International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I10P122

Volume 72 Issue 10, 148-159, October 2024
© 2024 Seventh Sense Research Group®

Original Article

Real-Time EHR Analysis and Predictive Healthcare Decisions: A Novel API Framework

Saigurudatta Pamulaparthyvenkata[1], Ramesh Babu Radhakrishnan[2]

[1]Independent Researcher, Bryan, Texas, USA.
[2]Independent Researcher, Principal Engineer, Texas, USA.

[1]Corresponding Author : saigurudatta.pamulaparthyv@gmail.com

Received: 08 September 2024 Revised: 10 October 2024 Accepted: 24 October 2024 Published: 31 October 2024

Abstract *- This research introduces HEALTHCARE API (Healthcare Analytics and Real-time Monitoring ON You), a framework designed for advanced Electronic Health Record (EHR) analysis, continuous patient monitoring, and real-time clinical decision-making. Built on the SMART on the FHIR platform, it ensures seamless integration across healthcare systems. A key technical contribution is using Python wrappers, simplifying API interactions by abstracting complexities and enabling efficient EHR data management. The API is based on Python libraries such as Pandas and NumPy to support solid integration of the data that could be cleaned, transformed, and analyzed over large healthcare datasets efficiently. Besides, it addresses interoperability issues, allowing free and complete data flows between systems while assuming incompatible data formats such as JSON, XML, and CSV by transforming them correspondingly. It optimizes performance, minimizing network latency to improve server response times and ensure reliable data synchronization. HEALTHCARE API complies with healthcare regulations, such as HIPAA, using secure data transmission and storage strategies. Additionally, the framework uses AWS EC2 instances for scalability, ensuring dynamic scaling to handle large volumes of data. Automation of data flows is achieved through the use of Apache Airflow, enhancing efficiency and operational reliability. Thereby, HEALTHCARE API integrates machine learning, predictive analytics, and real-time data processing, offering actionable insights that empower healthcare providers to make informed, personalized care decisions.*

Keywords *- HEALTHCARE API, Electronic Health Records, Predictive Analytics, Real-Time Patient Monitoring, Healthcare and Interoperability.*

1. Introduction

There are a number of possible advantages to switching between medical records on paper towards Electronic Health Records (EHRs). Redundancy mitigation is one benefit of sharing patient data throughout healthcare providers. Furthermore, the use of EHRs results in an improvement in the caliber of services. Doctors and specialists can respond more quickly by making more accurate and well-informed decisions when patient medical histories are shared [1-3]. However, the current heterogeneity of health record data and processes makes information system integration extremely difficult. Several internationally recognized norms have been proposed for the transfer of medical information: X73PHD [4] (an implementation based on ISO/IEEE 11073), NANDA-I [4], openEHR [5], ISO/IEEE 11073-a PhD [6,7], and X73PHD [8], which is an implementation of ISO/IEEE 11073.

The standard established by HL7 has gained recognition from all of these and is currently being extensively utilized in developing systems for information in the medical domain [9-16]. In fact, these standards were recognized by ANSI, an American National Standards Institute, in 2003 [17], and the HL7 organization was instituted in 1987 with a mission of enabling electronic sharing of data in the health area. HL7, 2024. Although HL7 v2.x and HL7 v3 have some structural and methodological differences, HL7 v3 has a higher usage rate in current clinical practice. The newest version, developed based on versions 2.x and 3.x, is HL7 Quick Healthcare Interoperability Resources, FHIR [17]. The simpler implementation of FHIR enables using the HTTP protocol and RESTful principles to build efficient services. Graphical interfaces are generated through HTML and cascading style sheets, which make the user interface flexible enough to integrate into the broader context. FHIR also allows users to choose from JSON, XML, and RDF to describe the detailed information in the resultant data. FHIR has been criticized technically despite its advantages, especially in creating apps for API interoperability, where its implementation might become complicated. RIM version 3 is criticized once more, this time for its implementation, usability, scope, and records, among marketing issues [18]. Three major contributions of the HEALTHCARE API framework include:

- The HEALTHCARE API leverages Python wrappers to simplify complex API interactions, abstracting technical details into more manageable functions. This streamlines EHR data integration, making it easier for developers to work without delving into intricate API specifics.
- The API enhances data cleaning, transformation, and analysis processes by adopting robust libraries like Pandas and NumPy in Python. These tools process large volumes of healthcare data efficiently. They can integrate a variety of data streams from wearables, remote monitoring, and connected health applications into continuous patient tracking.
- IAPI wrappers support easy data exchange between different healthcare systems for the correct, safe, and efficient transfer of patient data. Effective transformation techniques overcome the problem of data format incompatibility (JSON, XML, CSV) and enable integration without disturbances or lags. It offers support for various communication protocols like REST, SOAP, and GraphQL, enabling further flexibility in their integration.
- With proper planning and testing procedures, the API versioning process would not be difficult; hence, clients would upgrade smoothly. Overall performance will be improved through a reduction in latency in the networks, improvement in the responses of the servers, and proper data copying.
- Using cloud-based AWS EC2 instances offers scalability and flexibility to handle large datasets, with the automation of data flows enabled through Apache Airflow, enhancing overall system efficiency and reliability.

These will collectively impact the ability of healthcare providers to deliver personal data-driven care and improve overall patient outcomes.

2. Related Work

In order to give patients an accurate medical diagnosis and treatment, it was discovered following the COVID-19 pandemic that effective and secure medical information interoperability is crucial. Clinical records must be shared to increase the body of knowledge and treatment efficiency in the health industry. A person's health state is documented in electronic health records or EHRs. The difference between health information's technical implementation and portrayal is the main topic of this standard. As a result of this separation, electronic health systems can evolve over time and become more versatile and adaptive, further enhancing patient health [19]. This has a wide effect on public wellness because it's accustomed to making forceful diagnoses in healthcare treatments. From an international perspective, robust connectivity platforms and interoperability standards aim to overcome information integrity problems within the health industry [20]. The major challenge in integrating information systems is the coexistence of heterogeneity across data and health records procedures. Most countries' health systems are complex, and the use of standards to automate them needs to balance the complexity of the details in the clinical record that must be interoperable across the many methods systems making up that country's complex health system.

2.1. Medical Information Interoperability

The development of a mobile application for the HL7 protocol-based biosignal transmission in the medical industry utilizing interoperable API. The suggested method uses ubiquitous computing technologies to assess health at home and is tailored for a home healthcare (uHealthcare) setting. It is determined which are the primary elements of a pervasive medical system: identification, communication, diagnosis and prescription. This paper highlights a strategy for creating an HL7 Application Programming Interface with LabVIEW, focusing on the data transformation and radio frequency transmission of medical information to other systems. In this app, data are sent via a serial method from the biosignal of the SpO2 sensor to a mobile device. The raw data is converted into the HL7 standard in the cell phone, which then sends it via Bluetooth or a WLAN to other PCs.

The abstract message's structure was established in the course of system development as a part of the message-modeling content. This field of study has moved from basic logic based on wearable readings from sensors to sophisticated data processing in the area of health monitoring systems to provide end users with more useful information. The work of [21] made strong emphasis on the crucial function data mining methods play in the eliciting of insightful information from wearable sensors in systems for monitoring health. These methods incorporate the identification of anomalies, the forecasting of future patterns from past sensor data, the formulation of well-informed decisions regarding the health state of each individual, and the processing of continuous time series observations. These developments do, however, also present issues and worries, such as guaranteeing the correctness and dependability of data gathered, optimizing data processing effectiveness, and validating in real-world healthcare environments.

HL7 mapping feature, v3-RIM version, was prototyped in the work of Viangteeravat et al. using R-MIM classes in order to combine remote clinical data sources. A prototype utilizing the Clinical Data Management Systems clinical databases was interfaced to this module. The authors mention that because much of the information is already available in CDMS, there should be considerable improvements in information dissemination during testing using the existing CDMS. Root element enables the distinct identification of the injection event, converse, message identification and recipient duties. The information from the interaction code IN was enclosed in a wrapper or decorator design pattern in the system architecture. Message type, sender, recipient, decorator templates of the control act, and injector event are the components of IN that appear in the real message. Along with encapsulating the real message, the initializing event and the control act decorators also contain information about the date, time, and persons responsible for the event. One of the main advantages of HL7 v3 is using the XML data model, which brings data and information into one standardized form and acts like a real-time protocol. In this setting, however, it needs to be recalled that transaction latency also remains an essential component in clinical database interactions. The accessibility issue in the population-based patient registers was addressed in the work of [22]; the application of a federated semantic MDR was proposed. It was highlighted that one can hardly believe that one single health data standard would be sufficient to solve all interoperability problems because of different systems for health data, healthcare infrastructures, and the requirement to communicate with antiquated systems. The authors claim that this suggested framework can function across standards, promote data federation, ease data linking between disparate systems, and do away with the requirement for centralized data collecting procedures. The framework's potential is emphasized, although certain challenges are also listed. Second, consensus among CDEs at the inter-domain level is necessary to achieve interoperability between PBPR domains. Further, the establishment of a regional semantics master data repository and configuration of RESTful services for each metadata and data source could be difficult for lesser or underfunded registries. Another framework has been proposed. It provides semantic compatibility on a platform, allowing various technologies, platforms and guidelines to efficiently coexist and interact.

This makes it easier to create multiplatform services and apps, which encourages the sharing and uptake of multiplatform apps and services for a healthy and engaged aging (AHA). The framework's functionality can be accessed through a single point of entry, the AIoTES API. Access to internal elements, the data analytics layer, data lake, and semantic interoperability layer is made possible via its uniform RESTful operations. Furthermore, it provides unified access to the tool interfaces, guaranteeing transaction security. The architecture presented in [23] focuses on the implementation of an API service that is RESTful. This API checks the status of key study personnel, IRB project approval, permission in the REDCap project, and individual access rights in EHRs each time there is a registration by a last user within a RedCap undertaking at the time of access. Using the HL7 FHIR standard that was implemented [24], a new module was developed that makes gathering data in REDCap for CRF research, registries, and repositories of records more easily. The suggested module was shared among institutions using REDCap, and its use was found in several high-profile investigations at the University Medical Center of Vanderbilt.

2.2. Application Programming Interface Challenges

Application Programming Interface, or API, is a collection of guidelines and sources that are utilized to communicate various software applications with each other. Using APIs, the developers can integrate various kinds of software components by the methods and data described in APIs. The systems that make use of an API see it as a kind of "black box" as their internal details are wrapped up. Still, it gives access to a set of protocols and functionalities that are supposed to provide device, platform, and information system compatibility. Representational State of Transfer (REST) is the base architectural concept of RESTful API, according to [25]. The base of guidelines consists of sending the representation of resources, most commonly in XML or JSON format, and allowing interaction with resources by URLs using basic HTTP protocols: GET, POST, PUT, DELETE.

The "imaging framework" is suggested by [26] as one that might solve problems of extensibility and interoperability within Java frameworks for image processing applications. Due to its modular and extensible design, platform freedom, and broad operating system support, Java is a good, flexible option for all kinds of jobs involving images. Moreover, computer vision and image processing algorithms executed on GPU-accelerated frameworks make Java even more efficient. Although Python has more popularity in many domains, it also has a really strong and great big community. In turn, [27] investigated the characteristics of certain languages that cooperate with Java in the context of API development with the use of that language. The most salient advantages are based on object-oriented programming concepts and syntactic similarities promoting interoperability.

The focus of Saleh's proposal is to create a solution with respect to using the "wrapper" style of design, which will run Java programming in Eolang. Further, it was noted that having the capacity to host certain syntax is the primary means of fostering interoperability. Also, in promoting interoperability with Eolang, extending the set of atoms by including Java wrappers is recommended. It is also important to consider chronic scenarios, although norms related to data interoperability in healthcare domains have been defined. Heterogeneity at software and hardware levels can include a medical hub of everything from small cell phones to cloud-integrated systems. Effective memory management can be necessitated by the unpredictability of transaction volume that can occur in a single query. APIs are, therefore, required to be portable and support limitless transactions dynamically and effectively, which is the needed ingredient of interoperability frameworks. In this proposal, API was constructed using concepts of the substandard HL7-CDA version 3. As the architecture suggests, object-oriented design patterns coupled with a dynamic memory strategy are used for efficiency. In general, integration among design patterns like wrapper and abstracts factory is quite critical for information management in regard to medical records. Our approach increased the efficiency by optimizing both memory and temporal resources spent during object construction throughout its execution in various contexts.

3. Design of the API Model

The architecture of the suggested API was created to share patients' initial clinical encounter medical records or records from the general medicine speciality. The API layers integrated the DICOM guidelines [28] for imaging study documentation, predicated on the HL7 criteria for clinical documentation. The work identifies four big perspectives, represented by the Figure 1 use case diagram: patients, healthcare providers, government agencies, and medical facilities.

3.1. Healthcare Professionals View

In this approach, all patient-related information can be accessed and updated by the medical staff at the hospitals or health centres starting, including doctors, paramedics, nursing staff, and other types of healthcare staff. This will involve electronic medical records, laboratory results, and requests for medication. The patient will be the central focus of this perspective. It will, therefore, have access to the imaging area and share patient results based on a unique identifier for each study, activity, or message. This shall be according to the official specifications of CONF-DIR and in line with the DICOM imaging research standard. It proposes that the data collected should be filled in by the doctor using a form available on the API, and it should follow the HL7 standard's CDA clinical document architecture.

General personal information regarding a patient would be included in the header. The number of sections making up the body depends upon the research that needs to be done and collected for the medical professional to arrive at the correct diagnosis. Any ancillary information the medical expert believes is essential for the whole patient file will also be part of the document's body. The doctor can include as many sections as possible, provided each comes with a different number.

3.1.1. Patient View

The patient’s participation in API management will be minimal. They won’t have the ability to modify their medical records. They will only be able to use the system for inquiries, downloading certifications from their general practitioner, and updating their personal details as necessary.

3.1.2. Healthcare Institutions View

The hospital or health center staff members will be admitted by this perspective, enabling them to communicate with the medical data system. This would make it easier for clinical data to be transferred between various healthcare facilities, essential for coordinated and continuous medical care.

3.1.3. Governmental Organizations View

This supports the collection of standardized information for epidemiological and public health research. It will also simplify the evaluation of healthcare quality and formulation of health policies on both national and global scales.

3.2. API’s Architectural Design

It has devised a cloneable and distributable design for the health care interoperability API whose instances would link back to hospitals or medical centers to obtain data regarding the patient's medical history. Figure 2 shows the proposed architecture, which was designed using patterns for wrapper and abstract factory designs. An abstraction of a factory (Factory_HCE) that creates runtime objects of type HCE based on the quantity and qualities of medical facilities or medical centers required to get in touch with and recover the patient's medical information allows the API to replicate itself in the event that a patient arrives and needs a record. Based on established connections to medical centers, the information provided by the Medical History Wrapper Course shall include information from the patients, encapsulated through references to the Patient Identity Data class, references to records of medical documents, all details pertaining to incidents connected to the ailment for which a search was carried out, under the supervision of the CDA classes. The HL7 CDA protocol format aggregates these medical records for patients who contact hospitals or medical institutions. Previous ailments or circumstances; medical procedure records; reports on imaging investigations (MRIs, X-rays, etc.); vaccination history documentation of immunizations received, including dates of administration; Nota clinical: documentation of doctor appointments and consultations; observations accompanied by remarks from the medical team. Prescriptions, therapies, medications, and information about specific therapies.

Insurance and billing information (financial, billing, and health insurance information about the patient). Contact information for emergencies, including information about who to contact in the event of a disaster. Lastly, the Lab Outcomes and Studies course is part of a group that contains links to resources for learning about imaging studies. According to the use case diagram, asking the viewpoint of a healthcare expert could start an API action. The progress structure for the HCE class based on the medical background is displayed in Figure 3. Figure 3 depicts the various states of an API operation: the data validation stage, the processing of the data stage, and the update record stage. The validation state is created by the set function of the validation data get_Patient_Identification_Data. It finds information about verified users in the medical system through a method called obtain_List_Of_Previous_Conditions_Or_Illnesses. The wrapper would update and encapsulate the records in the HL7 format utilizing the getter, setter, and get_Record_Of_Performed_Medical_Procedures functions in the event that patient data is discovered. Only links within the imaging area would be considered in investigations that include images. This process sends an SMS message with the complete record. In this way, the abstract factory will create each instance depending on the records that the patient will have at different hospitals or clinics. The information acquired from the instances of every medical center goes to the asking medical center and is combined for the request, as the interoperability action in Figure 4 illustrates. Table 2 illustrates the format of an XML document generated by the Medical Staff Members View when it asks for details regarding a patient's medical history containing the following: slight hypoventilation with wheezing, the inguinal lymph node along with aberrant behavior, a heart rate of 95, a respiratory rate of 28, and a periscapular inguinal lymphatic node in normal symptoms. Table 1 outlines the specifications that follow the perspectives with regard to functional needs.

Table 1. Descriptions of the API's functional requirements

ID	Description
FR1	Authorized users can log into the system using their username and password for authentication.
FR2	The system must accommodate user profiles based on roles and organizational structure, offering four distinct views: patients, healthcare institutions, healthcare professionals, and governmental entities.
FR3	Healthcare Professionals: Those working in hospitals or health centers can access and modify patient data, including adding, deleting, and reviewing studies and Electronic Health Records (EHRs).
FR4	Patients: This user can only perform searches, download studies and certificates, and review studies and EHRs.
FR5	Healthcare Institutions: Users in this category can update information about a health center's or medical unit's staff, including registering medical professionals and reviewing research.

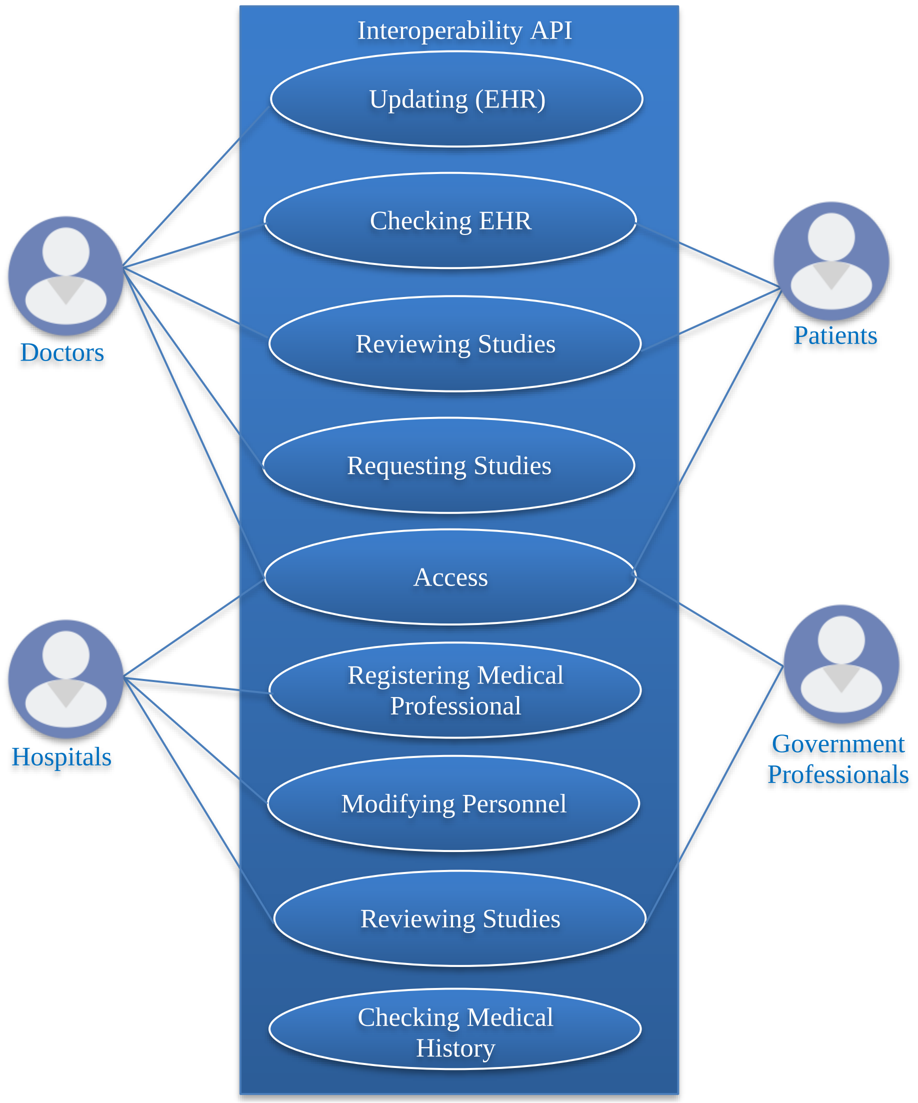
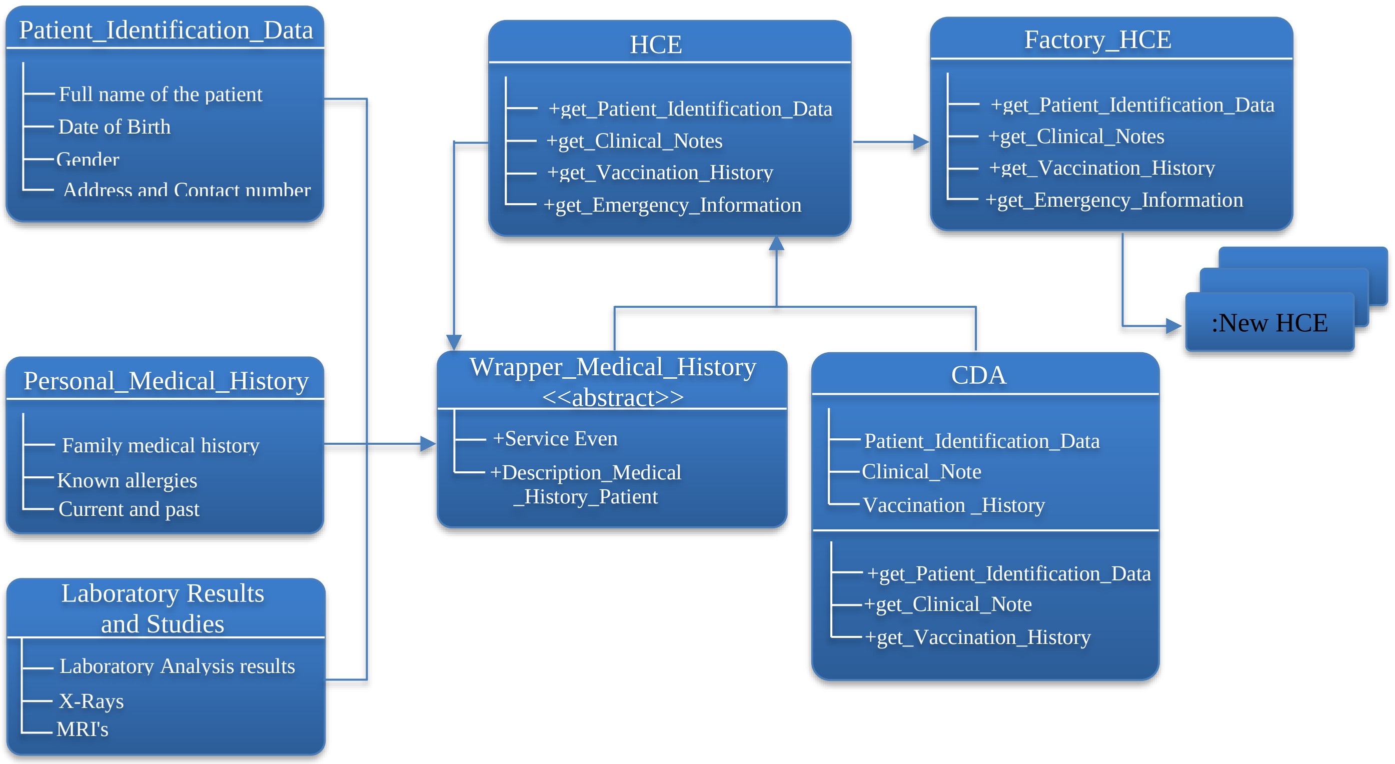
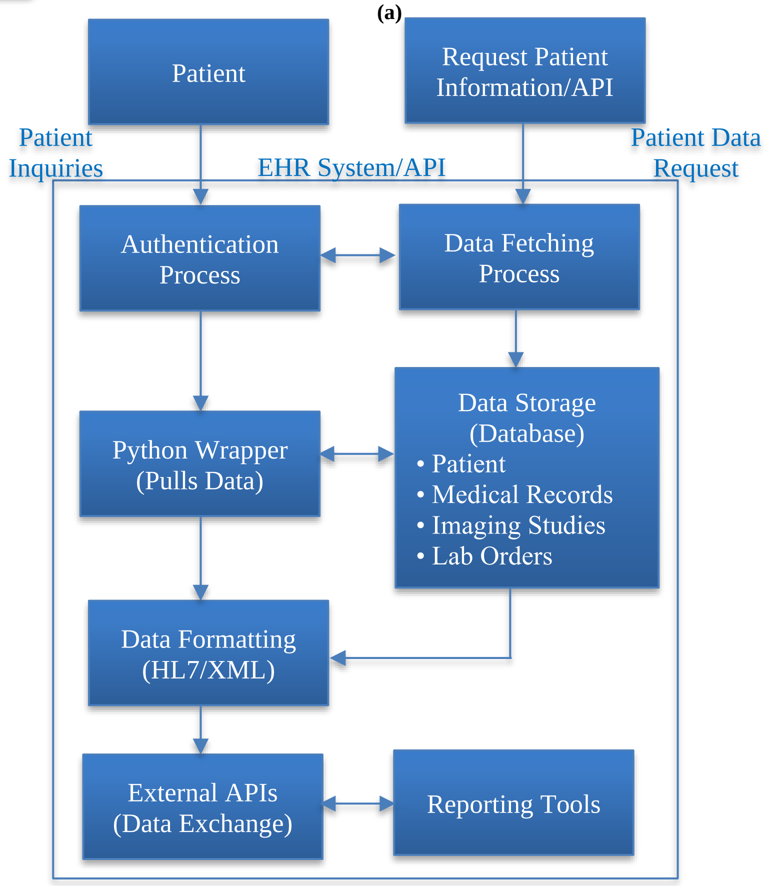


Fig. 1 Interoperability APIs use-case



(a)



(b)

Fig. 2 Flow diagram (a) Architecture for Interoperability API Design, (b) Flow of Data pipeline and its Components

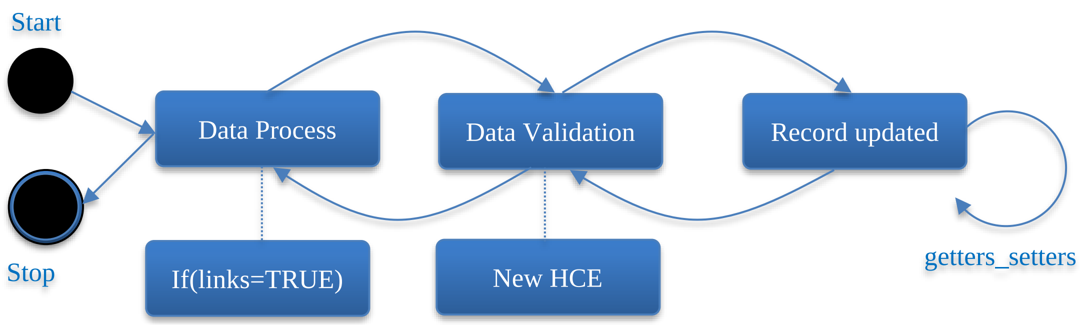


Fig. 3 Design of HCE state diagram

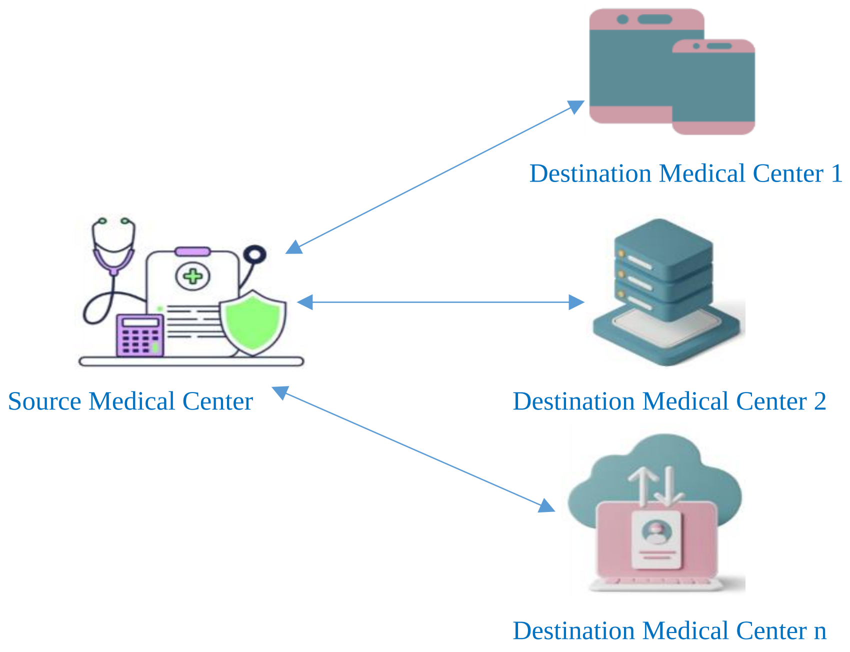

Destination Medical Center 1
Source Medical Center
Destination Medical Center 2
Destination Medical Center n

Fig. 4 Operation of Interoperability APIs

Table 2. XML answer with the objective description of the patient

Output XML Document
<?xml version="1.0" encoding="UTF-8"?> <ClinicalDocument xmlns="urn:hl7-org:v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <!-- Document Header --> <typeId root="2.16.840.1.113883.1.3" extension="POCD_HD000040" /> <templateId root="2.16.840.1.113883.10.20.22.1.1" /> <!-- Patient Information --> <recordTarget> <patientRole> <!-- Patient Details --> <patient> <name>Patient Name</name> <administrativeGenderCode code="F" codeSystem="2.16.840.1.113883.5.1" /> <birthTime value="Date of Birth" /> <!-- Signs and Symptoms Details --> <observation classCode="OBS" moodCode="EVN"> <!-- Hypoventilation --> <code code="267036007" codeSystem="2.16.840.1.113883.6.96" displayName="Hypoventilation" />

```
<value xsi:type="ST">Mild</value>
</observation>
<observation classCode="OBS" moodCode="EVN">
<!-- Heart Rate -->
<code code="8867-4" codeSystem="2.16.840.1.113883.6.1" displayName="Heart rate" />
<value xsi:type="PQ" value="95" unit="/min" />
</observation>
<observation classCode="OBS" moodCode="EVN">
<!-- Respiratory Rate -->
<code code="9279-1"codeSystem="2.16.840.1.113883.6.1"
displayName="Respiratory rate" />
<value xsi:type="PQ" value="28" unit="/min" />
</observation>
<!-- Lymph Nodes Details -->
<observation classCode="OBS" moodCode="EVN">
<code code="102942000" codeSystem="2.16.840.1.113883.6.96" displayName="Lymph node" />
<value xsi:type="ST">Inguinal with Abnormal Behavior</value>
</observation>
<observation classCode="OBS" moodCode="EVN">
<code code="102942000" codeSystem="2.16.840.1.113883.6.96" displayName="Lymph node" />
<value xsi:type="ST">Preescapular with Normal Signs</value>
</observation>
</patient>
</patientRole>
</recordTarget>
</ClinicalDocument>
```

4. Methodology for Evaluating the Architectural Design of API's Architecture System using Predictive Modeling and Graph Theory-Based Metrics

The effectiveness of the architectural design in the proposed system was evaluated using the proposal from [28]. Software architectures could be analyzed in the design phase based on a predictive model and graph theory using software metrics in an excellent paradigm for evaluating object-oriented systems. The evaluation of object-oriented systems, as proposed, would focus on the aspects of intricacy and connection. The following actions were supposed to be done for a system containing all classes: Patient Identification Data, Laboratory Results, Personal Medical History, CDA, Factory HCE, HCE, and Wrapper Medical History.

4.1. Methodology Applied

4.1.1. Creation of the ACG (Architectural Complexity Graph)

The ACG was developed by envisioning the architectural design for each of the six system types in Figures 2 and 3. With the support of UML diagrams, the construction of ACG was performed.

4.1.2. Assignment of the Complex Attribute on the ACG

A complex characteristic was assigned to the ACG using CCM measures as weights (see Figure 5). Applying the Floyd-Warshall algorithm, an estimate of critical pathways disclosed 18 critical paths.

4.1.3. Assignment of the CBO Metric on Localized Paths

The configuration of localized pathways was achieved by assigning the CBO metric. The computation of the correlation between the CBO and CCM data sets using Spearman for every localized pathway followed the proposal in determining the performance architectural parameters for critical paths. Table 3 indicates some path outcomes: The quality factor results are shown in the last column, with a path in the first column and the CCM and CBO metrics in the second column. Values from Spearman's correlation between -1 and +1 show how CCM and CBO connect to one another as ordinal variables. A score nearer +1 indicates a strong relationship and performance, whereas a value nearer -1 indicates poor performance and a weak relationship.

4.1.4. Inference of Performance Parameter

Some examples of the outcomes are shown in Table 3, along with sequencing of components with CMM and CBO metrics and quality criteria for six key approaches: 1, 2, 3, 4, 5, including 18. Figure 6 Scatter dispersion figure, using all the results as input data. This research involved the course Factory_HCE. As a conceptual course, it was given metrics of 0 for the CBO and 1 for the CCM. Eighteen key paths were discovered.

According to the scatter plot, only one road has a zero value, and the effectiveness evaluation parameter had values near +1 on the seventeenth of those roads. Only four dynamic and active nodes were reached finally by these pathways. This modified technique emphasizes a mapping view of the system's architectural design quality, comprising six components. The technique is based on metrics and critical routes for deductions about the architectural performance of the system, with special concern for coupling and complexity.

4.2. Discussion

An object-oriented strategy was used in this proposal to improve the software framework for the medical interoperability API, along with the incorporation of design patterns. The main challenge was handling the circumstances of inherited medical interoperability in this instance, which involves hospitals or medical centers using equipment with a constrained patient data memory. Using dynamic objects within memory created at runtime to optimize the design was emphasized. Unlike other approaches, wherein all system operations are put into memory, this one could potentially have a detrimental impact on the operational dynamics of the API because of memory exhaustion on the execution devices.

In order to measure and evaluate the suggested design approach, it was executed using an integrated software architectural evaluation method that considered the quality of design criteria like complexity and coupling. When coupling increases in complexity, and good performance is a desired feature for application performance, the chosen proposal for analysis focuses on the relation between coupling and complexity.

The lack of source code during this phase meant the design assessment was static. UML diagrams, algorithms, along other design elements were used in the evaluation process. Thanks to cost savings, earlier test prioritization according to components found to have a greater failure risk, and fault localization and remedy prior to code development, evaluating the design was favorable. This design and evaluation produced a prediction model and a formal method for evaluating the caliber of object-oriented systems.

The performance parameter was configured using the architectural complexity graph created for a deterministic analysis. Indeed, stronger correlations between coupling qualities and component sequence complexity are seen in sequences whose parameter values are near +1, indicating high-performance levels. In this case, 94.5% of the sequences under analysis were around +1, indicating a good degree of connection and complexity.

Figure 6, which displays the dispersed performance of the API architectural layout, reflects this. This was through the use of an abstract factory in addition to wrapper pattern designs in the proposed API architecture. We therefore deemed this approach, which had to do indirectly with software design architecture performance, as conceptually valid.

Table 3. Determine the crucial routes assessed using API architecture performance metrics

ID Path	Sequence	Performance Parameter
1	v0 → v1	**0.9428**
	CBO: 2 → 3	
	CCM: 1 → 1	
2	v0 → v1 → v2	**0.7745**
	CBO: 2 → 3 → 1	
	CCM: 1 → 1 → 1	
3	v0 → v1 → v2 → v3	**0**
	CBO: 2 → 3 → 1 → 2	
	CCM: 1 → 1 → 1 → 1	
4	v0 → v1 → v2 → v4	**0.7745**
	CBO: 2 → 3 → 1 → 1	
	CCM: 1 → 1 → 1 → 1	
5	v0 → v1 → v2 → v5	**0.7745**
	CBO: 2 → 3 → 1 → 1	
	CCM: 1 → 1 → 1 → 1	
...	...	...
18	V2 → V6	**1**
	CBO: 1 → 0	
	CCM: 1 → 0	

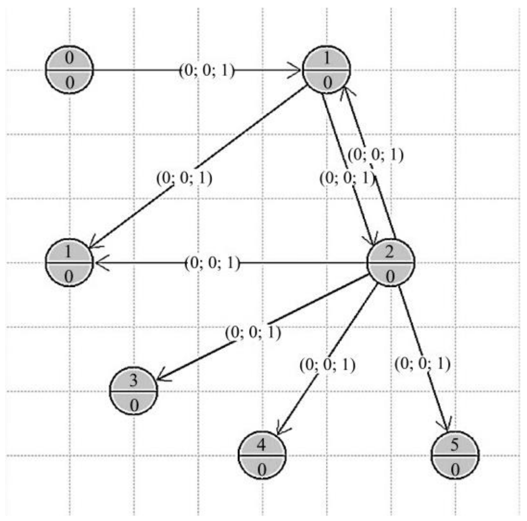


Fig. 5 ACG of Interoperability APIs

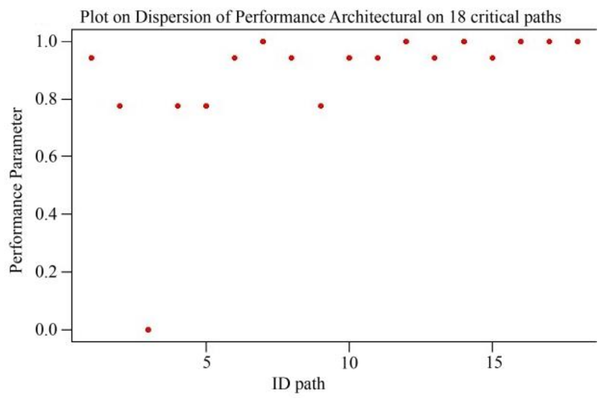


Fig. 6 API architecture's dispersion performance

4.3. Addressing Challenges and Considerations with the Proposed API

4.3.1. Compliance

The proposed API includes security elements due to their rigid nature that will ensure compliance with health regulations, especially HIPAA. This ensures health information stays safe from hacking in the course of transition and storage. The design of the API involves: It only grants access to the system to authorized users, through secure login protocols like OAuth2. Its access is further restricted to only healthcare professionals and institutions with rights of access to patient data.

- Data Encryption: All sensitive data across the API transmission with connected systems must be encrypted using industry-standard protocols, for example, TLS/SSL, to protect against unauthorized access to data over transmission.
- Audit Trails: The API has a clean audit log of data access and modifications. This way, healthcare organizations can keep tabs on and track unauthorized access attempts.
- Security Audits: The API is developed with regular security audits to detect vulnerabilities and constantly comply with the evolving regulatory compliance standards.

4.3.2. Data Quality and Standardization

A number of strategies are employed for data quality and standardization by the proposed API:

- Use of Standard Data Formats: It makes use of known standards such as HL7 and FHIR, which are otherwise referred to as Fast Healthcare Interoperability Resources, for data exchange. This is to mean that there is always consistency in formats across different systems to enhance interoperability.
- Data Cleansing Procedures. In preparation for the integration of data coming from different sources, the API devises data cleansing processes. This helps to eliminate duplicates, correct entry errors, and ensure that data comes in standardized formats, making the information stored in the EHR system accurate and reliable.
- Validation Rules: The API will enforce validation rules so that it gets accepted into the system only if the incoming data meets the predefined criteria. This ensures very high data quality by avoiding the entry of erroneous information.
- Feedback Mechanisms: The API carries features to enable healthcare professionals to report discrepancies or issues with data quality. This feedback can then be used to improve data cleansing processes and refine standardization practices over time.

Thereby, the proposed API addresses critical issues of regulatory compliance and data quality in the context of robust security and standardization. It integrates Python wrappers with EHR systems to achieve improved interoperability, better patient care, higher productivity and operational efficiency, innovation facilitation, and compliance with foundational standards to ensure improved healthcare results.

4.3.3. Research Limitations and Future Trends

The proposal addresses the pressing need for clinical data interoperability, which has grown even more important in the post-COVID-19 environment. Interoperability is essential for appropriate patient diagnosis and treatment. The main goal was to use object-oriented design patterns and seamless integration to improve the architecture of a new clinical interoperable API program. In order to guarantee compatibility with low-memory devices, the design was customized. The project also created a strong framework for quick assessment and enhancement of the software's architectural design, which eventually reduced expenses during the API development process. However, determining the suggested architecture's practical utility proved to be a substantial issue during the design phase.

A significant portion of the examination was based primarily on static predictive models and formal approaches. Moreover, resource constraints, platform compatibility issues, and the complexity of interfacing with existing healthcare systems could hinder the real-world implementation of the architecture. Java was among the computer languages' most promising candidates to support the specified design. It targeted interoperability, from its use of standards, abstractions and a common virtual machine and actively supported portability and integration on many systems and services. The analysis model, including the suggested GCA, provided an outline for guidance during the testing stage. This approach defined the operational steps of the API and provided preliminary information to make the test matrix, in conjunction with functional analysis that is weighted based on design metrics.

The proposed work followed the HL7 CDA standards [13] based on certain clinical papers. The latter possesses a data model that consists of segments and fields and needs these to be combined and exchanged in messages. Moreover, the strong structure of CDA leaves far less flexibility over different scenarios and different data requirements, especially when dealing with a huge quantity of data. Other standards, such as OpenEHR, are also highly adopted and offer added functionality with regard to health information management. This standard, in essence, focuses on generating and managing EHRs with the semantic representation of clinical data. A type-based data model in OpenEHR is defined as a semantic structure that describes the organization and meaning of clinical data. This makes the representation of health data much more sound and organized. This is the case where the functionality of the API depends on its ability to interrelate with other standards.

5. Conclusion

Interoperability in medicine is urgently needed at a global level. Apart from the interoperability aspect, the big problem is the diversified nature of the systems used by various hospitals and medical centers in maintaining patient data. Since the proposed API has to function seamlessly on all devices, starting from the cloud computing server to a small medical gadget called a mobile phone, memory management becomes a major challenge. This proposal chose integrated design patterns and object-oriented methodology to enhance the software architecture of the clinical interoperability API. The proposed API architecture included design patterns like abstract factory and wrapper, which allowed the system to replicate itself. It created instances on the fly and sent each to a hospital where patient data was required. Once contact has been established via medical centers, this API recovers the patient's records by encasing details about controlled personal data, recognizing a patient, recognizing illnesses or circumstances, documents of medical procedures performed, and indicating images like X-rays, MRI tests, etc. The information collected from the events at each facility is summarized for the request and submitted to the requesting medical center. 94.5 percent of the study's participants performed well in the architectural design review. The evaluation of the operation sequence, which demonstrated a good level based on the complexity and connectedness, served as an indirect proof of this. It can be inferred from the outcomes that the method maximizes temporal and memory resources, enhancing item construction efficiency as it's being executed in diverse circumstances. This method improves reaction times and memory use in electronic records interchange, influencing how well hospital care responses are executed.

This concluded that HL7 is a good standard for handling interoperability as a result. It has to be enhanced and augmented to facilitate the management of semantic modeling in Electronic Health Records (EHRs). It is recommended that this suggested design be further validated in real settings in subsequent work to ensure practicality. Other protocols, like OpenEHR, frequently used in developing medical interoperability application programming interfaces, should be incorporated into the suggested Java architecture format. The section provided the design with the means to adjust to the various current medical interoperability standards. Using the builder design pattern is one possible need to be met in this respect.

References

[1] Clara I. Valero et al., “AIoTES: Setting the Principles for Semantic Interoperable and Modern IoT-Enabled Reference Architecture for Active and Healthy Ageing Ecosystems,” *Computer Communications*, vol. 177, pp. 96-111, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Wajahat Ali Khan et al., “Achieving Interoperability among Healthcare Standards: Building Semantic Mappings at Models Level,” *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, New York, United States, pp. 1-9, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[3] Tim Benson, *Principles of Health Interoperability HL7 and SNOMED*, 1st ed., Springer London, pp. 1-263, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[4] The Future of Health & Care is Open, OpenEHR. [Online]. Available: https://www.openehr.org/

[5] Der-Fa F Lu et al., “Standardized Nursing Language in the Systematized Nomenclature of Medicine Clinical Terms A Cross-Mapping Validation Method,” *Computers, Informatics, Nursing*, vol. 25, no. 5, pp. 288-296, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[6] J.D. Trigo Vilaseca et al., “Standards-Based Real-Time ECG Transmission: Harmonization of ISO/IEEE 11073-PHD and SCP-ECG,” *Conference: XXVII Annual Congress of the Spanish Society of Biomedical EngineeringAt* Cadiz, Spain, pp. 756-759, 2009. [Google Scholar] [Publisher Link]

[7] Mustafa Yuksel, and Asuman Dogac, “Interoperability of Medical Device Information and the Clinical Applications: An HL7 RMIM Based on the ISO/IEEE 11073 DIM,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 4, pp. 557-566, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[8] M. Martínez-Espronceda et al., “*INTENSA: Monitoring System for Patients with Insufficiency Cardiac Based on ISO/IEEE11073 Standard*,” Universidad de Zaragoza, pp. 1-5. [Google Scholar] [Publisher Link]

[9] Miroslav Koncar, *Implementing the HL7 v3 Standard in the Croatian Primary Healthcare Domain*, Studies in Health Technology and Informatics, vol. 105, pp. 325-336, 2004. [Google Scholar] [Publisher Link]

[10] HL7 in Europe, hl7. [Online]. Available: https://www.hl7.eu/

[11] Mhd Adnan Jouned, “*Development of an Interoperable Exchange, Aggregation and Analysis Platform for Health and Environmental Data*,” Master’s Thesis, University of Applied Sciences Technikum Wien, Wien, Austria, pp. 1-81, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Ricardo Adolfo Aguilar Bolaños, and Diego M. López, “HL7 Implementation Guide for Public Health Reporting Systems in Colombia,” *Sistemas & Telematica*, vol. 7, no. 14, pp. 13-32, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ivett E. Fuentes Herrera, Damny Magdaleno Guevara, and María Matilde García Lorenzo, "Intelligent Decision Making from Medical Records Stored in CDA-HL7," *Cuban Journal of Medical Informatics*, vol. 8, no. 1, pp. 109-124, 2016. [Google Scholar] [Publisher Link]

[14] Victor Velasquez et al., "Electronic Health Record Interoperability Model Using HL7-CDA Based on Cloud Computing," *Research in Computing Science*, vol. 108, pp. 37-44, 2015. [Google Scholar] [Publisher Link]

[15] D.I. Shin, P.J. Pak, and S.J. Huh, "The Mobile Implementation of HL7 API for u-Healthcare Devices," *World Congress on Medical Physics and Biomedical Engineering*, Munich, Germany, vol. 25/5, pp. 110-111, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[16] Teeradache Viangteeravat et al., "Clinical Data Integration of Distributed Data Sources Using Health Level Seven (HL7) v3-RIM Mapping," *Journal of Clinical Bioinformatics*, vol. 1, pp. 1-10, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[17] Suresh Koduru, PVGD Prasad Reddy, and Padala Preethi, "A Novel Key Exchange Algorithm for Security in Internet of Things," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 3, pp. 1515-1520, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[18] Barry Smith, and Werner Ceusters, "HL7 RIM: An Incoherent Standard," *Studies in Health Technology and Informatics*, pp. 133-138, 2006. [Google Scholar] [Publisher Link]

[19] Thomas Beale, "Archetypes: Constraint-Based Domain Models for Future-Proof Information Systems," *OOPSLA 2002 Workshop on Behavioural Semantics*, pp. 1-18, 2002. [Google Scholar] [Publisher Link]

[20] Nicholas Nicholson, and Andrea Perego, "Interoperability of Population-Based Patient Registries," *Journal of Biomedical Informatics*, vol. 112, pp. 1-14, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi, "Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges," *Sensors*, vol. 13, no. 12, pp. 17472-17500, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[22] A.C. Cheng et al., "REDCap on FHIR: Clinical Data Interoperability Services," *Journal of Biomedical Informatics*, vol. 121, pp. 1-13, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] P. Rekha et al., "Smart AGRO Using ARDUINO and GSM," *International Journal of Emerging Technologies in Engineering Research*, vol. 5, no. 3, pp. 38-40, 2017. [Google Scholar] [Publisher Link]

[24] Matthias Biehl, *RESTful API Design: Best Practices in API Design with REST*, API-University Press: Rotkreuz, Switzerland, 2016. [Google Scholar]

[25] Christoph Praschl et al., "Imaging Framework: An Interoperable and Extendable Connector for Image-Related Java Frameworks," *SoftwareX*, vol. 6, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[26] N. Sandhya et al., "Smart Attendance System Using Speech Recognition," *2022 4th International Conference on Smart Systems and Inventive Technology*, Tirunelveli, India, pp. 144-149, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] Implementation Guide for CDA Release 2: Imaging Integration Levels 1, 2, and 3 v1.0 Basic Imaging Reports in CDA and DICOM Diagnostic Imaging Reports (DIR) - Universal Realm Based on HL7 CDA Release 2.0, v1.0, Australian Digital Health Agency, 2015. [Online]. Available: https://developer.digitalhealth.gov.au/standards/implementation-guide-for-cda-release-2-imaging-integration-levels-1-2-and-3-v1-0

[28] Leticia Davila-Nicanor et al., *Performance on Software Architecture Design to Serious Games for Mobile Devices*, Software Engineering for Games in Serious Contexts, pp. 63-84, 2023. [CrossRef] [Google Scholar] [Publisher Link]