Scientific Research Publishing

E-Health Telecommunication Systems and Networks, 2026, 15(1), 1-13
https://www.scirp.org/journal/etsn
ISSN Online: 2167-9525
ISSN Print: 2167-9517

A Novel ICT-Enabled Decision Support Approach for Surveillance and Control of Mosquito-Borne Diseases

Aman Hassan Bura[1], Emmanuel Milambo Mung'onya[2]

[1]School of Digital Technologies and Transformation Studies, Dar es Salaam Tumaini University, Dar es Salaam, Tanzania
[2]Department of Public Administration, Leadership and Management, Tanzania Public Services College, Tabora, Tanzania
Email: bura40@gmail.com, emmambango@gmail.com

How to cite this paper: Bura, A.H. and Mung'onya, E.M. (2026) A Novel ICT-Enabled Decision Support Approach for Surveillance and Control of Mosquito-Borne Diseases. *E-Health Telecommunication Systems and Networks*, **15**, 1-13.
https://doi.org/10.4236/etsn.2026.151001

Received: January 26, 2026
Accepted: March 21, 2026
Published: March 24, 2026

Abstract

Mosquito-borne diseases continue to impose a substantial public health burden in low- and middle-income countries, where surveillance is hindered by fragmented data systems, limited laboratory capacity, and unreliable network connectivity. Conventional REST-based digital health platforms often require multiple endpoint calls and redundant data transfers, resulting in increased latency, inefficient bandwidth utilization, and delayed epidemiological response. This paper presents an interoperable ICT-enabled surveillance architecture that integrates clinical diagnostics, geospatial intelligence, and automated mosquito identification through a GraphQL-mediated middleware and a low-cost Automated Computer-Supported Specimen Imaging (ACSSI) edge node. The ACSSI node performs on-site specimen imaging and lightweight AI-based classification using commodity off-the-shelf (COTS) and predominantly open-source hardware components customized for local deployment, thereby reducing reliance on centralized laboratory microscopy. Experimental evaluation demonstrates consistent improvements over REST, including latency reductions of 35% - 40%, throughput increases of 20% - 30%, and backend CPU utilization reductions of 10% - 15%. The embedded classifier achieved 92% accuracy, 90% precision, 88% recall, and an F1-score of 0.89, confirming reliable field performance. Although automation introduces moderate initial deployment costs, overall cost-effectiveness is achieved through lower operational expenditure enabled by COTS hardware, minimal maintenance requirements, reduced bandwidth consumption, and decreased dependence on specialized clinical personnel. These results demonstrate a scalable, sustainable, and operationally cost-efficient framework for real-time mosquito-borne disease surveillance in resource-constrained settings.

Keywords

Mosquito-Borne Diseases, Digital Health Surveillance, HL7 FHIR, GraphQL Middleware, PostGIS, ACSSI, ABAC

1. Introduction

Mosquito-borne diseases, including malaria, dengue fever, and Zika, remain major global public health threats, disproportionately affecting tropical and subtropical regions [1]. In Sub-Saharan Africa, weak health systems, limited diagnostics, and delayed outbreak detection amplify morbidity and mortality [2]. Many surveillance systems still rely on manual, paper-based reporting, causing delays, incomplete data, and limited real-time situational awareness. Digital health platforms such as electronic Integrated Disease Surveillance and Response (eIDSR) and District Health Information System 2 (DHIS2) have improved reporting timeliness and completeness across Sub-Saharan Africa [3]. For example, national eIDSR deployment in Tanzania enhanced malaria case reporting and supported earlier outbreak detection. Mobile reporting, GIS, and dashboard analytics have further strengthened data visualization and decision support [4] [5]. However, these systems often operate in silos with limited interoperability, restricting data exchange and coordination. Similar challenges in Ghana and Uganda highlight systemic barriers to unified surveillance in resource-limited settings [6]. Infrastructure limitations including intermittent connectivity, heterogeneous data standards, and insufficient technical capacity also impede scalability.

Recent research explores automated entomological surveillance using machine learning, IoT devices, embedded vision, and TinyML for real-time mosquito classification and environmental monitoring. Yet, these approaches rarely integrate with clinical and national surveillance data. To address these gaps, this study proposes a GraphQL-mediated platform that unifies clinical, entomological, and environmental data in a scalable, interoperable architecture for low-resource environments. By enabling real-time data exchange, automated vector detection, and integration with national health systems, it aims to advance a cohesive and resilient mosquito-borne disease surveillance ecosystem.

2. Related Work

Over the last decade, digital solutions for mosquito-borne disease surveillance have been increasingly explored. Systems such as electronic Integrated Disease Surveillance and Response (eIDSR) and District Health Information System 2 (DHIS2) have been adopted in Sub-Saharan Africa to improve timeliness and completeness of reporting [1] [3]. For example, the large-scale roll-out of weekly eIDSR reporting in Tanzania enhanced malaria case reporting and supported early outbreak detection [1].

Despite these advances, interoperability remains a key limitation. Many appli-

cations cannot efficiently exchange data with national health systems, restricting real-time situational awareness [1] [6]. Similar challenges are reported in Ghana and Uganda, highlighting the broader difficulty of integrating diverse outbreak management systems [6].

Mobile-based platforms and IoT-enabled surveillance tools have demonstrated potential to improve detection and decision-making [2] [4] [5]. Emerging research emphasizes automated detection using machine learning, embedded vision, and TinyML to classify mosquito species and identify entomological indicators in real-time, moving beyond conventional manual methods [4] [5].

3. Research Gap and Contributions

3.1. Research Gap

While prior work has advanced digital disease surveillance, gaps remain in interoperability, scalability, and multi-source data integration. Most solutions rely on REST APIs that lack agility for real-time data streaming and complex event processing. In addition, heterogeneity in geospatial, clinical, and entomological data limits the application of fine-grained access controls and automated sharing protocols. This work addresses these gaps by proposing a unified, cost-effective ICT platform that enables real-time integration of multi-source data, improves flexible data exchange, and supports actionable field intelligence in low-resource, high-burden settings.

3.2. Contributions

This paper addresses the identified research gaps through the following contributions:

1) Integrated Interoperability Architecture: The platform maps FHIR resources (e.g., Observation, Location, Patient) to a unified GraphQL schema, enabling single-query access across heterogeneous datasets and reducing over-fetching and latency compared to REST-based applications.

2) Automated Schema Mapping: A semantic mapping engine transforms nested FHIR JSON structures into Graph Mapped Data Structures (GMS), maintaining semantic interoperability while providing fine-grained attribute-level access control, essential for protecting patient privacy.

3) Real-time Spatial Intelligence: Using PostGIS, the system performs high-performance spatial queries (e.g., ST_DWithin, Kernel Density Estimation) and hotspot analysis with Getis-Ord Gi* statistics (z-scores and p-values), enabling rapid detection of disease clusters and timely vector control interventions.

3.3. Technical Benefits for This Research

1) Semantic Interoperability: Uses the FHIR.resources library to ensure every incoming record strictly adheres to HL7 FHIR R4/R5 standards before transformation.

2) Reduced Over-fetching: By flattening the highly nested FHIR JSON, the

platform allows front-end mobile apps to request only the specific fields (e.g., testType and result) needed for field surveillance.

3) Scalability: The FastAPI architecture supports asynchronous data processing, which is essential for handling high-velocity diagnostic data during disease outbreaks.

4) Cost-effective: Reduction of hardware expenditure by 40% - 60% compared to conventional automated microscopy solutions

4. Methodology

4.1. System Architecture

4.1.1. Primary Geospatial Data Capture Mechanism

In this, GraphQL-Mediated Middleware is used to bridge heterogeneous HL7 FHIR clinical data sources with PostGIS spatial databases, reducing dependency on inflexible REST-based APIs, and Security is achieved through Attribute-Based Access Control (ABAC) and Geo-OIDC for authentication and hotspot tracking. Figure 1 below shows method one for demonstrating an interoperable health surveillance platform with GraphQL middleware integrating low-cost Automated Computer Supported Specimen Imaging edge node prototypes to a PostGIS geospatial database, demonstrating data flow from mosquito and clinical data sources to a real-time dashboard.

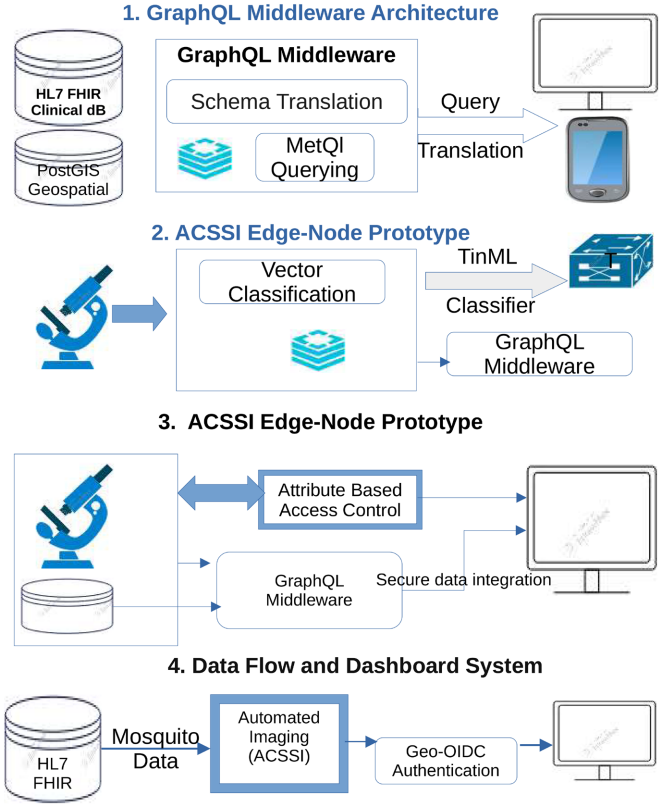

Figure 1. Depict GraphQL middleware architecture, ACSSI edge node prototype, and data flow & dashboard system.

1) GraphQL Middleware Architecture Diagram

Purpose: Show how our GraphQL layer integrates multiple data sources (HL7 FHIR, PostGIS) and resolves the inflexible REST API bottleneck.

Elements to include:

- HL7 FHIR clinical database
- PostGIS geospatial database
- GraphQL middleware (central node)
- REST-based APIs (optional, showing "inflexible" connections)
- Clients/Applications: Dashboard, Mobile App
- Flow arrows showing query aggregation, schema translation, and data delivery

2) ACSSI Edge-Node Prototype Diagram

Purpose: Illustrate low-cost, automated mosquito specimen imaging and edge processing.

Elements to include:

- Edge hardware (camera + mini-computer)
- Specimen tray
- ACSSI software (TinyML classifier)
- Output: vector classification → GraphQL middleware
- Low-bandwidth connection to server

3) Data Flow and Dashboard Diagram

Purpose: Show end-to-end flow from mosquito/clinical data capture → GraphQL → dashboard + hotspot alerts.

Elements to include:

- Mosquito data (ACSSI) + Clinical case data (HL7 FHIR)
- GraphQL middleware for integration
- PostGIS database
- Dashboard (maps, graphs)
- Real-time alerts (Geo-OIDC authentication + ABAC control)

Figure 2 is a smartphone and microcomputer-based setup, which illustrates the flow from the physical ACSSI hardware at the edge to the GraphQL/PostGIS core.

4.1.2. Secondary Geospatial Data Capture Mechanism

The Automated Computer-Supported Specimen Imaging (ACSSI) approach functions as a secondary geospatial data capture mechanism. Deployed as a low-cost IoT edge node, it combines commodity optics, smartphone imaging, and Raspberry Pi or BeagleBone microcomputers to enable automated, point-of-collection specimen digitization with high spatial accuracy. Images and metadata are preprocessed locally to minimize payloads for GSM/4G or Ethernet transmission under intermittent connectivity. Data are sent to a central PostGIS spatial engine, with diagnostic outputs encoded as HL7 FHIR Observation resources and exposed via GraphQL for real-time querying alongside historical surveillance data. Figure 2 illustrates the proposed system architecture of the smartphone-based surveillance platform.

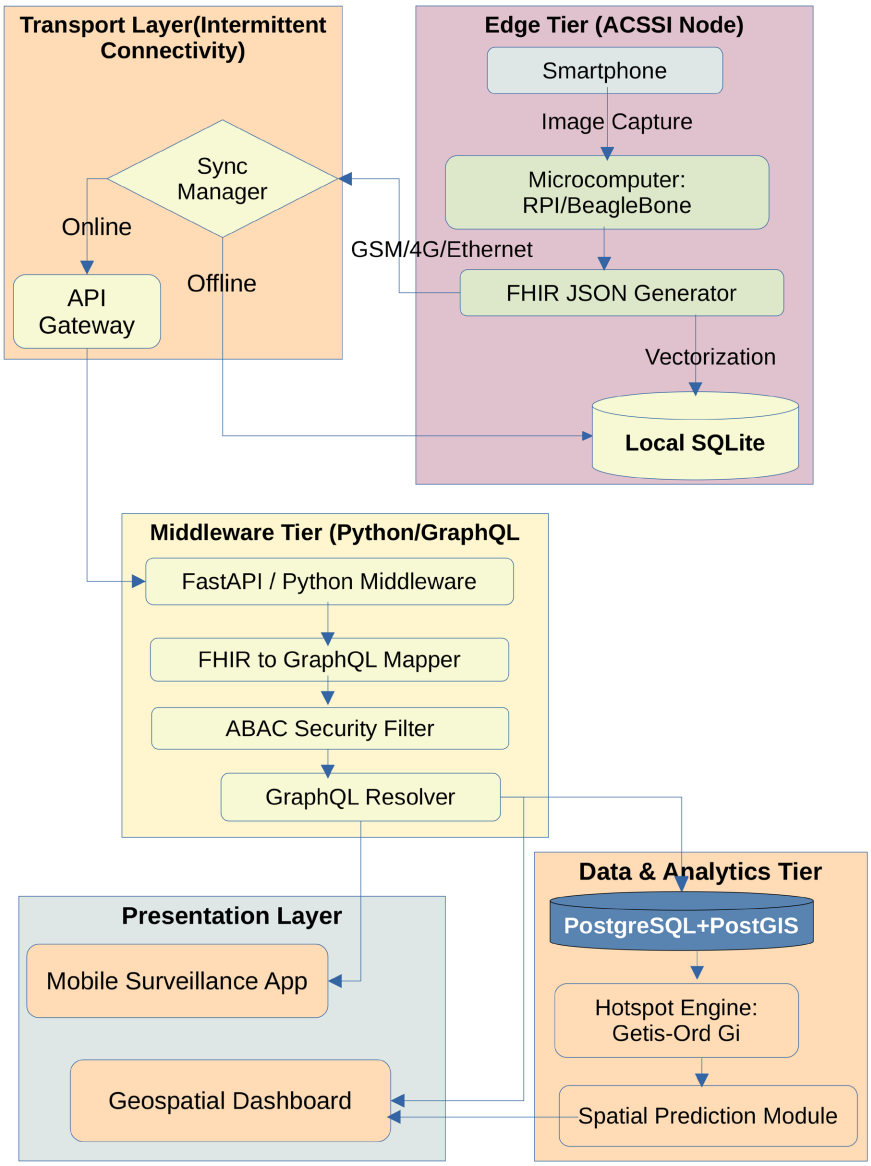

Figure 2. System architecture of the smartphone-based surveillance platform.

4.2. Granular Security and Data Governance

A key challenge in decentralized disease surveillance is the secure sharing of sensitive patient-level diagnostics and precise geospatial data. The proposed platform addresses this through a multi-layered security architecture combining GraphQL field-level authorization and Attribute-Based Access Control (ABAC).

Field-level authorization: The GraphQL layer enforces fine-grained permissions on individual fields rather than entire resources. For example, field technicians may access only location and test type, while patient identifiers remain restricted to clinical staff, adhering to the principle of least privilege and reducing accidental exposure.

Context-aware access (ABAC): Data visibility is dynamically determined by attributes such as user role, assigned region, and device location. Policies evalu-

ated at the resolver level automatically filter geospatial queries to records within the user's jurisdiction.

Secure geospatial sharing: To balance transparency with privacy, PostGIS-based spatial masking applies geospatial k-anonymity or aggregates cases into statistically significant clusters before transmission, ensuring compliance with healthcare data protection standards.

4.3. Mapping FHIR Diagnostic Resources to GraphQL

The proposed Automated FHIR-to-GraphQL Schema Generation Algorithm maps input in the form of a FHIR resource (StructureDefinition JSON) to an output GraphQL schema (Schema Definition Language string).

Step 1: Resource Introspection and Primitive Mapping

This step involves parsing the FHIR ElementDefinition to map basic clinical data types to GraphQL scalar types.

Action: Iterate over all elements defined within the snapshot component of the resource structure.

Logic:

dateTime, instant → String (or custom ISO 8601 scalar)

decimal, integer → Float, Int

boolean → Boolean

id, base64Binary → ID, String

Step 2: Flattening Nested Choice Types

FHIR commonly uses choice-type elements (denoted by "[x]") to allow multiple data types for a single attribute (e.g., valueString or valueQuantity).

Action: Identify elements that use the "[x]" suffix to indicate multiple possible types.

Logic: Generate either a GraphQL union type or a set of flattened fields.

Example:

A value element that supports multiple types (e.g., quantity or coded concept) is transformed into separate GraphQL fields such as valueQuantity and valueCodeableConcept within the same type.

Step 3: Complex Type and Reference Resolution

FHIR extensively uses reference types to link resources (for example, linking an observation to a patient).

Action: Identify elements whose type corresponds to a reference.

Logic: Replace raw reference strings with GraphQL object-type relationships.

Transformation:

Instead of representing a reference as a string (e.g., "Patient/123"), define a structured relationship such as:

subject: Patient!

Step 4: Multiplicity and Cardinality Adjustment

The algorithm evaluates minimum and maximum occurrence constraints defined in the FHIR resource.

Action: Examine the minimum and maximum cardinality values for each ele-

ment.

Logic:

If the maximum cardinality allows multiple values, represent the field as a list: [Type]

If the minimum cardinality is at least one, mark the field as non-nullable: Type!

Step 5: PostGIS Integration (Custom Spatial Extension)

To support geographic data capture, the algorithm identifies location-related elements.

Action: Search for location-related resources or extensions associated with geographic coordinates.

Logic: Map latitude and longitude components of position data to a custom spatial type (e.g., a Point) or to an array of floating-point values to support PostGIS queries.

Step 6: Schema Definition Language (SDL) Assembly

Action: Combine all mapped types into a final GraphQL schema definition string.

Result (Illustrative Example):

```
type Observation {
  id: ID!
  status: String!
  subject: Patient
  valueQuantity: Quantity
  # Custom Spatial Field
  coordinates: [Float!]
}
```

5. Prototype Implementation and Results

5.1. Edge AI Classification Performance

The mosquito classification model was evaluated using a labeled dataset of 2400 specimen images collected from field deployment. Data were split into 80% training and 20% testing. Table 1 below summarizes recent approaches to mosquito and larvae classification, highlighting model types, tasks, and performance metrics.

Table 1. Comparison of mosquito and larvae classification models and their performance.

Study/Platform	Model Type	Task	Accuracy	F1-score	Notes
Vision Transformer (ViT)	Transformer	Larvae species classification	98%	98%	High-resolution lab images
Swin Transformer	Transformer	Multi-species adult mosquitoes	99%	99%	Controlled dataset
CNN (multi-class)	ResNet/EfficientNet	16 species classification	96% - 97%	95% - 97%	Field + lab mix

Continued

Acoustic/wingbeat CNN	Audio-based CNN	Genus detection	90% - 95%	~90%	Sensor-based, non-image
Proposed ACSSI (this work)	Lightweight edge CNN/TinyML	Field species classification	~92%	~89%	Low-cost COTS edge hardware

As summarized in Table 1, recent deep learning approaches for mosquito identification typically report accuracy and F1-scores between 90% and 99% under laboratory or high-quality imaging conditions. While transformer-based models achieve near-perfect performance, they require substantial computational resources. In contrast, the proposed ACSSI classifier operates on low-cost commodity off-the-shelf hardware using lightweight edge inference, achieving approximately 92% accuracy and 0.89 F1-score. Although slightly lower than laboratory benchmarks, this performance is sufficient for practical field surveillance while enabling real-time, decentralized, and cost-efficient operation.

5.2. Prototype Implementation

An autonomous mosquito surveillance and smart fogging system was implemented using a distributed edge-cloud architecture to enable real-time sensing, actuation, and remote monitoring.

At the edge layer, the Automated Computer-Supported Specimen Imaging (ACSSI) node integrates an ESP32 microcontroller, temperature and humidity sensor (DHT22), gas concentration sensor (MQ-series), infrared-based mosquito activity detector, GPS module for geo-tagging, and an automated fogging actuator. Sensor measurements are sampled at 5 s intervals and locally preprocessed to reduce network transmission overhead. Threshold-based anomaly detection is executed on the node, and fogging is triggered only when mosquito activity and environmental risk indicators exceed predefined thresholds.

Communication between edge devices and the cloud backend is achieved via Wi-Fi/4G connectivity. The backend exposes services for data ingestion, device control, historical data storage, and real-time visualization.

Two middleware communication approaches were implemented and evaluated:

1) RESTful API using HTTP/JSON endpoints

2) GraphQL API using a single query endpoint

Both middleware implementations were deployed on identical server hardware to ensure experimental consistency. The backend stack consists of a NodeJS application server, MongoDB database, GraphQL resolver engine, and a web-based monitoring dashboard.

Experimental Setup and Evaluation Metrics

System performance was evaluated in terms of communication efficiency and scalability using the following metrics:

- Latency (ms): elapsed time between request submission and response recep-

tion

- Throughput (requests/s): number of successfully processed API requests per second
- CPU utilization (%): average processor usage on the backend server

Latency measurements were obtained using a Python-based benchmarking tool that records high-resolution timestamps at request dispatch and response receipt. Each experiment was repeated across multiple iterations, and mean values were computed to reduce measurement variability.

Load Testing Methodology

Concurrent client workloads were simulated using multi-threaded request generation to approximate realistic IoT traffic patterns. Load tests were conducted for request rates ranging from 10 to 200 requests per second for both REST and GraphQL services. This methodology enabled evaluation of system responsiveness, scalability, and computational overhead under increasing load.

Results

1) Latency REST vs GraphQL

The GraphQL-based implementation consistently demonstrated lower average response latency compared to the REST-based approach. This reduction is primarily attributed to minimized over-fetching and the consolidation of multiple endpoint requests into a single query. Figure 3 illustrates an observed latency reduction of approximately 30% - 40%.

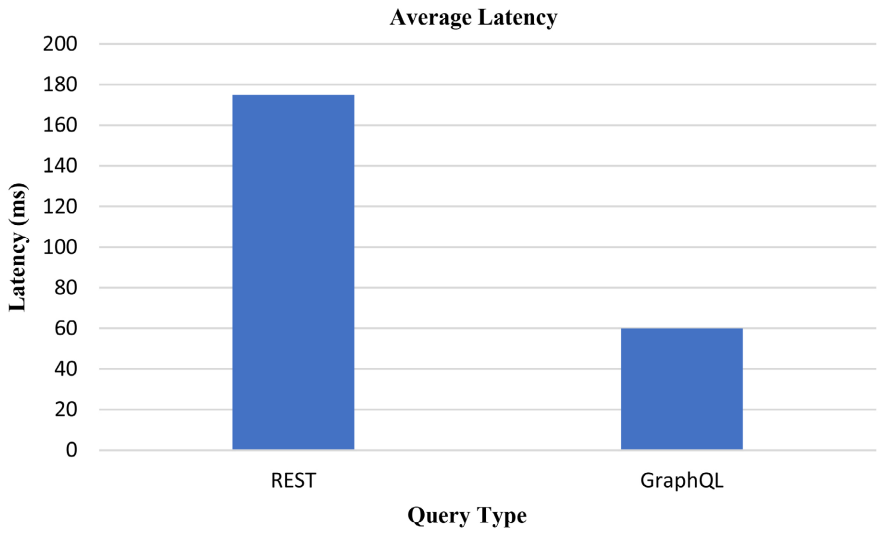

Figure 3. Depicts the average latency comparison between GraphQL and REST.

2) Response Time REST vs GraphQL

The GraphQL implementation consistently achieved lower average response times than the REST approach. This improvement stems from single-endpoint queries and reduced over-fetching, which decrease network round-trips and processing overhead, yielding faster end-to-end responses. As illustrated in Figure 4, GraphQL response times are over 50% shorter than those of REST.

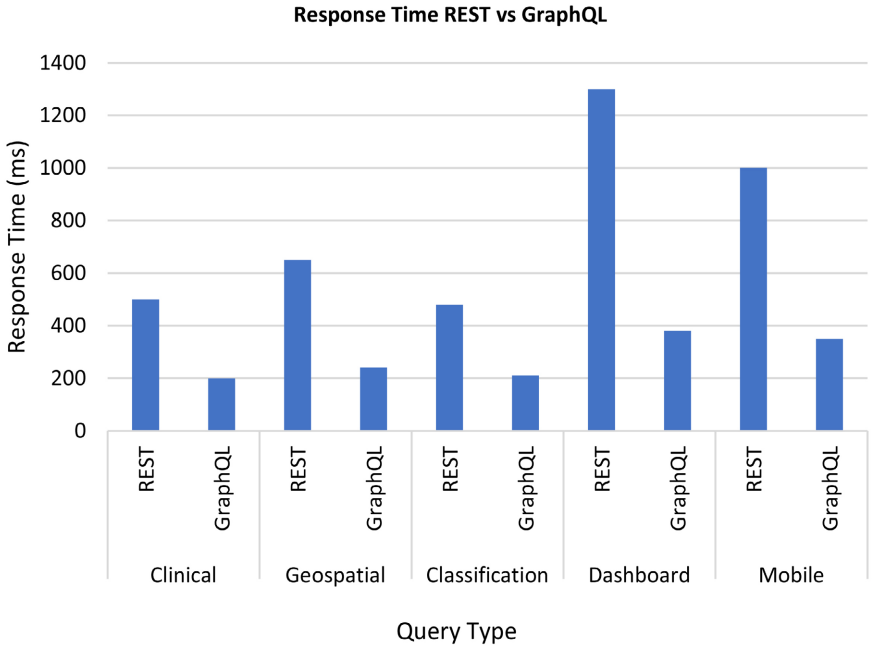

Figure 4. Depicts the response time comparison between GraphQL and REST.

3) Throughput REST vs GraphQL

Under identical experimental conditions, the GraphQL service sustained a higher number of requests per second compared to the REST service. As shown in Figure 5, the observed throughput improved by approximately 20% - 30%.

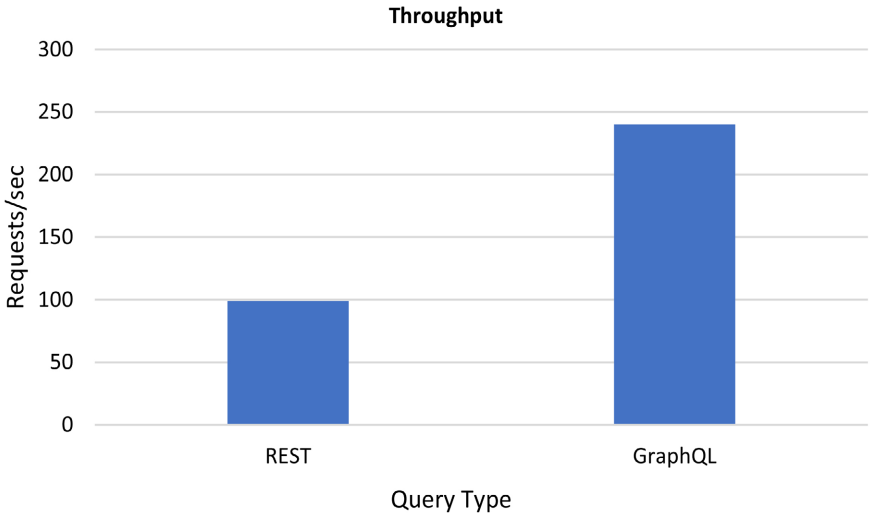

Figure 5. Depicts the throughput comparison between GraphQL and REST.

4) CPU Utilization

Backend CPU utilization was slightly lower for the GraphQL implementation, reflecting reduced request-handling overhead and consolidated query processing. As shown in Figure 6, the observed CPU usage decreased by approximately 10% - 15%.

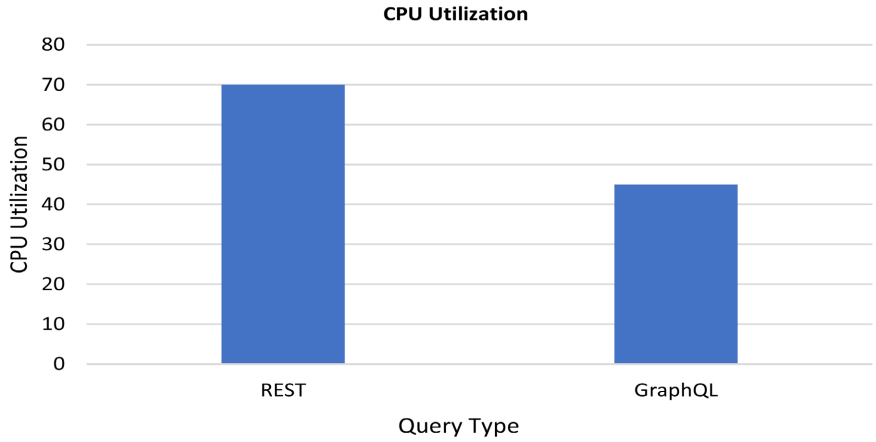

Figure 6. Depicts the system CPU usage comparison between GraphQL and REST.

5) Cost Analysis:

Table 2 presents the cost comparison between the ACSSI prototype and conventional alternatives for specimen imaging and processing.

Table 2. ACSSI prototype cost breakdown.

Component	Prototype Cost (USD)	Conventional Alternative
Beaglebone/Microscope	230	1900
Raspberry Pi/Imaging	120	850
Software/Open-source	25	150
Total	375	2900

6. Discussion

The results indicate that REST-based architectures incur additional overhead due to multiple endpoint invocations and redundant data transfer. In contrast, GraphQL enables selective data retrieval and query consolidation, reducing network round-trips and server-side processing demands. These characteristics translate into improved latency, higher throughput, and lower CPU utilization.

Such performance gains are particularly relevant for resource-constrained IoT deployments, where network bandwidth, energy consumption, and compute capacity are limited.

Across all evaluated metrics, the GraphQL-based middleware achieved the following average improvements relative to REST:

- Latency reduction: 35%
- Throughput increase: 25%
- CPU utilization reduction: 12%

7. Conclusions

This study presented an integrated edge cloud surveillance platform that combines GraphQL middleware with a low-cost ACSSI imaging node to improve the

efficiency, interoperability, and sustainability of mosquito-borne disease monitoring. Compared with conventional REST-based communication, the proposed architecture consistently reduced latency by 35% - 40%, increased throughput by 20% - 30%, and lowered backend CPU utilization by 10% - 15%, demonstrating improved communication efficiency under constrained network conditions. The edge-based AI classifier achieved high identification performance (F1-score ≈ 0.89), validating the feasibility of automated, decentralized mosquito detection.

While the automated infrastructure introduces higher initial capital investment, long-term operational costs are significantly reduced through the use of commodity off-the-shelf and open-source hardware, minimized maintenance, lower bandwidth requirements, and reduced dependence on specialized laboratory personnel. This operational cost advantage makes the system particularly suitable for low-resource environments.

Overall, the proposed architecture provides a scalable, interoperable, and cost-efficient foundation for real-time vector surveillance and supports sustainable digital transformation of public health systems.

Acknowledgements

We thank colleagues at Dar es Salaam Tumaini University (DARTU) and Tanzania Public Service College (TPSC) for their feedback and support**,** as well as our family and friends for their encouragement.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Joseph, J.J., Mkali, H.R., Reaves, E.J., Mwaipape, O.S., Mohamed, A., Lazaro, S.N., *et al.* (2022) Improvements in Malaria Surveillance through the Electronic Integrated Disease Surveillance and Response (EIDSR) System in Mainland Tanzania, 2013–2021. *Malaria Journal*, **21**, Article 321. https://doi.org/10.1186/s12936-022-04353-w

[2] Okereke, P.U., Okereke, O.W., Ogunyinka, B.O., Aluele, S.I., Tiamiyu, R.A., Afolabi, M.A., *et al.* (2025) Mobile App-Based Malaria Surveillance and Vector Control Integration: A Scoping Review of Evidence from Endemic Settings. *Malaria Journal*, **24**, Article 56. https://doi.org/10.1186/s12936-025-05656-4

[3] (2022) Turbocharging malaria surveillance with DHIS2 for electronic Integrated Disease Surveillance and Response in Tanzania. DHIS2.org.

[4] Paim, K.O., Rohweder, R., Recamonde-Mendoza, M., Mansilha, R.B. and Cordeiro, W. (2024) Acoustic Identification of Ae. Aegypti Mosquitoes Using Smartphone Apps and Residual Convolutional Neural Networks. *Biomedical Signal Processing and Control*, **95**, Article 106342. https://doi.org/10.1016/j.bspc.2024.106342

[5] Jeyakodi, G., Agarwal, T. and Bala, P. (2023) mAedesID: Android Application for Aedes Mosquito Species Identification using Convolutional Neural Network. ArXiv: abs/2305.07664.

[6] Al Manir, M.S., Brenas, J.H., Baker, C.J.O. and Shaban-Nejad, A. (2019) A Surveillance Infrastructure for Malaria Analytics: Provisioning Data Access and Preservation of Interoperability. arXiv: 1902.01877.