



# Doxa: A Web Based User-Friendly Document Oriented Database Management System with Integrated File Handling

Arawole M.O.

<sup>1</sup>( Department of Computer Science, Ladoke Akintola University of Technology, Ogbomosho, Oyo State, Nigeria)

Adetunji A.B.

<sup>2</sup>( Department of Computer Science, Ladoke Akintola University of Technology, Ogbomosho, Oyo State, Nigeria)  
Corresponding Author: Arawole M.O.

**ABSTRACT:** Managing databases can be challenging for non-technical users due to complex interfaces and cumbersome file-handling processes. Existing database management systems (DBMS) such as the Object-Oriented SGML/HYTime Compliant Multimedia DBMS and CRIUS have attempted to address these challenges by providing structured data management and user-friendly interfaces for non technical users. However, some require knowledge of object-oriented programming or lack integrated file-handling capabilities.

This paper introduces Doxa, a user-friendly, web-based document-oriented database management system with integrated file handling, designed to simplify data and file management for users of all technical backgrounds. The paper is structured into Introduction, Literature Review, Methodology, Results and Discussion, and Conclusion, covering system requirements definition, design, implementation, testing, and cloud deployment. The developed application features a streamlined interface with simplified file viewing, upload, and download functionality, making it highly accessible.

By addressing the limitations of existing database management systems, this paper aims to bridge the gap between complex database systems and non-technical users, ensuring accessible and efficient data management for a broader audience.

**KEYWORDS:** Database Management System, Data Management, Object-Oriented Programming, Document-Oriented Database, Non-Technical Users, File Handling, Cloud Deployment.

Received 14 June., 2025; Revised 24 June., 2025; Accepted 29 June., 2025 © The author(s) 2025.

Published with open access at [www.questjournals.org](http://www.questjournals.org)

## I. INTRODUCTION

Data plays a vital role in analysis, problem-solving, and innovation across fields like science, business, education, and healthcare. It helps people make decisions, improve processes, and spot patterns, showing just how important it is in today's world.

In the past, data was managed using physical documents stored in filing cabinets. While this method is still used in some places, it has many drawbacks, such as being time-consuming to search through, taking up a lot of space, and being at risk of damage or loss. The introduction of database management systems (DBMS) in the 1960s changed the way data was handled by offering digital tools to store, organize, and access information more efficiently. Building on this advancement, DBMS brought several important capabilities that transformed data management practices. These systems allowed the storage of large amounts of data in a structured format, making it easier to retrieve and update information as needed. They support multiple users at the same time without compromising data accuracy and provide tools to control access, ensuring that only authorized individuals can view or modify sensitive data. DBMS also introduced features like data backup and recovery to guard against data loss, as well as tools for maintaining data consistency and integrity across applications. Overall, these capabilities made data handling more reliable, efficient, and secure across various industries.

However, most DBMS are designed for technical users like programmers, leaving non-technical individuals struggling to use them effectively. These systems are filled with complex features and technical terms,

making them hard to navigate and most require the knowledge of a query language. Another issue is file management. Many DBMS don't store files directly but instead require file references like paths or URLs. This process, usually handled by developers in applications, is not user-friendly for non-technical users. Even when files are stored, they are often converted into formats like Binary Large Objects (BLOBs) and also require the knowledge of a query language such as SQL (Structured Query Language) making it difficult for regular users to access or view.

These challenges show the need for a simpler, more user-friendly DBMS that everyone can use, regardless of their technical skills. This paper introduces Doxa, a web-based document-oriented database management system designed to make data and file management easy and accessible for all users.

## **II. LITERATURE REVIEW**

Data can essentially be defined as an information that is raw and unprocessed and in itself is often meaningless but when collected, organized, processed and interpreted, it can provide valuable and useful insight and knowledge that can be utilized in decision making, enhancing productivity and so on. Elmasri and Navathe defined data as "known facts that can be recorded and have implicit meaning" [1]. Data is raw material information and is collected in a database [2].

### **2.1 Data Management**

Data management involves collecting, storing, organizing, and maintaining data to ensure its accuracy, accessibility, reliability, and security. The goal is to manage data effectively throughout its lifecycle, from creation to disposal.

### **2.2 Database**

Watt defines a database as a shared collection of related data used to support the activities of a particular organization [3]. A database is a collection of interrelated data, typically stored according to a data model [4]. A database is a structured collection of data that is stored electronically and managed through a database management system to enable efficient access, organization, and retrieval.

### **2.3 Database Management System (DBMS)**

A database management system is a dedicated software that facilitates the creation, organization, management, and manipulation of databases. It acts as an intermediary between user applications and the underlying database, providing an efficient and secure way to interact with the data stored in the database. The database management system is a set of rules or procedures which are used to create, organize and manipulate the data [5].

### **2.4 Relational Database Management System (RDBMS)**

A relational database management system (sometimes called SQL databases) is one that organizes data into structured tables with rows and columns. They adhere to the ACID (availability, consistency, isolation, and durability) attribute and are designed for structured data [6].

Each column is representing a field, each row is representing a record and each table is representing a specific entity. All the values in a column are of the same datatype. Relationships between tables are established using keys and every table in the database has a key field which uniquely identifies each record in the table. It uses a query language called SQL (Structured Query Language) for manipulating data. Its advantages include support for complex relationships between data tables, guarantees that transactions are atomic, consistent, isolated, and durable. It also has its own disadvantages such as it requires skilled knowledge to utilize, the schema is not flexible to easily adapt to changes in data entry. Examples are Oracle, MySQL, PostgreSQL.

### **2.5 Non-relational database Management system**

It is also referred to as NoSQL DBMS (pronounced "no sequel DBMS") which means Not Only SQL DBMS. It refers to a category of database management systems that doesn't require the use of SQL, designed to handle various types of data and data models beyond the tabular, structured data that RDBMS excels at. NoSQL databases management systems are characterized by their flexibility, scalability, and ability to work with unstructured or semi-structured data. They are particularly well-suited for applications that require handling large volumes of data, real-time data processing, and flexible data models. Non-relational databases don't require fixed table schemas and their ability to store structured, semi-structured and unstructured data makes them easy to scale and manage.

NoSQL database management system key features are scalability, flexible schemas, fast queries due to data model, ease of use, and it doesn't require SQL.

## **2.5 Types of Non-relational Database Management Systems**

### **2.5.1 Graph Database Management System**

A graph DBMS also called graph-oriented DBMS is a NoSQL database management system that organizes and stores data in a structure that resembles a graph, where data entities are represented as nodes, and the connections or relationships between them are represented as edges.

The main characteristic of a graph database is that the data are conceptually modeled and presented to the user as a graph, that is, the data structures (data and/or schema) are represented by graphs, or by data structures generalizing the notion of graph [7].

Many graph databases are schema-less, allowing for flexible data modeling and the addition of new relationships without altering existing data. Examples are Neo4j, Microsoft Azure Cosmos DB, OrientDB, ArangoDB.

### **2.5.2 Wide-Column Database**

A wide-column database also known as column family database, It stores data in tables, rows and flexible columns. The word flexible is used to depict the fact that the name and the format of the column can vary across rows, even within the same table. Examples are Google Big Table, ScyllaDB, Apache HBase.

### **2.5.3 Key-Value Database Management System**

Key-value database management system is a NoSQL database system that stores data as simple key-value pairs, making them efficient for fast data retrieval. They are known for their high performance and scalability which can be used for cases like caching, session management, and real-time analytics. Examples of key-value DBMS include Redis, Memcached, and Amazon DynamoDB, and they are very useful in applications where quick and straightforward data access is a priority.

### **2.5.4 Document Database Management System**

It is also known as a document-oriented DBMS, is a type of NoSQL DBMS that is designed to store, retrieve, and manage data in a flexible and semi-structured document format. In a document database, data is stored as documents, which can be in various formats like JSON (JavaScript Object Notation), XML, BSON (Binary JSON), or others. In each document, there are pairs of fields and their corresponding values which can either be a string, a number, a boolean, an array, or an object. Document databases are commonly used in applications where data is semi-structured, dynamic, and schema-less. They are well-suited for scenarios like content management systems, e-commerce platforms, catalogs, and applications with variable and evolving data structures. MongoDB and Couchbase are examples of popular document-oriented databases.

Some of the key features of document databases include documents to have their own flexible structure or schema. This flexibility makes them very efficient for handling data where the structure can vary from one document to another. Another feature is the organization of data into collections and documents which can be likened to a record in a relational database and documents are grouped into collections which can be likened to a table in a relational database. Also, they are document-centric, that is data is organized into documents, where each document contains all the data related to a specific entity. Their performance is also high because they offer efficient and fast retrieval of data, making them suitable for read-heavy workloads and unlike relational databases, document databases typically avoid complex joins and instead store related data within a document or use references for relationships.

## **2.6 Related Works**

The Object-Oriented SGML/HYTime Compliant Multimedia Database Management System, developed by Özsü, Iglinski, Szafron, El-Medani, and Junghanns is a database management system that stores and manage SGML/HYTime compliant multimedia documents [8]. Although the DBMS doesn't necessarily store all the multimedia data (which includes text, images, videos and so on) some are stored in traditional files but ensures that user's access to the data is managed by the database. However, this system has its limitations. It is an object-oriented database management system requiring the knowledge of object oriented programming and it also uses a multimedia object query language based on the standard query language. This system will definitely be challenging for non-technical individuals.

CRIUS developed by Qian, LeFevre, and Jagadish is a user-friendly database management system that supports flexible schema and data modification [9]. It has a graphical user friendly interface (spreadsheet-like) enhancing direct manipulation of data. It permits users to create and refine schemas in a flexible way over time. It permits the creation and modification of schema through a simple drag-and-drop interface making it a very user friendly database management system that can be used by anyone. However, CRIUS is not developed to handle files.

The Document Management and Exchange System developed by Egredzija and Kovacic is a system that enables easy publishing and exchange of documents making users not worry about their distribution and storage

[10]. It is an open source technology and is deployable on popular web software platforms. It permits the storage of documents and access to documents. However, it is not a database management system and lacks the functionalities of a database management system.

### **III. METHODOLOGY**

#### **3.1 Defining the System Requirements**

The requirements necessary to outline the functionalities, features, and constraints of the database management system were defined. These requirements were categorized into functional and non-functional requirements.

##### **3.1.1 Functional Requirements for Doxa**

For Users:

1. The database management system must be available on the internet.
1. The user interface must not be bulky and complex.
2. It must authenticate users for protection of data.
3. Users must be able to create, edit and delete databases, collections and documents and they must also not find them challenging to create. They must also be able to link up collections through the use of nested data,
4. Users must be able to add other users to their databases.
5. The database must provide the opportunity for users to upload and download files by the use of buttons and icons.
6. If files are images or videos, users must be able to view them.
7. Users must be able to make data queries by searching.
8. Users must be able to deactivate their accounts.

For the Admin:

1. The admin must be able to login and logout of the admin section of the database management system.
2. The admin must be able to see the number of the database management system users and their details.
3. The admin must be able to see the number of databases, collections, documents existing in the database management system but not their contents.
4. The admin must be able to block the account of any user thereby making their databases uneditable.

##### **3.1.2 Non-Functional Requirements for Doxa**

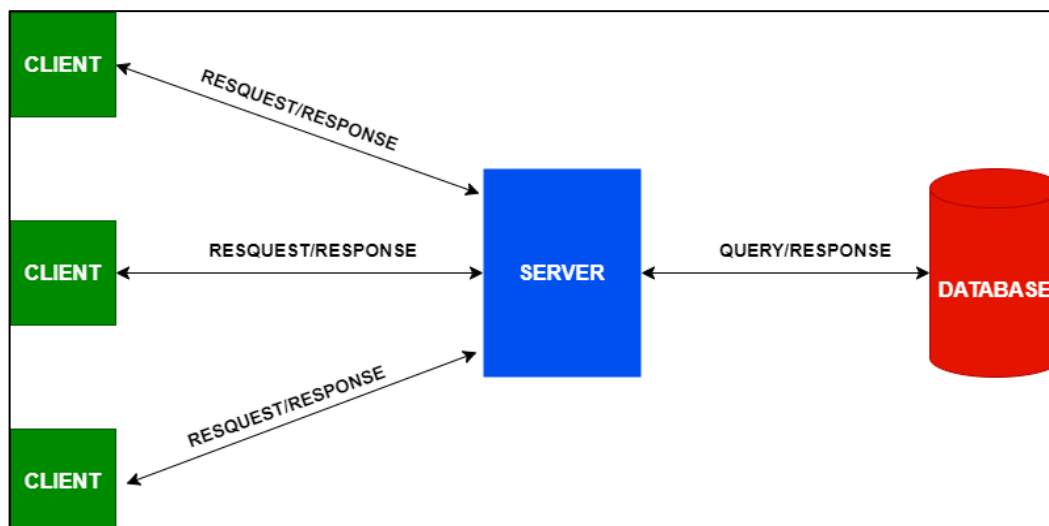
1. Resources must only be accessed by authorized and authenticated users.
2. All incoming data must always be verified and validated to ensure proper performance of the system.
3. The user's password must be protected in the database by encryption.
4. All elements (icons, colors) of the user interface must be consistent to promote familiarity.
5. System error must be well handled and the error notification should be user friendly.
6. The system must be available for use 24 hours per day.

#### **3.2 System Design**

At this phase the requirements were transformed into a blueprint for development by designing the necessary designs which included the architecture design, the logical design and the pseudocode.

##### **3.2.1 Architecture Design**

A three-tier client-server architecture was adopted which includes the client tier which handles the user interface and interactions, The server tier which manages application logic and processes HTTP requests and the database tier which stores and retrieves data. Figure 1 is a representation of the three-tier client-server architecture.



**Figure 1:** The Three Tier Client Server Architecture

### 3.2.2 Logical Design

The system was divided into two broad modules:

#### **User Module:**

1. **User Interface:** This component renders a user-friendly interface to facilitate user interactions with the system. This interface includes visually appealing forms, menus, and navigation tools, ensuring a smooth and enjoyable user experience.
2. **User Authentication and Management Component:** This component handles user registration and login. It also handles user profile update, account deactivation and query of user details. It interacts with the database management component to store and validate user credentials.
3. **Database Management Component:** Responsible for storing and retrieving data from the database. It is used by various other components to perform CRUD (create, read, update, delete) operations on data. Interacts with the file handling component to manage data related to files and file metadata.
4. **File Handling Component:** Handles file uploads, storage, and retrieval. It relates with the database management component for storing file metadata.
5. **Security Component:** This component ensures the security of data and user interactions. Collaborates with the user authentication component for secure authentication. Works with the user interface components for input validation and secure data transmission.
6. **Error Handling Component:** This component handles errors and provides error feedback to the user interface components for displaying error messages to users.

#### **Admin Module:**

1. **Administrative Interface Component:** This component provides an overview of the system required by the admin to manage and oversee system operations. The administrative interface provides a dedicated space for the admin to access system management features.
2. **Administrative User Management Component:** This component allows the admin to view all users, view the number of their databases, view the number of their collection, view the number of their documents, block users accounts and make their databases uneditable.

### 3.2.3 Pseudocode and FlowChart

The pseudocode outlines the sequence of actions a user performs when interacting with the system. Figure 2 is the flowchart of the system which shows each step a user will take in utilizing the system.

#### **Step 1: User Sign Up**

- If new user, register else, proceed to step 2.
- If registration is successful, proceed to step 2.
- If registration fails, return to sign-up.

#### **Step 2: User Login**

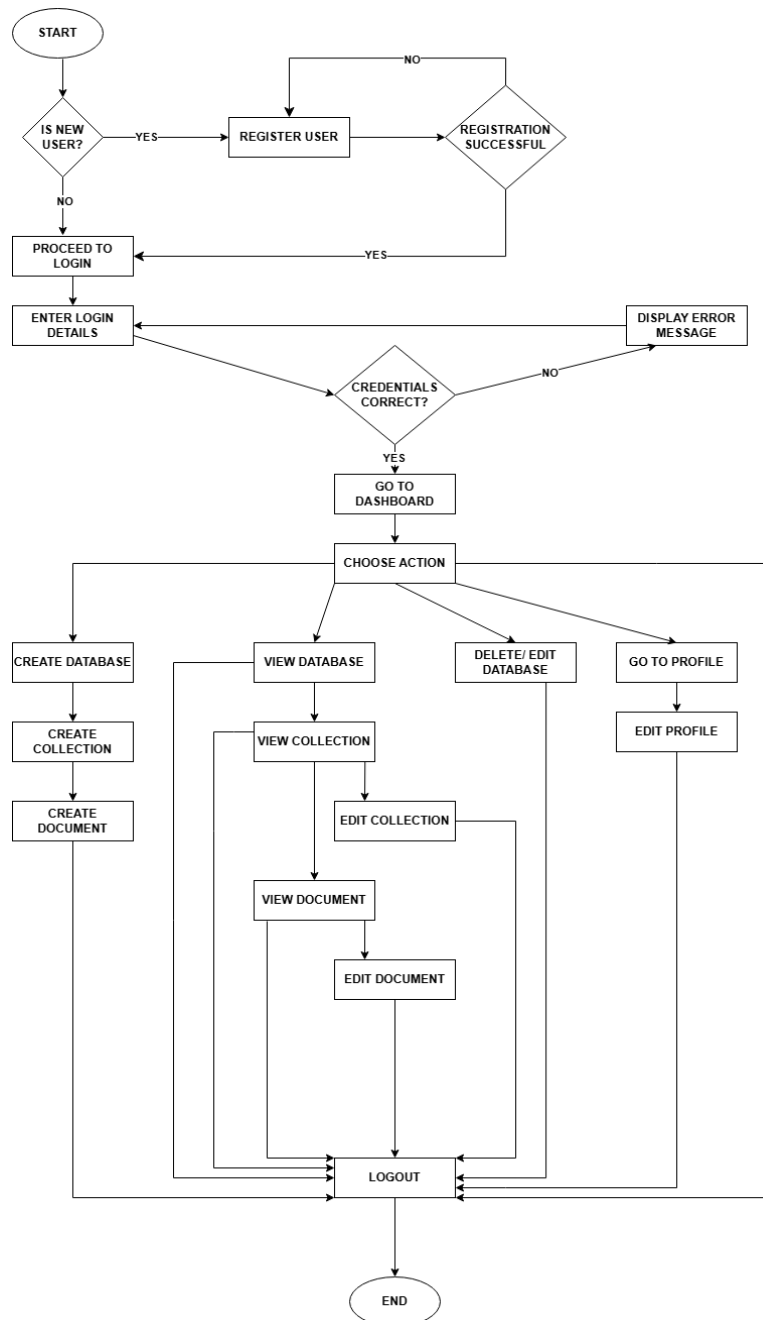
- Enter login credentials.
- If credentials are correct, proceed to step 3.
- If credentials are incorrect, display errors and return to step 2.

#### **Step 3: Dashboard**

- Choose an action:
  - Option 1: Create a new database.

1. Input database details.
  2. Create the database.
  3. Display database dashboard
  4. Add a new collection and define its schema
  5. Add a new document
  6. Return to database dashboard
- Option 2: View an existing database, collection or document
- Option 3: Delete a database, collection or documents.
1. Select a database, collection or documents.
  2. Confirm database, collection or documents deletion.
  3. Return to the Dashboard (Step 3).
- Option 4: Update Profile.
- Access and update user profile information and return to the Dashboard (Step 3).

**Step 4: Logout.**



**Figure 2:** System Operation Flow Chart

### 3.3. Implementation

JavaScript, a widely used programming language was utilized for the development of the software. During implementation, the software was divided broadly into two parts: the user interface (front-end) and the server (back-end).

Major software tools were utilized during this phase such as; React a JavaScript library for building user interfaces, Node.js a runtime environment that allows developers to execute JavaScript on the server-side, TypeScript a superset of JavaScript that adds static typing to JavaScript, Express a minimal and flexible Node.js web application framework, MongoDB a NoSQL document oriented database management system, Cloudinary a cloud-based media management platform.

### 3.4. Testing

Unit testing was conducted to validate the functionality of individual components, including: User Interface, User Authentication and Management, Database Management and File Handling.

### 3.5. Deployment

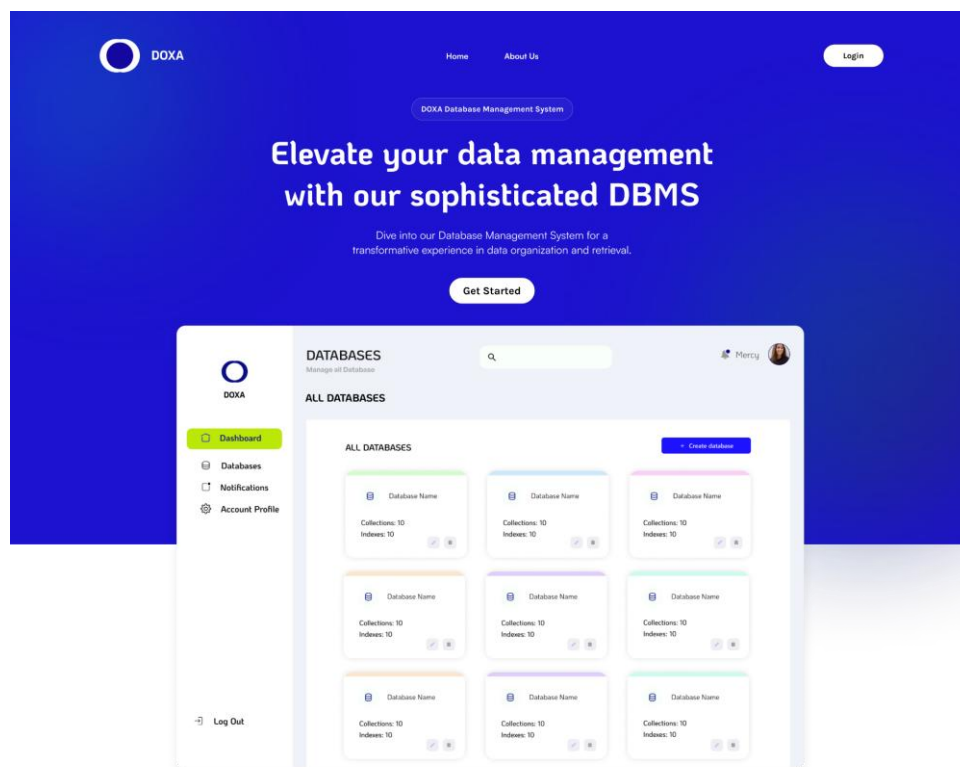
The system was deployed to a cloud platform (Render) to ensure accessibility over the internet.

## IV. RESULTS AND DISCUSSION

The system was developed following the system design and the resulting implementation is discussed in the following section.

### 4.1 User Authentication

Figure 3 is the landing page of the database management system. The landing page is the first interface users encounter when accessing the database management system via a website browser. It serves as the entry point, providing users with essential options to interact with the system. The page features intuitive buttons for registration, login, and accessing information about the system. These elements are designed to ensure a seamless and user-friendly experience, allowing users to quickly navigate and begin using the database management system.




**Figure 3:** Doxa Landing Page.

**Get started on DOXA!**

First Name


Last Name

Email

Phone

Password


**REGISTER**

**Figure 4:** Doxa Login Page.


**Figure 4** is the page for registering users. It receives the user's first name, last name, phone number, email address and password. Once the user clicks the register button a token is sent to their mail to verify their email and activate their account.

**Welcome back to DOXA!**

Email address

Password

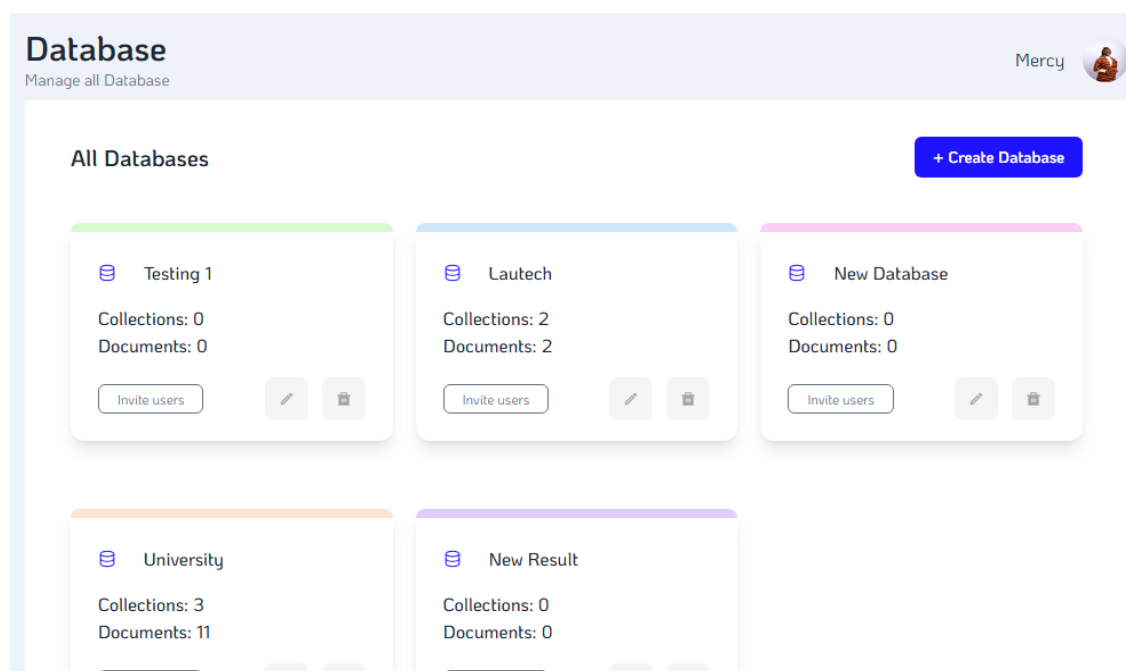
☐ Remember me [Forget password?](#)

**LOGIN**

**Figure 5:** Doxa Login Page.



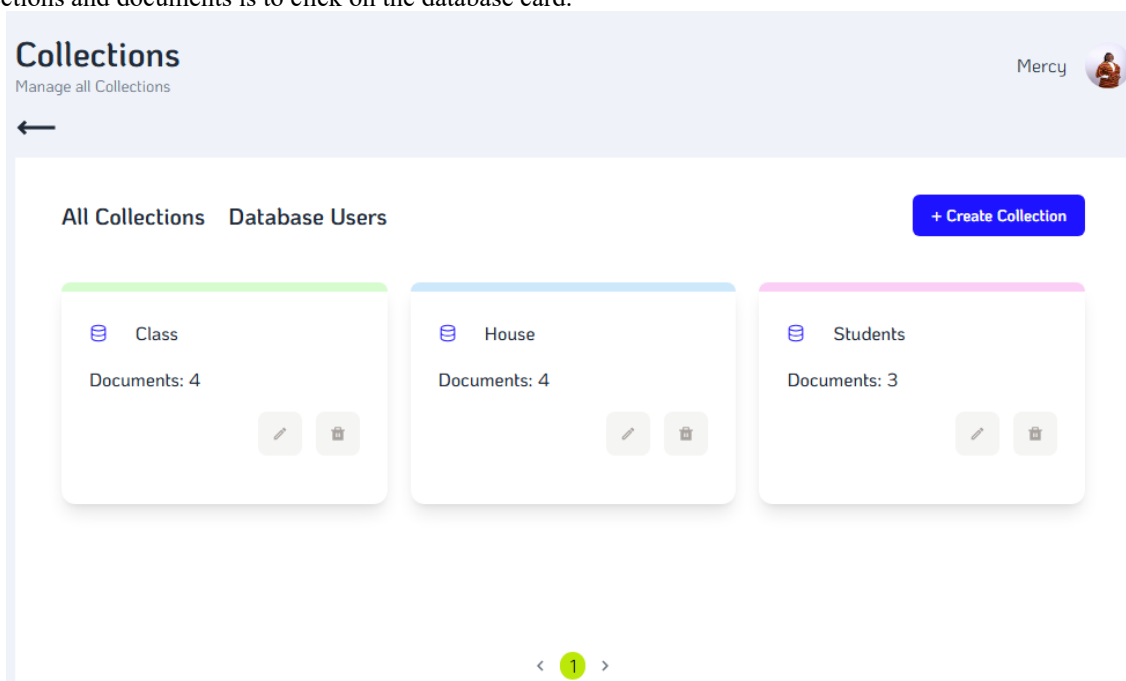
Figure 5 is the login page. It receives the user email and password and validates it before giving the user access to the application.



**Figure 6:** List of Created Databases.

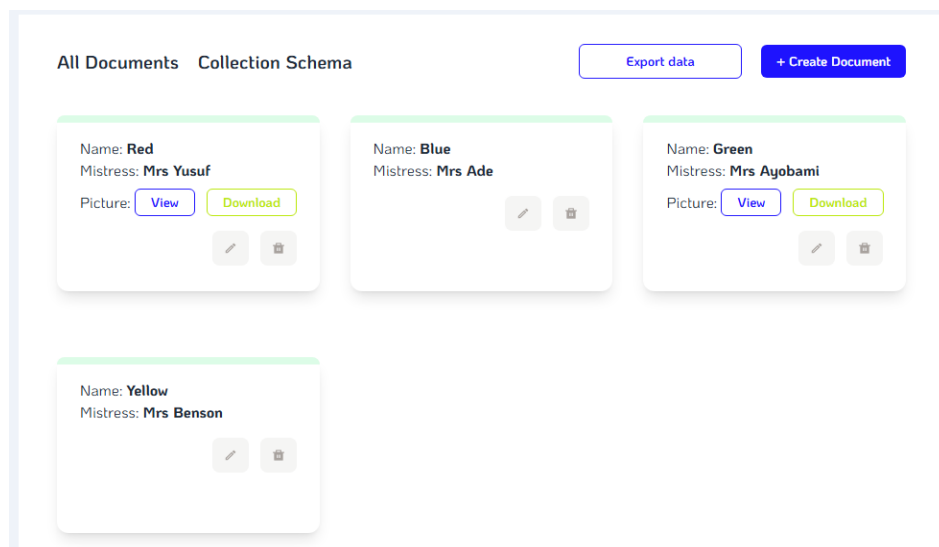
#### 4.2 Database Management Component

Figure 6 provides a list of the databases already created by the user. All they have to do to view a single database collections and documents is to click on the database card.



**Figure 7:** List of Created Collections.

Figure 7 provides a list of collections created under a particular database. To get to this page users need to click on a database card in the database page.



**Figure 8:** List of Created Documents.

Figure 8 provides a list of documents created under a particular collection. To get to this page users need to click on a collection card in the collection page.

## V. CONCLUSION

In summary, this paper has tackled the challenges common in conventional database management systems, addressing issues related to complex user interfaces and intricate file handling. The primary objective was to develop a more accessible and user-friendly solution, resulting in the creation of Doxa, a web-based, document-oriented database management system with integrated file handling.

Doxa's key features, including adaptability for users with varying technical expertise and a focus on efficient file handling, position it as a solution poised to redefine user experiences in database management. In essence, Doxa reflects a significant step forward in creating a more accessible and efficient database management system.

The following is a list of potential areas for improvement and future developments: Data backup and recovery to protect against data loss and enable document recovery, Document editing and viewing by adding built-in tools for editing and viewing documents, Offline mode so as to allow users to work without an internet connection, Advanced analytics by integrating tools for better data visualization and analysis.

## REFERENCES

- [1]. R. A. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 6th ed. Boston, MA: Addison-Wesley Longman Publishing Company, 2010.
- [2]. A. S. Susanto and M. Meiryani, "Database management system," Int. J. Sci. Technol. Res., vol. 8, no. 6, pp. 26–22, 2019.
- [3]. A. W. Watt and N. E. Eng, Database Design, 2nd ed. Victoria, B.C.: BCcampus, 2014. [Online]. Available: <https://opentextbc.ca/dbdesign01/>
- [4]. T. Taipalus, "Database management system performance comparisons: A systematic literature review," arXiv, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2301.01095>
- [5]. D. V. S. Vadivoo, S. Shanthini, A. Vinora, and G. M. Priya, "An overview of database management systems and their applications along with the queries for processing the system," SSRG Int. J. Comput. Sci. Eng., vol. 4, no. 3, pp. 1–4, 2017. [Online]. Available: <https://doi.org/10.14445/23488387/IJCSE-V4I3P101>
- [6]. W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments—A systematic literature review," Big Data Cogn. Comput., vol. 7, no. 2, p. 97, 2023. [Online]. Available: <https://doi.org/10.3390/bdcc7020097>
- [7]. R. Angles and C. Gutiérrez, "An introduction to graph data management," Graph Data Management, 2017.
- [8]. M. T. Özsu, P. I. Iglinski, D. S. Szafron, S. E. El-Medani, and M. J. Junghanns, "An object-oriented SGML/HYTIME compliant multimedia database management system," in Proc. 5th ACM Int. Conf. Multimedia (MULTIMEDIA '97), Seattle, WA, USA, Nov. 1997, pp. 239–249. [Online]. Available: <https://dl.acm.org/doi/10.1145/266180.266375>
- [9]. L. Q. Qian, K. L. LeFevre, and H. V. Jagadish, "CRIUS: User-friendly database design," Proc. VLDB Endow., vol. 4, no. 2, pp. 81–92, 2010.
- [10]. E. E. Egredzija and B. K. Kovacic, "Document management and exchange system - supporting education process," Int. J. Emerg. Technol. Learn. (iJET), vol. 5, no. SI2, pp. 46–49, 2010.