Edge Hill University

The Department of Computer Science

CIS4503 Databases

Coursework 1

2024/2025

Database Feasibility and Legal Compliance for Building a Robust Movie Search Engine

Student Name: Sunday Idika

Student ID: ****

Table of Contents

1. Introduction

1.1 Background Information

Project Overview:

The project at hand involves the development of an **online movie search engine** designed to provide users with an efficient way of searching and exploring movie metadata. The goal is to create a tool that offers detailed information about movies, such as titles, genres, budgets, directors, actors, and other associated data. To effectively manage and support the diverse and dynamic nature of this movie data, the system will require a robust and scalable **database solution**.

The platform will begin by handling **5,000 movie records**, but as the user base grows, it is expected to scale significantly, managing millions of movie entries. Thus, selecting an appropriate database system is crucial to ensuring fast, efficient query performance, data consistency, easy scalability, and a smooth user experience. Additionally, given the nature of the sensitive information involved, including potential personal data of users, the system must comply with **data protection regulations** to ensure privacy and security.

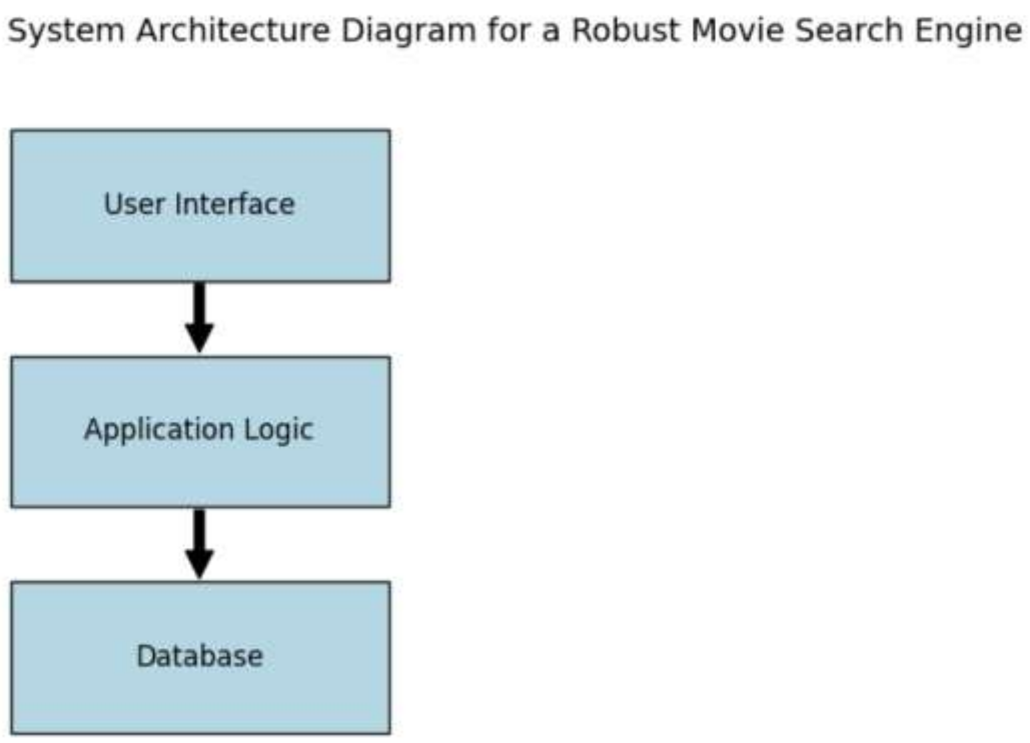
System Architecture Diagram for a Robust Movie Search Engine
User Interface
Application Logic
Database

Figure 1

Figure 1 shows the basic system architecture for the movie search engine. Cemil Abis and Murat Osman Unalir (2017) provided a detailed study stating the importance of enhancing the search capabilities of document management systems.

Explanation:

- **User Interface:** Users access the system through a browser or application interface.
- **Application Logic:** The backend processing layer handles user requests.
- **Database:** The database layer stores and retrieves data, interacting with the backend.

- **Flow of Data:** Arrows illustrate data flow between components (e.g., user queries -> application -> database).

1.2 Purpose and Objectives

Purpose of the Feasibility Study

- **Problem Statement**: The feasibility study seeks to evaluate and identify the most suitable **database type** to store and manage an online search engine's extensive and evolving movie metadata. The challenge is to choose between **relational databases** and **document-based databases**, assessing each for suitability in terms of scalability, flexibility, and compliance with data protection laws.

Objectives:

- Analyze the strengths and weaknesses of at least **two types of databases** (Relational and Document Databases).
- Evaluate how each database type aligns with key project requirements, such as **dynamic schema updates**, **scalability**, **availability**, and **data consistency**.
- Ensure each database meets **legal requirements** like compliance with **GDPR**, **CCPA**, and other data protection frameworks.
- Recommend a database type that best suits the project's goals, including a specific **database engine** (e.g., MySQL, MongoDB).

1.3 Scope

Database Types

This study will focus on **Relational Databases** (e.g., MySQLSQL) and **Document Databases** (e.g., MongoDB), analyzing their potential benefits and limitations.

- **Relational Databases** (RDBMS) are typically used for structured data with fixed schema constraints, where the relationships between data entities are well-defined.
- **Document Databases** (e.g., MongoDB) are highly flexible databases that store data as documents (e.g., JSON-like formats). These are ideal for handling dynamic, unstructured, or semi-structured data that may evolve over time.

Assumptions and Limitations

- The initial dataset will consist of **5,000 movie records**, with the potential to scale to millions of records in the future.
- The study assumes that both database types can be adequately scaled to handle large datasets, and that appropriate legal and security measures will be implemented.
- Performance benchmarks such as query speed and cost of scaling are important but will depend on specific configurations.
- The study will not consider **NoSQL** database options beyond document-based databases.

2. Literature Review

2.1 Overview of Database Types or Alternatives

Description of Options

- **Relational Database**: A **relational database** organizes data into tables with predefined schema, where relationships between data are established through keys (primary, foreign). Data is stored in rows and columns, and the database uses Structured Query Language (SQL) for data management (Do et al., 2022).
    - **Examples**: MySQL, PostgreSQL, Oracle DB.
    - **Use Cases**: Relational databases are ideal for applications requiring structured, consistent data management, such as **banking systems**,

enterprise resource planning (ERP) systems, and **inventory management** (Hosen, et al., 2024).

- **Document Database**: A **document database** is a type of NoSQL database where data is stored in flexible, schema-less documents, often in JSON or BSON formats. These databases provide the ability to handle semi-structured or unstructured data (MongoDB, 2024).
    - **Examples**: MongoDB, CouchDB, RavenDB.
    - **Use Cases**: Document databases are particularly effective for **content management systems**, **e-commerce platforms**, **real-time analytics**, and applications with dynamically changing datasets (Ngcobo et al., 2024).

Core Features

- **Relational Databases**:
    - **Structured schema**: Requires predefined tables with fixed columns (Martins et al., 2023).
    - **Strong consistency**: Relational databases use ACID properties to ensure transactions are fully completed or fully rolled back (Guo et al., 2022)).
    - **SQL querying**: Users query data using a structured language (SQL).
- **Document Databases**:
    - **Flexible schema**: Allows documents to have different structures, which is especially beneficial for applications with dynamic data (ALI et al., 2023)
    - **Scalable architecture**: Optimized for scaling out via horizontal distribution across multiple nodes (MongoDB, 2024).
    - **JSON-based queries**: Queries typically use a structure similar to the JSON format.

The summary of the database comparison is provided in Table 1.

2.2 Table 1: Strengths and Weaknesses

Feature	**Relational Database**	**Document Database**
Schema	Fixed; predefined table structure.	Flexible; schema-less; can store dynamic data structures.
Ease of Use	Requires design up front and strict data definitions.	Easy to change or evolve schema; however, more complex operations may require deep knowledge of the document structure.
Scalability	Vertical scaling (limits depend on hardware), difficult to horizontally scale.	Horizontal scaling; suitable for distributed systems.
Data Consistency	Strong consistency (ACID compliant)	Eventual consistency (BASE model); more tolerant to downtime.

Performance	Slower on large-scale operations or when data structure changes.	Optimized for high-performance read/write on large-scale, distributed data.
Maintenance	Mature tools, well-defined indexing and optimization strategies.	Complex replication management; suitable for agile development with frequent changes.
Query Language	SQL-based query language with complex querying capabilities.	Typically uses JSON-like queries, more suited to simpler queries.

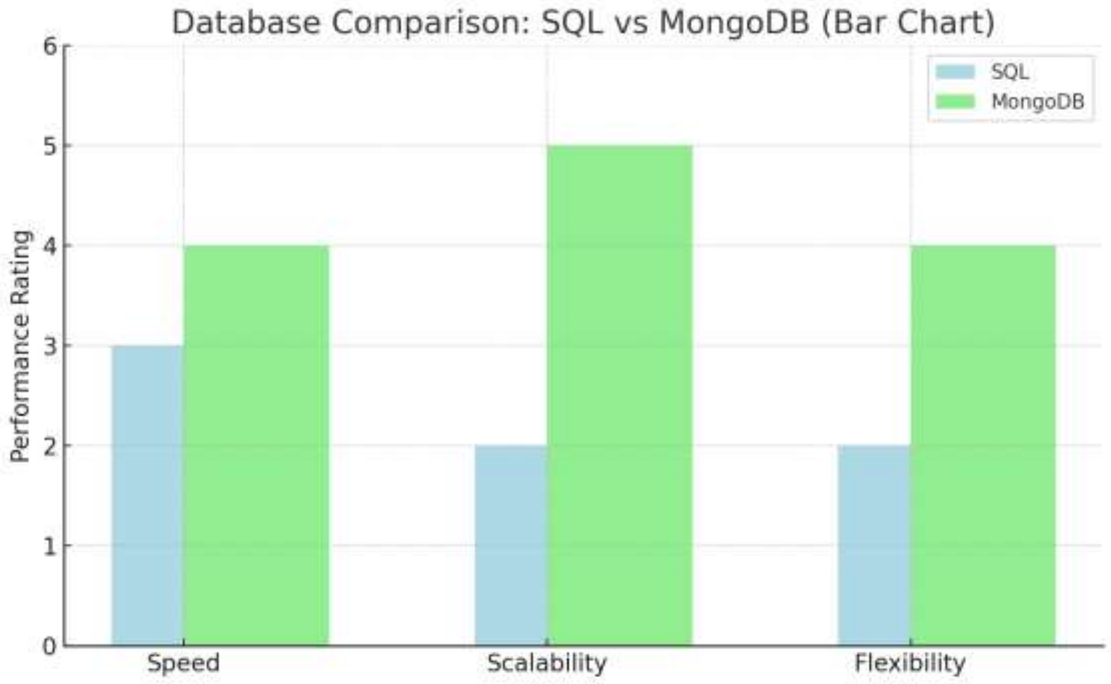

Figure 2

Figure 2 is a **bar chart** comparing SQL and MongoDB across the attributes of **Speed**, **Scalability**, and **Flexibility**:

- **SQL** performs moderately on speed but falls behind in scalability and flexibility.
- **MongoDB** demonstrates strong performance in scalability and flexibility, with slightly better speed.

2.3 Legal and Regulatory Considerations

Outline of Legal Requirements for Storing, Accessing, and Managing Data

Regulatory frameworks such as **GDPR (General Data Protection Regulation)**, **HIPAA (Health Insurance Portability and Accountability Act)**, and **CCPA (California Consumer Privacy Act)** dictate specific requirements on how data should be protected, stored, accessed, and processed (See Table 2). These include aspects such as data encryption, access control, reporting requirements, and data residency.

GDPR: It mandates explicit consent from data subjects to collect their personal data and requires the ability to delete this data when requested (right to erasure). It also requires businesses to maintain detailed access logs and auditing processes (Kamaruddin et al., 2023).

- **CCPA**: Similar to GDPR, it enforces consumer rights to access, delete, and opt out of personal data collection. It is primarily aimed at consumers in California, USA (Nortwick and Wilson, 2022).
- **HIPAA**: Primarily governs the healthcare sector, enforcing data privacy and security requirements for health information, with explicit requirements for data storage, encryption, and access controls (Syed and Faiza Kousar E S, 2023).

Table 2: Summary of the Legal and Regulatory Consideration for the Database

Requirement	**Relational Database**	**Document Database**
GDPR Compliance	Supports structured consent records but less flexible; manual handling may be required for consent withdrawal.	Flexible in accommodating user-specific data requests like consent withdrawal.
Data Encryption	Typically includes support for encryption (at rest, in transit).	Provides encryption support at both storage (at rest) and transit levels.
Data Residency	Stored in specific server locations; easy to manage under jurisdiction laws.	Distributed clusters complicate residency and may require configurations for compliance.
Access Control and Breach Reporting	Advanced RBAC support, logs, and audit trails are available to meet compliance standards.	Role-based access control and audit logging are available; robust replication and monitoring required for compliance.

2.4 Critical Analysis

- **Gaps or Limitations**:
  - Relational databases generally excel with structured, consistent data, but their ability to scale horizontally is limited, which may pose challenges for handling large, dynamic datasets (Ranjitha and Santhirakumar, 2022). Also, adapting to schema changes (e.g., adding new movie attributes) can be cumbersome.
  - Document databases, although highly flexible and scalable, face challenges in ensuring strict data consistency (since they follow the BASE model instead of ACID), which may be a risk in applications requiring full transactional consistency (Reddy, et al., 2022).
- **Alignment with Project Needs**:
  Based on prior research and the project requirements (handling millions of movie records with flexible, dynamic schema updates), **Document databases** appear more aligned with the needs of this online movie search engine. MongoDB, for example, provides high scalability and schema flexibility, making it ideal for an evolving dataset. Additionally, its strong support for data

consistency (even with eventual consistency) and scalability through horizontal sharding makes it suitable for handling large and distributed data (MongoDB, 2024).

- **Scalability, Schema Updates, and Legal Compliance**:
  Document databases like MongoDB align well with **scalability** and **schema updates** by allowing easy changes to data structure and distributing the data load across servers (MongoDB, 2024). However, careful configuration will be required to handle legal compliance, especially data residency and encryption standards, but MongoDB can be set up in compliant regions and supports necessary security features (encryption, role-based access controls).

In conclusion, while relational databases have their advantages in terms of data consistency and query capabilities, **document databases** (specifically MongoDB) provide the flexibility, scalability, and easier schema evolution necessary for this project to manage large volumes of dynamic movie metadata while ensuring compliance with relevant regulations.

3. Addressing the Requirements

This section critically evaluates how each examined database type—relational Databases (RDBMS) and Document Databases (specifically MongoDB)—addresses the key requirements outlined for the online movie search engine project.

3.1 Requirement 1: Schema Flexibility

- **Relational Databases**
  Relational databases are typically characterized by rigid, predefined schemas. Each table in a relational database needs to define columns in advance, and these columns cannot be altered without significant downtime and restructuring. While it is possible to add new columns, doing so can disrupt the existing data structure, requiring schema migrations and making it more cumbersome to introduce changes (Khan et al., 2023). This becomes increasingly problematic as new fields or data types need to be incorporated into the schema.

For this project, where the movie metadata may change (e.g., adding new fields for new movie attributes), the relational database’s static schema design would be a challenge to maintain flexibility over time. Hence, **Relational Databases do not fully satisfy the requirement of supporting dynamically updated schemas**.

- **Document Databases (MongoDB)**
  MongoDB, a document-oriented NoSQL database, offers a schema-less design, where each document (e.g., representing a movie) can have different structures. This makes it easy to add new fields or even completely new types of data without altering existing documents (MongoDB, 2024). New attributes such as new categories for movie genres or additional metadata can be added as the application evolves, supporting the project's need for flexibility without affecting the entire database structure.

Thus, **Document Databases like MongoDB excellently satisfy the requirement of dynamically updating the schema**.

3.2 Requirement 2: Scalability

- **Relational Databases**
  While relational databases are efficient for handling moderate amounts of structured data, they typically scale vertically (i.e., adding more powerful

hardware to a single machine) (Khan et al., 2023). This limits their scalability, especially when large-scale data growth is expected (i.e., an increase from 5,000 to millions of records). To scale horizontally across a distributed system, relational databases require complex sharding strategies, and even then, it remains challenging to maintain performance and consistency at large scales (Diaz Erazo et al., 2022).

Thus, **Relational Databases do not offer an optimal solution for horizontal scalability**, which is required for a project with expected large data growth and the need for distribution across multiple servers.

- **Document Databases (MongoDB)**
  MongoDB, by design, is optimized for horizontal scaling. It supports **sharding**, which allows data to be distributed across multiple machines and thus enables the database to scale horizontally. Each "shard" in a MongoDB cluster is responsible for a subset of data, and as the dataset grows from 5,000 to millions of records, MongoDB can continue to scale efficiently by adding new nodes to the cluster without significant reconfiguration or performance loss (MongoDB, 2024). This makes MongoDB highly suitable for handling massive datasets and for scaling out across large computer clusters.

Hence, **MongoDB perfectly satisfies the scalability requirement**, as it can support large-scale collections through horizontal distribution.

3.3 Requirement 3: Data Consistency

- **Relational Databases**
  Relational databases follow the **ACID** (Atomicity, Consistency, Isolation, Durability) model, ensuring **strong consistency**. This model guarantees that transactions are processed in a way that leaves the database in a consistent state, with no partial or conflicting updates (Elmasri & Navathe, 2015). This strong consistency makes relational databases a good fit for applications where data integrity and correctness are critical (e.g., banking, financial systems).

For the movie search engine project, **relational databases satisfy the need for strong data consistency**. The use of transactions and ACID guarantees ensures that all systems in the cluster will maintain a consistent view of movie data.

- **Document Databases (MongoDB)**
  MongoDB uses the **BASE** (Basically Available, Soft state, Eventual consistency) model rather than ACID, meaning that it emphasizes high availability and partition tolerance, and the system may temporarily allow inconsistent states while asynchronously converging towards consistency (Cabral et al., 2023) In MongoDB, each read operation might not necessarily reflect the most up-to-date state of data (especially in sharded setups), which could be a risk if strict consistency is required at all times.

Although MongoDB **supports configurable consistency guarantees** for specific operations (such as read and write concerns), it typically provides **eventual consistency**, which might be a limitation for the project where real-time data consistency across the system is crucial. If strict consistency is needed, additional mechanisms and configurations such as **multi-document transactions** can be leveraged (Khan et al., 2023), but they may have some performance overhead.
In conclusion, **Relational Databases fully satisfy the need for strong consistency**, whereas **MongoDB provides eventual consistency by default**, but can achieve stronger consistency with configuration changes.

3.4 Requirement 4: High Availability

- **Relational Databases**
  While high availability can be achieved in relational databases through replication and clustering (e.g., PostgreSQL with synchronous replication, MySQL with master-slave replication), it usually requires complex configurations and management (Do et al., 2022). For this specific project, where major updates will happen rarely and the demand for **continuous availability** is not extreme, relational databases are capable of achieving high availability as long as proper disaster recovery and backup strategies are in place.

Therefore, **Relational Databases can support high availability but require more complex setup and maintenance compared to document databases**.

- **Document Databases (MongoDB)**
  MongoDB is designed for **high availability** from the ground up. It uses **replica sets**, where data is automatically replicated across multiple servers. If one node goes down, another replica can take over, ensuring minimal service disruption (Carvalho, Sá and Bernardino, 2023). MongoDB's high availability features are inherent to its design, and it simplifies disaster recovery by replicating data across nodes with automatic failover.

Since high availability is a desired feature for this project, MongoDB **supports it excellently** with built-in high availability features that are easy to implement.

4. Recommendation

4.1 Restate the Objective

The purpose of this study was to evaluate different database types for managing movie metadata in an online search engine and recommend the most suitable solution based on the project's requirements, including schema flexibility, scalability, data consistency, availability, and legal compliance.

4.2 Summarise Key Findings

- **Relational Database:**
  - **Strengths**: Provides strong data consistency with ACID compliance, making it ideal for structured data and transactional consistency. It excels in handling complex queries using SQL.
  - **Weaknesses**: Lacks schema flexibility, making it challenging to modify the database structure as the project evolves. Vertical scaling, limited to hardware upgrades, is often less efficient than horizontal scaling for large datasets.
- **Document Database (MongoDB):**
  - **Strengths**: MongoDB offers flexible, schema-less data storage, allowing easy updates and the ability to handle dynamic data. It is horizontally scalable, efficiently managing growth by distributing the data across servers. High availability features, like replica sets, add reliability.
  - **Weaknesses**: MongoDB uses eventual consistency by default, which can complicate scenarios requiring immediate consistency. However, tunable consistency options can be adjusted for specific use cases.

4.3 Recommendation of Database Type

A **Document Database (MongoDB)** is recommended for this project, as it best meets the project's requirements:

- **Schema Flexibility**: MongoDB's schema-less design is ideal for the evolving structure of movie metadata, enabling easy addition of fields without requiring database downtime.
- **Scalability**: MongoDB is designed for horizontal scaling through sharding, allowing it to grow with the dataset. As the database transitions from 5,000 to millions of records, MongoDB can scale efficiently across distributed systems.
- **Data Consistency**: MongoDB offers tunable consistency models, providing eventual consistency by default, but with options to configure stronger consistency for critical operations.
- **High Availability**: MongoDB's replica sets automatically handle failover, ensuring the database remains available even in the event of system failures, aligning with the project's need for reliability during major updates.

4.4 Recommendation of Database Engine

MongoDB is the recommended database engine due to:

- **Performance**: It is optimized for both read and write-heavy applications, ensuring quick queries and data insertion, crucial for an online search engine with frequent updates.
- **Industry Adoption**: MongoDB is widely adopted in various industries, offering substantial community support and mature ecosystem tools.
- **Feature Alignment**: With built-in sharding, replication, and flexible schema, MongoDB aligns perfectly with the scalability and flexibility needs of this project.

4.5 Legal Implications

MongoDB supports legal compliance through:

- **Encryption**: MongoDB offers encryption at rest and in transit to ensure sensitive data protection.
- **Access Control**: Its role-based access control (RBAC) secures data access, adhering to legal frameworks like GDPR.
- **Audit Logging**: MongoDB enables audit logging to track data changes, supporting compliance and traceability.

4.6 Expected Benefits

Using MongoDB enhances:

- **Scalability**: It can handle millions of records and scale seamlessly as the database grows.
- **User Experience**: Faster data retrieval and updates improve overall user experience.
- **Cost-Effectiveness**: Horizontal scaling makes MongoDB a cost-effective, long-term solution.

4.7 Conclusion

MongoDB is the optimal choice for the project, providing scalability, flexibility, and compliance with legal requirements. Its adoption ensures a reliable, efficient, and future-proof movie search engine.

5 Reference

1. ALI, A., NAEEM, S., ANAM, S., and AHMED, M., 2023. A State of Art Survey for Big Data Processing and NoSQL Database Architecture. *International Journal of Computing and Digital Systems*. 14 (1), pp. 297–309.
2. ANDERSON, B. and NICHOLSON, B., 2024. SQL vs NoSQL. *Ibm.com* [online]. Available from: https://www.ibm.com/think/topics/sql-vs-nosql? [Accessed 21 Dec 2024].
3. CABRAL, J.V.L., NOGUERA, V.E.R., CIFERRI, R.R., and LUCRÉDIO, D., 2023. Enabling schema-independent data retrieval queries in MongoDB. *Information Systems*. 114, p. 102165.
4. CARVALHO, I., SÁ, F., and BERNARDINO, J., 2023. Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB. *Algorithms*. 16 (2), p. 78.
5. DIAZ ERAZO, A.D., RAUL MORALES MORALES, M., PINEDA CHAVEZ, V.K., and LEONARDO MORALES CARDOSO, S., 2022. Comparative Analysis of performance for SQL and NoSQL Databases. *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*.
6. DO, T.-T.-T., MAI-HOANG, T.-B., NGUYEN, V.-Q., and HUYNH, Q.-T., 2022. Query-based Performance Comparison of Graph Database and Relational Database. *The 11th International Symposium on Information and Communication Technology*.
7. GUO, R., LUAN, X., XIANG, L., YAN, X., YI, X., LUO, J., CHENG, Q., XU, W., LUO, J., LIU, F., CAO, Z., QIAO, Y., WANG, T., TANG, B., and XIE, C., 2022. Manu: A Cloud Native Vector Database Management System. *arXiv:2206.13843 [cs]* [online]. Available from: https://arxiv.org/abs/2206.13843.
8. HOSEN, M.S., ISLAM, R., Naeem, FOLORUNSO, E.O., CHU, T.S., AL MAMUN, M.A., and ORUNBON, N.O., 2024. Data-Driven Decision Making: Advanced Database Systems for Business Intelligence. *Nanotechnology Perceptions*. 20 (3), pp. 687–704.
9. KAMARUDDIN, S., MOHAMMAD, A.M., SAUFI, N.N.M., ROSLI, W.R.W., OTHMAN, M.B., and HAMIN, Z., 2023. Compliance to GDPR Data Protection and Privacy in Artificial Intelligence Technology: Legal and Ethical Ramifications in Malaysia. *IEEE Xplore* [online]. Available from: https://ieeexplore.ieee.org/abstract/document/10150615?casa_token=pKKunH042SUAAAAA:RzYPYZLdHWI-di3PZ95UNr-ENGHg7dbyWxfGmA7mDQkGM_tpRYkN4H8E8W7RA8WpeuYdr5wMWg [Accessed 6 Sep 2023].
10. KHAN, M.Z., ZAMAN, F.U., ADNAN, M., IMROZ, A., RAUF, M.A., and PHUL, Z., 2023. Comparative Case Study: An Evaluation of Performance Computation between SQL and NoSQL Database. *Journal of Software Engineering* [online]. 1 (2), pp. 14–23. Available from: http://sjhse.smiu.edu.pk/sjhse/index.php/SJHSE/article/view/42.
11. MARTINS, P., ALTIGRAN DA SILVA, AFONSO, A., CAVALCANTI, J., and EDLENO DE MOURA, 2023. Supporting Schema References in Keyword Queries Over Relational Databases. *IEEE Access*. 11, pp. 92365–92390.
12. MONGODB, 2024. What Is NoSQL? NoSQL Databases Explained. *MongoDB* [online]. Available from: https://www.mongodb.com/resources/basics/databases/nosql-explained.

13. NGCOBO, K., BHENGU, S., MUDAU, A., THANGO, B., and LERATO, M., 2024. Enterprise Data Management: Types, Sources, and Real-Time Applications to Enhance Business Performance - A Systematic Review. *Enterprise Data Management: Types, Sources, and Real-Time Applications to Enhance Business Performance - A Systematic Review*.
14. NORTWICK, M.V. and WILSON, C., 2022. Setting the Bar Low: Are Websites Complying With the Minimum Requirements of the CCPA? *Proceedings on Privacy Enhancing Technologies* [online]. Available from: https://petsymposium.org/popets/2022/popets-2022-0030.php [Accessed 27 Dec 2024].
15. RANJITHA, P. and SANTHIRAKUMAR, S., 2022. The impact of covid - 19 on households' gem mining industry: a study based on Pelmadulla divisional secretariat, Ratnapura district. *Seu.ac.lk* [online]. Available from: http://ir.lib.seu.ac.lk/handle/123456789/6892.
16. REDDY, REDDY, JONNALAGADDA, R., SINGH, P., and GOGINENI, A., 2022. Analysis of the Unexplored Security Issues Common to All Types of NoSQL Databases - Academic Digital Library. *Article4sub.com* [online]. 14 (1). Available from: http://publications.article4sub.com/id/eprint/285/.
17. SYED, F.M. and FAIZA KOUSAR E S, 2023. Leveraging AI for HIPAA-Compliant Cloud Security in Healthcare. *Revista de Inteligencia Artificial en Medicina* [online]. 14 (1), pp. 461–484. Available from: http://redcrevistas.com/index.php/Revista/article/view/146.
18. CEMIL ABIS and MURAT OSMAN UNALIR, 2017. A metamodel-based search engine for document management systems. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP) [online]. pp. 1–8. Available from: https://ieeexplore.ieee.org/abstract/document/8090177?casa_token=skl69dbGll4AAAAA:lQathdoBhfFuIOcD4W9vl0coB50FHUIJ4PD23tduT19dh3leGy2fzXd4gboQhi3viV4db4CWDrMg [Accessed 11 Jan 2025].