DEVELOPMENT OF AN IMPROVED TECHNIQUE FOR PATIENT DATA HANDLING

(A research project report submitted in fulfillment of the requirements for the Research and Extension Center for the fiscal year 2024-25 of Jatiya Kabi Kazi Nazrul Islam University Trishal, Mymensingh-2224, Bangladesh)

Submitted By:

Prof. Dr. Uzzal Kumar Prodhan
Dept. of Computer Science & Engineering
Jatiya Kabi Kazi Nazrul Islam University
Trishal, Mymensingh, Bangadesh

Submitted To:

Dr. Md. Habibur Rahman
Director
Research and Extension Center
Jatiya Kabi Kazi Nazrul Islam University

JATIYA KABI KAZI NAZRUL ISLAM UNIVERISITY

JULY-2025

DECLARATION

I hereby declare that the research project titled **"Development of an improved technique for Patient Data Handling"** is done by me. I have completed the project in time and within the budget allocated to me from the research and extension center of Jatiya Kabi Kazi Nazrul Islam University. This project is unique and haven't done before in the Jatiya Kabi Kazi Nazrul Islam University.

(Prof. Dr. Uzzal Kumar Prodhan)
Department of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University
Trishal, Mymensingh, Bangladesh
Email: uzzal_bagerhat@yahoo.com
Mobile: 01718045304

ACKNOWLEDGEMENT

This research work is supported by the University Grants Commission Bangladesh. I was funded by the Research and Extension Center of Jatiya Kabi Kazi Nazrul Islam University (JKKNIU), Trishal, Mymensingh, Bangladesh. I am grateful to the authorities of Research and Extension Center for their support.

I am also very grateful to Professor Dr. Md. Sujan Ali, honorable head of the department of Computer Science and Engineering of JKKNIU. I would like to acknowledge the entire faculty members of this department for their supports.

My project team members from the Department of CSE, Jatiya Kabi Kazi Nazrul Islam University, Trishal, Mymensingh, Bangladesh helped me a lot to do this project. I shall give my heartfelt thanks to them for his restless support, effort and suggestions. Prof. Dr. Subrata Kumar Das has given me the direct instructions, suggestions and observations for the timely development of the project. Sraboni Sarker Mukti and Jannatul Naim who were two of my master's student has helped to write and develop the research work.

Department of Computer Science and Engineering of Bangladesh University and German University of Bangladesh are also connected with me with their teams for data collection, processing and developing this project. Without their direct co-operation it would be very difficult to complete my project in time.

ICT cell of Jatiya Kabi Kazi Nazrul Islam university help me for the technical support and finally with their assistance, I can conclude my project in practical stage.

Lastly, I want to give my deepest thanks to my family members for whom I am finally able to complete this research work.

(Prof. Dr. Uzzal Kumar Prodhan)
Department of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University
Trishal, Mymensingh, Bangladesh
Email: uzzal_bagerhat@yahoo.com
Mobile: 01718045304

DEDICATION

Dedicated to my lovely parents and family members for whom I am now in a position.

Table of Contents

Abstract ..... **Error! Bookmark not defined.**
CHAPTER 1 10
INTRODUCTION 10
1.1. Overview 100
1.2. Objectives 100
1.3. Problem Statement 111
1.4. Motivation 111
1.5. Organization of This Project 122
CHAPTER 2 133
BACKGROUND STUDY 133
2.1 Literature Review 133
2.2 Related Works 143
2.3 Comparative Study 143
2.4 Scope of the Problem 154
2.5 Challenges 155
2.6 Primary Survey **16**
CHAPTER 3 177
SOFTWARE REQUIREMENT & SPECIFICATION 177
3.1 Methodology 177
3.2 Requirement Collection and Analysis 18
3.3 Functional Requirements 18
3.4 Non-functional Requirement 18
3.5 Deployment Diagram 19
CHAPTER 4 210
DESIGN & IMPLEMENTATION 210

4.1 What is Design & Implementation ? 210
4.2 Implementation Tools & Technology 210
User Interface & Technology 210
ImplementationTools&Platform 210
4.3 Design API 221
4.4 System Architecture 232
4.5 Activity Diagram 254
4.6 Interface to share data from databases 267
4.7 Mathematical model 29
4.8 Json Sample Data 30
4.9 Summary 333
CHAPTER 5 344
RESULT & DISCUSSION 344
5.1 Result 344
5.2 Limitation 411
5.3 Future Work 421
5.4 Conclusion 433
5.5 References 444

List of Figures

Figure 1: Proposed Diagram 19

Figure 2: Architectural Diagram 233

Figure 3: Activity Diagram 255

Figure 4: System Intefacing 27

Figure 5 : Mathematical Model 29

Figure 6 : Sql Database 344

Figure 7 : SQLite database 355

Figure 8: MongoDB Database 36

Figure 9 ; MariaDB database 37

Figure 10: Find patient data on 4 medicals 38

Figure 11: Find patient data on 3 medicals 39

Figure 12: Find patient data on 2 medicals 400

Abstract

General practitioners and other professionals record healthcare data from various sources and locations. Integrating and exchanging electronic health data from many databases is crucial for providing patients with timely, high-quality care. However, because of the diversity of structural, semantic, and querying syntaxes, exchanging patient data remains a significant challenge. A unified perspective for data exchange is provided by the research project "Development of an improved technique for Patient Data Handling." This research conducted a primary survey on 83 health organizations and found that 42.6% organizations use My SQL, 23.1% use other database, 11.5% use Access database. 42.7% organization told that their health records cannot be shared with other health organizations. Survey found that 53.8% organization collect health record through customized software This research developed an improved model to handle patient data from diverse database. The architectures of the system functionality are tested using our customized database. The system that has been put in place extracts and combines data from many databases effectively and efficiently. The suggested architecture resolves issues without changing the current systems, facilitating the consolidation and exchange of health data. Users or health professionals could exchange data from dispersed source databases to improve the efficacy of their actions and judgments for patients. The cost of the patients' care would also be decreased by the availability of their prior medical records.

CHAPTER 1

INTRODUCTION

1.1. Overview

In order to share prior patient information with doctors or other users, healthcare organizations digitally preserve patient data. Patients' disease symptoms, various diagnostic tests, past and present medical conditions, employment history, presence of harmful habits, and other information are typically included in this electronic health data. They also save the records pertaining to past prescription drugs and medications. In order to provide better healthcare services and high-quality care, remote practitioners should share electronic health data. According to Anand et al., prior patient data shows that healthcare providers make the best decisions for their patients and provide a foundation for efficient treatment and the execution of treatment plans. According to Kumari et al. (Kumari, 144–149) and Roehrs et al. (Roehrs, 2017), electronic medical records are necessary for thoughtful urgent patient care and quality treatment.

1.2. Objectives

The aim of this research " Development of an improved technique for Patient Data Handling " is to design and develop an effective architecture that efficiently integrates and exchanges electronic health data from Distributed Databases (DDBs). The primary goals include:

- Establish a Single view of the patient data across multiple sources and locations to support interoperability across multiple healthcare systems

- Develop and test a system which is capable of extracting data from different databases with good precision and merging the information without disturbing the system in which it exists.

- Design the solution in such a manner that it can integrate with the existing healthcare data systems without making heavy alterations in the original system so that it will be easy to adapt and use.

- Minimize the cost of treatment towards patients by making the documents easily available to the medical staff. This in turn avoids the repetition of tests and medical procedures on patients.

- Improve the action and decision making of medical personnel by furnishing them with solid and ample information of their patients which in turn guarantees the quality of healthcare given to patients.

If these goals are achieved, the project will have delivered a sound and practical system for handling patient data, which will increase data availability, enhance healthcare treatment and decrease costs without compromising integrity and security of the system.

1.3. Problem Statement

In healthcare domain, patient data are gathered by general practitioners and other clinicians from diverse sources and locations. These data are typically stored in diverse electronic health repositories, providing key information to support decision making with the aim of delivering the right patient treatment at the right time.

Data integration techniques currently used are often inefficient and demand deep changes to the existing systems, leading to high costs and disruptive modifications. An urgent need exists for an effective solution to integrate and disseminate patient data stored in heterogeneous distributed databases (HDDBs) with minimal changes to the current systems.

The aim of this project is to overcome these limitations by investigating and propose an architecture capable of materializing the notion of data reconstructed from different data sources into a global as-view. The global as-view notion should allow for the information interchange about a given patient among different healthcare information systems without needing to worry about the differences among the data repositories and the healthcare providers. The main expectation is that the easy access to documents previously created during the patient treatment will significantly decrease costs and time spent for handling the patient data, leading to an efficient and effective healthcare treatment.

1.4. Motivation

Numerous studies focus on finding semantic solutions to combine data from various sources. Reconstruction data from distant databases, however, is scarce due to structure and querying syntactic variability mitigation.

To combine patient data from sources that preserve information on particular illness symptoms (such as skin cancer, Alzheimer's disease, sports injuries, or

tuberculosis), a few ways are presented. The method to aggregate data from NoSQL or RDBMS is also introduced in a few published works.

This study offers an architecture for health professionals or other users to share data from current databases. The suggested method addresses the problems of structural variability, querying syntactic variability, and semantic heterogeneity while integrating data from various data sources. Our approach for this project would enable native searches for many source databases to overcome the querying syntax variability.

1.5. Organization of This Research Work

Chapter1: Discuss about the report's summary, goals, problem statements, inspiration, and structure.

Chapter2: Discuss about the project's background investigation. This covers relevant literature, comparative analyses, problem scope, and project scope issues.

Chapter 3: The software requirement specification is discussed. Included in this are functional requirements, nonfunctional requirements, requirements for requirement analysis and design, and so on.

Chapter4: Discuss about the system's design and implementation. An API that has been implemented to evaluate data extraction from distributed, heterogeneous databases. presents the suggested architectural design. The suggested building shown mathematically.

Chapter5: Discuss about the project's outcome, limitation, future work, discussion & references. This covers the conversation, resolution, and subsequent execution phase.

CHAPTER 2

BACKGROUND STUDY

2.1 Literature Review

In order to combine and obtain health data from several heterogeneous databases and distribute it internationally to end users, researchers are working on this project. A distributed system for integrating heterogeneous, multidimensional patient data on skin cancer was proposed by Maglogiannis et al (Balelli, 2021).They provided an extension of the existing healthcare systems to data in the form of a layered analytical platform. Almeida et al. (Almeida, 2020) released a paper that used multiple operations simulation over healthcare data spread across many virtual locations to gather and validate data from solely NoSQL. A middleware was proposed by Bezerra et al. (Bezerra, 2020) for health data integration and interoperability across different applications.

Their first effort was to facilitate data transfer between the sports injury clinic and authors alone, but they later built an open data combining platform from databases. Li (Cosío-León, 2018) presented an architecture for data integration and interoperability among healthcare systems to facilitate collaboration, albeit its performance was subpar. In order to address semantic heterogeneity, Cosió-León et al. developed a message template for data extraction from embedded devices. Kiourtis et al. introduced an additional ontology-based method that incorporated syntactic and semantic data similarity and reserved into a triplestore to exchange data (Kiourtis, 2019).

For the eHealth area, an ontology-based system that solely offers semantic interoperability among heterogeneous IoT devices was mentioned (Carbonaro, 2018). It also makes patient data integration and exchange easier. The suggested course of action investigated a standard communication technique for information representation and exchange. An overview of integrating biomedical data based on big data technologies and semantic ontology systems was presented by Messaoudi et al. (Messaoudi, 2021). Additionally, they presented an integrated proteomics data system (IPDS) that processes queries and extracts data via a mediator technique (zarm, 2023).

The middleware requirements and constraints were highlighted by Singh and Bacon (Singh, 2022) in light of an enforced personalized, preventative, and omnipresent healthcare vision on supporting infrastructure. Their research proved the systems strategy's viability in generating new functional opportunities (Yuksel, 2023) through high-level and user-specified preferences actively.

2.2 Related Works

As per the survey paper in this field, data reconstruction from heterogeneous distributed databases face three main challenges: structural heterogeneity, syntactic variability of queries and semantic heterogeneity. Most of the recently published papers in this field are trying to find techniques to overcome semantic heterogeneity but there is limited work on structural heterogeneity and syntactic variability of queries.

Some of the authors aggregated patient information on a particular disease (e.g., TB, sports injury, skin cancer, Alzheimer's disease) from numerous data sources. The authors also aggregated some data from RDBMSs or only NoSQL databases.

So, based on the above discussion on current techniques, it can be observed that the previous researchers worked on a specific disease or data model, or tried to solve only one problem among the multiple problems.

However, appropriate techniques to solve problems like syntactic variability of queries and structural heterogeneity are not present. and merging information from the databases. To measure the response time of the different group of databases, the nodes holding the different databases were deployed in homogenous and heterogeneous topologies.

2.3 Comparative Study

In our research "**Development of an improved technique for Patient Data Handling**", we can take various factors as comparison points to evaluate and examine contrasting methodologies or systems for handling patient data.

Even system architectures like centralized v/s decentralized, cloud v/s on-premises, proprietary v/s open-source can be taken as a comparison point to identify factors like scalability, interoperability and cost. Some security controls, front-end and back-end user interface developments may also be taken as comparison point to determine ease of use and implementation. Standards and compliance for privacy and security like HIPAA and GDPR should also be compared to determine ease of use and implementation.

2.4 Scope of the Problem

The scope of the problem in " **Development of an improved technique for Patient Data Handling** " includes many vital factors that must be taken into consideration in order to manage patient data effectively & efficiently. The major contents included in the scope are:

- Data Collection and Entry
- Data Storage and Security
- Data Integration and Interoperability
- Data Retrieval and Accessibility
- Data Accuracy and Quality
- User Experience and Usability
- Technological Advancements
- Case Studies and Real-World Applications

The above content provides vast area to be worked upon and to achieve this goal/project the following projects are planned which will help in understanding the current scenario and to propose certain innovative techniques to tackle the problems and to increase the efficiency and effectiveness of the healthcare system as a whole.

2.5 Challenges

A major consideration on our project "Development of an improved technique for Patient Data Handling" is the integration of data from different sources. Data in healthcare systems are stored as records of different types. These types have structures that vary across healthcare systems making it quite difficult to standardize them for integration. What's more, is that these types (or structures) use terminologies and coding schemes that have semantic variations. These variations necessitate the need for efficient mapping and translation to guarantee consistency and accuracy of the data.

User adoption and training could be an issue as healthcare providers may resist changing to new systems and will most likely require significant training and support. There may also be technological limitations such as the need to integrate with legacy systems and provide scalability. Financial and other resource costs for implementation and ongoing maintenance must also be considered.

There are healthcare regulations that need to be considered in order to ensure compliance in all aspects of the system. Patient consent should be verified while preserving their trust in the healthcare providers and designers. The system should

be easy to use from a user perspective with friendly interfaces and minimal disruption to their workflow. Clear policies should be defined and continuous monitoring should be in place to assure data integrity. Technological changes occur rapidly in this field and a good balance between innovation and practical implementation should always be maintained. These are some of the concerns that need to be addressed in order to develop a system that effectively handles patient data to enhance healthcare delivery and ensure data security.

2.5 Primary Survey

In this section we have prepared a questionnaire on the health organizations to know about the present conditions of health organizations. In the questionnaire, there were 24 questions on different matters. We have collected 83 responses for this study. After processing those data, we have found that 42.6% organizations use My SQL, 23.1% use other database, 11.5% use Access database. 42.7% organization told that their health records cannot be shared with other health organizations. Survey found that 53.8% organization collect health record through customized software. Data were collected from both rural and urban areas. Survey guides that there are huge opportunities to connect the diverse database through an advanced framework. If we can connect the different databases through our developed system, then it would be possible to share the patient's health records with the health professionals. As a result, we can easily able to access the records and health cost will be reduced. Treatment can be given to patients easily. Patients don't need to carry the health records. Advanced health services can be offered to them through the developed model.

CHAPTER 3

SOFTWARE REQUIREMENT & SPECIFICATION

3.1 Methodology

The project's goal is to solve structural, querying syntax, and semantic problems in order to integrate and exchange patient data from remote databases. Our suggested design enables access to different data sources from a heterogeneous setting. The primary responsibility is gathering information from many sources, merging it, and distributing it. Thus, when creating the architecture components, we take into account multiple factors (structural variability, query syntax, and semantic heterogeneity). A module containing the architecture was incorporated to address the heterogeneity in query syntax. Additionally, the architecture included a few parts for delivering the native query to the local database management system (LDBMS) on an individual basis. Different LDBMSs obtained the known query via those components, the structural variability could not be a matter to access data.

Additionally, in order to recover the problems with semantic heterogeneity, we integrated a knowledge-based repository. In order to receive queries from end users and gather information from source databases in response, the architecture was created to act as a mediator between the users. To verify the functionality of the system, we created an interface based on the specified architecture. On the chosen PCs, various NoSQL and SQL databases were installed. WiFi was used to link the Ubuntu computers so they could access and integrate database data. To assess the response time of distinct groupings of databases, the nodes holding the various databases were deployed in both homogenous and heterogeneous patterns.

- **Requirements Gathering:** specifications for the patient data handling system, taking into account aspects like usability, scalability, interoperability, and data security.

- **System Design**

- **:** Create data models, schemas, and storage systems to efficiently organize and handle patient data.

- **System development:** Creating a productive system that can manage patient data with ease for future applications.

- **Monitoring and Maintenance:** To maintain the system's continued dependability and security, perform routine maintenance tasks like database backups, software updates, and security patches.

- **Continuous Improvement:** Maintain the patient data handling system's relevance and efficacy throughout time by keeping an eye on industry trends, regulatory changes, and technology improvements.

3.2 Requirement Collection and Analysis

Requirements analysis (alternatively called requirements engineering) is the activity of discovering user expectations of a new or changed product. These properties are alternatively called criteria and have to be accurate, relevant and quantifiable. These requirements specifications are often called functional specifications in the context of software engineering. Requirements analysis is a key aspect of any project management analysis.

3.3 Functional Requirements

Gathering and analyzing requirements is one of the most critical pre-requisites for building an application. There are two types of deployment requirements: functional requirements and non-functional requirements. Any specification stating calculations to be performed, technical specifications, data manipulation and processing, and other technical capabilities defining what a system is supposed to perform can be termed as functional requirements. Use cases include behavioral requirements that enumerate all the scenarios where the system implements the functional requirements.

3.4 Non-functional Requirement

Non-functional requirements ensure added productivity, optimized performance, memory usage, hassle-free operation and the fastest possible load time of the application. In order to provide an excellent user experience, the user interface of an application should be beautiful and intuitive. The system architecture includes details of how the non-functional requirements will be implemented.

3.5 Deployment Diagram

A block diagram for the target architecture of the research " Development of an improved technique for Patient Data Handling " is shown below. This diagram illustrates the major building blocks and their interactions for the integration and exchange of patient data from source databases.

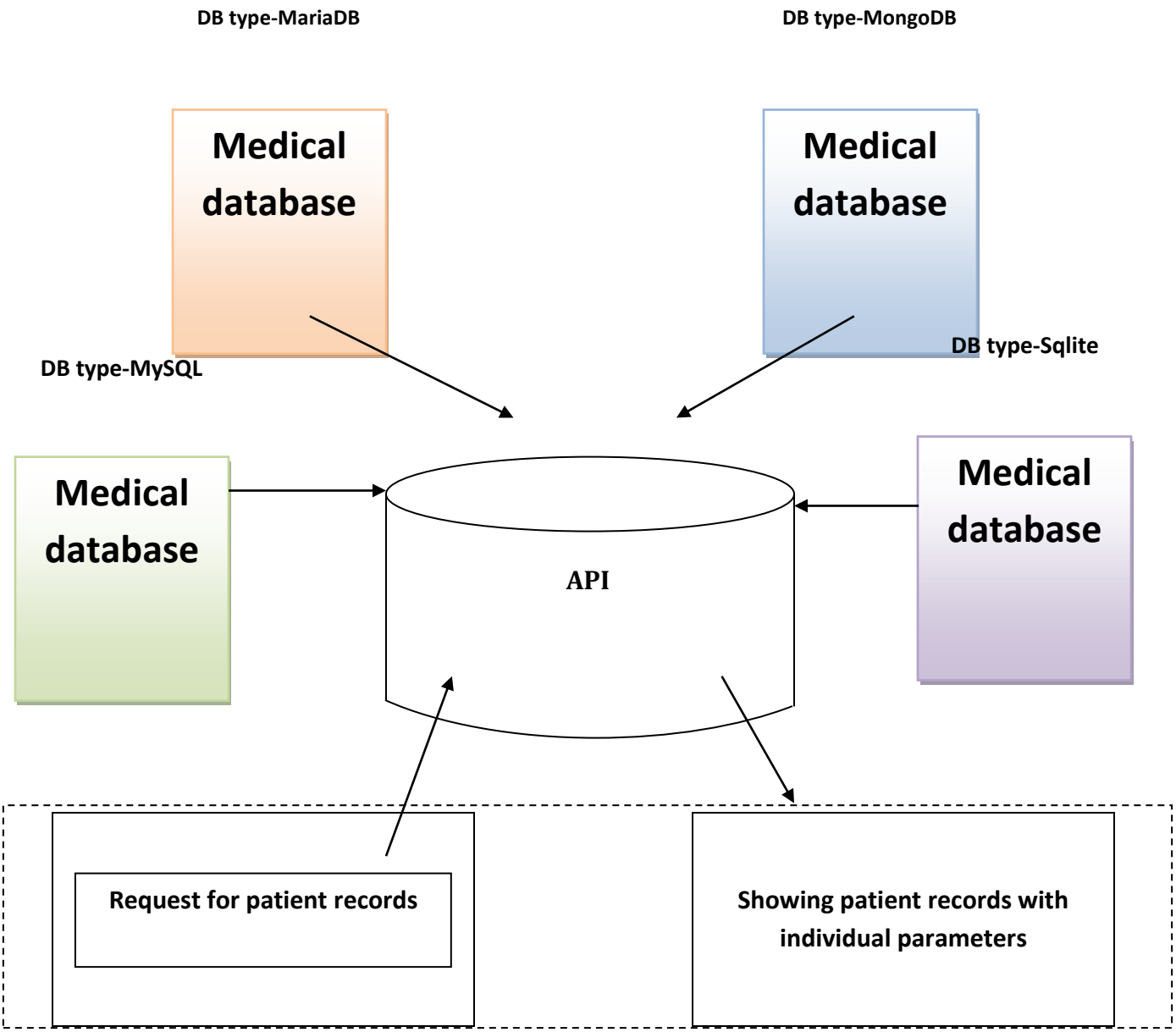


Figure 1: Developed Model

A deployment diagram for the target architecture of the project " Development of an improved technique for Patient Data Handling " shows the physical components and their distribution of the run-time system. This includes the sources databases, integration server, application server, and user devices. The source databases containing the electronic health records (EHR), lab results, etc. are located on separate database servers. Each database server is responsible for one type of the patient data sources. The integration server contains the data extraction, transformation, and consolidation modules and plays a major role in the system.

The application server components are the Query and Retrieval Engine, the User Interface (UI), and the Data Presentation Layer. The Query and Retrieval Engine provides an efficient mechanism to access the integrated data. The UI offers a convenient interface for health care providers to access the patient data. The Data Presentation Layer represents the retrieved data in an unambiguous format for better decision making by health care providers. The user devices include PCs, tablets, and mobile devices, which are used by doctors and nurses to access the patient data via the UI.

This type of architecture permits the integration and exchange of patient data from different databases. This furnishes health care providers with extensive, timely, and accurate patient information, which enhances the quality of patient care. Also, it reduces cost by sparing the need of repetitive tests and procedures, streamlines the workflow, and increases the overall system throughput.

The system was designed to be interoperable, to maintain data security, and to be user accessible. Thus, the research work " Development of an improved technique for Patient Data Handling " can be employed in modern health care management.

CHAPTER 4

SYSTEM DESIGN & IMPLEMENTATION

4.1 What is Design & Implementation?

Design in the context of software development refers to the process of planning and defining the structure, components, interfaces, and overall architecture of a system before it is built. The design phase involves creating detailed specifications and blueprints that guide the actual coding and implementation of the software.

Implementation in software development refers to the process of translating the design specifications into actual code and building the software application. It involves the actual creation of the software components, integration of those components, and validation that they work together as intended.

4.2 Implementation Tools & Technology

User Interface & Technology

- Laravel Framework
- Blade Template
- Model, View and Controller Design Structure

Implementation Tools & Platform

- Operating System: Windows 10
- Platform: VS Code, Open Standard
- Language: PHP
- Database: MySQL, SQLite, MongoDB, MariaDB

4.3 Design API

Health data is currently stored by healthcare companies using a variety of Relational and NoSQL databases from different developers. Those databases have different query languages and different data models. In accordance with our suggested model, we create an API to get data from databases.

In order to reduce the database technology's dependency on the application source code, the system is designed to isolate it from the data persistence layer. The interface may separate the technical specifics of each module and retrieve data from the many database systems described below. Every new database technology is also made possible by the interface without requiring a lot of programming work.

To acquire the desired answer, the user creates a query and sends it to the system. The user's message is accessed by the system, which then establishes a connection with each database using its own native query language. The system integrates the data and gives it to the user in response to their request after receiving it. This test evaluates the API's ability to extract data from a variety of database management systems, including NoSQL databases (MongoDB) and relational databases (MySQL, SQLite, MariaDB). MongoDB is an open-source NoSQL database management system that supports JSON, while MySQL, SQLite, and MariaDB are open-source relational database management systems (RDBMS).

4.4 System Architecture

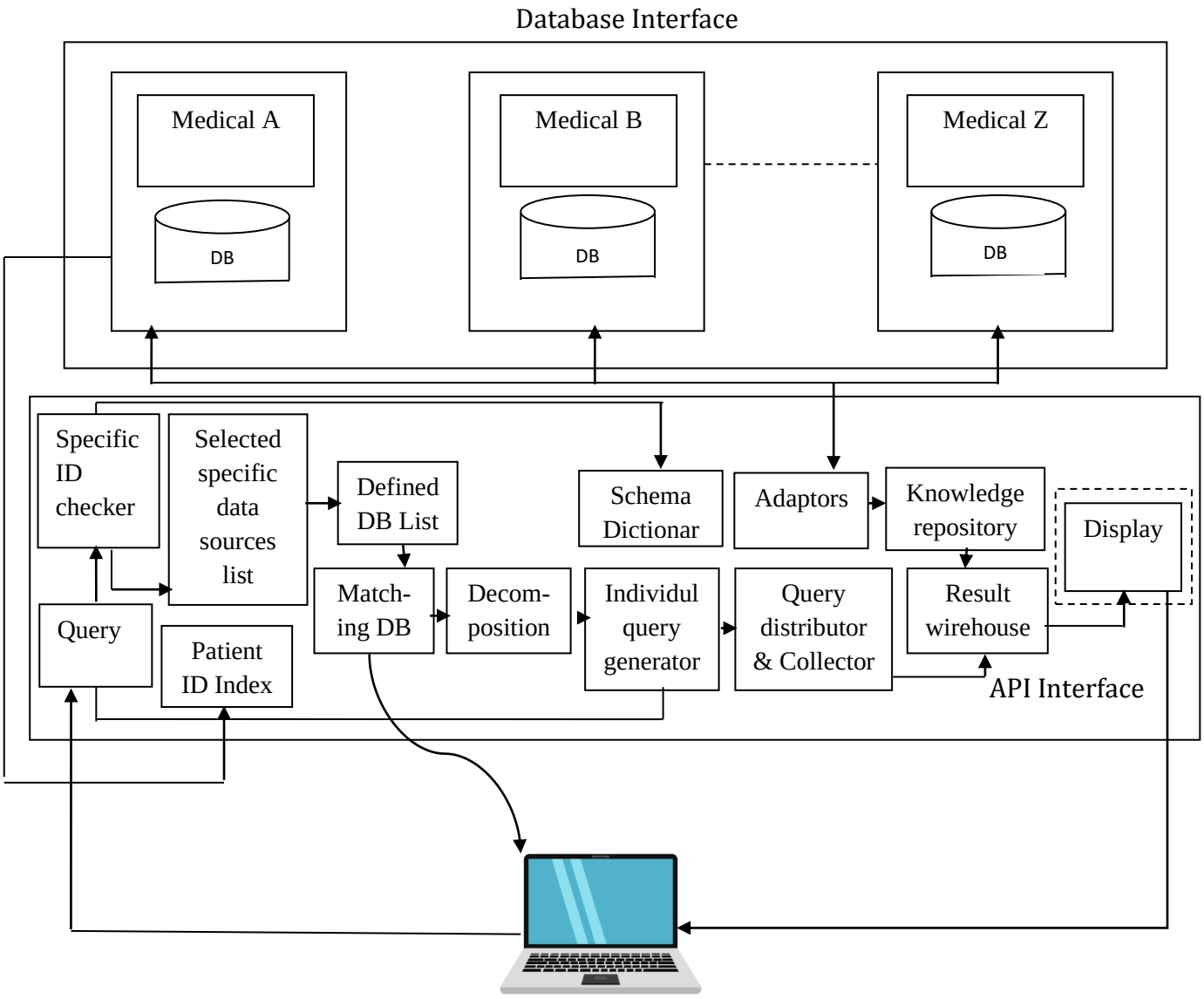


Figure 2: Architectural of the Developed Model

This architecture ensures that patient data from various sources can be effectively integrated and shared, providing healthcare professionals with comprehensive, accurate, and timely information to enhance patient care.

Specific ID Checker: This module is responsible for finding the patient ID from the generated XML file. Finding the patient ID creates a list of health databases by choosing from the 'Patient ID Index.' The 'Patient ID Index' consists of the list of the source node containing DBMSs, their types, and their addresses.

Defined DB List (DDL): This component keeps the definition of the databases having diverse data models. The 'Patient ID Index' consists of the location addresses of the source databases and their categories against patient ID according to the architecture. Most medical organizations generally use the NoSQL databases to store data instead of using any standards (HL7, DICOM, etc.).

Matching DB List (MDL): This module finds the types of the databases by taking information from the 'Patient ID Index' whether 'Defined DB List' already included their definition or not. If it gets the details of all targeted databases from the DDL module, it then performs the later steps for all databases.

Decomposition: This component is optional in our proposed architecture. The decomposition module works when the system requires to yield a sub-query from the user query.

Individual Query Generator (IQG): This part of the program is responsible for making an individual query for the various source data models got from MDL taking support from QXML and DDL. NoSQL systems do not have a standard SQL interface; instead, it processes data with the assistance of internal API.

Query Distributor & Collector (QDC): This section transfers the native query to the respective local database management system to retrieve patient data. It distributes the query to the targeted databases via the adaptors to get a response from them. The adaptors of the system can use the 'Knowledge Repository' and 'Schema Dictionary' that contains synonyms for finding the corresponding words and a pattern or sub-pattern of attributes.

Schema Dictionary (SD): This module is significant for schema-based databases, because it could keep a short schema from database objects. The end-users do not care about the data structure and what databases are stored data. The users are also transparent about the database's name, their entities, and fields or similar objects.

Result Warehouse: This component collects the results from the QDC received data from multiple distributed databases and integrates them to display data later.

Data Display: This module is responsible for displaying or transforming the result to the user-end.

4.5 Activity Diagram

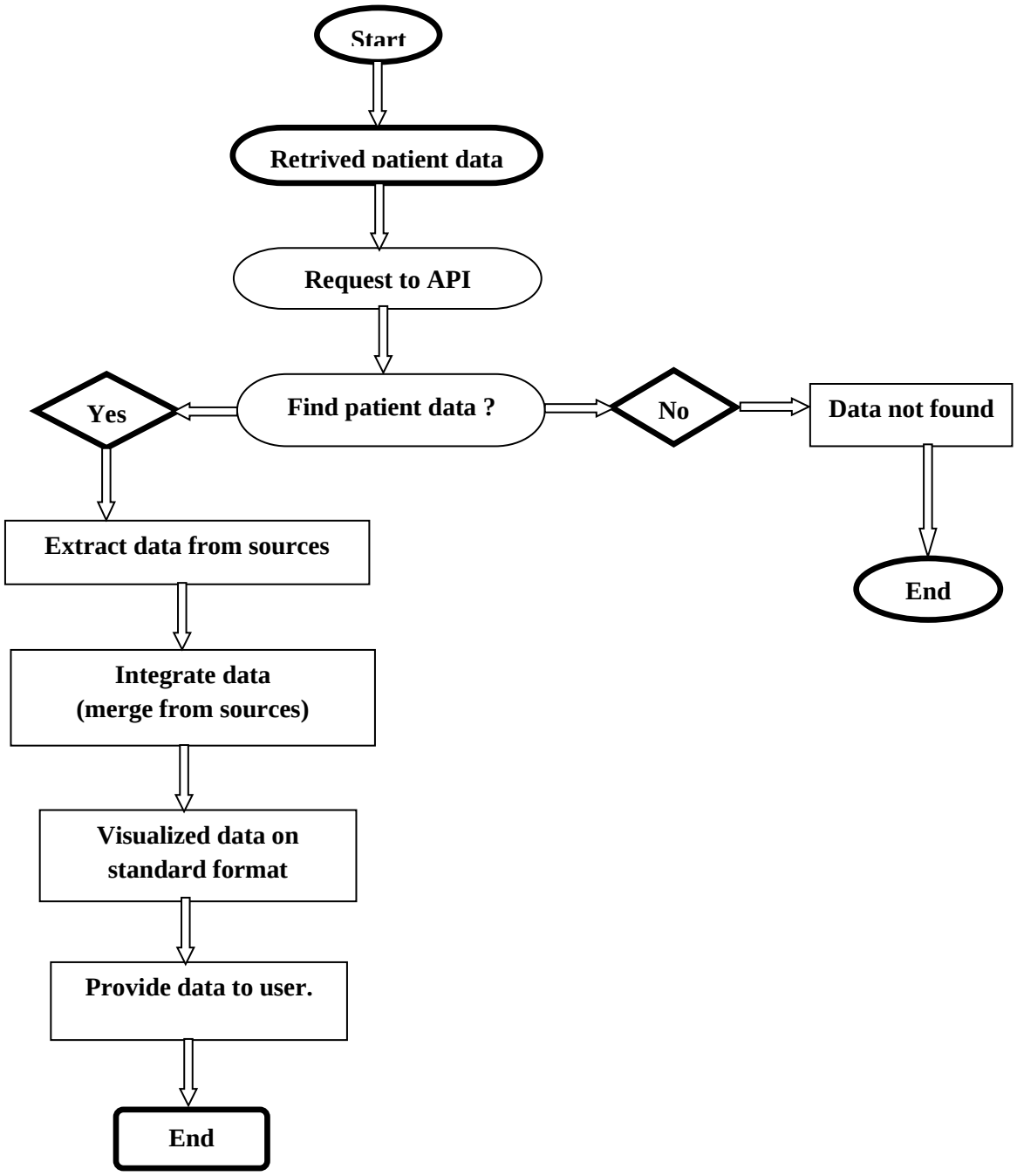


Figure 3: Activity Diagram

Flow Chart Steps

- **Start:** Ingestion process starts
- **Retrieve Patient Request:** Patient data request is received by the system.
- **Find Patient Data?:**
  Yes: Patient data found, go to next step.
  Data Not Found (Log Error): Patient data not found, log error and terminate process.
- **Extract Data from Sources:** Extract patient data from data sources (EHRs, LIS, Imaging Systems, Pharmacy Information Systems and other databases and registries.)
- **Integrate Data (Merge from Sources):** Transformed data from each source are integrated into a single dataset.
- **Visualize/Analyze Data:** Data visualization tools and analytics engines are used to analyze and visualize the data.
- **Provide Data to User:** Integrated and secure data is provided to the user (healthcare provider or other authorized users.
- **End:** Ingestion process terminates.

4.6 Interface to share data from databases

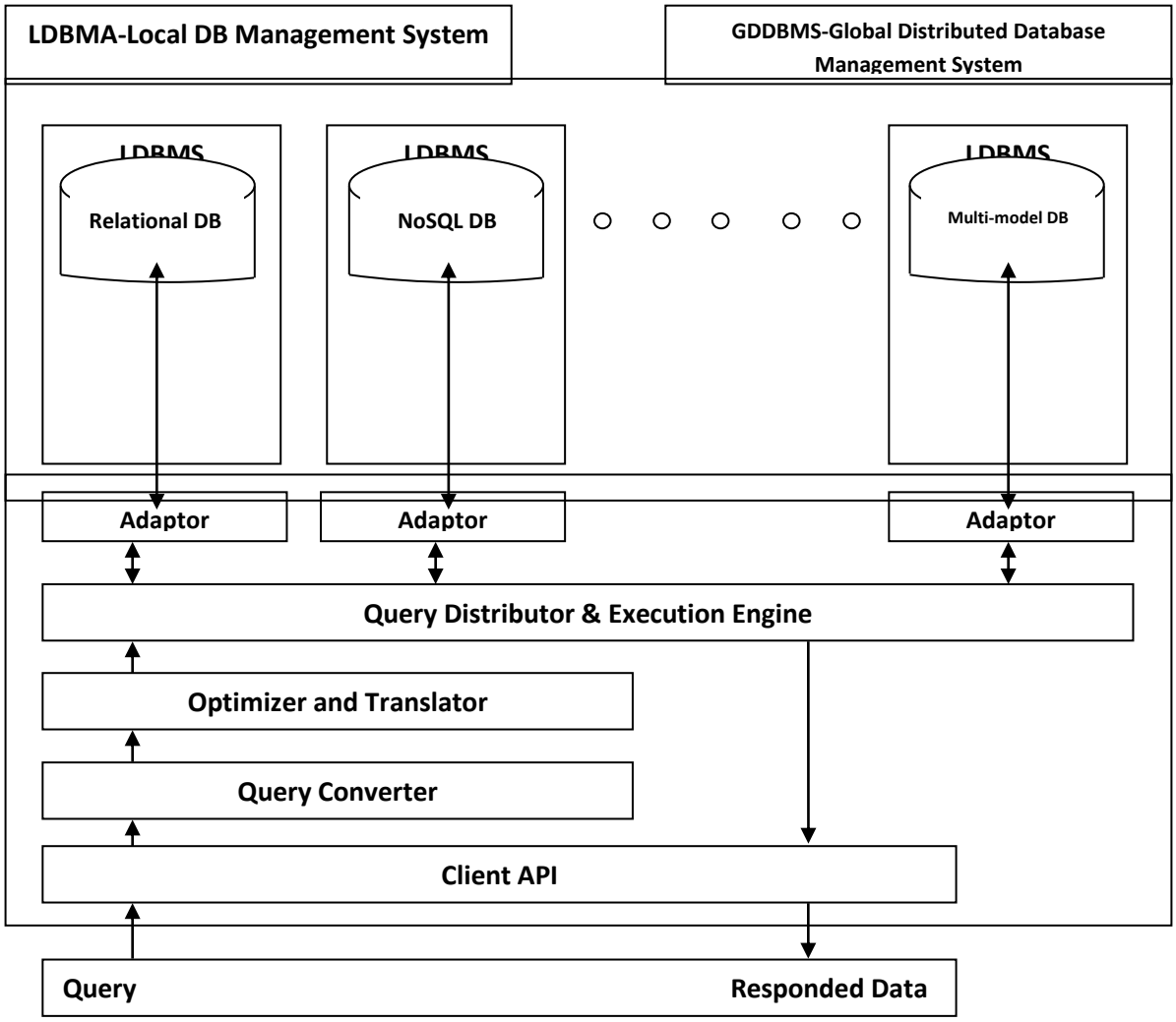


Figure 4 : System Interfacing

The required data was successfully fetched by the middleware API from databases connected in various network topologies. We test several requests to gauge the response. the duration of the inquiries made. The identical query was run multiple times for every network made up of various DDBs connected in both homogeneous and heterogeneous ways.

Query Distributor & Execution Engine: Execution Engine performs the queries distributed by Query Distributor in a distributed computing environment. Query fragments are evaluated on individual nodes or servers. Intermediate results may be retrieved, processed and aggregated by the Execution Engine. Finally, the final output query generated is returned.

Optimizer and Translator: Optimizer creates an optimal query execution plan based on the SQL query generated and database schema, statistics and available indexes. The main goal of Optimizer is to minimize the time taken to execute the query and to use minimum resources while producing correct results.

Query Converter: Syntax translation, data type mapping and function mapping are three main features of a Query Converter. It addresses various challenges around SQL variations, complex queries and impact on optimization. It provides consistency in query structure and efficient execution in all environments.

Client API: Client API resides between frontend UI and backend server or services. It provides necessary abstraction from the backend operations and data access layer to the client applications.

4.7 Mathematical model

Three levels make up the suggested architecture: the database layer, the middle layer, and the user layer. Fig. 5 shows a Petri Net-based mathematical model. The model illustrates how different system layers work together to retrieve data from many sources. The model displays both the sequence of events and the larger system in which they happen simultaneously. The model comprises eight places (p3, p4, p5, p6, p7, p8, p11, p12) and transitions (t3, t4, t5, t6, t7, t8, t11, t12) for middleware, two places (p9, p10) and transitions (t9, t10) for the databases layer. The model also includes five places (p1, p2, p13, p14, p15) and four transitions (t1, t2, t13, t14) for users. Every state in the three layers of the model has a certain purpose.

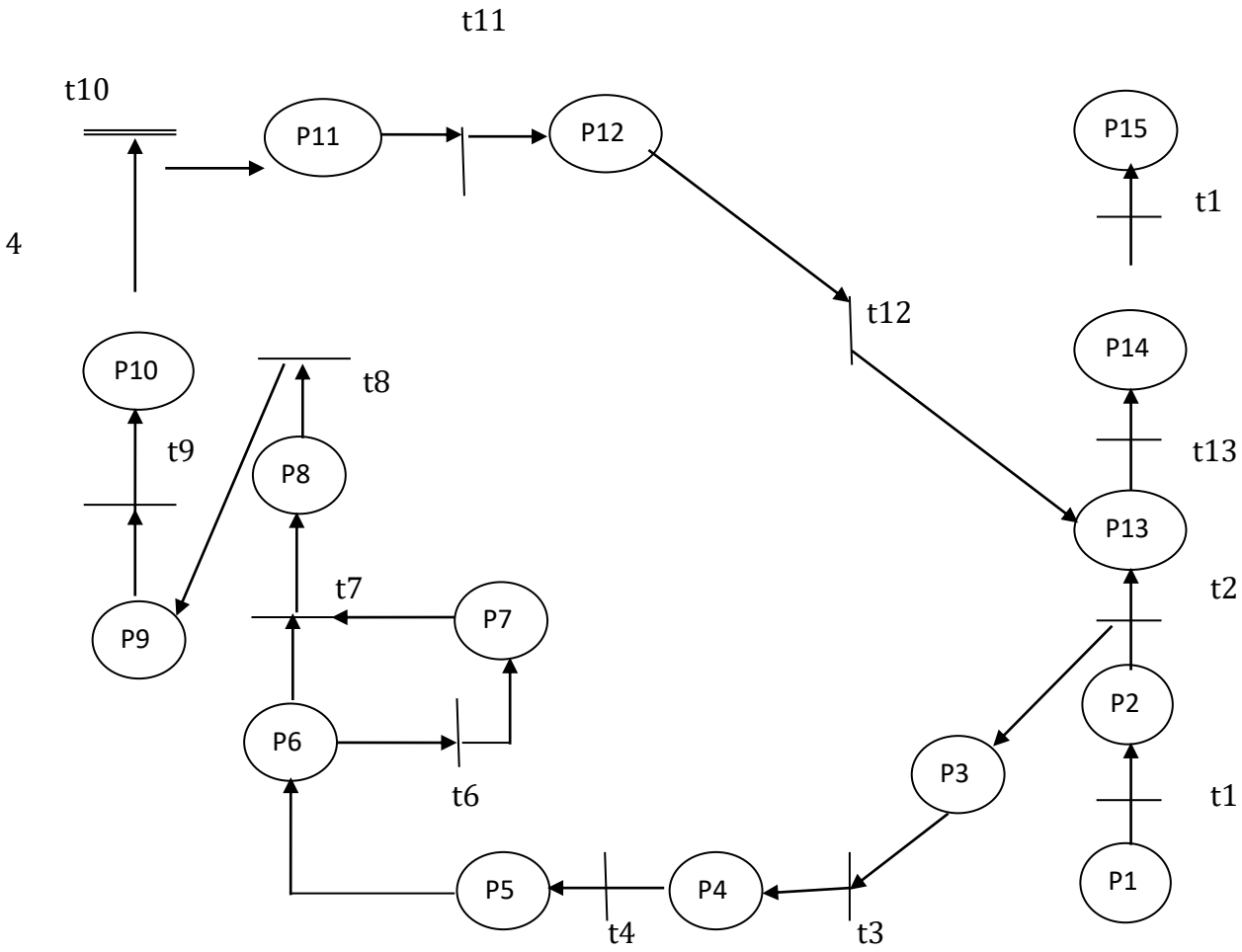


Figure 5 : Mathematical Model

4.8 Json Sample Data

JSON (JavaScript Object Notation) is a light-weight data-interchange format, easy for humans to read and write, and familiar to programmers in the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These characteristics make JSON a good choice as a data-interchange language.

JSON consists of two constructs:

- **A collection of name/value pairs:** In other languages, this is implemented as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- **An ordered list of values:** This is implemented as an array, vector, list, or sequence in most languages.

In "Designing & Developing an Efficient Technique for Patient Data Handling" project, JSON plays a very important role.JSON (JavaScript Object Notation) is a data format and it is considered as a must for patient data handling project because of its interoperability, convenience, light-weight, standardization and web technology compatibility nature. Using JSON format, patient data handling project would be providing a seamless integration and effective data exchange facility between varied healthcare databases and systems for the benefit of quality patient care and optimized healthcare delivery.


```
{
    "patient_id" : 01,
        "name" : "tamal",
    "age" : 35,
    "sex" : "male",
    "prescriptions" : [
                        {
                            "date" : "15/09/2020",
                            "bp" : { "diastolic" : 70 , "systolic" : 120},
                            "sugar" : 6.7,
```

"weght" : "67 kg",
"drugs" : [
{
"name": "paracetamol",
"type" : "tablet",
"unit" : 1,
"dosage" : "twice a day",
"time" : "after meal"
},
{
"name": "ursocol",
"type" : "tablet",
"unit" : 1,
"dosage" : "twice a day",
"time" : "before meal"
},
{
"name": "rectacare",
"type" : "ointment",
"unit" : 1 ,
"dosage" : "use once a day",
"time" : "any time"
}
],
"diet_to_follow" : "don't take prawn; red meat",

```
                "physician" : {
                                "id" : 10102,
                                "name" : "Mr. Tanmoy",
                                "specialist" : "medicine"
                        }
        },
{
                "date" : "11/11/2020",
                "bp" : { "diastolic" : 85 , "systolic" : 135},
                "sugar" : 5.7,
                "weght" : "68 kg",
                "drugs" : [
                                {
                                  "name": "metronidazol",
                                  "type" : "tablet",
                  "unit" : 1,
                                  "dosage" : "trice a day",
                                  "time" : "after meal"
                        },
                                {
                                  "name": "antacid",
                                  "type" : "tablet",
                  "unit" : 1,
                                  "dosage" : "twice a day",
                                  "time" : "before meal"
```

```
                }
                ],
        "diet_to_follow" : "don't take allergic food",
        "physician" : {
                "id" : 10111,
                "name" : "Mr. Raoshan",
                "specialist" : "medicine"
                }
},
```


4.9 Summary

Design phase is basically the planning and drawing phase where in you think through and draw the details of the entire system. Nothing should be assumed or left out of scope. The system components, architecture, interfaces, data and security should all align with the requirements. A successful implementation lies ahead if the design is clear and precise with all the details marked.

Implementation phase is the actual building of the system. Building it from code, integrating the components together, testing it thoroughly, ensuring all requirements are met and many such activities. This is the phase which materializes the design into a working and ready to ship software system. A lot of diligence and discipline in coding and integration during implementation phase along with the clear thoughts in design phase and handed over details leads to a robust, efficient and easy to use software system.

CHAPTER 5

RESULT & DISCUSSION

5.1 Result

It was observed that, with the exception of the MySQL-formed network, the suggested architecture performed better than alternative homogenous distributed databases. Due to differences in their functional characteristics, tree structures, data distribution strategies, and other internal database architecture, different databases may produce different outcomes. The outcome thus showed that the built API could effectively and efficiently extract and combine the data from various databases.

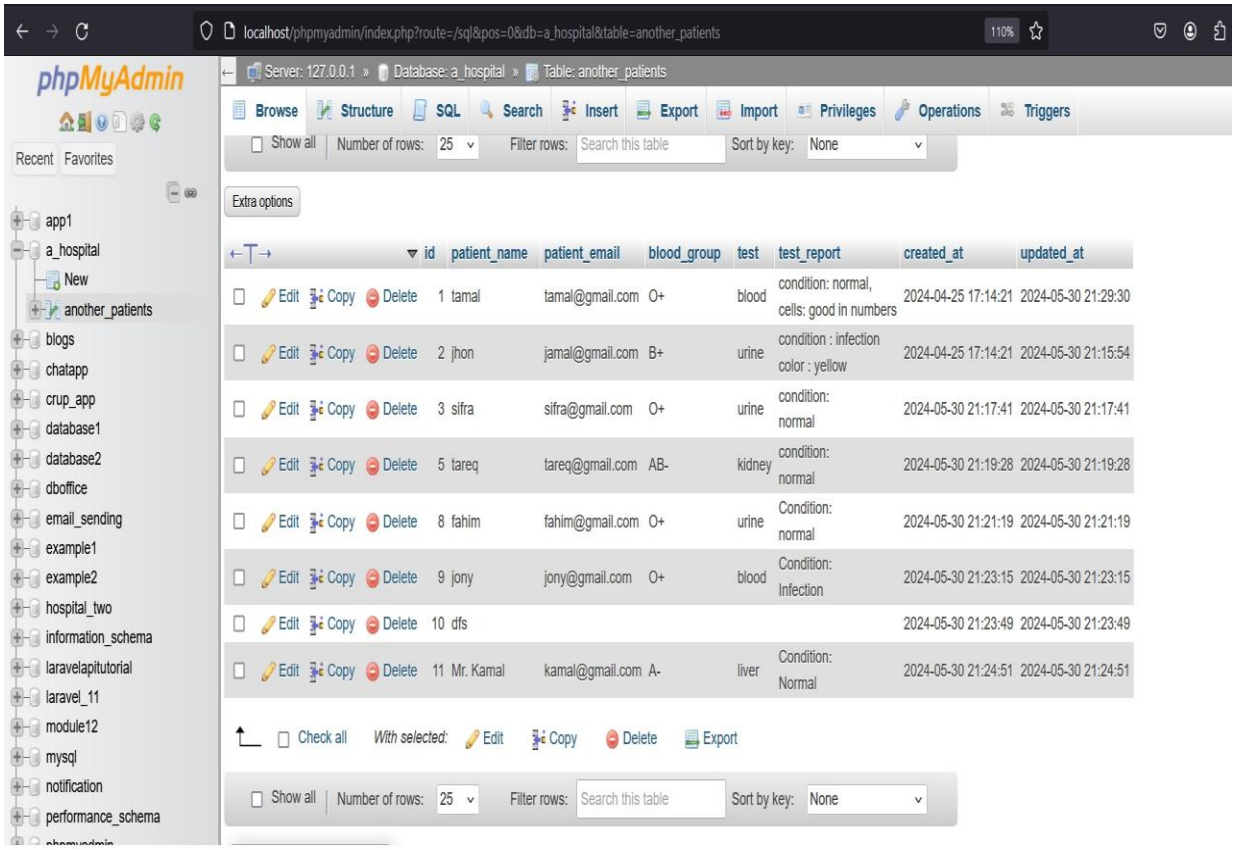
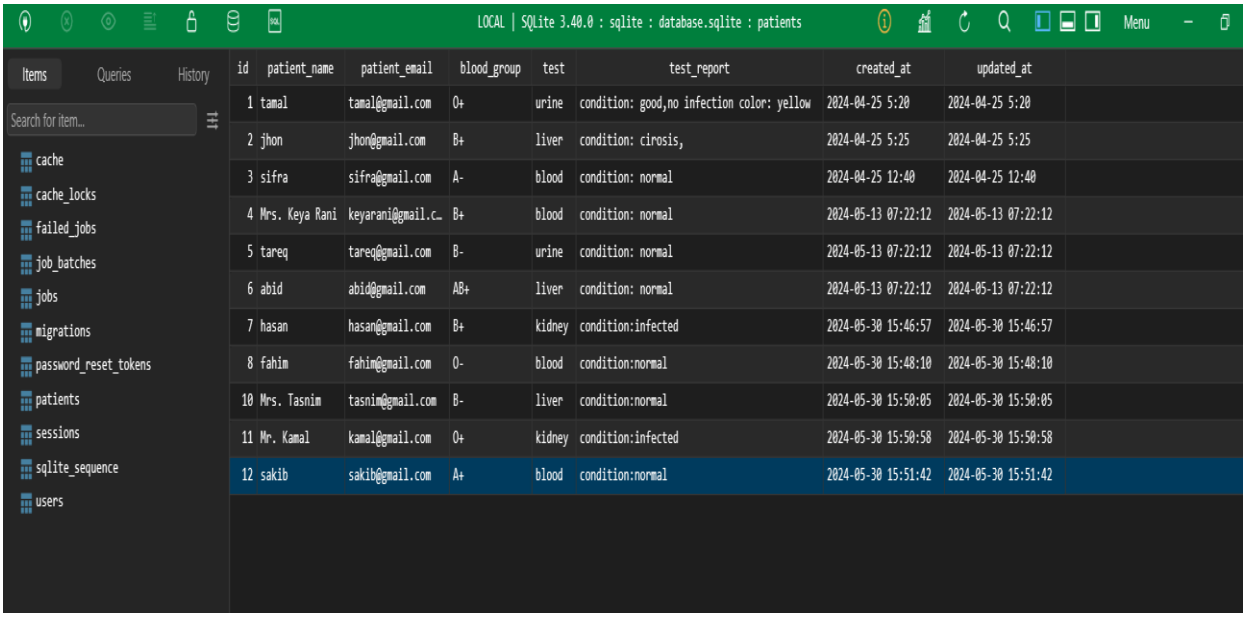


Figure 6 : Sql Database

Figure 6 demonstrates that medical information is stored in an SQL database. It consists of those details about the patient such as patient_Id, Patient_name, blood group, test repot & recording date. A relational database (SQL) is a database management system (DBMS) that stores data in tables and enables users to retrieve, manipulate, and control their data with SQL.



id	patient_name	patient_email	blood_group	test	test_report	created_at	updated_at
1	tamal	tamal@gmail.com	O+	urine	condition: good,no infection color: yellow	2024-04-25 5:20	2024-04-25 5:20
2	jhon	jhon@gmail.com	B+	liver	condition: cirosis,	2024-04-25 5:25	2024-04-25 5:25
3	sifra	sifra@gmail.com	A-	blood	condition: normal	2024-04-25 12:40	2024-04-25 12:40
4	Mrs. Keya Rani	keyarani@gmail.c…	B+	blood	condition: normal	2024-05-13 07:22:12	2024-05-13 07:22:12
5	tareq	tareq@gmail.com	B-	urine	condition: normal	2024-05-13 07:22:12	2024-05-13 07:22:12
6	abid	abid@gmail.com	AB+	liver	condition: normal	2024-05-13 07:22:12	2024-05-13 07:22:12
7	hasan	hasan@gmail.com	B+	kidney	condition:infected	2024-05-30 15:46:57	2024-05-30 15:46:57
8	fahim	fahim@gmail.com	O-	blood	condition:normal	2024-05-30 15:48:10	2024-05-30 15:48:10
10	Mrs. Tasnim	tasnim@gmail.com	B-	liver	condition:normal	2024-05-30 15:50:05	2024-05-30 15:50:05
11	Mr. Kamal	kamal@gmail.com	O+	kidney	condition:infected	2024-05-30 15:50:58	2024-05-30 15:50:58
12	sakib	sakib@gmail.com	A+	blood	condition:normal	2024-05-30 15:51:42	2024-05-30 15:51:42

Figure 7 : SQLite database

In the illustration below, it is evident that patient information is stored in a SQLite database consisting of fields such patient_Id, Patient_name, blood group test repot recording date. SQLite is a light weight self-contained SQL database engine noted for its simplicity, portability and single user nature. Different from the conventional database systems, SQLite does not require a separate server process and hence can be embedded in applications directly. The serverless design and few needs of configuration make SQLite a good choice even for small-sized and middle-sized projects and for prototyping or developing mobile apps. SQLite may be called a lightweight system but it still has features like a well implemented transaction mechanism, strict SQL adherence, and high speed in most scenarios.

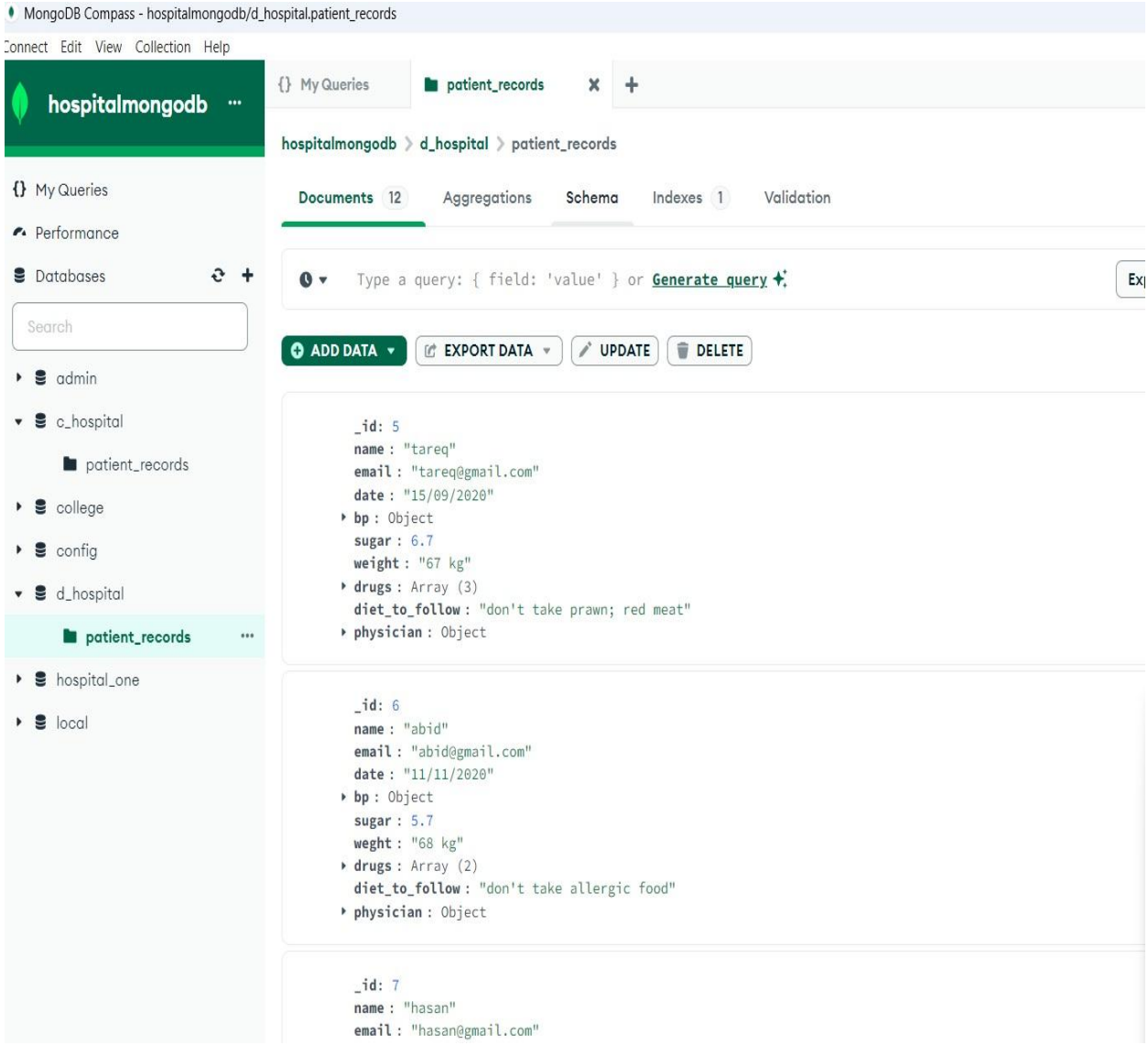
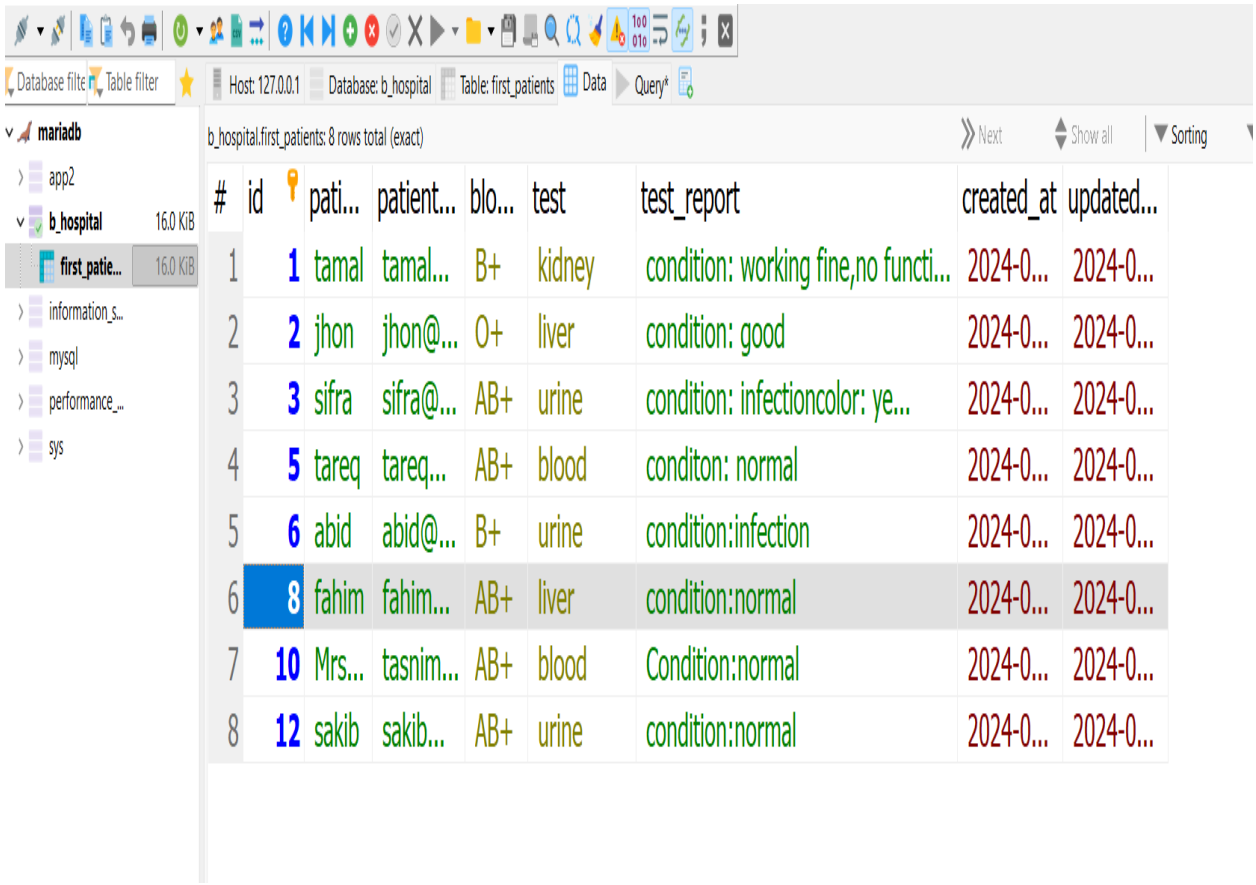


Figure 8: MongoDB Database

MongoDB shines in the world of databases because it uses a new way to store data in JSON-like files. This method lets data be kept in a more flexible format instead of the fixed setup of regular databases. This lets coders work with changing or complex data

easily without needing a set plan first. MongoDB is also great at growing. It can spread out over many servers by splitting and copying data. This means it can handle lots of data and many users without losing speed or being unreliable.



#	id	pati...	patient...	blo...	test	test_report	created_at	updated...
1	1	tamal	tamal...	B+	kidney	condition: working fine,no functi...	2024-0...	2024-0...
2	2	jhon	jhon@...	O+	liver	condition: good	2024-0...	2024-0...
3	3	sifra	sifra@...	AB+	urine	condition: infectioncolor: ye...	2024-0...	2024-0...
4	5	tareq	tareq...	AB+	blood	conditon: normal	2024-0...	2024-0...
5	6	abid	abid@...	B+	urine	condition:infection	2024-0...	2024-0...
6	8	fahim	fahim...	AB+	liver	condition:normal	2024-0...	2024-0...
7	10	Mrs...	tasnim...	AB+	blood	Condition:normal	2024-0...	2024-0...
8	12	sakib	sakib...	AB+	urine	condition:normal	2024-0...	2024-0...

Figure 9 ; MariaDB database

MariaDB, a free relational database management system, keeps working like MySQL but adds new things and better parts. This makes it easy for users to switch from MySQL to MariaDB without messing up current apps and setups. In image 9, it shows that health data is kept in a MariaDB database. The database holds patient ID, patient name, blood type, test report, and the date it was recorded.

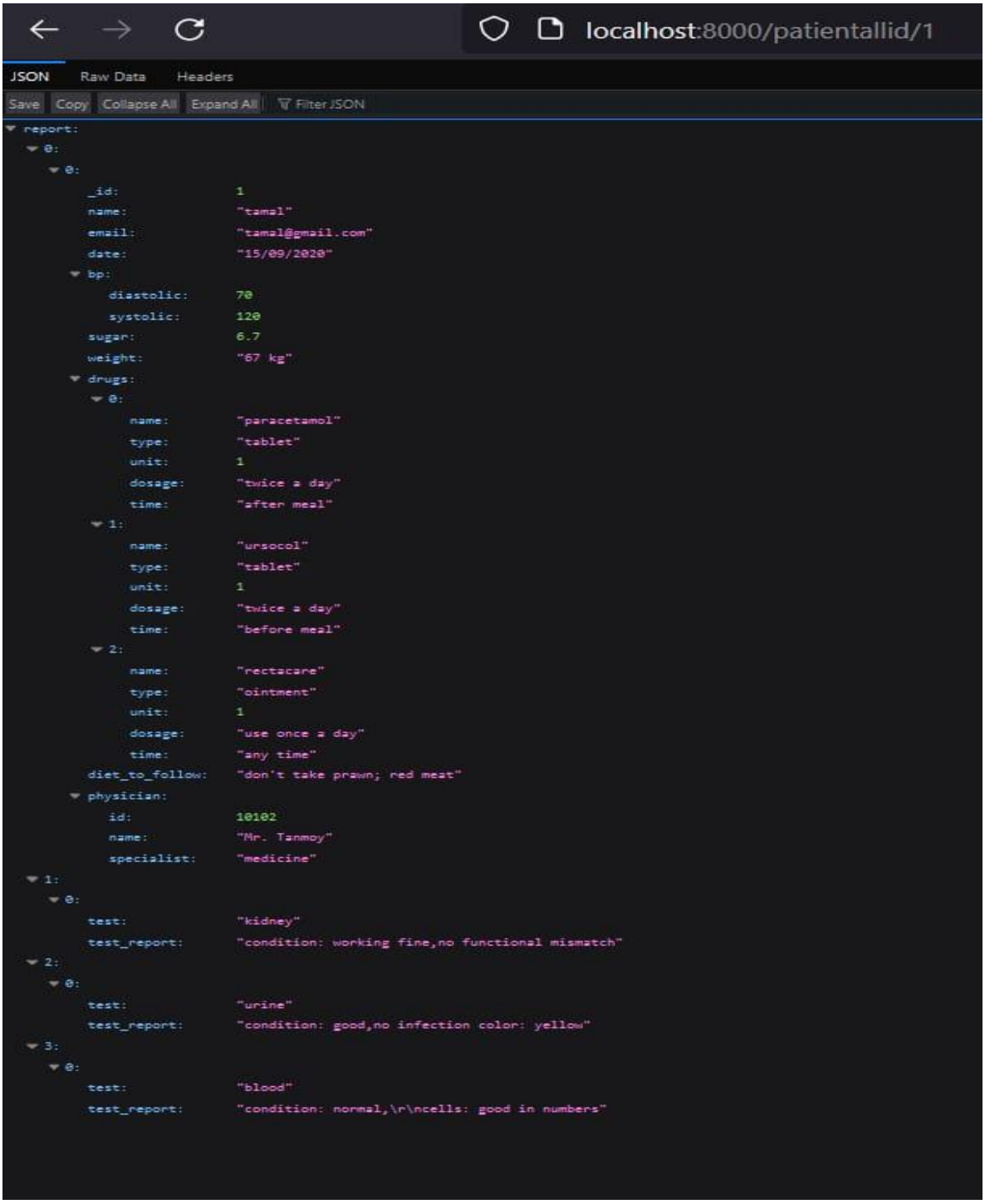


Figure 10: Find patient data on 4 medicals

Figure 10 shows how to look for patient details across four health records or systems. It shows a search box where users put in search details or patient IDs. The picture has symbols or dots for the different health sources, all linked to the search box with arrows or

lines. These links show how the search question goes to each data source to get information back.

Figure 11: Find patient data on 3 medicals

Figure 11 displays that the described patient information could be matched in three medical sources only from among other available sources. It contains a search interface where a user can type in his/her search parameters or patient's registration details. After this, three medical sources can be chosen by the users from any of the available options

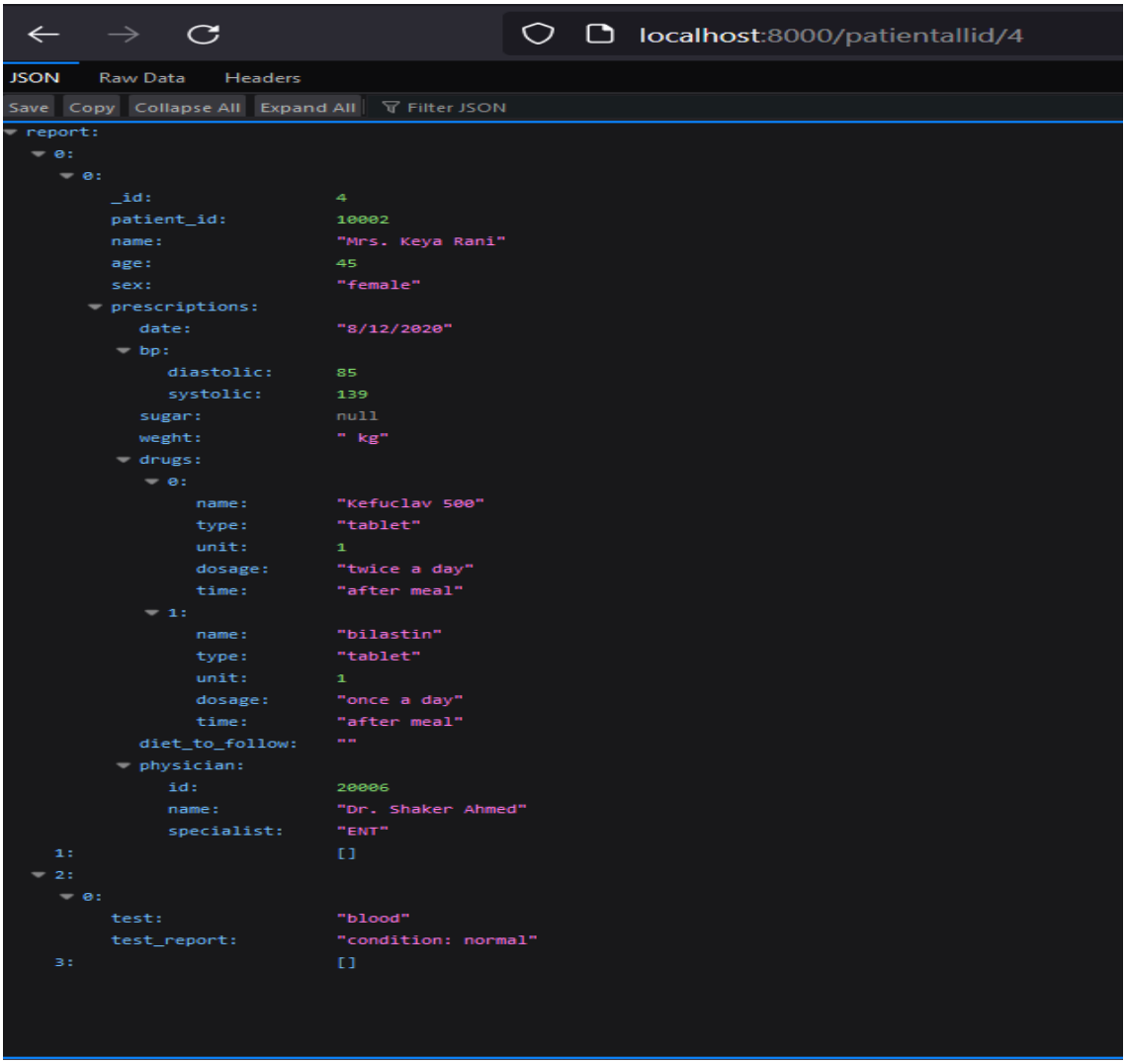


Figure 12: Find patient data on 2 medicals

In Figure 12, you can see a particular search situation. In this case we are trying to find medical records of a patient from two different medical sources. After entering a query, the system connects to them on the network. This operation involves checking IDs, formulating queries, transmitting them, and processing responses for each of them. In the end, obtained patient details from these two sites may be combined together or displayed as is presented in the diagram.

The result screen of the following figures show the efficiency and effectiveness of retrieving patient data from multiple medical sources. The result screen of this system shows the ability of the system to aggregate all patient information such as personal information, medical history, diagnosis reports, treatment plan, appointments and so on. The aggregated data enables the healthcare professionals to get an overall idea about the patient health status and patient journey, and thus allow them to make appropriate decisions and provide personalized care.

The result screen of this system also demonstrates the scalability of the application, and allows the users to customize their search based on their requirements, and to select the number of desired medical sources in each query. This shows the flexibility of the system to work with different configurations and settings according to the user preferences. In addition, the automated and semi-automated process of retrieving the data from multiple medical sources, and the user-friendly interface enhances the usability and accessibility for the healthcare professionals, and allows them to easily navigate through the different features of the system. The result summary demonstrates the importance of this system in aiding the healthcare professionals in providing high quality care, by enabling them to retrieve complete patient data from multiple medical sources in time.

5.2 Limitation

Patient data is stored in Hospital Information System (HIS) and many other database systems follows a schema. Some NoSQLs like Cassandra also use logical schema to store data. For schema-based databases, our proposed architecture may match sub-patterns or patterns of entities to perform data retrieval operations. For source databases that do not have schema, “Schema Dictionary” will not be able to create logical schema. The proposed method does not match schema-free database sub-pattern or pattern. In future, we want to add standards or other components to the middleware architecture for schema-less databases to solve these issues.

5.3 Future Work

Here, in "Design & Developing an Efficient Technique for Patient Data Handling" some possible directions for future work could be researched & implemented:

- **Integration of Emerging Technologies:**
  Explore how the emerging technologies like artificial intelligence (AI), machine learning (ML) and blockchain could be integrated with patient data handling systems to make it more efficient, secure and data analytics oriented.

- **Enhanced Data Analytics:**
  Data analytics techniques to be explored to get much more actionable insights from the patient data like predictive analytics based on patient data to detect the diseases at early stage, personalized recommendation of treatments based on patient data, population health management using patient data.

- **Mobile and Remote Healthcare Solutions:**
  Mobile and remote healthcare solutions could be explored and developed to provide secure access to patient data, telemedicine services, remote monitoring and virtual care delivery to meet the needs of patients and providers.

- **Collaborative Research and Implementation Initiatives:** Academia, industry, healthcare providers, policy makers and patient advocacy groups to be collaborated to encourage interdisciplinary research, innovation and implementation of efficient techniques for patient data handling.

By considering these directions for future work, the researchers and practitioners can surely contribute their bit in making the patient data handling practices better and better day by day for better healthcare outcomes, improved patient experiences and advancements in healthcare delivery.

5.4 Conclusion

Healthcare systems distribute and store patient data in multiple locations with varying structures (semi-structured, organized, and unstructured). The variety of data makes it difficult for health-related businesses to decide on a single database. Various DBMSs with heterogeneous architectures are used by the various organizations depending on the type and structure of data. Medical services are adversely affected when health data from disparate distributed systems is not accessed and shared.

In order to guarantee data integration and sharing from healthcare databases, this research suggested an architecture. In addition, to evaluate the functionality of the architecture, we connected several databases (Relational and NoSQL) and created a Client API based on our suggested architecture. Based on our suggested architecture, we created an interface for several database management systems that could successfully access data from the DDBs for integration.

As a result, the architecture may be advantageous in several ways. Health organizations might be able to exchange data in this way without having to modify their current data models in accordance with external standards. Health data from international remote databases dispersed across many sites could be accessed and shared by the users. Through the verification of their prior medical reports, such data sharing and accessibility might help the doctors treat the patients with high-quality care. The cost of patient care may be reduced if prior medical records are made available.

5.5 References

Almeida, A. O. (2020). Nosql distributed database for dicom objects. *IEEE International Conference on Bioinformatics and Biomedicine(BIBM)* , 1882–1885.

Balelli, I. S. (2021). A probabilistic framework for modeling the variability across federated datasets. *International Conference on Information Processing in Medical Imaging* , 701-714.

Bezerra, C. d. (2020). An hl7-based middleware for exchanging data and enabling interoperability in healthcare applications. *17th International Conference on Information Technology-New Generations (ITNG 2020)* , 461–467.

Carbonaro, A. P. (2018). *Integrating heterogeneous data of healthcare devices to enable domain data management.* Knowled: e-Learn.

Cosío-León, M. O.-C.-H. (2018). The use of standards in embedded devices to achieve end to end semantic interoperability on health systems. *Comput. Standards Interfaces* , 57, 68–73 .

Kiourtis, A. N. (2019). Aggregating the syntactic and semantic similarity of healthcare data towards their transformation to hl7 FHIR through ontology matching. *Int. J. Med. Inform* , 132, 104002.

Kumari, R. N. (144–149). ntegration of blockchain in WBAN. *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* , 2019.

Messaoudi, C. F. (2021). A semantic mediator based system using spark for the integration of heterogeneous proteomics data sources. *Concurr. Comput. Pract* , 162-176.

Roehrs, A. D. (2017). Personal health records: a systematic literature review. *J. Med.nternet Res* , 19(1), e13.

Singh, J. B. (2022). *On middleware for emerging health services.* Appl. 5(1): Internet Serv.

Yuksel, M. D. (2023). A case for enterprise interoperability in healthcare it: Personal health record systems. *E-Health and Telemedicine* , 1073–1096.

zarm, M. B. (2023). Breaking the healthcare interoperability barrier by empowering and engaging. *the healthcare system. Proc* , 113, 326–333.