IADIS International Journal on Computer Science and Information Systems
Vol. 18, No. 2, pp. 113-129
ISSN: 1646-3692

LEVERAGING MOBILE TECHNOLOGY FOR ENHANCED DIAGNOSIS OF TROPICAL FEBRILE DISEASES IN RESOURCE-CONSTRAINED SETTINGS

Kingsley Attai[1], Okure Obot[2], Daniel Asuquo[2], Ekemini Johnson[1], Kiirya Arnold[3], Mercy Edoho[2], Moses Ekpenyong[2], Christy Akwaowo[4], Owoidoho Udoh[5], Victoria Usen[5] and Faith-Michael Uzoka[6]
[1]*Ritman University, Ikot Ekpene, Akwa Ibom State, Nigeria*
[2]*University of Uyo, Uyo, Akwa Ibom State, Nigeria*
[3]*Kyambogo University, Kampala, Uganda*
[4]*University of Uyo Teaching Hospital, Uyo, Akwa Ibom State, Nigeria*
[5]*Primary Health Centre, Ukana Ikot Ide, Essien Udim, Akwa Ibom State, Nigeria*
[6]*Mount Royal University Calgary, Canada*

ABSTRACT

Febrile diseases have contributed to a very large extent to the high mortality and morbidity rate in the tropical regions especially in the low-to-middle-income countries (LMIC). Among other reasons is the confusion arising from the differentiation of the various symptoms that look alike and overlapping. This is in addition to the fact that there is an acute shortage of experienced medical personnel to diagnose cases of febrile diseases. A solution is proffered to this problem in this study by developing a medical decision support system (MDSS) for the diagnosis of febrile diseases. The system is based on the Agile software life cycle model that allows all the stakeholders to interact in the course of developing the software. Here, the medical doctors, the frontline health workers (FHWs), and the information scientists meet regularly to elicit the requirements, and plan, design, and implement the system. The Analytic Hierarchy Process (AHP) model was used to aggregate the knowledge elicited from the medical doctors which they gathered after interaction with 3253 patients with febrile diseases. The AHP result showed a 90% accuracy with the diagnosis conducted by 60 medical doctors from the collected patients' data. The developed MDSS is an app deployed for use by the FHWs in an Android-based device with a user-friendly graphical user interface.

KEYWORDS

Febrile Disease, Tropical Disease, Analytic Hierarchy Process, Medical Decision Support System, Agile Methodology, Low-to-Middle-Income Countries

1. INTRODUCTION

Reusing the design concepts and codes of previously built software has enabled the quick and affordable construction of software. This concept has been promoted by software developers as well as academic researchers. Scientific journals have promoted reproducibility and reusability by mandating the Findability, Accessibility, Interoperability, and Reusability (FAIR) principle (Hauschild et al., 2022). These parameters are driven by software engineering (SE), a disciplined and principled approach to developing software. At the heart of SE is the Software Development Life Cycle (SDLC) model, defined by IEC 62304 as a conceptual structure including its lifetime from the requirements' definition until its deployment that describes processes, tasks, and activities involved in developing a software product with the ordered interdependencies. There are many SDLC models including the Agile model. The choice of a model in software development depends on the application area, size of the application, and availability of technology. The healthcare application needs to map the hospital model appropriately to fit the existing processes as well as align with the healthcare providers and the services rendered (Lalband and Kavitha, 2019). In the Agile model, the implementation is done iteratively and incrementally (Stoica et al., 2013). It focuses on the customer (patients) and the end-users (health care providers) satisfaction through regular meetings to elicit requirements and to understand their needs. It emphasizes users' feedback, reviews, and continuous testing of the program. The end-users are made to contribute immensely to the interface development since this is their primary point of contact with the patients.

In the design phase, the knowledge of the experts and the datasets of the patients as elicited at the requirement phase are used in building the engine of the software. The diagnosis processes used in healthcare are complicated and include several specialists with different subjective preferences and viewpoints, which may impede the final choice about the optimization of healthcare systems for various applications (Obot et al., 2023a). This necessitates the use of a model that can handle multi-criteria parameters as found in the Analytical Hierarchical Process (AHP). The professionals provide pair-wise comparison ratings based on their practical knowledge and assist in identifying the symptoms (criteria) associated with each febrile condition (alternative).

In the tropical and sub-tropical regions of the world, Nunthavichitra et al. (2020) claim that high humidity, high temperatures, and a lot of rain create a suitable habitat for agents of febrile diseases. Poor sanitation and vector control are the causes of this. Self-diagnosis and drug resistance have made things more difficult, particularly in low- to middle-income countries (LMIC). The symptoms of febrile diseases can be confusing and occasionally overlap, making it challenging for an inexperienced doctor to make an appropriate differential diagnosis. (Attai et al., 2022). The lack of skilled medical professionals in LMICs has led to a reliance on Frontline Health Workers (FHWs) to deliver essential medical services. According to WHO (2016), the acute worldwide physician shortage has increased interest in FHW training and utilization as a means of providing healthcare services. The majority of the time, patients appear with several diseases and multiple symptoms, making it difficult for an FHW to accurately diagnose their conditions.

This work aims to create an Agile SE lifecycle-based Medical Decision Support System (MDSS) for the diagnosis of febrile illnesses. The MDSS is an Android app with useful features and components that may be used on a variety of platforms and devices with or without internet connectivity. This paper is an extension of the work of Obot et al. (2023b) originally presented at the International Conference on e-Health. It emphasizes domain knowledge and end-user involvement while adhering to design thinking principles. The subsequent actions were taken in order to accomplish this goal; The development of an app for differential diagnosis of febrile diseases by FHWs includes the following steps: i) extraction of practical knowledge from medical experts in the diagnosis of febrile diseases; ii) application of agile SE methodology; iii) development and validation of an AHP-based model for the differential diagnosis; and iv) development of functional graphical user interfaces in collaboration with the FHWs who will ultimately use the app.

The remaining parts of the paper are structured as follows. The associated literature is discussed in Section 2, and the study's methodology is covered in Section 3. Results of the system deployment are reported in Section 4, and the study's conclusion and recommendations are presented in Section 5.

2. RELATED WORKS

The framework and methodology of SE for machine learning (ML) in health informatics (SEMLHI), which Moreb and Ata suggested in 2019, is a unique framework for health informatics. The study examines how SE and ML interact with health systems. The innovative method clarifies its characteristics, enables users to research and examine user needs, and establishes the functions of system-related objects as well as the ML techniques that must be used with the dataset. Three years' worth of datasets was gathered from a hospital run by the Palestinian government. The proposed framework was used to compare ML methods that predict test laboratory results on the datasets. This study put out a theoretical framework made up of four modules: software, an ML model, ML algorithms, and health informatics data. Three system engineering methodologies, Vee, Agile, and SE for ML in health informatics were used to compare the new technique. The outcomes demonstrated the effectiveness of the new one-shot delivery mechanism. The Support Vector Classifier (SVC) had an accuracy of approximately 0.57 according to the data.

Hasan et al. (2023) provide a fresh approach to SE for healthcare programs. The study proposes a new Secured Agile Software Engineering Methodology. To ensure the integrity of data and source, extra security layers have been introduced into the proposed agile method. The main focus of the newly proposed agile method was on design and preventive threat modeling. In the design phase of the proposed Secured Agile Model, four sequential subphases have been introduced and those are security requirements analysis, forming secure coding structure, threat modeling, and developing security architecture. The reason behind the sequential approach is to ensure effectiveness in protecting every phase from cyber criminals, attackers, phishing, threats, and malware.

By modifying the work of Kaur et al. (2018), Udo et al. (2022) added the software team productivity element to the Constructive Cost Model (COCOMO) for Software Development Effort Estimation. The software team productivity component was included in the COCOMO II model as a result of the research, and an Adaptive Neuro-Fuzzy Inference System (ANFIS) model was created to predict the software development effort using information from the PROMISE repository. The model was trained and the results were compared to an existing neuro-fuzzy constructive cost model by Kaur et al. (2018). Based on the results of the trials, it was concluded that the ANFIS model (with the productivity factor) generated better estimates than the Kaur et al., (2018) model and the Back Propagation model with six (6) inputs.

According to the requirements and capacities of research organizations, Hauschild and colleagues (2022) suggested a set of guidelines for academic software life cycle processes. Their analysis suggests a subset of components that they are confident will deliver a large benefit while keeping the effort within a practical range, even though the full execution of a software life cycle by commercial norms is not achievable in scientific work. The research demonstrated how new quality controls for academic software development might hasten the application of academic innovations in clinical practice.

In a survey article published in 2019, Lalband and Kavitha assess existing SE models and recommend the optimal SDLC model for healthcare apps that are focused on quality improvement. Another survey aspect is finding SE research topics for smart apps. The survey underlines the requirement for healthcare applications to accurately map the hospital model to meet the existing processes and align with the practitioner's specialty.

To develop Internet of Things - Data Analytics (IoT-DA) driven applications, a methodology that can be used to methodically investigate the impact of SE procedures and their underlying practices was empirically developed by Ahmad et al. in 2021. The project first utilized existing taxonomies and frameworks to build an evaluation framework for software processes, methods, and other SE artifacts for IoT-DA. Second, the study conducted a systematic mapping investigation to qualitatively choose 16 processes of SE for IoT-DA (from academic research and commercial solutions). The study then used the evaluation framework that had been created based on 17 different criteria (process activities) to conduct a fine-grained analysis of each of the 16 SE processes. A case study of how the suggested framework may be used to create an IoT-DA healthcare application was demonstrated. The paper concludes by highlighting major issues, suggested procedures, and knowledge gained through supporting process-centric software engineering of IoT-DA.

A systematic SE method based on a tailored rational unified process was presented by Abd Ghani et al. (2018) to manage the development of an integrated telehealth system. The study also suggested using literate modeling to analyze systems and create models that are understandable and accessible to users. One of the major advantages of these approaches, according to the study, is that the complexity of the telehealth system can be addressed by designing it incrementally, and system maintenance can be handled appropriately. On the other hand, literate modeling may enhance communication between the technical team and domain team (users such as doctors, nurses, and medical assistants) during system modelling, assisting the users in comprehending the proposed model of system requirements and design, which is typically presented in a technical language such as unified modeling language (UML) notation. The following issues with software development projects in the healthcare industry are emphasized before outlining the suggested method.

A study conducted by Ansari et al. (2020), aimed to present security experts' perspectives on the relative importance of the criteria for selecting an effective Software Requirement Engineering (SRE) method by utilizing the multi-criteria decision-making methods. The study was designed and carried out to determine the best SRE approach for the production of high-quality and reliable software based on the expertise and knowledge of the security expert. The fuzzy TOPSIS model was used to evaluate the hierarchical model. In pairs, efficient SRE selection criteria were contrasted. 25 security professionals were contacted and asked to complete a pair-wise criterion comparison form. A quantitative evaluation of the effects of the acknowledged selection criteria for efficient security requirement engineering methodologies has been conducted. Based on the results of the 25 participants' form replies comparison matrices for each of them were created. It was discovered that the consistency ratios (CR= 9.1%) were less than 10%. According to the results of pair-wise comparisons, the STORE methodology is the best security requirements engineering strategy for reliable healthcare software development, with a closeness coefficient (Ci) of 0.842. The results of this study's investigation show numerous elements that influence the choice of a trustworthy approach for security requirement engineering. This important work combines multi-criteria decision-making tools, particularly fuzzy TOPSIS, to compare several SRE approaches for developing safe and reliable healthcare applications.

In Port Harcourt, a city in Nigeria's Niger Delta, Adehor and Burrell (2008) created an intelligent MDSS for quick diagnosis of malaria and typhoid fever. Based on users' reactions to physical examinations for symptoms, their algorithm identified illnesses. The responses were used to calculate the degree of certainty of the condition, or the certainty factor (scaled from 0 to 100). They found that the certainty factor of typhoid was closer to the physicians' certainty factor. In another research, Sharma et al. (2013) used fuzzy logic to create a symptom-based MDSS for diagnosing malaria. Over 200 rules were developed using the knowledge acquired in this study, which included information from medical professionals, textbooks, and online resources. Internal medicine experts helped to construct the fuzzy rules. The suggested technique employed patient-identified symptoms to identify dengue fever and malaria. An expert diagnostic method for identifying human diseases was created by Patra, Sahu, and Mandal (2010). In the planned rule-based system, information was gathered from secondary sources, including specialist books, databases, and websites, as well as medical experts. Symptoms were categorized into three groups (key, sub, and unexpected group) and each group of symptoms was mapped against (newly) observed symptoms. Hence, indicating disease presence. However, several properties of their model require further investigation. Similar to this, Abiola et al. (2017) used data gathered from observation of medical records to construct a rule-based Lassa fever diagnosing system (LFDS). To develop rules and inputs for intelligent diagnosis, patient symptoms were used. These experiments' failure to provide predictive learning, however, limited the ability of their algorithms to forecast muddled symptoms. To assess the signs of tropical diseases in Nigeria, Omoregbe et al. (2020) created a chatbot service telehealth system based on fuzzy logic principles and fuzzy inference. The chatbot, the system, and the SMS subscriber were all connected via Twilio AP and the Telegram Bot Application Programming Interface (API), respectively. To accurately forecast the disease based on the inputted symptoms and to notify the user once the diagnosis procedure is complete, a fuzzy support vector machine (SVM) was used.

Oguntimilehin et al. (2013) proposed an ML approach for quick diagnosis of typhoid fever. A set of typhoid fever conditional variables was labeled to generate explainable rules classifying five different levels of severity for diagnosing the disease. Their implementation was carried out using Visual Basic as the front end and MySQL as the back end. Data on typhoid fever cases were collected from reputable hospitals in Nigeria, and a knowledge base was constructed from clinical records of selected diagnosed patients. With a total of 18 conditional symptoms, they generated rules using the rough set theory. A classification model was finally developed using the Learning from Examples Module (LEM2) algorithm (Grzymala-Busse, 2012). In a related study, Oguntimilehin (2020) created a smartphone app for diagnosing malaria utilizing 19 conditional symptoms and instances of datasets corresponding to the medical records of chosen patients. The symptoms were classified and labeled using a data mining method known as non-nested generalized exemplar (NNGE). To build the inference engine for an approachable mobile diagnosis application, rules were subsequently developed by the NNGE. Symptom datasets were learned, and classification results demonstrated great performance, but the datasets lacked confirmatory laboratory testing to validate the symptoms.

An expert system for diagnosing and advising malaria therapy based on user/patient-provided symptoms and confirmatory (blood test) data was developed by Adebayo et al. (2013). The expert system was developed using information from medical experts gathered through organized interviews and a thorough literature study. They used the waterfall software development approach for their design. An expert system named XpertMalTyph was proposed by Fatumo, Adetiba, and Onolapo (2013) to identify the side effects of typhoid and malaria. They developed the expert module of XpertMalTyph using MySQL, Java, and Java Expert System Shell (JESS). The approach was designed to allow patients with minor typhoid complications and mild instances of malaria to receive treatment without going to the hospital. However, there was no intelligent modeling or prediction component in these systems.

Ayo et al. (2020) proposed a bioinformatics-based DSS (BBDSS) for malaria, typhoid, and malaria-typhoid diagnosis, using a hybrid of expert systems and global alignment with a constant penalty. BBDSS accepts input diagnosis sequences of binary input patterns and benchmarks these patterns with patient signs/symptoms and domain knowledge experts. Using a set of IF-THEN rules guiding the diagnosis, decisions guiding the diagnostics were generated. An optimal alignment to determine the disease condition of the patient was finally achieved through alignment scores for three benchmarked diagnosis sequences. The datasets used by the authors were binary transforms (0 and 1), making it highly unlikely that they could accurately predict the degree or severity of the disease(s).

Using secondary data, Maidabara et al. (2021) created an expert system for detecting typhoid and malaria. Support vector machine (SVM), artificial neural network (ANN), and naive Bayes (NB) were the three ML algorithms they used. They discovered that the best classification accuracy was provided by SVM and NB. However, they did not use confirmatory (laboratory) testing to resolve confusable symptoms or corroborate doctor suspicions; instead, they completely relied on patients' medical histories and physical examinations. The authors' datasets also underwent binary changes (0 and 1), making it highly improbable that they could accurately predict the degree or severity of the disease(s).

Ozkan, Koklu, and Sert (2018) used examination data and diagnosis outcomes from 59 UTI patients to apply artificial intelligence methods (decision tree, SVM, random forest, and ANN) in the diagnosis of urinary tract infection (UTI). Among the suggested models, ANN had the highest UTI diagnosis accuracy score of 98.3%. A mobile-enabled plasmonic ELISA-based tuberculosis (TB) detection method employing cellphones and an ensemble classifier (random forest) was presented by Shabut et al. (2018). The Samsung Galaxy S7 edge was used to build and deploy the TB detection, and the results revealed 98.4% accuracy.

Our research incorporates collaborative crowdsourcing of input symptoms or data, a community-based strategy that collects both patient and physician viewpoints before a symptom is rendered, thereby removing diagnostic biases. This approach was inspired by the aforementioned literature. To address illness-specific ecosystems, additional experiential knowledge is also captured in the form of disease-specific thresholds establishing the universe of discourse (UoD) or cutoffs to disease burden estimation.

3. METHODOLOGY

Agile software development techniques with an emphasis on iterative user interaction were used to create the application for identifying febrile disorders. Based on the ideas of empathy in design thinking, the Agile model interweaves the processes with a considerable amount of iterative user participation. Through the use of empathy in design thinking, we were able to better understand the needs of potential users (FHWs), gain a thorough understanding of how FHWs interact with the health system, standing orders, and patients during the diagnosis process, recognize the potential impact that a diagnosis may have on patient's lives, and investigate the motivations and thought processes used by physicians and FHWs during the diagnosis process. The FHWs, who are the system's end-users, iteratively described the functional requirements of the system at each incremental phase.

3.1 Agile Software Development Methodology

The steps of the Agile software methodology used to construct the app for our diagnostic system are displayed in Figure 1 and are discussed below.

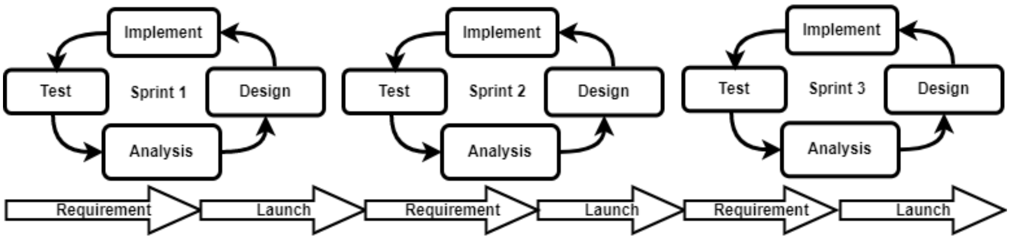

Figure 1. Iterative Process in Agile Software Development

3.1.1 Requirements Gathering

Relevant stakeholders and users were included in the iterative process to ensure a complete grasp of the proposed system features. The stakeholders' input on the functions was gathered through the many meetings that were held at this stage. For the elicitation, documentation, and comprehension of the system features by the stakeholders and users, three main and overlapping sub-processes were used. Major and minor use cases were gathered from stakeholders and users during requirement elicitation. The elicited needs were then organized and documented using the proper software engineering format as part of the requirement documentation process. The interaction between the users, FHW, facility head, record officer, patient, and administrator is shown graphically in the use case diagram in Figure 2. The patient assessment, appointment and follow-up, report module, patient clinical information, and personnel information categories make up the use case model. In three of the four modules, the four main stakeholders—the FHW, Facility Head, Record Officer, and Patient—share a variety of use cases.

3.1.2 Requirement Analysis and Validation

The acquired requirements were examined and confirmed through multiple organized meetings to make sure that all the stakeholders and team members had a consistent understanding of the product to be produced. The meetings included additional requirement elicitation and editing of the documentation created by taking into account the advantages and disadvantages of each need. The output from these sessions is a set of agreed-upon specifications for the proposed system, which is graphically represented in a context diagram as illustrated in Figure 3. The context diagram represents the interface between the system's environment and its boundary by depicting the entities that engage with it. Additionally, a Standing Order is a documented policy that directs FHWs in the primary healthcare setting during the diagnosis and treatment of illnesses. The app was created by integrating the Standing Order components for the index diseases. The steps for using a standing order to treat a sickness are as follows: Greeting the client and offering him/her a seat; Describing yourself to the client, reassuring them to relax; assembling the past records; doing a physical exam and making notes, if necessary, conduct more interrogations to confirm your findings; Adapt your treatment.

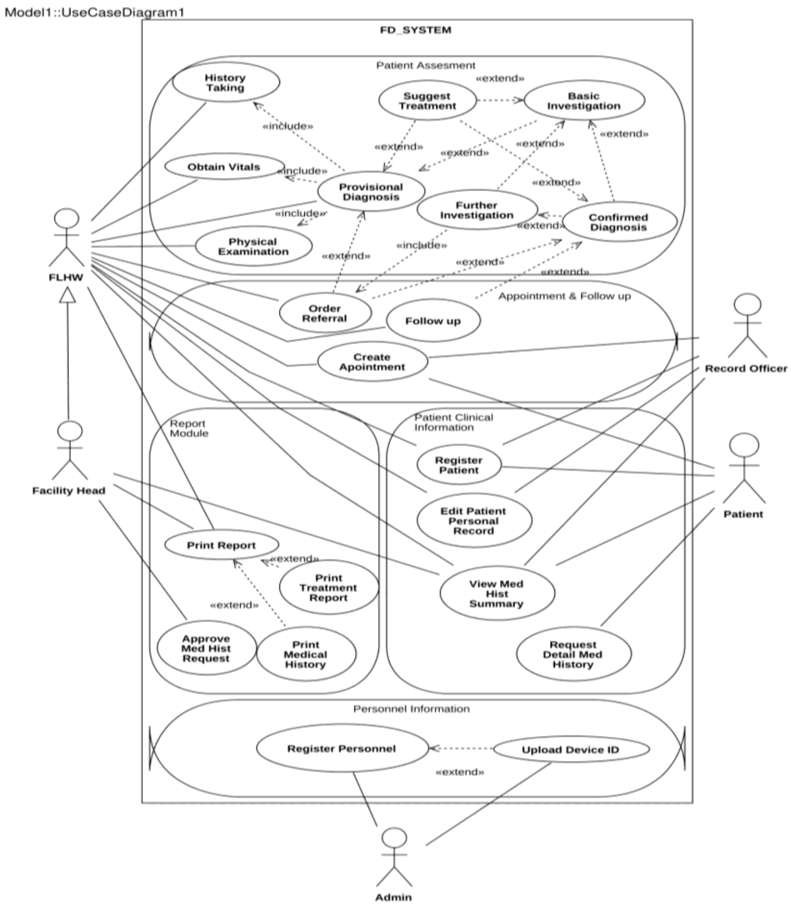

Figure 2. Use Case Diagram of the Febrile Disease Diagnostic System

3.1.3 Design

The primary design elements in this study are the user interface, knowledge base, and diagnostic engine. The interactions between the software (which includes the data storage, access logic, application logic, and presentation logic) and the hardware (the mobile device and the server) are also taken into account. In Figure 4, the architecture is displayed. The medical professionals, patients, user interface, knowledge base, diagnostic system and its sub-components (AHP diagnostic engine, decision support filters), diagnosis, and therapy are all included in the architecture. Sixty-two (62) doctors in secondary and tertiary healthcare facilities who are experts in the field of diagnosing febrile disorders and their related symptoms have contributed their experience-based knowledge to the knowledge base.

Additionally, it covers adequate patient assessment and test results, as well as risk factors that can make a patient more susceptible to various febrile disorders. This data aids in the creation of a model for several diseases with multiple symptoms that calls for their representation in an AHP structure to create the AHP diagnostic engine. The knowledge base's contents are supplied to the diagnostic system, where the decision support filters use the data to mimic how a skilled doctor would use the AHP diagnostic engine to start the relevant individual febrile disease model(s) for more research. The linear consensus models in the AHP engine determine which illness should be activated by the decision support filters. Based on the patient's symptoms and risk factors, the AHP diagnostic engines help identify the suspected illness and discriminate between febrile illnesses' symptoms, making it easier to recommend the best course of treatment. The FHW can also use the system's projected diagnosis to visualize it on the user interface in order to make fast and precise healthcare decisions on medical counsel and treatment.

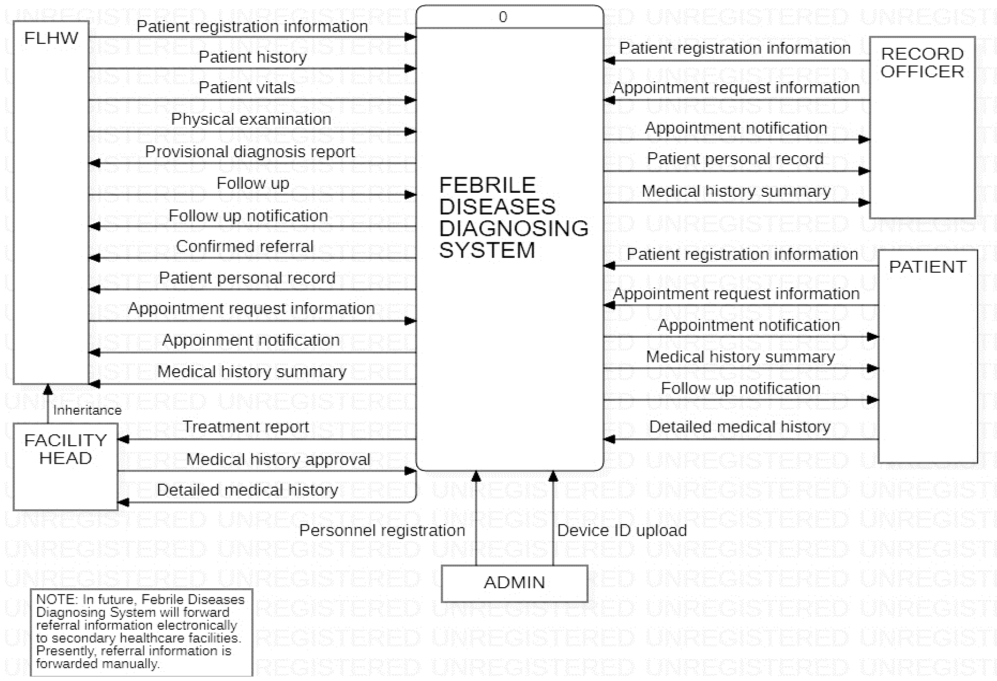

Figure 3. Context Diagram of the Febrile Disease Diagnostic System

The decision support filter of the malaria (MAL) diagnostic system employs a systematic method that is presented as follows. While fever (FVR) as a symptom is related to all eleven (11) febrile diseases, twenty (20) symptoms were interrelated with two or more febrile diseases. Based on the significance of each symptom to a condition and using the consensus preferences and confidence levels of clinicians, the system makes an educated diagnosis. A threshold was set for the prediction of each disease by the AHP engine with consideration of overlapping symptoms.

Utilizing a conventional AHP comparison matrix with each leading diagonal set to 1, the criteria-by-criteria pair-wise comparison for each illness alternative was assessed. Once the composite weight (final score) for each alternative is determined, the AHP approach calculates the relative weights of each criterion and aggregates the eigenvectors for each comparison matrix. The final weight vector's values represent the relative weights assigned to each alternative in relation to the choice problem's objective. The eigenvectors are then used by a DM to rank the options to choose the best choice. After obtaining the normalized vector matrix, the relative weight, also known as the weighted eigenvector, is computed as the matrix's row average. Each performance matrix's consistency ratio (CR) was calculated to determine the degree of inconsistency resulting from the DM's subjective assessment. If CR 0.1, the degree of consistency is acceptable; otherwise, the degree of consistency is high, and the DM is suggested to alter the matrix's components to achieve a more consistent matrix.

According to the linguistic score of each febrile disease's symptoms (1=absent; 2=very low; 3=low; 4=moderate; 5=high; 6=very-high) and the weighted Eigenvector (priority score) produced by the AHP process, the probability of occurrence of each febrile disease was calculated. Equation (1) represents the ADFI for MAL, and equation (2) represents the probability of occurrence (P_i). According to the computed ADFI, Table 1 displays occurrence probability ranges (Uzoka et al., 2017). The actors, roles, processes, and activities necessary for the system's design, documentation, comprehension, and deployment were described using diagrams created using the unified modeling language (UML).

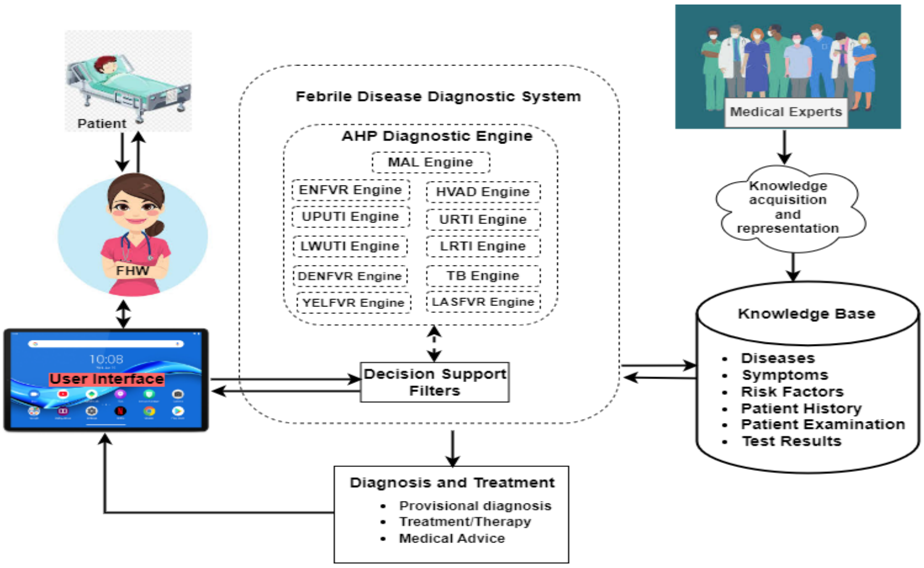

Figure 4. System Architecture

ADFI_MAL= (FVR_Score * 0.132092708) + (HGDFVR_Score * 0.124526459) + (CHLNRIG_Score * 0.115693613) + (GENBDYPN_Score * 0.114760331) + (FTG_Score * 0.108827326) + (JONPN_Score * 0.107823379) + (HDACH_Score * 0.120826663) + (PRTN_Score * 0.086728546) + (BITAIM_Score * 0.088720975) (1)

$$P_i = \frac{ADFI}{5} * 100 \qquad (2)$$

Table 1. Occurrence probability ranges

Uniform Rating	ADFI Range	Probability range (%)
1	0.000 - 1.000	0.01-20.00
2	1.001 – 2.000	20.01 – 40.00
3	2.001 – 3.000	40.01-60.00
4	3.001 – 4.000	60.01 – 80.00
5	4.001 – 5.000	80.01– 100.00

4. SYSTEM IMPLEMENTATION RESULTS

Xtensible Markup Language (XML) was the software tool used for front-end development, creating the graphical user interface (GUI) for mobile layouts, while SQLite, Python, and SQL were used for the back-end, creating application programming interfaces (APIs). The main Integrated Development Environment (IDE) for creating, compiling, and debugging the system app using Java as the programming language is Android Studio Dolphin 2021 v3.3. The binaries needed to create, maintain, test, and debug the system App are part of the Android Software Development Kit (SDK), Version 28. A library for Android that produces graphical representations of calculations and data is called MPAndroidChart: v 3.0.3. *Git, a source code management program used to keep track of changes in the code and make it possible for programmers to collaborate on this nonlinear development, and Adobe Illustrator, a vector-based graphics program used to produce icons and images that are inserted in the app layouts to provide users additional information.

The different screen outputs of the created app for the febrile disease diagnostic system are shown in Figures 5 and 6. The FHW must enter their worker identity (ID) on a login screen to gain access to the system. This will limit access and stop unauthorized users from accessing the system. When the FHW enters their ID information correctly, they may access the main dashboard in Figure 5 and register a new patient, look for an existing patient, or browse a list of all the patients who have already registered in the system. The patient account dashboard, depicted in Figure 6, allows the FHW to view prior and/or load current medical accounts for each patient, which may include vital signs, diagnoses, follow-ups, and appointments. The dashboard offers the workspace that the FHW utilizes to access various system modules.

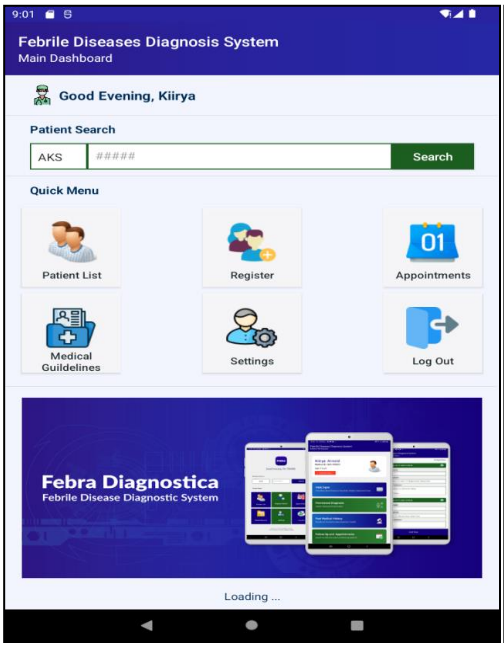

Figure 5. User Main Dashboard

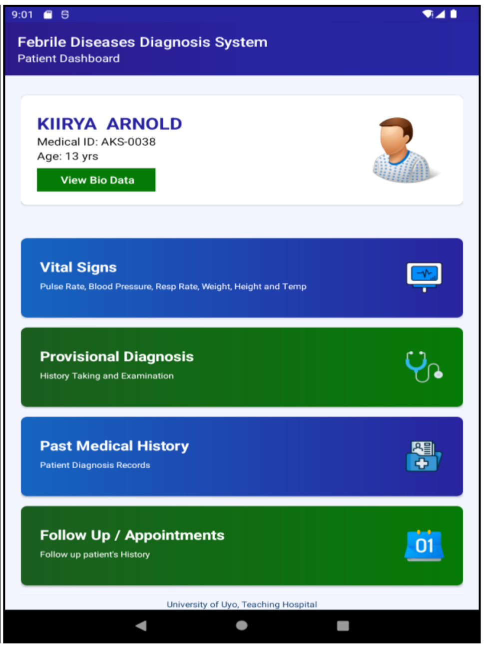

Figure 6. Patient Account Dashboard

Figure 7 is a snapshot that shows how the DSS stores patients' past and present vital records, and Figure 8 is a screenshot of the health status assessment that demonstrates how the FHW collects data on a patient's felt symptoms using sliders on a 5-point scale. Using a knob or lever that is moved horizontally to adjust a variable, sliders are GUI control elements. It merely enables FHWs to choose an approximate value (rather than an exact value) or range from a predetermined list of choices. To maintain focus, manage material, and prevent distraction, the FHW merely moves the seek bars corresponding to each symptom and sign level on the GUI. Our strategy made sure that this element's usability was not compromised by its attractive appearance. Additionally, aesthetic control makes sure that consumers make the right choices without exerting too much effort to obtain an exact value. Additionally, the slider labels are exposed below so they can be seen when the user chooses a value. Risk factors are loaded after sliding to the input data, which helps to improve the diagnosis procedure for precise decision-making. To assess ADFI and the likelihood of each disease diagnosis, these values are coupled with the eigenvalues for each symptom. Figure 9 demonstrates how the system takes into account the potential effects of specific risk factors that could prompt additional research for a specific febrile disease's accurate diagnosis while the initial and final diagnoses (ID and FD) for each confirmed febrile illness are shown in Figure 10 of the diagnosis report, together with any relevant risk factors. Results from the examination of risk factors show that the AHP model's performance in terms of diagnosis accuracy closely resembles that of domain experts.

Figure 7. New Vital Records

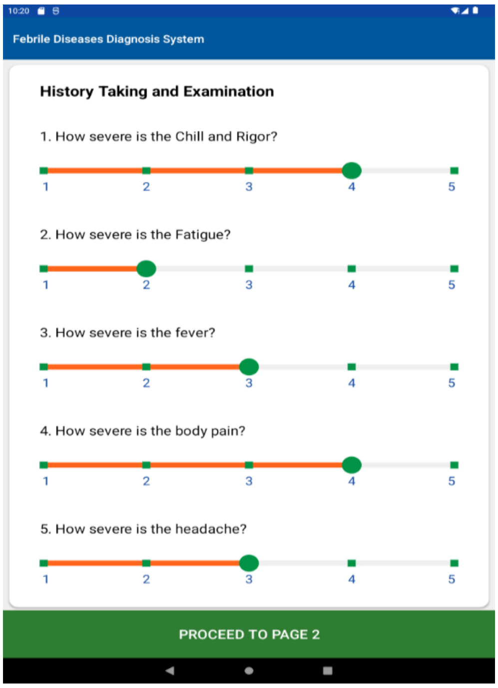

Figure 8. History Taking and Examination

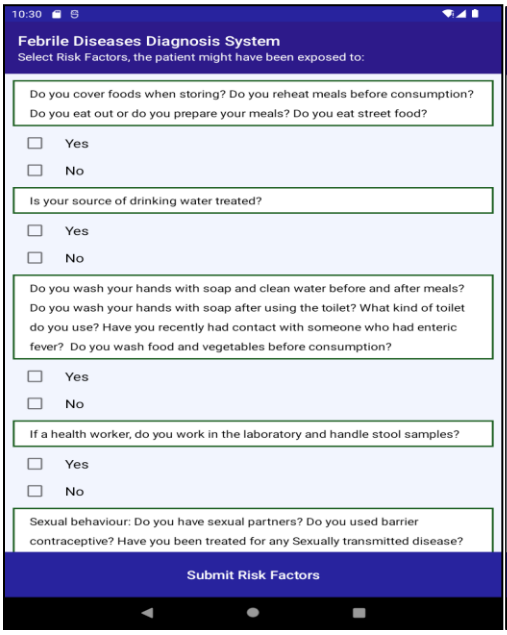

Figure 9. Risk Factors Assessment

Figure 10. Diagnosis Report

Figure 11 shows the basic investigations, where the FHW can carry out Rapid tests in the laboratory, and fill in the correct results by selecting Positive, Negative, or Not done for cases when there are no rapid test kits available. The FHW at this point can also generate a referral note by clicking on the “Make referral “button before finishing the process and Figure 12 shows the medical record for that specific day which can be printed out.

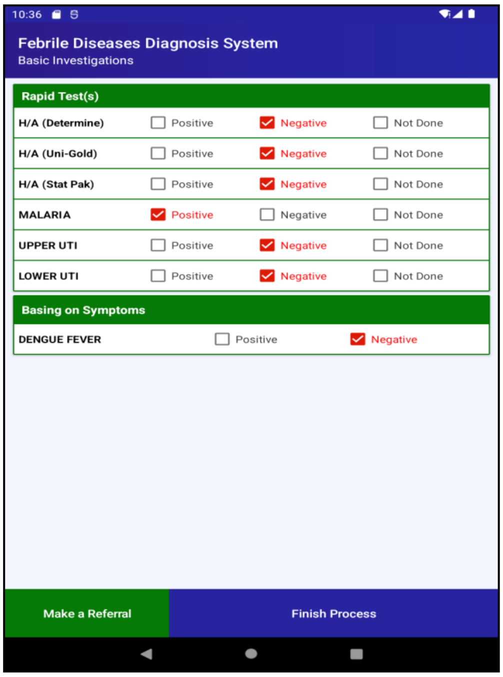

Figure 11. Basic Investigation

Figure 12. Diagnosis Report

5. DISCUSSION OF RESULTS

The AHP model used in this work shows a diagnosis accuracy and precision of 90% and 91% respectively computed from the confusion matrix. A high accuracy signifies that a large percentage of the predictions are accurate by evaluating how accurately the model's predictions are made in general. In addition, high precision signifies that when the model predicts a positive case, it is likely to be correct. Febra Diagnostica takes the vital signs of patients and obtaining vital signs during diagnosis is a crucial component of medical evaluation that gives important information for prompt diagnosis, therapy, and patient monitoring. It enables medical practitioners to make wise choices, enhance patient outcomes, and guarantee the provision of high-quality care (Zhou, 2023). The consideration of risk factors during the diagnostic process is also crucial for thorough and efficient patient care because it takes into account both the patient's current health situation and probable future health trajectory. This app will also create a large dataset that can contribute to medical research studies, especially in the area of febrile disease diagnosis.

Data collected for the study is a mixture of both female and male patients' records in addition to those of vulnerable populations such as pregnant women and those with disabilities. The team of medical doctors, the FHWs, and the information scientists involved in the development of the app are in the ratio of 3:2 for males and females respectively. The app is not to be directly used by a patient but by the FHWs who apply the diagnostic procedure and guidelines they are trained on in attending to the patients. The app serves as an MDSS leaving the final decision for the FHWs to make sometimes in consultation with a medical doctor. In most cases, laboratory investigation is recommended by the app and the results of such investigation are fed into the system and reprocessed for the final diagnosis.

The dearth of experts on febrile disease diagnosis in the temperate regions calls for concern of travelers from the tropics to the regions. Such travelers at times experience symptoms of febrile diseases with no physicians to treat them. The developed app could serve as a tool in such a situation. This will curtail the rate of morbidity and mortality caused by febrile diseases.

In the course of analyzing the datasets obtained for this study, it was observed that some of the symptoms contribute insignificantly to the diagnosis of the associated disease. Diagnosis results for the disease are the same irrespective of the presence or absence of such symptoms. Thus, the study could serve as a guide on the specific symptoms that give an accurate diagnosis without the stress of eliciting other symptoms.

6. CONCLUSION

The development of a health information system such as an MDSS requires the involvement of key players in the healthcare sector along with information scientists. The interactions between the information scientists and health care providers assist in eliciting the requirements needed in the design phase of the development. The agile software methodology provides the needed structure for the planning, design, implementation, and regular review of the stages until a desired app is obtained. The elements of design thinking to ensure excellent user experience were also incorporated in building the mobile app with functional features and components usable across a range of devices and platforms.

The app is specifically built for the operations by FHWs as the eventual users to conduct primary patient diagnosis in rural communities of LMICs. With 3253 datasets gathered from 16 hospitals in the Niger Delta region of Nigeria, the AHP was used as a modelling tool which produced 90% and 91% accuracy and precision respectively when compared with the conventional diagnosis carried out by the medical doctors in the hospitals. Armed with such results, the AHP model was used to build the engine of the mobile app. Testing and subsequent deployment of the app have proved successful in the task of differentiating febrile diseases and accurate diagnosis of these diseases by the FHWs in the Niger Delta regions. The febrile diseases diagnosed by this app are known to be responsible for high mortality in the tropical regions of the world (Macicame et al., 2023; Magalhães et al., 2023; Dean et al., 2023). Further studies in the use of machine learning are suggested with the hope of getting better results than what is obtained in this study.

REFERENCES

Abd Ghani, M. K. et al. (2018). Proper Software Engineering Process in Developing an Integrated Telehealth System. *International Journal of Engineering & Technology*, Vol. 7, No. 3.30, pp. 441-450.

Abiola, H. M., Alaba, A. A. and Luka Joy, D. (2017). Expert system for Lassa fever diagnosis using rule-based approach. Annals. Computer Science Series, Tome 15, Fasc. 2, pp. 68-74.

Adebayo, A. O., Fatunke, M., Nwankwo, U. and Odiete, O. G. (2013). The design and creation of a malaria diagnosing expert system. *Compusoft*, Vol. 2, Issue 12, 442-449.

Adehor, A. B. and Burrell, P. R. (2008). An intelligent decision support system for the prompt diagnosis of malaria and typhoid fever in the malaria belt of Africa. *IFIP International Conference on Artificial Intelligence in Theory and Practice*. Springer, Boston, MA., pp. 287-296.

Ahmad, A. et al. (2021). Software engineering for IoT-driven data analytics applications. IEEE Access, Vol. 9, pp. 48197-48217.

Ansari, M. T. J., Al-Zahrani, F. A., Pandey, D. and Agrawal, A. (2020). A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development. *BMC Medical Informatics and Decision Making*, Vol. 20, No. 1, 1-13.

Attai, K. et al. (2022). A Systematic Review of Applications of Machine Learning and Other Soft Computing Techniques for the Diagnosis of Tropical Diseases. *Tropical Medicine and Infectious Disease*, Vol. 7, No. 12, p. 398.

Ayo, F. E. et al. (2020). A decision support system for multi-target disease diagnosis: A bioinformatics approach. *Heliyon*, Vol. 6, No. 3, p.e03657.

Dean, L. et al. (2023). A health-systems journey towards more people-centred care: Lessons from neglected tropical disease programme integration in Liberia. Health Research Policy and Systems, Vol. 21, No. 29.

Fatumo, S. A., Adetiba, E. and Onolapo, J. O. (2013). Implementation of XpertMalTyph: an expert system for medical diagnosis of the complications of malaria and typhoid. *IOSR Journal of Computer Engineering*, Vol. 8, No. 5, pp. s34-40.

Grzymala-Busse, J. W. (2012). An empirical comparison of rule induction using feature selection with the LEM2 algorithm. *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, Berlin, Heidelberg, pp. 270-279.

Hasan, M. et al. (2023). Software Engineering Methodology for Smart Healthcare Security and its Application in Bangladesh. *American Journal of Science and Engineering*, Vol. 3, Issue 4, pp. 26-32.

Hauschild, A. C. et al. (2022). Guideline for software life cycle in health informatics. Iscience, Vol. 25, No. 12.

IEC (2015). *IEC 62304:2006/Amd1:2015 Medical Device Software - Software Life Cycle Processes: Ammendment 1*. https://www.iso.org/standard/64686.html

Kaur I. et al. (2018). Neuro Fuzzy - COCOMO II model for Software Cost Estimation. *International Journal of Information Technology*, Vol. 10, No. 2, pp. 181-187.

Lalband, N. and Kavitha, D. (2019). Software engineering for smart healthcare applications. International *Journal of Innovative Technology and Exploring Engineering*, Vol. 8, Issue 6S4, pp. 325-331.

Macicame, I. et al. (2023). Countrywide mortality surveillance for action in Mozambique: Results from a national sample-based vital statistics system for mortality and cause of death. The American Journal of Tropical Medicine and Hygiene, Vol. 108, 5 Suppl., pp. 5-16.

Magalhães, A. R. et al. (2023). Neglected tropical diseases risk correlates with poverty and early ecosystem destruction. *Infectious Diseases of Poverty*, Vol. 12, No. 1, pp. 1-15.

Maidabara, A. H., Ahmadu, A. S., Malgwi, Y. M. and Ibrahim, D. (2021). Expert system for diagnosis of malaria and typhoid. *Computer Science & IT Research Journal*, Vol. 2, No. 1, pp. 1-15.

Moreb, M. and Ata, O. (2019). *Software Engineering for Machine Learning in Health Informatics*. Independently published.

Nunthavichitra, S. et al. (2020). Case report: COVID-19 presenting as acute undifferentiated febrile illness — a tropical world threat. *The American Journal of Tropical Medicine and Hygiene*, Vol. 103, No. 1, p. 83.

Obot, O. et al. (2023a). Development of a mobile app diagnostic system for tropical febrile diseases. *Proceedings of the International Conference on e-Health (EH 2023)*, Porto, Portugal, pp. 15-17.

Obot, O. et al. (2023b). Modelling Differential Diagnosis of Febrile Diseases with Fuzzy Cognitive Map. *Tropical Medicine and Infectious Disease*, Vol. 8. No. 7, p. 352.

Oguntimilehin, A. (2020). A mobile malaria fever clinical diagnosis system based on non-nested generalized exemplar (NNGE). *International Journal of Emerging Trends in Engineering Research*, Vol. 8, No. 2, pp.259-264.

Oguntimilehin, A., Adetunmbi, A. O. and Abiola, O. B. (2013). A Machine Learning Approach to clinical diagnosis of typhoid fever. International Journal of Computer and Information Technology, Vol. 2, Issue 4, pp. 671-676.

Omoregbe, N. A. et al. (2020). Text messaging-based medical diagnosis using natural language processing and fuzzy logic. *Journal of Healthcare Engineering*. https://doi.org/10.1155/2020/8839524

Ozkan, I. A., Koklu, M. and Sert, I. U., (2018). Diagnosis of urinary tract infection based on artificial intelligence methods. *Computer Methods and Programs in Biomedicine*, Vol. 166, pp. 51-59.

Patra, P. S. K., Sahu, D. P. and Mandal, I. (2010). An expert system for diagnosis of human diseases. *International Journal of Computer Applications*, Vol. 1, No. 13, pp. 71-73.

Shabut, A. M. et al. (2018). An intelligent mobile-enabled expert system for tuberculosis disease diagnosis in real time. *Expert Systems with Applications*, Vol. 114, pp. 65-77.

Sharma, P., Singh, D. B. V., Bandil, M. K. and Mishra, N. (2013). Decision support system for malaria and dengue disease diagnosis (DSSMD). *International Journal of Information and Computation Technology*, Vol. 3, No. 7, pp. 633-640.

Stoica, M., Mircea, M. and Ghilic-Micu, B. (2013). Software development: Agile vs. traditional. *Informatica Economica*, Vol. 17, No. 4, pp. 64-76.

Udo, E. N., Obot, O. U. and Obike, P. G. (2022). A Comparative Analysis of Adaptive Neuro-Fuzzy Inference System Back Propagation and Hybrid Learning for Software Development Effort Evaluation. *International Journal of Software Engineering (IJSE)*, Vol. 8, Issue 7, pp. 23-38.

Uzoka, F. M. E. et al. (2017). AHP model for diagnosis of tropical confusable diseases. *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, pp. 1758-1763.

WHO (2016). *Global Strategy for Human Resources for Health: Workforce 2030*. https://apps.who.int/iris/bitstream/handle/10665/250368/9789241511131-ng.pdf?sequence=1&ua=12

Zhou, Y. (2023). *Integrated Stretchable Circuits and Machine Learning for Human Vital Signs Monitoring*. University of California, Irvine.