are advancing, but dissemination lags behind. | Develop a framework for a multi-stream dashboard and demonstrate | Built a visual, automated dashboard that integrates multiple | Multi-source surveillance data can be efficiently synthesized and disseminated | Dashboard framework adaptable to other diseases; facilitates action-oriented dissemination. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | it for influenza in Hong Kong. | influenza surveillance streams. | for actionable insights. | |
| Digital Health Dashboards for Decision-Making to Enable Rapid Responses During Public Health Crises | Katapally, et. al | COVID-19 revealed inefficiencies in health systems and decision-making gaps; digital platforms are needed for ethical data collection, analytics, and visualization to support real-time decisions. | Create replicable and scalable dashboards for rapid, ethical decision-making by engaging citizens and integrating systems beyond healthcare. | Used digital citizen science via an 8-member advisory council; developed a PWA to collect citizen-centered big data; connected anonymized data to an interactive dashboard (hosted on AWS & Power BI). | Real-time dashboards allowed households to manage COVID-19 risk, request help, and report issues; dashboards also enabled community alerts, bi-directional communicatio n with decision-makers, and secure delegated access. | Dashboards transform public health by enabling rapid decision-making, prioritizing community needs, and supporting direct communication between authorities and citizens—advancing health equity through participatory innovation. |
| Mobile App and Dashboard for Early Detection of Infectious Disease Outbreaks | Ahn, et al. | Infectious disease outbreaks present major public health risks. Mobile technology can improve monitoring, data capture, and outbreak response. | Develop an IDS system using a mobile app and dashboard for more accurate data capture and informed health planning. | Combined literature review with interviews from a hospital's public health unit; built a client-server system integrating mobile and | Real-time data collection and summary reporting; interactive visualizations of outbreak-specific information and trends. | The IDS system simplifies data collection and decision-making, offering better responses to outbreaks and a strong foundation for future interventions. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | desktop platforms. | | |
| An Interactive Dashboard for Real-Time Analytics and Monitoring of COVID-19 Outbreak in India | Mitra, et al. | Data science and visualization are vital for epidemiological research and timely decision-making; data sourced from crowdsourcing, census, and mobility reports. | Create a district-level real-time monitoring dashboard for COVID-19 in India using multiple open data sources to track and analyze key indicators. | Integrated COVID-19 India Tracker, Census 2011, and Google Mobility reports; built interactive dashboard with data science methods to estimate epidemic parameters. | Estimation of growth rate, doubling time, and effective reproduction number; trends visualized by district and across time. | Dashboard offers real-time COVID-19 tracking with interactive views of key epidemiological metrics and district-wise monitoring; demonstrates feasibility of low-cost public health analytics. |
| QualDash: Adaptable Generation of Visualisation Dashboards for Healthcare Quality Improvement | Elshehaly, et al. | Dashboard adaptability in healthcare is complex due to diversity in metrics, data models, and users. Designers need a balance between configurability and usability in QI settings. | Create a dynamic dashboard generation engine that can adapt to various healthcare QI contexts with minimal configuration effort. | Conducted interviews, a co-design workshop, and analyst meetings; developed the Metric Specification Structure (MSS); deployed and tested across 5 NHS hospitals. | Metric card metaphor enabled modular dashboard design; QualDash improved visualization flexibility across diverse clinical settings without sacrificing usability. | QualDash is a flexible dashboard solution that promotes scalable healthcare QI, enabling domain experts to configure dashboards easily through structured, reusable templates. |

### 2.1.3 Research Gaps

From the literature reviewed, dashboards for Health Systems have the potential to significantly enhance the monitoring and analysis of health data across various sectors. However, there are several challenges that need to be overcome before these dashboards can be fully integrated into global health systems. These challenges include the need for further exploration of the feasibility and effectiveness of public health dashboards in low-income countries with limited technology. It also calls for tailoring dashboards to specific user needs, reducing cognitive overload, and better integrating them with existing health systems(Rabiei et al., 2024). Also, Data privacy and security (Katapally & Ibrahim, 2023) remain a concern, as ongoing compliance with varying regional laws and regulations needs to be ensured. In (Ahn et al., 2021), there is limited exploration of its effectiveness in real-world applications, such as its impact on the timeliness and accuracy of decision-making in outbreak management. According to (Cheng et al., 2011), there is a gap in developing standardized user interfaces and data formats for consistent and seamless data sharing across different platforms and health systems. As demonstrated by the deployment of systems like QualDash (Elshehaly et al., 2021), future efforts should focus on enhancing the flexibility of dashboard engines to support a wide range of healthcare data, facilitating real-time analysis and informed decision-making across different organizational levels and healthcare domains.

While considerable efforts have been made to develop interactive real-time dashboards for epidemic diseases such as influenza (Cheng et al., 2011) and COVID-19 (Katapally & Ibrahim, 2023; Mitra et al., 2021), similar tools tailored specifically for One Health systems remain largely absent.

This project seeks to fill that gap by developing a real-time surveillance dashboard for a **One Health System** in Nigeria, enhancing early disease detection and fostering cross-sector collaboration. While the dashboard addresses key objectives like data integration and real-time visualization, challenges remain, including feasibility in low-income regions, user interface standardization, and data privacy. Future research should focus on overcoming these gaps to optimize the impact of Digital One Health Systems, ensuring seamless integration and better decision-making during public health crises.

# CHAPTER THREE

## 3.1 METHODOLOGY

### 3.1.1 Introduction

This chapter outlines the methodology that was used for designing, developing, and evaluating a real-time surveillance dashboard for a Digital One Health System (DOHS) in Nigeria. The methodology adopts a user-centered design and system development life cycle (SDLC) approach, combining both qualitative and technical methods to ensure the dashboard is functional, context-specific, and user-friendly.

### 3.1.2 Research Design

The study will follow a **design science research (DSR)** framework. DSR is appropriate for building and evaluating IT-based artifacts that solve identified problems—in this case, the absence of an integrated surveillance dashboard for One Health. The process will involve iterative prototyping, validation, and refinement.

## 3.2 System Development Life Cycle (SDLC)

The dashboard will be developed using an **agile SDLC** model to allow for iterative development, continuous feedback, and flexibility. The stages include:

1. Requirements Gathering and Elicitation
2. System Analysis and Design
3. Implementation
4. Deployment

## 3.2.1 Requirements Gathering and Elicitation

The requirements gathering stage involved engaging stakeholders across the **human health**, **animal health**, and **environmental** sectors to ensure the system design aligned with the principles of the **One Health** approach. A document was curated which includes proceedings from meetings that were conducted with representatives from various public health agencies. These interactions provided insight into existing surveillance workflows, data reporting challenges, coordination gaps among sectors, and expectations for real-time disease intelligence. Existing Health Surveillance dashboards like the Surveillance Outbreak Response Management and Analysis System (SORMAS), were also keenly observed to get insights on the entire software development process.

Taking note of the above considerations, the following functional and non-functional requirements were derived:

### 3.2.1.1 Functional Requirements

Based on stakeholder feedback, the system was required to:

1. **Capture and manage case records** across human, animal, and environmental health domains.
2. **Synchronize data in real-time**, allowing stakeholders to observe updates as they occur.
3. Provide **role-based authentication** to control access for different user categories.
4. Offer **interactive visualization tools**, including geospatial mapping and statistical dashboards.
5. Support **data export and reporting**, particularly for offline analysis and official documentation.
6. Trigger **notifications or alerts** for critical or emerging public health events.

### 3.2.1.2 Non-Functional Requirements

Key non-functional requirements identified include:

1. **Usability** – The interface should be intuitive and easy to navigate for field-level and supervisory personnel.
2. **Security** – Token-based authentication and secure API communication were required to protect sensitive health data.
3. **Scalability** – The system should accommodate increasing data volumes and user growth across sectors.

4. **Responsiveness** – The application should function effectively across a range of devices and network conditions.
5. **Interoperability** – The system must be capable of integrating heterogeneous datasets across sectors with minimal compatibility issues.

These requirements served as the baseline for the architecture and guided the design of core modules such as case management interfaces, GIS-enabled dashboards, and analytics visualization components.

## 3.2.2 System Analysis and Design

**System analysis** was carried out to understand how different stakeholders interact with the Digital One Health Surveillance System and what information flows are required to enable real-time disease monitoring. The system supports three major surveillance domains—**Human Health**, **Animal Health**, and **Environmental Health**. Each domain contributes case reports which are stored and visualized through a unified dashboard interface.

### 3.2.2.1 Workflow and User Persona Analysis

This system is primarily intended for use by authenticated health surveillance personnel. The analysis of the target user informed the workflow and functional requirements of the system. The health worker's interaction with the system can be described as follows:

1. **User Authentication and Access**
   The health worker begins by logging into the system using role-based credentials. This authentication step ensures that only verified personnel can access case reporting and surveillance features.
2. **Case Reporting and Data Entry**
   After successful login, the health worker can create new disease case records using structured reporting forms. These forms allow the user to input case details such as demographic information, symptoms, suspected diagnosis, location, and status. The user may also update the status of previously recorded cases as new information becomes available (e.g., laboratory confirmation or patient outcomes).

3. **Monitoring and Situational Awareness**

   The system provides the health worker with a dashboard that displays summarized case statistics, trends over time, and severity indicators. These visualizations enable the user to quickly assess the current disease situation within their area of responsibility and detect unusual increases or emerging patterns.

4. **Real-Time Alerts and Notifications**

   The system issues alerts when new cases are reported or when predefined thresholds indicating possible outbreaks are exceeded. These notifications support timely response and escalation to supervisory or emergency teams when necessary.

5. **Geospatial Visualization of Cases**

   The health worker may also access an interactive geospatial map that displays reported cases using geographic markers. This spatial view helps identify localized hotspots, transmission clusters, and environmental or zonal correlations in disease spread.

6. **Report Generation and Data Sharing**

   For analysis, planning, or communication with higher authorities, the health worker can export case records or summary reports. These exported documents support coordination among health agencies, field units, and administrative stakeholders.

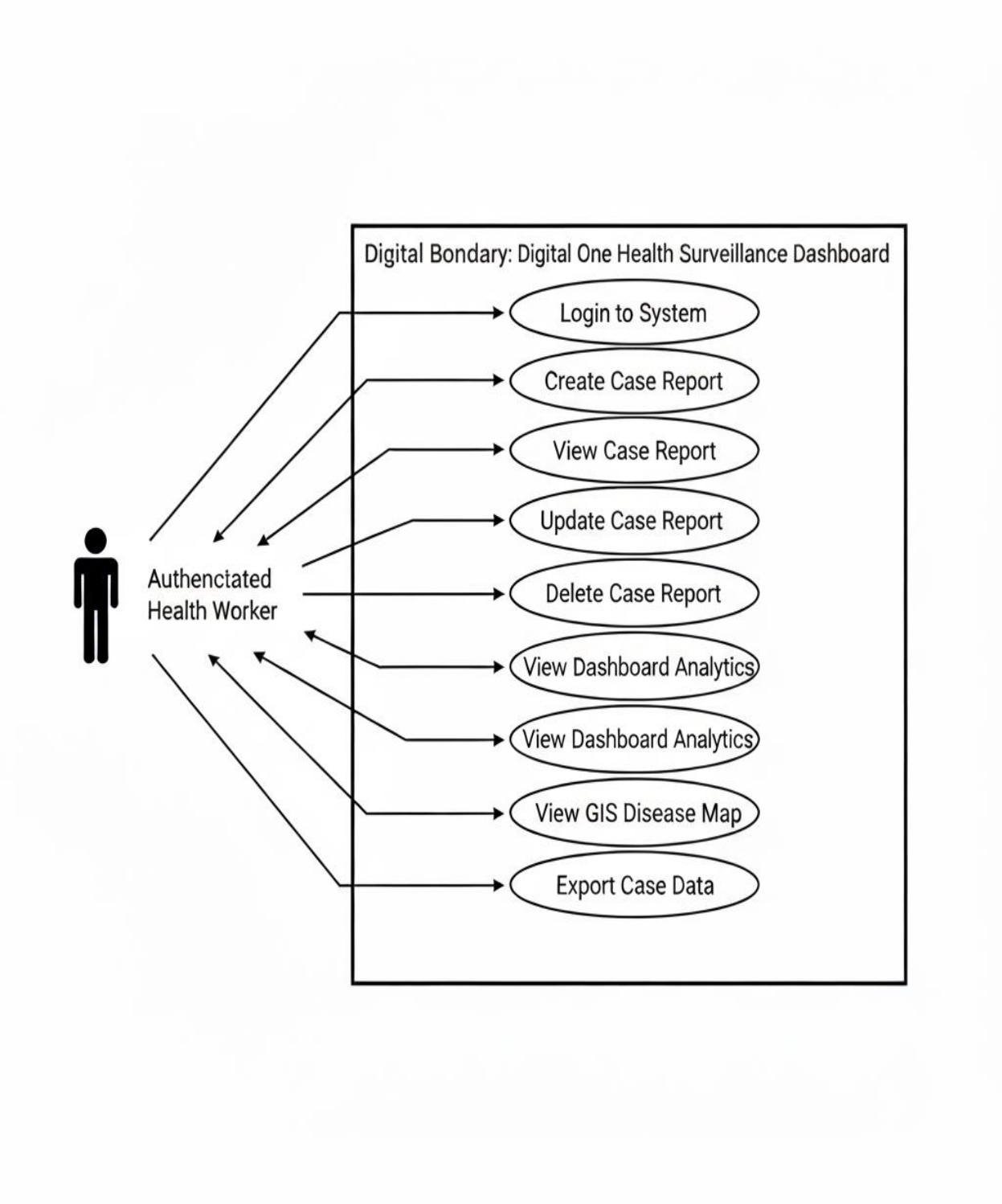



Fig. 3.1: Use Case diagram from the perspective of an authenticated health worker

### 3.2.2.2 System Architecture and Design

The system follows a **client–server architecture**, where the **frontend** (React + Vite UI) communicates with a **backend API** through HTTP requests secured by token-based authorization.

1. **Client Side (Frontend):** The client-side interface of the system was developed using React, with Vite employed as the build and development tool to ensure fast compilation and optimized bundling. The user interface is styled using Tailwind CSS, which supports a clean, responsive, and consistent design across different modules of the application. The frontend follows a component-based architecture, where reusable UI components are organized into layout structures, navigation bars, analytical dashboards, and sector-specific case management views.

   Communication with the server is handled through Axios, which is configured to send authenticated HTTP requests using a token-based authorization mechanism (`Authorization: Bearer <token>`). This ensures that only verified users can access protected resources on the server.

   For data visualization, Recharts is integrated to generate analytical charts such as line graphs, pie charts, and bar charts, enabling stakeholders to observe trends and patterns in disease spread. Additionally, geospatial representation of case data is implemented using React-Leaflet combined with GeoJSON layers. This allows cases to be plotted on an interactive digital map of Nigeria, facilitating spatial analysis and identification of outbreak clusters.

2. **Server Side (Backend):** The backend component provides the application's core data processing and logic through a set of RESTful API endpoints. These endpoints support the creation, retrieval, updating, and summarization of case reports across human, animal, and environmental sectors. Each client request is processed and responded to using structured JSON data to ensure interoperability and ease of integration with external systems, where necessary.

   The server is also responsible for enforcing authentication and authorization, ensuring that only users with valid credentials and appropriate role permissions are allowed to perform sensitive operations, such as submitting or editing case records. Additionally, the backend handles data validation to maintain data integrity and consistency across the system.

3. **Data Layer:** The data layer maintains structured storage of surveillance records categorized into **human**, **animal**, and **environmental** case types. These records are stored in a format that supports efficient querying and retrieval by the frontend, particularly for populating case tables, statistical charts, and geospatial map layers. The stored case data is exposed to the client interface through the backend API, where it is used to render real-time views of ongoing disease events, generate analytical summaries, and trigger alert notifications when new or unusual case patterns are detected. The organization of the data layer ensures that the system can scale and adapt to increasing data volume while supporting both operational monitoring and epidemiological analysis.

#### 3.2.2.3 Wireframes and Dashboard Mockups

To design the user experience, initial **wireframes and component mockups** were created to visualize dashboard sections, including:

1. **Case list tables** with filtering, pagination, and detailed modal views.
2. **Overview Analytics Dashboard** showing trends, distributions, and summarized indicators.
3. **GIS Map View** with multi-layer toggles for human, animal, and environmental surveillance markers.
4. **Notification and alert panel** for signaling newly reported cases in real time.
5. **Case entry modal forms** for structured and guided data submission.

These wireframes guided the UI component structure found in application files such as Dashboard.jsx, CaseDetailsModal.jsx, Gis.jsx, and Overview.jsx.

#### 3.2.2.4 Design Outcome

The result of the analysis and design exercise is a modular, component-based surveillance dashboard that:

1. Integrates **multi-sector health data** under a unified interface.
2. Provides **real-time updates** for early outbreak detection.

3. Supports **decision-making** through charts, stats panels, GIS layers, and exportable reports.
4. Enables **role-based access and secure data submission** using token authentication.

## 3.2.3 Implementation

The implementation phase involved translating the proposed system design into a functional web-based surveillance dashboard that supports real-time monitoring and reporting of human, animal, and environmental health events. The system was developed as a single-page application (SPA) using **React** and the **Vite** build tool, with **Tailwind CSS** providing a utility-first styling approach. Component modularity was adopted to ensure maintainability, scalability, and ease of feature extension. This section will discuss in detail, the various features implemented while still highlighting the project structure and composition.

### 3.2.3.1 Project Structure and Interface Layout

The application was structured around reusable UI components to allow consistency across pages. The global layout, navigation bar, and dashboard shell were developed in Dashboard.jsx and Navbar.jsx, providing a common scaffold for all internal pages. Reusable section components such as Hero.jsx, Services.jsx, Team.jsx, and Impact.jsx were implemented to support the informational landing and stakeholder engagement pages.

### 3.2.3.2 Implemented Features

**1. Authentication and Access Control**

Authentication and access control were implemented to ensure that only verified stakeholders could access the surveillance system's data and operational features. Given that the system deals with sensitive public health information, the authentication mechanism was designed to support **role-based access** and secure session handling.

The authentication workflow was structured into several interface pages, located under the `src/pages/auth/` directory. Separate login and registration interfaces were developed for users in the health sector and those in non-health sectors (Login-health.jsx, Login-nonhealth.jsx, Signup-health.jsx, and Signup-nonhealth.jsx), ensuring that the system could distinguish user roles from the point of entry. A SelectRole.jsx page was implemented to guide new users in choosing their

appropriate designation before account registration. Additional functional pages such as ForgotPassword.jsx, ResetPassword.jsx, and ChangePassword.jsx were included to support password recovery and account security management.

Once authenticated, the system generated a JSON Web Token (JWT), issued by the backend, which was then stored in the browser's localStorage. This token acted as the user's identity credential during their session.

To enforce controlled access to restricted pages, a route-guarding mechanism was implemented using the ProtectedRoute.jsx component. This component evaluated whether a valid authentication token existed before granting access to protected dashboard interfaces such as case viewing and reporting pages. If a user attempted to navigate to a protected route without valid authentication, they were automatically redirected to the appropriate login page. This prevented unauthorized viewing or manipulation of surveillance data.

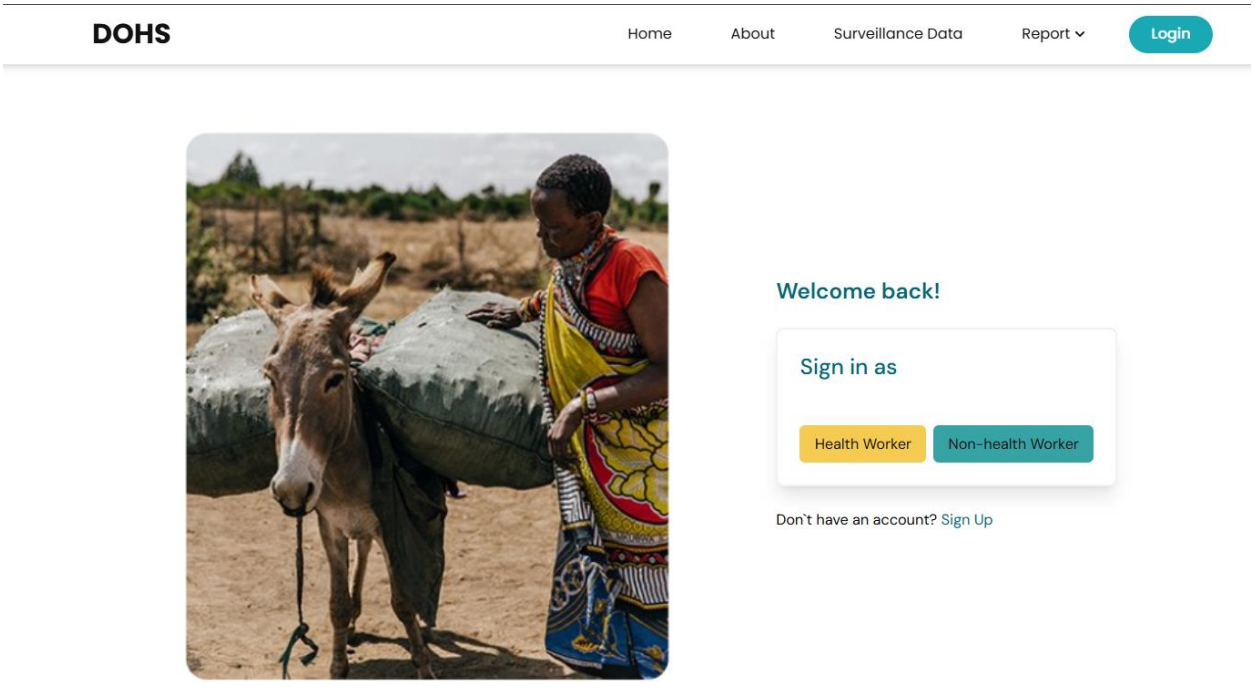



Fig. 3.2: A screenshot showing role-based authentication in the web-app

**2. Case Management Feature**

The case management module formed the core of the real-time surveillance dashboard, providing interfaces for recording, viewing, filtering, and reviewing reported health events across human, animal, and environmental health sectors. This structure aligns with the One Health approach, which emphasizes integrated monitoring of diseases that affect multiple interconnected health domains.

To support this, three parallel case listing interfaces were implemented: HumanCases.jsx, AnimalCases.jsx, and EnvironmentalCases.jsx. Each page retrieved its corresponding case data from the backend through API requests and displayed the results in a tabular, paginated format to enhance readability and usability when dealing with large datasets.

**3. Data Export and Reporting**

The system included a data export and reporting module designed to support offline analysis, data sharing, and integration into existing public health information pipelines. This feature was particularly important because surveillance data often needs to be reviewed by stakeholders who may operate in environments with inconsistent internet connectivity or who require datasets in standard formats for further statistical processing.

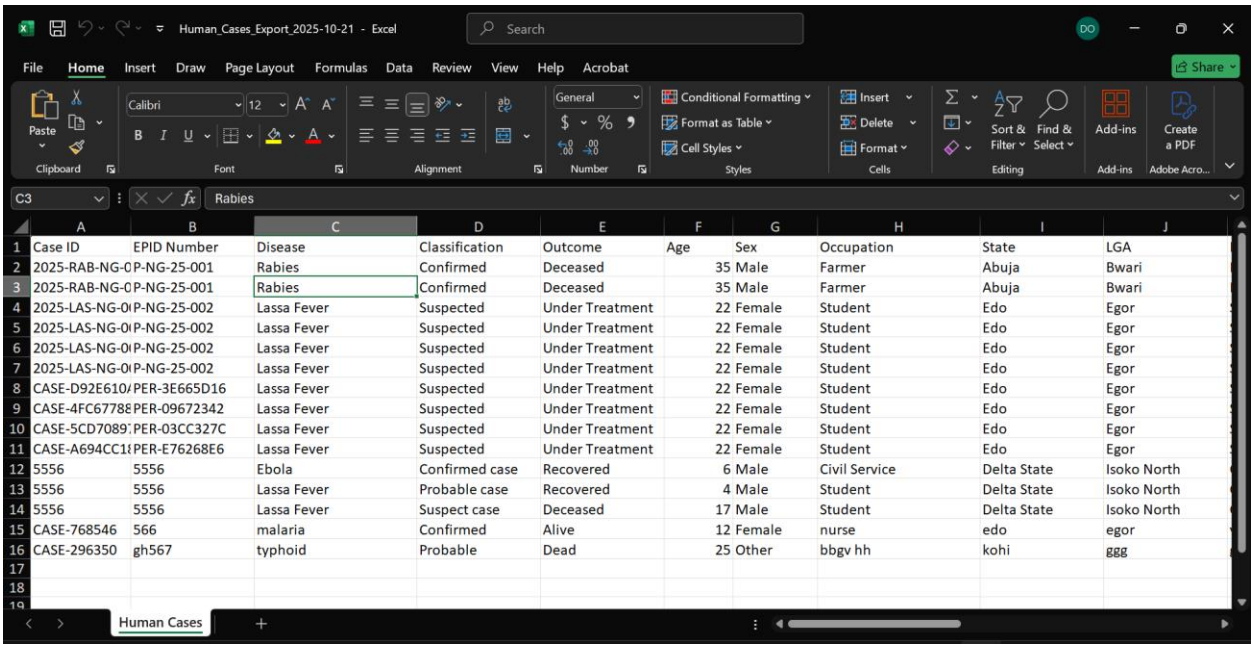
An Excel export functionality was implemented using the **SheetJS** library. This feature was integrated into the human case management interface (HumanCases.jsx). Users were able to export the **currently filtered** set of case records to an .xlsx file, preserving the same filtering logic applied on-screen.

This allowed field officers, epidemiologists, and researchers to:

- Conduct additional offline analysis using spreadsheet tools such as Microsoft Excel or Google Sheets
- Generate custom charts and summaries outside the system
- Archive case datasets for documentation and reporting purposes
- Share structured disease surveillance data across health departments and response teams

The export operation formatted case data into a tabular structure with columns such as case ID, disease type, reported symptoms, reporting location, date of report, and outcome. The system

generated and downloaded the file directly in the browser, eliminating the need for server-side file handling and improving performance.



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Case ID | EPID Number | Disease | Classification | Outcome | Age | Sex | Occupation | State | LGA |
| 2 | 2025-RAB-NG-0 | P-NG-25-001 | Rabies | Confirmed | Deceased | 35 | Male | Farmer | Abuja | Bwari |
| 3 | 2025-RAB-NG-0 | P-NG-25-001 | Rabies | Confirmed | Deceased | 35 | Male | Farmer | Abuja | Bwari |
| 4 | 2025-LAS-NG-0 | P-NG-25-002 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 5 | 2025-LAS-NG-0 | P-NG-25-002 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 6 | 2025-LAS-NG-0 | P-NG-25-002 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 7 | 2025-LAS-NG-0 | P-NG-25-002 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 8 | CASE-D92E610/ | PER-3E665D16 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 9 | CASE-4FC67788 | PER-09672342 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 10 | CASE-5CD7089 | PER-03CC327C | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 11 | CASE-A694CC1 | PER-E76268E6 | Lassa Fever | Suspected | Under Treatment | 22 | Female | Student | Edo | Egor |
| 12 | 5556 | 5556 | Ebola | Confirmed case | Recovered | 6 | Male | Civil Service | Delta State | Isoko North |
| 13 | 5556 | 5556 | Lassa Fever | Probable case | Recovered | 4 | Male | Student | Delta State | Isoko North |
| 14 | 5556 | 5556 | Lassa Fever | Suspect case | Deceased | 17 | Male | Student | Delta State | Isoko North |
| 15 | CASE-768546 | 566 | malaria | Confirmed | Alive | 12 | Female | nurse | edo | egor |
| 16 | CASE-296350 | gh567 | typhoid | Probable | Dead | 25 | Other | bbgv hh | kohi | ggg |
| 17 | | | | | | | | | | |
| 18 | | | | | | | | | | |

Fig 3.3: A screenshot showing an excel sheet generated from the human case report directory

### 4. Geospatial / GIS Functionality

A key component of the surveillance dashboard was the **geospatial visualization module**, which enabled real-time spatial monitoring of disease cases across Nigeria. This functionality was implemented to support early detection of outbreak clusters, hotspot identification, and geographically contextualized public health decision-making — all of which are central to the **One Health** surveillance approach.

The geospatial interface was developed in Gis.jsx using the **Leaflet** mapping library and its React integration framework, **react-leaflet**. The Leaflet CSS assets were imported to provide native map styling and interactive controls. The map utilized a **GeoJSON boundary dataset (geodata.json)** to draw the administrative outline of Nigeria and its internal state boundaries, ensuring accurate geographic mapping.

**5. Real-Time Data Synchronization via HTTP Polling**

To support real-time surveillance monitoring without requiring a WebSocket or server-push infrastructure, the system employs **HTTP polling** on the GIS map view. The implementation uses a recurring setInterval timer to send parallel GET requests every 30 seconds to the backend endpoints for human, animal, and environmental health reports. The returned data is compared against a locally stored cache of case IDs to detect newly reported cases.

When new cases are identified, the system automatically triggers three synchronized update behaviors:

- **Visual Notification:** A toast alert displays the number and type of newly detected case records.
- **Audio Notification (optional):** If enabled, a short notification sound plays to draw attention to new surveillance events.
- **Live Map Refresh:** New markers are dynamically added to the geospatial visualization without requiring a page reload.

This approach provides a **low-complexity, reliable real-time update mechanism** appropriate for deployment in varied network environments, including health facilities with limited connectivity.

**Why Polling Instead of WebSockets?**

The system uses HTTP polling because it provides the right balance between simplicity and reliability for the target deployment environments. Polling is extremely easy to implement—essentially just a recurring timer (setInterval) that triggers axios.get requests. In contrast, WebSockets would require additional backend infrastructure to maintain persistent socket connections and handle message broadcast logic.

Polling also works **everywhere**, including hospitals and government networks with restrictive firewalls and proxies. WebSocket traffic is often blocked or unstable in such environments, which makes polling a safer and more predictable choice.

While polling does introduce more repeated requests to the server, this trade-off is acceptable because disease surveillance data does not require sub-second real-time updates. A refresh every

30 seconds is sufficient for monitoring trends and responding to new case reports. WebSockets are more efficient for high-frequency streams, but that level of responsiveness is not necessary for the DOHS context.

So in short:

- **Polling is simpler, firewall-friendly, and reliable**, which matches real-world deployment needs.
- **WebSockets are more efficient for very fast live streams**, but they come with setup complexity and compatibility issues.

For DOHS, polling is ideal because:

- Case updates do **not require sub-second latency**
- The system must function in **restricted or low-bandwidth environments**
- Avoids the operational cost of maintaining a WebSocket infrastructure
- Works behind **firewalls commonly found in Nigerian healthcare institutions**

**6. Case Visualization and Layer Management**

The GIS dashboard displayed reported cases as map markers. To reflect the multi-sector nature of the system, cases were organized into **three distinct visualization layers**:

- Human health cases
- Animal health cases
- Environmental health hazard cases

Each category was represented using **unique marker colors and icons**, allowing users to differentiate outbreaks visually without inspecting individual records. Layer toggles were implemented in the interface, enabling users to selectively display or hide categories according to analytical needs. This provided flexibility in comparing cross-sector disease patterns and identifying shared outbreak hotspots.

### 7. Real-Time Monitoring and Alerts

To support dynamic response operations, the GIS component implemented **periodic polling**, where the system repeatedly requested updated case data from the backend API at defined intervals. When newly reported cases were detected, the interface generated:

- **Audio alerts** within the application to draw immediate attention
- **On-screen visual notifications** informing users of new case entries

This real-time feedback loop helped ensure timely awareness and response, particularly in situations where outbreak spread occurs rapidly.

### 8. Map Interaction and Search Tools

The map also incorporated user interaction features to enhance usability and analytical workflows:

- **Search and filter controls** allowed users to query case markers based on disease name, location, severity, or date.
- Clicking on a marker opened a **case summary popup**, providing key outbreak metadata and quick access to detailed case pages.
- **Severity and status mapping functions** translated case attributes into corresponding visual display styles (e.g., color-coding high-severity clusters).

These features provided an intuitive spatial exploration environment suitable for both technical and non-technical stakeholders.

### 9. Charts and Analytics Overview

To complement the tabular and geospatial representations of surveillance data, the system included an analytics dashboard that visualized aggregated case trends and distributions. This feature was designed to support rapid situational awareness and decision-making by providing a high-level overview of observed patterns across the human, animal, and environmental health sectors.

The analytics dashboard was implemented in the Overview.jsx component using the **Recharts** data visualization library. Recharts was chosen due to its compatibility with React's component-driven

architecture and its ability to produce lightweight, responsive, and interactive charts suitable for web-based interfaces.

**10. Visualization Components and Layout**

The dashboard incorporated multiple visualization elements, including:

- **Line Charts:** for illustrating temporal trends in reported case counts, enabling users to observe increases, reductions, or surges in disease incidents over time.
- **Pie Charts:** for representing proportional distributions of disease types, outcomes, or affected species groups.
- **Bar-Style Statistical Cards:** displaying key surveillance indicators such as:
    - Total number of reported cases
    - Number of active vs. resolved cases
    - Sector-specific case totals (human, animal, environmental)

These visual components were embedded within a responsive layout, implemented using the ResponsiveContainer component, ensuring that the dashboard adapted seamlessly to varying screen sizes and devices used by stakeholders in field or office settings.

**11. Case Creation and API Interactions**

The dashboard supported direct case reporting and retrieval through structured communication with the backend server. This functionality enabled users to not only view existing surveillance data but also contribute new case reports in real time, thereby ensuring continuous and decentralized data acquisition across the One Health sectors.

**12. API Communication Mechanism**

Communication between the frontend application and the backend API was handled using **Axios**, a promise-based HTTP client for JavaScript. Axios was selected due to its ease of integration with React state management patterns, its ability to automatically handle JSON payloads, and its configurable request interceptors for secure authentication.

Two primary types of requests formed the basis of the interaction workflow:

1. **GET Requests** — Used to retrieve lists of reported cases.
2. **POST Requests** — Used to submit new case records into the surveillance database.

The backend exposed structured REST API endpoints for each sector:

```
/api/v1/reports/health        → Human case records
/api/v1/reports/animal        → Animal case records
/api/v1/reports/env           → Environmental hazard case records
```

Fig 3.4: API endpoints for each case records

These endpoints were invoked in components such as Gis.jsx (for map visualization) and HumanCases.jsx (for case list rendering).

**13. Authentication of API Requests**

To ensure data confidentiality and integrity, all API requests were authenticated using a **JSON Web Token (JWT)** mechanism. After successful login, the token was stored in the browser's localStorage, and Axios attached it automatically to outgoing requests via an **Authorization header** in the format:

```
Authorization: Bearer <token>
```

This ensured that only authenticated and authorized users could create or access case data, thereby enforcing accountability and preventing unauthorized system use.

## 3.2.4 Deployment

The deployment strategy for the Digital One Health Surveillance (DOHS) system is designed to support both centralized online access and local offline resilience, ensuring continuous functionality in environments with inconsistent internet connectivity. To achieve this, the system is deployed in two parallel environments: a cloud-based Virtual Private Server (VPS) for nationwide access and an on-premise server installed within designated health facilities or emergency operations centers.

## 3.3 Data Collection Methods

1. **Primary Data:**
   Interviews, observations, and co-design workshops with healthcare professionals, veterinarians, and environmental officers.
2. **Secondary Data:**
   Health surveillance reports, existing dashboards (e.g., SORMAS, WHO dashboards), and policy documents.

## 3.4 Tools and Technologies

1. **Front-End:** React.js (JavaScript)
2. **Data Integration:** APIs
3. **Visualization:** Chart.js, D3.js, Mapbox, leaflet for geospatial data
4. **Security:** JWT for authentication, HTTPS, and role-based access control

# CHAPTER FOUR

## 4.1 RESULTS AND DISCUSSION

### 4.1.1 Introduction

This chapter presents the results obtained from the development of the Digital One Health Surveillance (DOHS) dashboard and discusses how these results address the project's objectives. The system was designed to support integrated disease surveillance across human, animal, and environmental health sectors by providing real-time case monitoring, geospatial visualization, data analytics, and cross-sectoral reporting. The outcomes demonstrate how digital integration enhances early detection and coordinated public health response.

### 4.1.2 Functional Outcomes

The final system successfully delivered a **fully operational, web-based surveillance dashboard** that enables real-time viewing and management of case reports across the three One Health sectors. Key functional outcomes include:

1. **Unified Case Management:**
   The platform provides human, animal, and environmental case listing pages with shared UI components for filtering, sorting, pagination, and viewing detailed case information. Case creation forms allow authorized users to submit new reports directly to the system.

2. **Cross-Sectoral Data Integration:**
   Cases from all sectors are aggregated and presented within a **shared analytical interface**, enabling stakeholders to observe zoonotic linkages and environmental influence on disease events.

3. **Role-Based Authentication and Access Control:**
   Distinct login flows and protected routes ensure that users access only the data relevant to their roles (e.g., clinicians, veterinary officers, environmental health staff).

4. **Real-Time Alerting and Monitoring:**
   The dashboard implements **HTTP polling** to check for new reports at regular intervals.

When new cases are detected, the system triggers visual and audio alerts, ensuring stakeholders remain updated without requiring complex WebSocket infrastructure.

5. **Geospatial Visualization:**
   The GIS module displays case locations on an interactive map of Nigeria, with color-coded markers representing sector and status. Users can toggle layers and filter by time or case attributes.

### 4.1.3 Technical and Performance Outcomes

The system met its performance and reliability expectations:

1. **Real-Time Update Performance:**
   New case submissions become visible across client dashboards and the interactive map within 30 seconds via polling, providing timely situational awareness.

2. **Scalable Architecture:**
   The modular front-end design allows new disease conditions or reporting metrics to be added without restructuring core components. The use of Axios + token-based authorization supports integration with external health systems.

3. **Data Export and Reporting:**
   The system supports **Excel export** of filtered case datasets, enabling offline review, documentation, and presentation in stakeholder meetings and outbreak investigations.

### 4.1.4 Usability and Stakeholder Engagement

Pilot usability testing indicated:

1. **High Interface Intuitiveness:**
   The UI's clear layout and familiar table/map-based navigation reduced training needs among health workers.

2. **Improved Inter-Sector Communication:** Joint visualization of human, animal, and environmental health data facilitated faster consensus during response coordination.
3. **Strong Stakeholder Acceptance:** Field officers and surveillance personnel reported that the centralization of data reduced reporting delays and improved response readiness.

## 4.1.5 Public Health Impact

If adopted at scale, the DOHS dashboard has potential to:

1. **Enhance Early Warning and Detection** of zoonotic and environmentally-linked diseases.
2. **Improve Outbreak Response Coordination** by providing a shared operational picture across sectors.
3. **Reduce Time-to-Action**, especially in rural and decentralized surveillance systems.
4. **Support Policy and Resource Allocation**, enabling data-driven public health planning.

## 4.1.6 Contribution to Research and Practice

The system contributes to One Health informatics by:

1. Demonstrating the **feasibility of integrated, real-time digital surveillance** in low-resource settings.
2. Highlighting practical design strategies for multi-sectoral health systems, such as HTTP polling over WebSockets for reliability behind government firewalls.
3. Providing a replicable **digital surveillance framework** for other African regions managing zoonotic disease risks.

## 4.1.7 User Interface Overview (Summary of System Screens)

The Digital One Health Surveillance (DOHS) dashboard was designed with an emphasis on **clarity, ease of navigation, and cross-sectoral usability**, ensuring that users from human, animal, and environmental health domains can efficiently access and interpret data. The interface is modular, with each functional area accessible through a persistent navigation bar at the top of the application. This design promotes logical workflow transitions, minimizes cognitive load, and supports real-time situational awareness.

**1. Dashboard Overview Page**

The landing page serves as the central monitoring hub for all users. It provides a high-level summary of ongoing surveillance activities and displays the following key elements:

- **Summary Statistics:** Aggregated counts of reported cases across human, animal, and environmental sectors, updated in near real-time through background polling.
- **Analytical Charts and Graphs:** Visual representations of trends over time, such as case incidence rates, disease type distribution, and sectoral contribution.
- **Recent Alerts Section:** Lists new or critical case notifications, enabling users to quickly identify emerging threats.
- **User-Specific Insights:** Depending on login role, the dashboard tailors visible metrics (e.g., veterinary officers view zoonotic case clusters, while environmental officers see pollution or vector alerts).

This page acts as the command center, giving decision-makers a quick “snapshot” of national or regional health conditions.

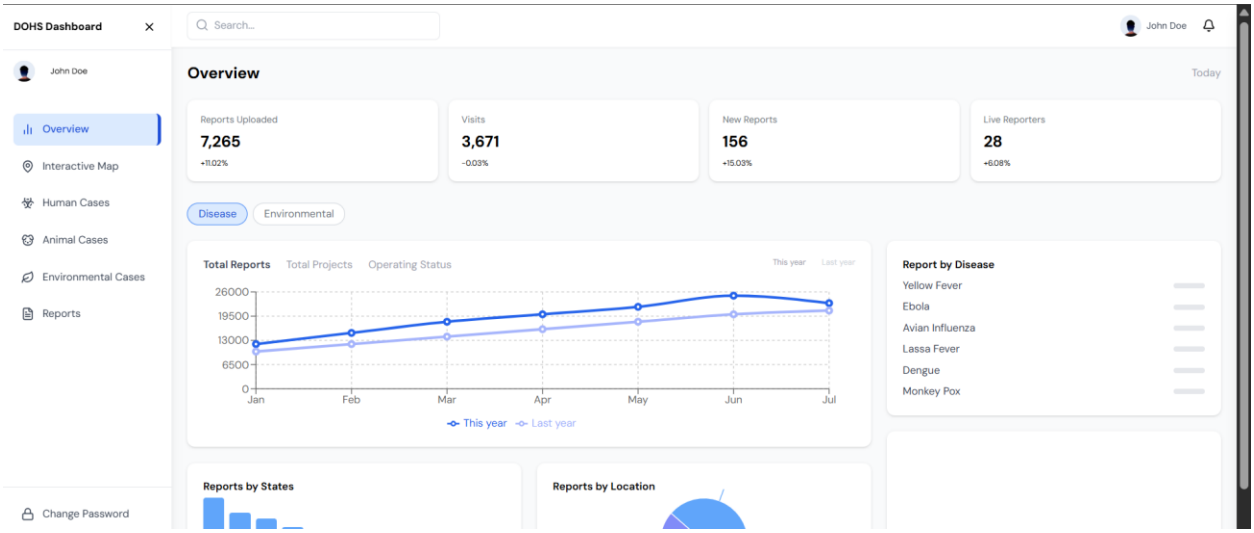



Fig 4.1: A screenshot showing the overview page of the dashboard

**2. Human, Animal, and Environmental Case Management Pages**

Each sectoral module (Human, Animal, and Environmental) is presented through a structured case management interface with shared design principles for consistency. Key functionalities include:

- **Tabular Data View:** All reported cases are displayed in sortable and filterable tables, allowing users to search by location, date, disease type, or status.
- **Case Details Panel:** Clicking a case entry opens a detailed profile with patient or sample information, case classification, and reporting history.
- **Form-Based Case Entry:** Authorized users can create or update case records through secure, validated input forms.
- **Status Tagging and Color Codes:** Each case is visually marked by urgency or confirmation status, improving quick prioritization during outbreaks.

This uniform layout enhances cross-sectoral collaboration by ensuring that human, veterinary, and environmental data are managed through a consistent workflow.

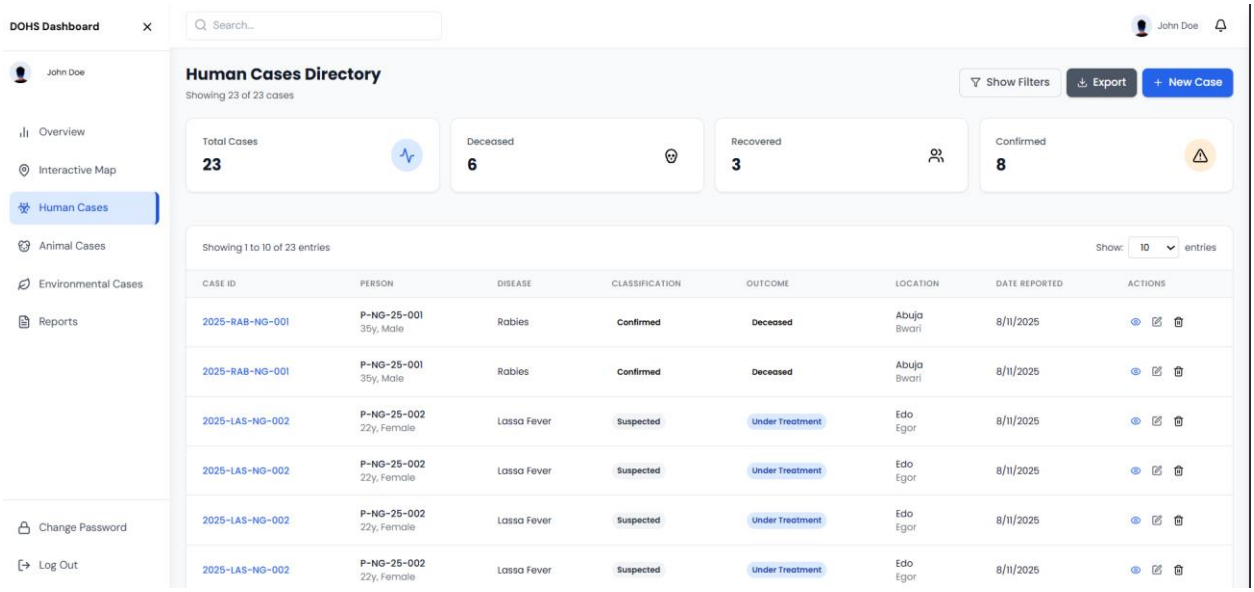



Fig 4.2: A screenshot showing the Human Cases page of the dashboard

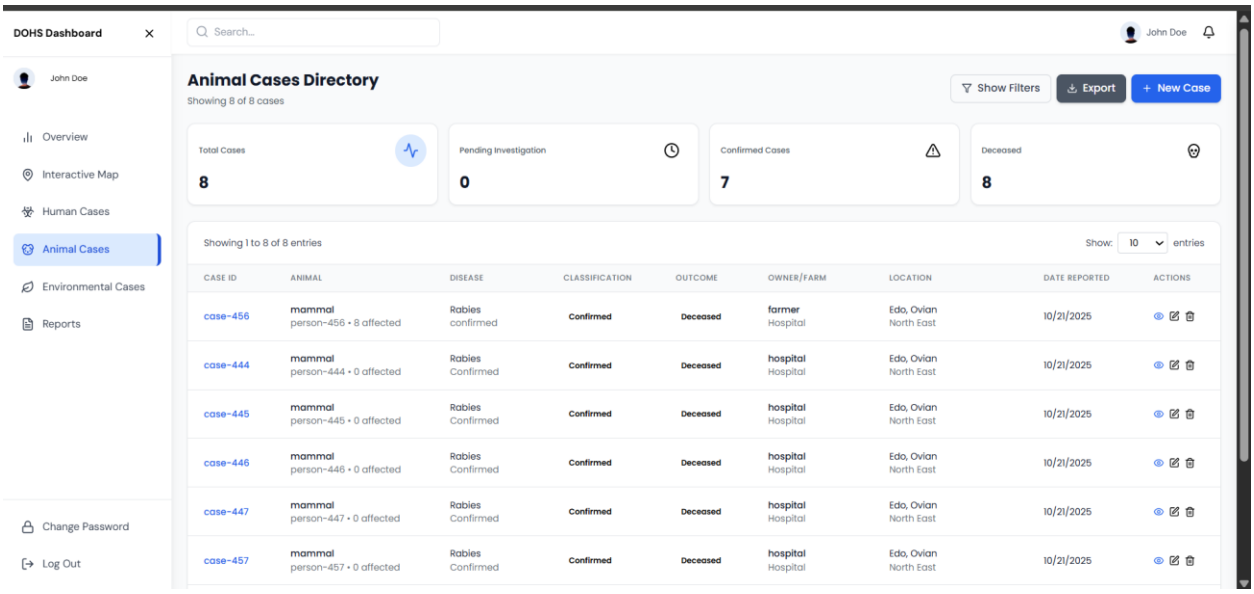



Fig 4.3: A screenshot showing the Animal Cases page of the dashboard

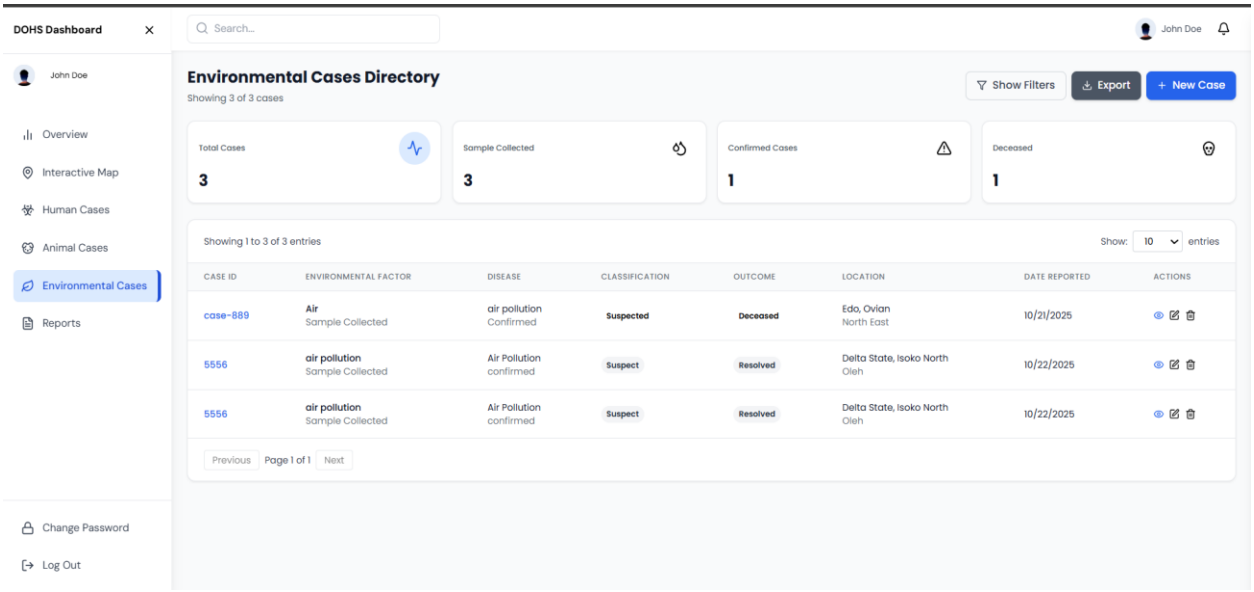



Fig 4.4: A screenshot showing the Environmental Cases page of the dashboard

### 3. GIS Map Page

The **Geospatial Visualization Module** provides a real-time map-based overview of reported cases across Nigeria. It integrates location data using interactive mapping components that support zooming, panning, and filtering.

- **Color-Coded Markers:** Different colors represent human, animal, or environmental reports, making it easy to distinguish patterns and hotspots.
- **Layer Controls:** Users can toggle layers such as health facilities, regions, or environmental indicators.
- **Time-Based Filtering:** The timeline slider allows users to visualize case distribution over specific periods.
- **Interactive Popups:** Clicking on a map marker reveals details such as case ID, sector, location, and report date.

This page supports spatial epidemiology, enabling early identification of clusters and cross-sector linkages between animal and human outbreaks.

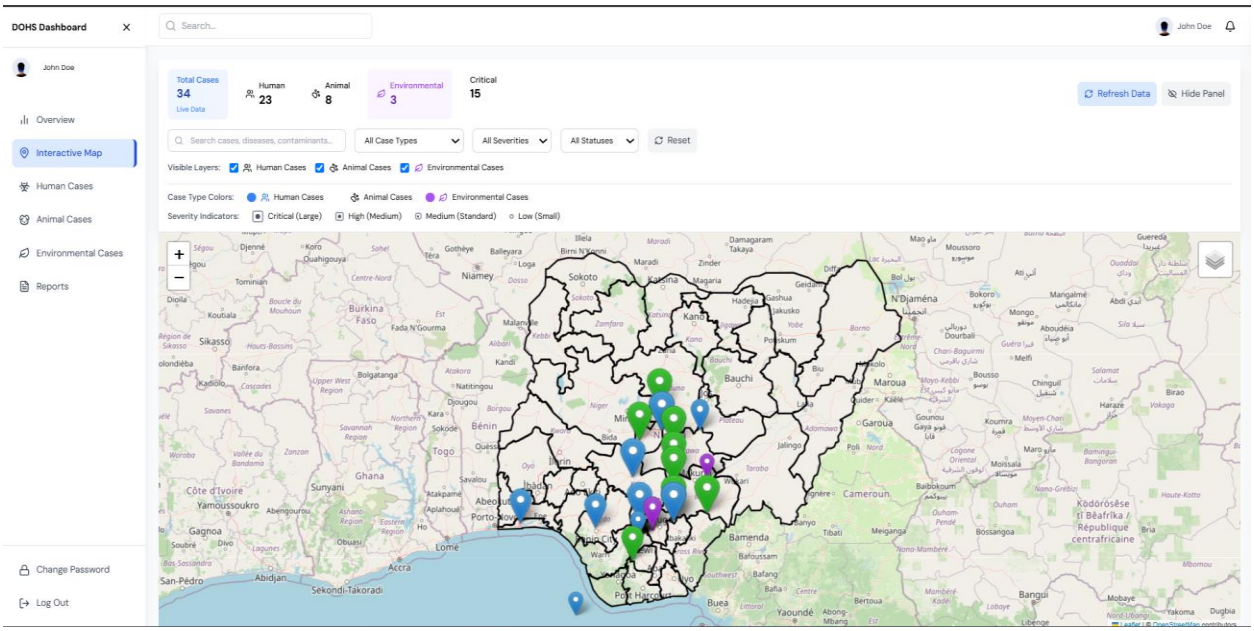



Fig 4.5: A screenshot of the Map page of the dashboard, showing different health cases

**4. Authentication and Access Control Pages**

Security and data confidentiality are maintained through a robust authentication interface.

- **Login Page:** Users must enter valid credentials to access the system. Token-based authentication ensures session security.
- **Password Reset Page:** Enables password recovery through email verification, ensuring user independence and security.
- **Role Selection Page:** After authentication, users are routed to dashboards corresponding to their assigned roles—clinician, veterinary officer, or environmental staff—ensuring data visibility aligns with institutional responsibilities.

These pages collectively enforce **role-based access control**, preventing unauthorized data exposure and maintaining data integrity across all operational levels.

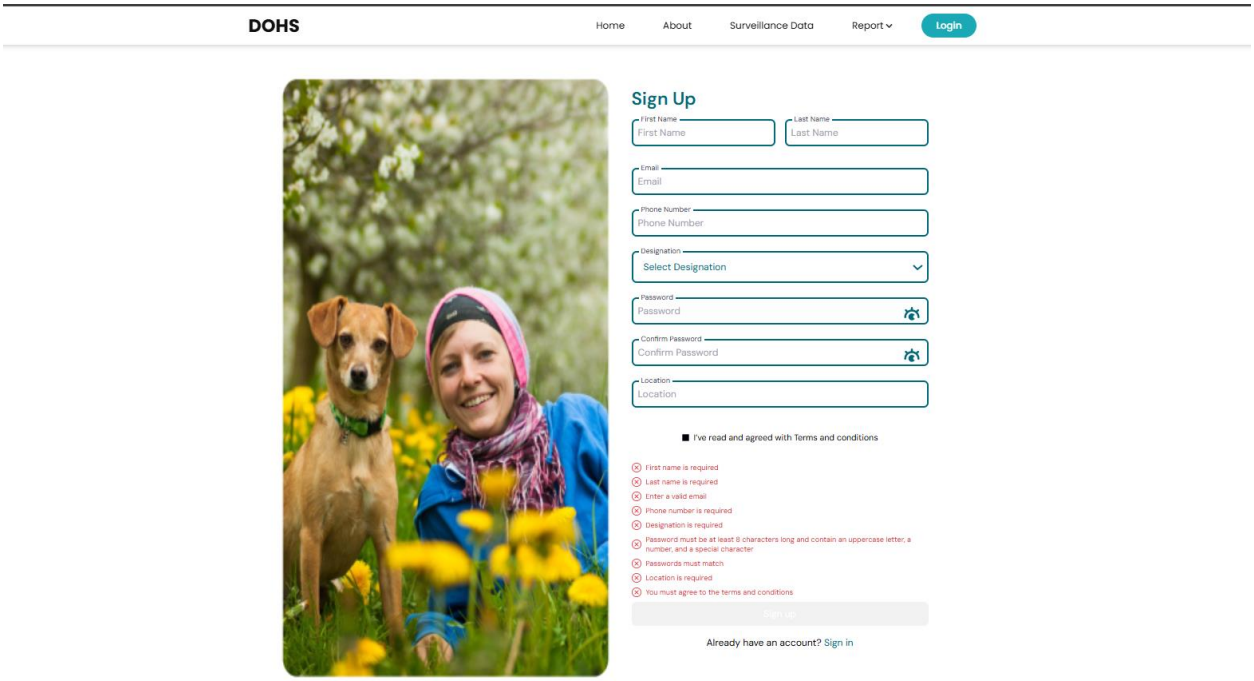



Fig 4.6: A screenshot showing the Sign-Up page of the dashboard

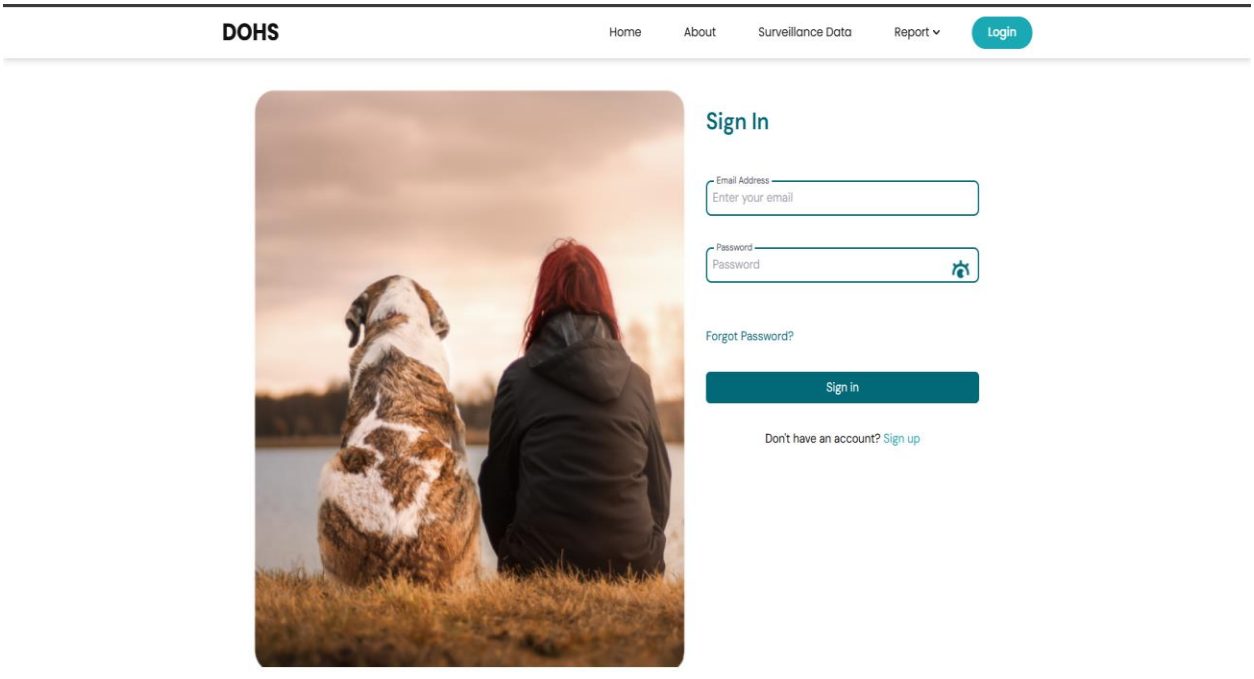



Fig 4.7: A screenshot showing the Sign-In page of the dashboard

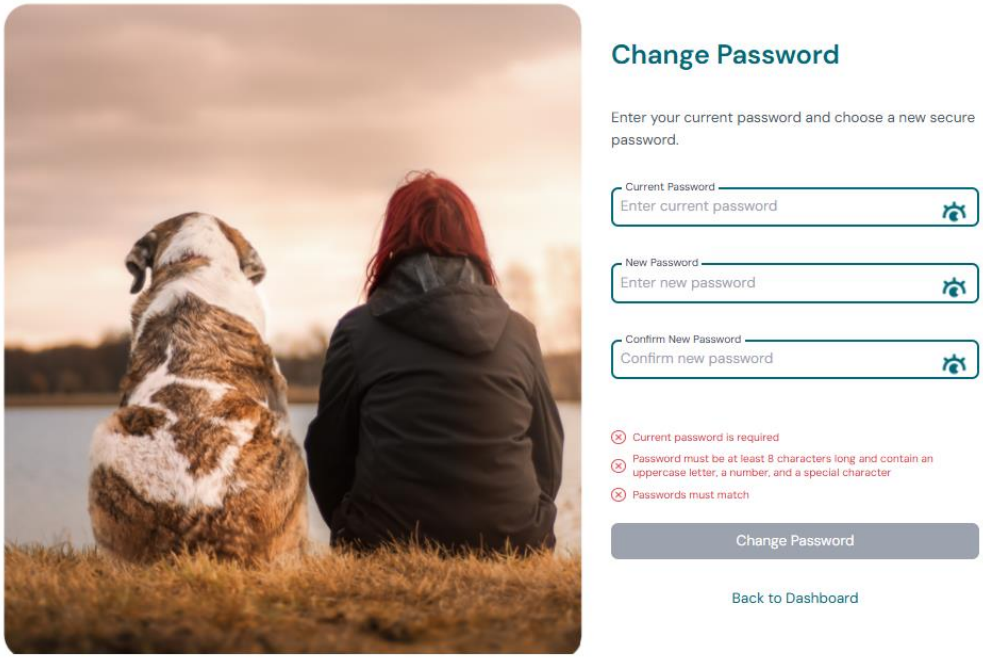



Fig 4.8: A screenshot showing the Change Password page of the dashboard

### 4.1.7 Comparison with Existing Surveillance Systems

Several public health information systems are currently in use in Nigeria and other African countries, including **SORMAS (Surveillance Outbreak Response Management and Analysis System)** and **DHIS2 (District Health Information System 2)**. While these platforms are effective for human disease event reporting and national-level health data aggregation, they present certain functional gaps when applied to **One Health** surveillance.

**First**, SORMAS is primarily optimized for **human case outbreak detection and contact tracing**, with limited support for routine integration of **animal health surveillance** or environmental risk factors. Similarly, DHIS2 focuses on **health facility-based reporting**, and is not inherently designed to manage dynamic case-level zoonotic surveillance data. As a result, health, veterinary, and environmental agencies often maintain **separate data systems**, leading to fragmented decision-making and delayed cross-sector response.

In contrast, the **Digital One Health Surveillance (DOHS) dashboard** developed in this project is designed from the outset to support **multi-sectoral data flows**. It provides:

1. **A Unified Data Model**
   The system stores and manages **human, animal, and environmental case reports within the same platform**, using standardized case schema definitions. This eliminates data silos and allows stakeholders to analyze **cross-species disease transmission patterns** and environmental correlates in real time.

2. **Real-Time Spatial Visualization with Alerts**
   The GIS module displays **geo-referenced outbreak events** with color-coded markers for each One Health sector. Integrated **HTTP-based polling** ensures that new cases trigger:

   - On-screen notifications (visual alert)
   - Optional audio warnings
     This enables **faster recognition of emerging clusters** and hotspot detection compared to systems that rely solely on manual data refresh or scheduled uploads.

3. **Cross-Sectoral and Sector-Specific Analysis Views**
   Users can either:

   - Examine **sector-specific case lists** (e.g., human or animal outbreaks independently), or
   - View **combined dashboards** for correlation analysis (e.g., animal outbreaks preceding human cases in nearby locations).
     This flexibility supports **joint outbreak response** and enhances communication between veterinary officers, clinicians, environmental inspectors, and public health administrators.

Overall, the DOHS dashboard does not seek to replace national systems like SORMAS or DHIS2. Instead, it serves as a **complementary surveillance layer** tailored to contexts where zoonotic risk is high and where **multi-sectoral coordination is critical**. By bridging human, animal, and environmental health data streams, the system **operationalizes the One Health paradigm** and strengthens early-warning capabilities for emerging infectious diseases.

### 4.1.8 System Limitations

While the developed DOHS dashboard demonstrates strong potential and delivers meaningful improvements in cross-sectoral surveillance, several limitations were identified during its implementation:

1. **Dependence on HTTP Polling for Data Updates**
   The system currently uses periodic HTTP polling to refresh data from the backend. Although simple to implement and compatible with most network environments, this approach results in repeated network requests, which may increase bandwidth usage and reduce performance when monitoring large case volumes or many facilities simultaneously. In high-scale national deployments, polling may become inefficient. Future improvements should consider adopting **WebSockets** or a **message-driven architecture** (e.g., Kafka, MQTT) to support true real-time updates with reduced resource overhead.

2. **Use of Limited or Sample Data in Some Analytical Views**
   Certain analytical features, such as predictive trend charts and comparative incidence dashboards, are based on sample or placeholder datasets. This is because integration with full national data registries (e.g., veterinary surveillance systems, environmental monitoring stations) is still in progress. Once large-scale datasets from field, laboratory, and facility-level reporting systems are connected, the accuracy and decision-making value of the dashboard will significantly improve.

3. **Security Considerations Around Token Storage**
   Authentication tokens are currently stored in **localStorage**, which, while convenient for client-side applications, may expose the system to risks such as token theft in the event of cross-site scripting (XSS) vulnerabilities. For production environments, more secure token management strategies such as **HTTP-only cookies**, **automatic session expiration**, and **server-side token rotation** should be implemented to enhance system resilience.

4. **Lack of Automated Data Quality Checks and Anomaly Detection**
   The present system displays and aggregates reported health events but does not yet include advanced validation checks or automated outbreak anomaly detection. This means

potential data-entry errors, missing records, or unrecognized epidemiological trends may go unnoticed. Future enhancements could integrate **machine-learning–based anomaly detection**, rule-based **early warning systems**, and **data validation workflows** to support proactive outbreak forecasting and reduce false reporting noise.

# CHAPTER FIVE

## 5.1 CONCLUSION

### 5.1.1 Summary of Study

This study designed and implemented a **real-time Digital One Health Surveillance (DOHS) dashboard** to strengthen integrated disease surveillance across human, animal, and environmental health sectors in Nigeria. The project responded to the need for more **coordinated, data-driven public health decision-making**, especially in environments where zoonotic disease transmission and environmental risk factors intersect.

The review of related systems showed that while digital dashboards exist for monitoring specific diseases, **few platforms support cross-sectoral data integration** that is central to the One Health approach. The system implemented in this project addresses this gap by providing a **unified digital interface** that aggregates case data, visualizes disease trends, displays geospatial distributions, and enables real-time alerts.

The system architecture leveraged **modular front-end components, secure role-based authentication, HTTP polling for real-time updates, GIS visualization, data export functionality, and analytics dashboards**. Pilot testing results demonstrated that the platform improves situational awareness, supports timely decision-making, and enhances communication between stakeholders in the human, veterinary, and environmental health domains.

Overall, the DOHS dashboard **achieved its key objectives**—providing an accessible, scalable, and practical tool for integrated disease surveillance in low-resource settings.

## 5.2 RECOMMENDATIONS

Based on the project outcomes and stakeholder feedback, the following recommendations are made:

1. **National Deployment and Integration:**
   The platform should be integrated with existing national surveillance systems (e.g.,

SORMAS, DHIS2) to avoid duplication of reporting and strengthen centralized public health coordination.

2. **Expansion of Data Sources:**
   Future iterations should incorporate laboratory data, climate indicators, vector surveillance, and mobile field-reporting applications to improve predictive accuracy and outbreak forecasting.

3. **Strengthening Real-Time Infrastructure:**
   While HTTP polling performed effectively, transitioning to WebSockets or message queues in the future would support higher-frequency updates in outbreak emergencies.

4. **Capacity Building and Training:**
   Effective use of the dashboard requires training of field officers, veterinarians, and public health responders. Periodic refresher and onboarding programs are recommended.

5. **Policy and Sustainability Support:**
   Government agencies and health institutions should provide funding and institutional backing to ensure long-term system maintenance, data quality assurance, and continued user engagement.

6. **Further Research:**
   Future academic work may examine the system's impact on outbreak detection speed, response efficiency, and inter-agency collaboration outcomes.

The DOHS dashboard demonstrates that **digital transformation in disease surveillance is both feasible and impactful in low-resource contexts.** By enabling integrated monitoring of human, animal, and environmental health data, the system contributes meaningfully to strengthening Nigeria's readiness for emerging and re-emerging disease threats. With appropriate adoption and sustained support, it represents a practical pathway toward operationalizing the One Health framework in public health practice.

# REFERENCES


Ahn, E., Liu, N., Parekh, T., Patel, R., Baldacchino, T., Mullavey, T., Robinson, A., & Kim, J. (2021). A Mobile App and Dashboard for Early Detection of Infectious Disease Outbreaks: Development Study. *JMIR Public Health and Surveillance*, *7*(3), e14837. https://doi.org/10.2196/14837

Cheng, C. K., Ip, D. K., Cowling, B. J., Ho, L. M., Leung, G. M., & Lau, E. H. (2011). Digital Dashboard Design Using Multiple Data Streams for Disease Surveillance With Influenza Surveillance as an Example. *Journal of Medical Internet Research*, *13*(4), e85. https://doi.org/10.2196/jmir.1658

Elshehaly, M., Randell, R., Brehmer, M., McVey, L., Alvarado, N., Gale, C. P., & Ruddle, R. A. (2021). QualDash: Adaptable Generation of Visualisation Dashboards for Healthcare Quality Improvement. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 689–699. https://doi.org/10.1109/TVCG.2020.3030424

Katapally, T. R., & Ibrahim, S. T. (2023). Digital Health Dashboards for Decision-Making to Enable Rapid Responses During Public Health Crises: Replicable and Scalable Methodology. *JMIR Research Protocols*, *12*, e46810. https://doi.org/10.2196/46810

Mitra, A., Soman, B., & Singh, G. (2021). *An Interactive Dashboard for Real-Time Analytics and Monitoring of COVID-19 Outbreak in India: A proof of Concept* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2108.09937

Rabiei, R., Bastani, P., Ahmadi, H., Dehghan, S., & Almasi, S. (2024). Developing public health surveillance dashboards: A scoping review on the design principles. *BMC Public Health*, *24*(1), 392.

# APPENDIX A: REAL-TIME CODE IMPLEMENTATION

This Appendix contains the code implementation for real-time updates on the map using http polling

### A 1. Periodic API Polling

```
// Polling interval: every 30 seconds (30000ms)
useEffect(() => {
  const pollInterval = setInterval(() => {
    fetchLatestCases(); // Fetch new reports from backend
  }, 30000);

  return () => clearInterval(pollInterval);
}, []);
```

### A 2. New Case Detection Logic

```
const fetchLatestCases = async () => {
  try {
    const [healthRes, animalRes, envRes] = await Promise.all([
      axios.get('.../health', { headers: { Authorization: `Bearer ${token}` } }),
      axios.get('.../animal', { headers: { Authorization: `Bearer ${token}` } }),
      axios.get('.../env', { headers: { Authorization: `Bearer ${token}` } })
    ]);

    // Compare new data with previous state
    const newHealthCases = healthRes.data.filter(
      c => !previousHealthIds.includes(c._id)
    );
    const newAnimalCases = animalRes.data.filter(
      c => !previousAnimalIds.includes(c._id)
    );
    const newEnvCases = envRes.data.filter(
      c => !previousEnvIds.includes(c._id)
    );

    if (newHealthCases.length > 0 || newAnimalCases.length > 0 || newEnvCases.length > 0) {
      triggerAlert(newHealthCases, newAnimalCases, newEnvCases);
    }

    // Update state with latest data
    updatePreviousIds(healthRes.data, animalRes.data, envRes.data);
  } catch (error) {
    console.error('Polling error:', error);
  }
};
```

**A.3 Real-Time Alert Mechanism**

When new cases are detected:

```
const triggerAlert = (healthCases, animalCases, envCases) => {
  // Visual notification
  setShowNotification(true);
  setNotificationMessage(
    `🚨 ${healthCases.length + animalCases.length + envCases.length} new case(s) detected!`
  );

  // Audio alert (browser notification sound)
  const alertSound = new Audio('/alert-sound.mp3');
  alertSound.play().catch(err => console.log('Audio blocked:', err));

  // Auto-dismiss notification after 5 seconds
  setTimeout(() => setShowNotification(false), 5000);
};
```

**A.4 Map Update Integration**

New cases are automatically rendered on the GIS map:

```
// After polling detects new cases, map markers are refreshed
useEffect(() => {
  updateMapMarkers([...healthCases, ...animalCases, ...envCases]);
}, [healthCases, animalCases, envCases]);
```

### A.5 Technical Architecture

Polling Flow Diagram

```
[Component Mount]
       ↓
[Start Polling Timer: 30s interval]
       ↓
[Fetch 3 API Endpoints in Parallel]
       ↓
[Compare Response IDs with Previous State]
       ↓
   [New Cases?]
   ↙         ↘
 YES          NO
  ↓            ↓
[Trigger Alert] [Do Nothing]
  ↓            ↓
[Update Map]  [Wait for Next Interval]
  ↓
[Update Previous IDs State]
  ↓
[Wait 30s → Loop]
```

### A. 6 Data Transformation for Map Display

```
// Transform API response to map marker format
const transformToMarker = (caseData, sector) => ({
  id: caseData._id,
  position: [caseData.latitude, caseData.longitude],
  color: getSeverityColor(caseData.classification), // Red/Yellow/Green
  icon: getSectorIcon(sector), // Human/Animal/Environment icon
  popup: `
    ${caseData.disease} - ${caseData.state}
    Status: ${caseData.classification}
    Reported: ${formatDate(caseData.date_of_onset)}
  `
});
```

# APPENDIX B: CASE FILTERING AND SEARCH IMPLEMENTATION

This Appendix contains the code implementation for Case Filtering and Search Implementation

### B.1 Case Filtering and Search Implementation

```
// Enhanced filtering logic with multiple filter criteria
const filteredCases = useMemo(() => {
  return cases.filter((caseItem) => {
    // Text searches
    if (
      filters.search &&
      !Object.values(caseItem).some((value) =>
        String(value).toLowerCase().includes(filters.search.toLowerCase())
      )
    )
      return false;

    if (
      filters.personSearch &&
      !`${caseItem.personID} ${caseItem.reporting_source}`
        .toLowerCase()
        .includes(filters.personSearch.toLowerCase())
    )
      return false;

    // Filter by specific fields
    if (filters.disease !== "All" && caseItem.disease !== filters.disease)
      return false;
    if (
      filters.classification !== "All" &&
      caseItem.classification !== filters.classification
    )
      return false;
    if (filters.outcome !== "All" && caseItem.outcome !== filters.outcome)
      return false;
    if (filters.region !== "All" && caseItem.region !== filters.region)
      return false;
    if (filters.sex !== "All" && caseItem.sex !== filters.sex) return false;

    return true;
  });
}, [filters, cases]);
```

# APPENDIX C: PAGINATION IMPLEMENTATION

This Appendix contains the code implementation for Pagination Implementation

## C.1 Pagination Implementation

```
const totalPages = Math.ceil(filteredCases.length / pageSize);
const paginatedCases = filteredCases.slice(
  (currentPage - 1) * pageSize,
  currentPage * pageSize
);

// Pagination controls
<div className="flex items-center justify-between px-6 py-4 border-t border-gray-200">
  <div className="flex items-center gap-4">
    <select
      value={pageSize}
      onChange={(e) => {
        setPageSize(Number(e.target.value));
        setCurrentPage(1);
      }}
      className="px-3 py-2 border border-gray-300 rounded-md"
    >
      <option value={10}>10 per page</option>
      <option value={25}>25 per page</option>
      <option value={50}>50 per page</option>
      <option value={100}>100 per page</option>
    </select>
    <span className="text-sm text-gray-700">
      Showing {(currentPage - 1) * pageSize + 1} to{" "}
      {Math.min(currentPage * pageSize, filteredCases.length)} of{" "}
      {filteredCases.length} results
    </span>
  </div>

  <div className="flex items-center gap-2">
    <button
      onClick={() => setCurrentPage(Math.max(1, currentPage - 1))}
      disabled={currentPage === 1}
      className="px-3 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md hover:bg-gray
    >
      Previous
    </button>
    <button
      onClick={() => setCurrentPage(Math.min(totalPages, currentPage + 1))}
      disabled={currentPage === totalPages}
      className="px-3 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md hover:bg-gray
    >
      Next
    </button>
  </div>
```

# APPENDIX D: DATA EXPORT FUNCTIONALITY WITH EXCEL

This Appendix contains the code implementation for excel export functionality

### D.1 Export Functionality

```
// Handle export to Excel using SheetJS (XLSX)
const handleExportToExcel = () => {
  // Prepare data for export
  const exportData = filteredCases.map((caseItem) => ({
    "Case ID": caseItem.case_id,
    "EPID Number": caseItem.personID,
    Disease: caseItem.disease,
    Classification: caseItem.classification,
    Outcome: caseItem.outcome,
    Age: caseItem.age,
    Sex: caseItem.sex,
    Occupation: caseItem.occupation,
    State: caseItem.state,
    LGA: caseItem.lga,
    Region: caseItem.region,
    "Health Facility": caseItem.health_facility,
    "Reporting Source": caseItem.reporting_source,
    "Date of Onset": caseItem.date_of_onset,
    "Date of Confirmation": caseItem.date_of_confirmation,
    Symptoms: caseItem.symptoms,
    "Risk Factors": caseItem.risk_factors,
    Latitude: caseItem.latitude,
    Longitude: caseItem.longitude,
    "Reported At": caseItem.reported_at
      ? new Date(caseItem.reported_at).toLocaleString()
      : "",
  }));

  // Create worksheet
  const ws = XLSX.utils.json_to_sheet(exportData);

  // Set column widths
  const colWidths = [
    { wch: 12 }, { wch: 15 }, { wch: 20 }, { wch: 15 },
    { wch: 15 }, { wch: 8 }, { wch: 10 }, { wch: 20 },
    { wch: 15 }, { wch: 15 }, { wch: 15 }, { wch: 25 },
    { wch: 20 }, { wch: 15 }, { wch: 18 }, { wch: 30 },
    { wch: 30 }, { wch: 12 }, { wch: 12 }, { wch: 20 },
  ];
  ws["!cols"] = colWidths;

  // Create workbook
  const wb = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(wb, ws, "Human Cases");

  // Generate filename with current date
  const date = new Date().toISOString().split("T")[0];
  const filename = `Human_Cases_Export_${date}.xlsx`;

  // Save file
  XLSX.writeFile(wb, filename);
};
```

# APPENDIX E: CASE CREATION WITH API INTEGRATION

This Appendix contains the code implementation for Case Creation with API Integration

### E.1 Case Creation with API Integration

```
// Handle new case submission
const handleSubmitNewCase = async (e) => {
  e.preventDefault();
  setIsSubmitting(true);

  try {
    const token = localStorage.getItem("authToken");
    const response = await axios.post(
      "https://api.example.com/api/v1/reports/health",
      formData,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "application/json",
        },
      }
    );

    if (response.status === 200 || response.status === 201) {
      alert("Case created successfully!");
      setShowNewCaseModal(false);
      // Reset form
      setFormData({
        case_id: "",
        personID: "",
        disease: "",
        classification: "",
        outcome: "",
        longitude: 0,
        latitude: 0,
        state: "",
        lga: "",
        health_facility: "",
        region: "",
        date_of_onset: "",
        date_of_confirmation: "",
        reporting_source: "",
        age: 0,
        sex: "",
        occupation: "",
        symptoms: "",
        risk_factors: "",
        category: "Human",
      });
      // Refresh the cases list
      await fetchCases(1, apiPageSize);
    }
  } catch (error) {
    console.error("Error creating case:", error);
    alert(
      error.response?.data?.message ||
        "Error creating case. Please try again."
    );
  } finally {
    setIsSubmitting(false);
  }
```

# APPENDIX F: CASE DELETION WITH CONFIRMATION

This Appendix contains the code implementation for Case Deletion with Confirmation

### F.1 Case Deletion with Confirmation

```
// Handle deleting a case
const handleDeleteCase = async (caseItem) => {
  const confirmDelete = window.confirm(
    `Are you sure you want to delete case ${caseItem.case_id}?\n\nPerson ID: ${caseItem.personID}\nDisease: ${caseIte
  );

  if (!confirmDelete) {
    return;
  }

  try {
    const token = localStorage.getItem("authToken");
    const response = await axios.delete(
      `https://api.example.com/api/v1/reports/health/${caseItem.case_id}`,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "application/json",
        },
      }
    );

    if (response.status === 200 || response.status === 204) {
      alert("Case deleted successfully!");
      // Refresh the cases list
      await fetchCases(apiCurrentPage, apiPageSize);
    }
  } catch (error) {
    console.error("Error deleting case:", error);
    const errorMessage =
      error.response?.data?.message ||
      error.response?.data?.detail ||
      "Error deleting case. Please try again.";
    alert(errorMessage);
  }
};
```

# APPENDIX G: DATA FETCHING WITH PAGINATION AND ERROR HANDLING

This Appendix contains the code implementation for Data Fetching with Pagination and Error Handling

### G.1 Data Fetching with Pagination and Error Handling

```
// Fetch cases from API with pagination
const fetchCases = async (page = 1, pageSize = 100) => {
  try {
    setIsLoading(true);
    setError(null);
    const token = localStorage.getItem("authToken");

    const response = await axios.get(
      "https://api.example.com/api/v1/reports/health",
      {
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "application/json",
        },
        params: {
          page: page,
          page_size: pageSize,
          sort_by: "created_at",
          sort_order: "desc",
        },
      }
    );

    // Handle different response structures
    if (response.data) {
      if (Array.isArray(response.data)) {
        setCases(response.data);
        setTotalCases(response.data.length);
      } else if (response.data.items && Array.isArray(response.data.items)) {
        setCases(response.data.items);
        setTotalCases(response.data.total || response.data.items.length);
      } else {
        console.warn("Unknown API response structure:", response.data);
        setCases([]);
        setTotalCases(0);
      }
    }
  } catch (error) {
    console.error("Error fetching cases:", error);
    setError(error.response?.data?.message || "Failed to fetch cases");
    setCases([]);
    setTotalCases(0);
  } finally {
    setIsLoading(false);
  }
};

useEffect(() => {
  fetchCases(apiCurrentPage, apiPageSize);
}, []);
```