ISSN 2581 – 3781

AJCSE
ASIAN JOURNAL OF COMPUTER SCIENCE ENGINEERING

Available Online at www.ajcse.info
Asian Journal of Computer Science Engineering 2022;7(4):1-9

REVIEW ARTICLE

A Deep Dive into Effective Database Migration Approaches for Transitioning Legacy Systems in Advanced Applications

Dhruv Patel*, Ritesh Tandon

Independent Researcher

Received on: 15-03-2022; Revised on: 01-05-2022; Accepted on: 20-05-2022

ABSTRACT

The problem with legacy systems in the age of digital transformation is that they are most of their enterprises and public sector services passing pipes, and the technology that serves as their back bone is most of the time too old and outdated to grow, as all of us know that most of their enterprises are growing, not only they but also so are the public sector services. This paper presents the urgent demand for updating these systems, advocating for moving relational databases to modern platforms. The characteristics of legacy systems are addressed, what motivates such transformation, several reengineering strategies such as rehosting, replacing, mitigation, and retargeting. The data migration strategy along with best practices for a successful execution is provided along with a complete overview of database migration processes (including schema translation and data transformation). The paper also brings up that advanced technologies such as the advanced message queuing protocol and DevOps practices play a positive part in smoothing the migration process. It analyzes the various migration approaches, that is, rehosting, refactoring, rebuilding, replacing and their cost, risk, alignment to organizational goals, and so forth. Through this study, it strives to provide the companies with the strategic framework for their realization of legacy infrastructure replacement, which should be of scalable, secured, and ready for future technology advancements systems.

Key words: Cloud adoption, Database migration, Legacy systems, Modernization strategies, Reengineering, Server migration

INTRODUCTION

Legacy systems are important, but fundamental systems, and these are not easy to keep up with the needs of current applications. With the rise of cloud-based and moving to completely distributed and artificial intelligence (AI)-powered platform, there is no room for less efficient database migration strategy.[1,2] Legacy databases, typically rigid, monolithic, and resource-intensive, pose significant challenges in terms of performance, integration, and maintainability.[3] Migrating these systems to modern environments enables organizations to harness enhanced scalability, improved security, real-time analytics, and operational efficiency.

Moving data from the source relational database (RDB) to the target RDB, including data transformation and schema translation, is known as RDB migration. Legacy migration has been the foundation of several solutions since 1990. Many software companies developed their own migration process solutions for their key products as information technology developed.[4] The main reason for the migration is to turn the current system into a developed system that meets the needs of the business. The next problem is redefining the current storage and database system in terms of hard-to-understand code.[5,6] There is a rule in the business that during the migration, the source and target databases will often have different structures or data will not match up across multiple data sources. Due to this issue, many studies and the creation of migration tools have continuously come up with a full answer for the data migration methodology for migration projects. The whole process of moving a database is broken up into several steps that are done one at a time. With this method, the database migration takes into account the number of rows, columns, and other information from the source database.

Address for correspondence:
Dhruv Patel
E-mail: dp270894@gmail.com

Legacy tools are also used to help the government provide services to the people. The methods that have changed over time are still used because they are good for business in the public sector.[7] It keeps years of data that are needed for daily operations and important tasks for public administration. But because technology changes so quickly, it's harder for the government to use these tools. People in the public sector now want to access information across organizations and countries, but old methods cannot keep up. People have pointed out that these systems make it harder for the government to come up with new ideas that are needed to keep up with changes in technology around the world.[8] As a result, these systems need to be improved so they can keep helping the public sector provide services.

The old programs were created using technologies such as mainframes, SAP, and others, and they still do important work for a business.[9] There have been big changes in technology lately, so old systems need to be replaced with new ones so that business applications can be made.[10] On the other hand, there is operational risk that could hurt the whole system if it is not handled well.

Structure of the Paper

The structure of this paper is as follows: Section II, understanding legacy systems and modernizing. Section III discusses the need for database migration, highlighting the benefits of transitioning from legacy databases to modern systems. Section IV presents the different strategies and tools available for migrating legacy systems. Section V reviews literature and case studies. Section VI: Conclusions with findings and future research directions.

UNDERSTANDING LEGACY SYSTEMS AND MODERNIZING

Legacy systems are outdated technologies that still perform critical business functions but are built on obsolete platforms. They often come with issues such as high maintenance costs, limited scalability, and security risks. Despite this, organizations continue to use them because they hold valuable data and support key operations. Modernizing these systems means upgrading, transforming, or replacing them to align with current business needs. This improves performance, enhances security, and allows integration with modern tools. The push for modernization is driven by digital transformation, cloud adoption, customer expectations, and regulatory compliance. Choosing the right approach, whether rehosting, refactoring, rebuilding, or replacing, depends on the company's goals and resources. Effective modernization helps that businesses become more agile, efficient, and future-ready while reducing operational risks.[11]

Definition and Characteristics of Legacy Systems

A legacy system is an old or out-of-date piece of technology, software, or hardware that an organization still uses because it still does what it was made to do.[12] Systems that are too old are usually not supported or managed anymore, and they cannot be improved much.

- A legacy system is outdated technology or software still in use because it fulfills its intended function
- These systems are often difficult to replace due to their critical role in operations, lack of documentation, and intertwined dependencies
- Over time, multiple changes by different personnel make system comprehension harder
- IT managers must evaluate which legacy systems are essential and how much maintenance or replacement they require to reduce operational risks.

Reengineering Strategies for Upgrading Legacy Systems

Using different reengineering processes to replace and update old systems with new software-based solutions.[13]

1. Re-hosting
   This is called "re-hosting," and it means using old software on a new server without making any changes. This will lower the cost of keeping old gear running.
2. Re-placing
   Once an existing system cannot meet all of an organization's needs, replacement methods are used. Rewriting old software or adding new features to it is how replacement is done in software legacy systems. This program has been used before and works well.

3. Mitigation
   New versions of software will have new features added to them to fix bugs in older versions. This method is called reduction. This is a way to move old systems to environments that are more flexible.[14]
4. Re-targeting
   The legacy system is being changed into a new system with some extra features and functions. Making a new hardware platform out of a system platform.

Transition from Legacy System to New System

Modernizing legacy systems is a strategic necessity for businesses striving to remain competitive, scalable, and secure in a technology-driven world. The process involves migrating to a new system that overcomes the shortcomings of legacy infrastructure while ensuring minimal disruption to existing business processes.
Modernizing a legacy system means making old software and programs work with today's technology and business needs, as shown in Figure 1. This does not always mean that the running systems or apps will change. Most of the time, old hardware is what gets in the way, which makes it easy for business-critical apps to fail.

- Modernization improves these systems by moving old ones to new platforms. Its benefits include streamlining IT processes, lowering maintenance costs, and improving performance.[15]
- These changes also improve the systems' ability to work with new technologies. Modernizing businesses prepares them to use new technologies such as AI, big data, and cloud computing.

Need for Modernization

This is one of the most common ways to update an app, and it's also the simplest way to make sure that the product will keep working for years to come. It requires moving the system (usually by re-hosting it using cloud solutions) and making a few small improvements. This includes making changes to the user interface (UI) and user experience (UX), improving speed, and moving the database. However, this method has some problems. The main business logic and design do not change much because these kinds of changes need a more invasive method. If the product's technology stack is pretty new and does not pose a threat to its future growth, modernization may only require a few small fixes or improvements. This could mean optimizing the design or refactoring the code.[16] No major changes were made to the product's business logic or UX when it was updated or its speed was improved. New features can be added to an existing product as soon as it is suitable. These could be modules built just for it or connections from a third party.

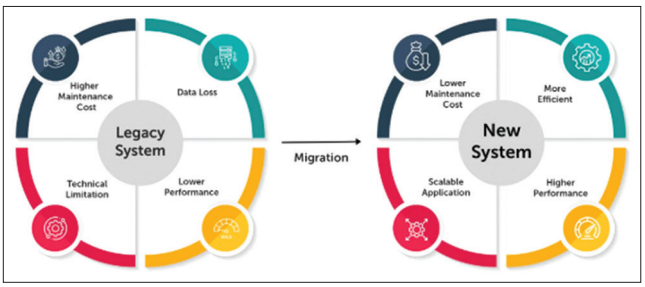

Figure 1: Migration from legacy system to new system

DATABASE MIGRATION: AN OVERVIEW

Moving a current database application to a different database management system (DBMS) or service provider is called database migration. Existing data are exported from the current DBMS and added to the new DBMS during this process. Although the migrated database will probably have the same data, the way it works with it will probably be different,[17] This is especially true when moving from very old systems or systems built on different technologies, or when the new database was not designed with the users in mind.[18]

Data Migration Strategy

A clear plan for moving data from one system to another should include dealing with legacy data, finding source data, and working with targets that are always changing, making sure data quality standards are met, coming up with the right project methods, and learning general migration skills.[19]
The primary factors and inputs for creating a data migration strategy are as follows:

- Plan for making sure that the migrated data are correct and full after migration
- Iteratively moving a logical group of data is possible with agile concepts
- Plans to deal with the problems that come with the quality of the source data right now as well

as the quality standards that the target systems expect[6]
- Plan and set up a migration system with the right checkpoints, controls, and audits so that errors and broken accounts can be found, reported, fixed, and closed
- A way to make sure that the right things are matched up at different stages to make sure that the migration is finished
- A way to pick the best tools and platforms for the tricky nature of migration.[20]

Best Practices for Successful Migration

A good move from older databases to Microsoft SQL server using SQL server migration assistant (SSMA) is more complicated than just following the steps of the move. There must be best practices that are followed to ensure efficiency, lower risks, and increased profits. Consider these important best practices as it moves.[21]

- Comprehensive pre-migration assessment: Before starting the transfer, it should carefully examine the current database setup. This includes things like knowing what the current database layout is, what other parts depend on it, and how to avoid potential problems
- Start with a pilot project: Before committing to a full-scale project, this might want to start with a trial migration to lower the risks. Choose a database that is not very important or a smaller amount of data to move first[22]
- Optimize the target environment: Before it moves the data, ensure that the target SQL server environment is optimized for speed and scalability
- Focus on data integrity: Data integrity is very important during any transfer. Use the data validation tools in SSMA to make sure that all the data are moved correctly and that there are no problems between the old system and SQL Server.

Advanced Message Queuing Protocol (AMQP)

The publish-subscribe paradigm, as described by OASIS, is what AMQP is based on. It is an open standard protocol that lets a lot of different applications and systems work together, even if they are not built the same way. It was first created for business messaging with the goal of providing a non-proprietary solution that could handle a lot of message exchanges that could happen quickly in a system.[23]

- This AMQP interoperability function is important because it lets platforms that are written in different languages send and receive messages. This can be especially helpful in heterogeneous systems. The two versions of AMQP, AMQP 0.9.1 and AMQP 1.0, are very different from each other and use very different ways to send messages.[24]
- The publish-subscribe paradigm is used by AMQP 0.9.1. It is based on the exchanges and the message queues, which are two important parts of an AMQP broker. The exchanges are a part of the broker that guides the messages that come in from writers.[25]

MIGRATION APPROACHES, CHARACTERISTICS, AND STRATEGIES

Methods to move old systems to newer platforms, like the cloud or newer architectures. The three approaches of rehosting, refactoring, rebuilding, or replacing have their own trade-off definition in terms of complexity, costs, risk, and time.[26] The rehosting is quick and the risk is minimal, whereas rebuilding or replacing brings more long-term benefits but demands more investment. That comes down to factors like system performance, scalability, and budget, as well as business needs, as to what strategy to choose. A good migration strategy causes little perturbation, is flexible, and plays to companies' digital transformation requirements by being synchronized to modern business objectives.[27] There are different migration techniques that use different.

Characteristics of Migration Approaches

There are a large number of critical characteristics that govern the selection of a suitable approach to migrating legacy systems and they vary based on the approach. The characteristics of these qualities assist in the organization's ability to manage cost, risk, performance, and business continuity against during the transformation process.

- Complexity: Depending on the chosen approach, migration is a complex process. Rehosting requires fewer changes as it is just changing the host, whereas rebuilding requires

more changes as it involves redesign and development[2]

- Cost: The method will vary the cost. However, rebuilds or replacements are very expensive due to the requirement of new development and testing and rehosting is cheaper
- Risk level: The chances of risk are different for rehosting carries low risk since it preserves the system as it is, but rewriting exposes to risk on codes, which may involve mistakes[28]
- Time to implement: Rehosting and encapsulation of the request are faster, execution wise. Due to redevelopment needs, rebuilding or replacing a system takes more time
- Scalability: Modernized systems are more scalable. System approaches involving code or architecture upgrades improve the system's ability to handle increasing demand[29]
- Performance improvement: Some methods do not boost performance immediately. Rehosting will have no impact on performance, but refactoring and refactoring tends to improve performance
- Business continuity: Approaches such as rehosting and encapsulation ensure minimal disruption. Full replacement may temporarily affect ongoing business operations
- Integration capability: Modern systems integrate easily with application program interface (APIs) and cloud tools. Legacy systems often need additional tools or custom connectors to work with new technologies[30]
- Security enhancements: Migration allows implementation of modern security standards. Older systems are more prone to vulnerabilities and lack updated protections.

Legacy System Modernization Approaches and Strategy

Modernizing old systems helps businesses stay competitive, efficient, and ready for the future. Let's look at the best ways to update old systems that people who make decisions should think about.

Rehosting or lift and shift

Rehosting, also known as "Lift and Shift," is the process of moving parts of a program to a different infrastructure (like the cloud or on-premises) without changing the code or features. A new system is only put in place for the hardware platform below.[31] Applications from the past will still work the same way they do now. People often use this method to quickly move old apps to the cloud and take advantage of its benefits without having to change the code.

Refactoring or re-architecting

Moving and improving the current code without changing how it works is a traditional way to update an old system. It is easier to adapt to new situations when refactoring a system. These steps should be used by companies moving from containers to microservices. Things change in big and small ways. However, they all make the system stronger.[32]

Rebuilding or rewriting from scratch

The process of rewriting means making a new application from scratch while keeping the old system's needs and functions. Due to this, current frameworks, codes, and tools can be used.[33]

Encapsulation

Encapsulation lets it reuse key system parts while getting rid of old code. These parts are linked to new access levels through APIs.[34] This way of updating old software gives current parts a new look and feel using the app's features. Some parts may need more work than others, so it is important to plan ahead. Encapsulation works well if all these want to do is change the interface.

API

API modernization makes it possible for old systems to connect to and use new services and apps. It adds to the features of an old system without making big changes to the code. The process is not always easy. There are easy and complex interfaces.[35]

Cloud Migration

Apps and data are moved to cloud technology when old systems are moved to the cloud. This cuts costs and makes it easier to grow and change. It often makes the machine run faster. Depending on what the system needs, the change can happen slowly or quickly.[36]

Service-oriented architecture (SOA)

SOA breaks down old systems into smaller services that can be used again and again. These can be built, put into use, and kept up to date individually. SOA makes systems more adaptable and scalable. It makes it easier to connect to other services. How hard it is to implement depends on the size and layout of the system.

Buying a new application

Finally, getting a new application is another good way for businesses to update an old system. They can replace the old system with the new application. In this method, it picks a solution and replaces it with the current program.

LITERATURE REVIEW

This section discusses the previous research on Database Migration Approaches for Transitioning Legacy Systems. Table 1 provides a summary of key studies on database migration approaches for legacy systems, highlighting methodologies, findings, and associated challenges. It offers insights into various frameworks, strategies, and practical applications across domains.

Althani and Khaddaj suggested that quality be built into the transfer process, which would have a big effect on risk and cost. This paper does a systematic review of different ways to update old systems, ranging from easy wrapping to full migration. When choosing a moderation approach to meet the needs of the migration, the quality aspects of the process need to be taken into account. To cut costs and make them more flexible, they need to be updated and moved to new technological settings.[37]

Wijaya and Akhmadarman to help with the data migration process, the suggested framework includes algorithm migration, model migration, and migration schemes. The proposed framework can move data in the real operating world after it has been tested and evaluated. To fix this problem,

Table 1: Literature review on database migration approaches for transitioning legacy systems in advanced applications

References	Study on	Approach	Key findings	Challenges and limitations	Future work
Althani and Khaddaj (2017)[37]	Quality integration in legacy system migration	Systematic review of modernization strategies (wrapping to full migration)	Emphasizes considering quality aspects in selecting a migration strategy to reduce cost and risk	Quality factors are often overlooked; they require a tailored strategy for each case	Develop adaptive frameworks that incorporate quality metrics dynamically into migration decision-making
Wijaya and Akhmadarman (2018)[38]	Development of a general data migration framework	Framework with algorithm migration, model migration, and migration schemes	Provides complete stages of data migration; validated in operational environment	Existing frameworks lack completeness; general frameworks still need real-world robustness	Extend framework testing across varied industry scenarios to enhance robustness and completeness
M'Baya *et al.* (2017)[39]	Assessment of legacy systems using quality metrics	Legacy System Assessment Conceptual Framework with tools	Helps maintainers choose modernization strategies using quality indicators	Full project lifecycle must be addressed; integration of the toolkit into the process can be complex	Streamline toolkit integration into standard software engineering workflows and lifecycle models
Khan *et al*. (2020)[40]	Secure cloud migration of industrial control systems	Minimal interruption, cloud-based migration framework	Suitable for real-time systems; ensures safety and continuity	Needs further validation in diverse industrial environments	Conduct real-world testing across various critical infrastructure domains for broader applicability.
Preti *et al*. (2021)[41]	Gradual migration of public safety legacy systems to microservices	Database sharing strategies during monolith-to-microservice transition	Demonstrates co-existence and gradual migration benefits	Complexity in managing shared databases during transition	Investigate advanced data synchronization and management techniques during transitional phases.
Schnappinger and Streit (2021)[42]	Cost-effective modernization with a limited budget	Custom transportation and alternative strategy under constraints	Enables legacy language migration without full re-engineering	Strategy selection is still a challenge due to budget, technical, and business constraints	Explore intelligent decision support systems for strategy recommendation under multi-constraint environments.
Martens *et al*. (2018)[43]	Scalable data decomposition for large-scale healthcare migration	Splitting monolithic data into independent tranches	Efficient for large-scale systems; applied in the healthcare sector migration	Technical and cost constraints in decomposing and migrating huge datasets	Research automation tools and heuristics for scalable and cost-effective data tranche decomposition.

they need a general migration system that covers all the steps of moving data. To help with moving data, many systems have been made. By providing a general data migration framework, the study results will help companies or developers who need help moving data.[38]

M'Baya *et al*. based on quality measure, a legacy system assessment conceptual framework (LSACF). To assist maintainers in the evolution process, LSACF provides a plan that includes a methodological approach and a functional tool. Another common problem for large businesses is updating old systems. Businesses need to update old systems and perform proper modernization because technology is changing so quickly. To adequately choose a modernization strategy and create an effective evolutionary system, it is necessary to look at the whole modernization project.[39]

Khan *et al*. offer a way for old industrial control systems to be moved to the cloud in a way that is both smooth and safe. It checks to see if the cloud can handle the real-time needs of control operations without putting system safety at risk. The suggested method is meant to keep industrial processes running as smoothly as possible during the cloud migration process, and it provides a general framework that can be used in various industrial sectors. Experiments with the cloud migration method look good for systems that need to work quickly, like synchrophasor technology.[40]

Preti *et al*. describe the transition plan that the Public Safety Secretariat of Mato Grosso is using to change its old, single-piece systems to a microservices-based design. Even though there are already standards for microservice projects, it is hard to find reports on how to successfully move from a monolithic architecture to a microservice design over time. This is especially true when it comes to splitting and separating legacy databases. Their findings when they used the database sharing techniques and migration process outlined in this paper during a time when monoliths and microservices lived together.[41]

Schnappinger and Streit to make things clearer explain why the current transfer plans did not work and suggest a different approach, keeping in mind the limited moving funds. The old code is instantly translated to a different programming language by custom translation. The economy depends on old software systems, but they are known to be expensive to run and keep up to date. Modern technology or languages are often used to move these kinds of systems in order to cut down on costs. Many transfer plans exist, but it is still hard to pick the best one or set of plans when there are technical, economic, and business issues to consider.[42]

Martens *et al*. show a way to decompose data in which the whole volume of data in a single business IT application is split into separate data migration chunks. The method for moving data explained here is being used in one of the biggest healthcare data migration projects in Europe right now. It involves millions of customer records. New enterprise IT applications must finally take the place of old ones, both because they are more cost-effective and better at what they do. Moving data is an unavoidable part of making this move.[43]

CONCLUSION AND FUTURE WORK

Modernizing legacy systems and executing robust database migration strategies have become critical imperatives in today's digital-first enterprise landscape. As demonstrated, legacy systems, despite their foundational roles, are often barriers to innovation due to their rigid architectures, outdated technologies, and high maintenance costs. Migration of RDBs is a complex and multi-phased process. However, it ensures that organizations fall in line with modern IT infrastructure, including the cloud native and smart one. Rehosting, replacing, mitigation, and re-targeting are practical ways of system modernization under reengineering approaches. Thorough pre-migration assessment, pilot testing, and ensuring data integrity tools such as SSMA all lower the risk and increase the success rate of migrations. As an extension to it, the integration of interoperable messaging protocols like AMQP makes the communication seamless after migration across diverse systems; thus, the extent of modernization is expanding. Legacy modernization as well as database migration not only aid in operational efficiency and scalability but also play an instrumental role in establishing a strong base for the digital transformation. An approach to move from obsolete systems to future-ready platforms with minimal disruption can be done by looking back with structured methodologies and best practices.

The understanding and analysis of legacy systems could be automated so that migration

projects would consume less time and money in future research on database migration. Finally, still further automating the migration process is the development of intelligent tools that are based on AI and machine learning for schema translation, data transformation, and anomaly detection. Moreover, database environments in modern times are becoming increasingly complex and comprise several other factors which should be taken into account during a database migration. For example, data security during a migration needs to be robust, and interoperability between heterogeneous systems also has to be enhanced, with database migration being cloud-based. Organizations will benefit from further advancements that make the process of migration more efficient and resilient, perfectly adapting to the needs of those organizations.

REFERENCES

1. Goerzig D, Bauernhansl T. Enterprise architectures for the digital transformation in small and medium-sized enterprises. Procedia Cirp 2018;67:540-5.
2. Abhishek, Khare P. Cloud security challenges: Implementing best practices for secure saas application development. Int J Curr Eng Technol 2021;11:669-76.
3. Flynn T, Triantafilis J, Rozanov A, Ellis F, Lázaro-López A, Watson A, *et al*. Numerical soil horizon classification from South Africa's legacy database. Catena 2021;206:105543.
4. Elamparithi M, Anuratha V. A review on database migration strategies, techniques and tools. World J Comput Appl Technol 2015;3:41-8.
5. Anju, Hazarika AV. Extreme gradient boosting using squared logistics loss function. Int J Sci Dev Res 2017;2:54-61.
6. Thokala VS. A comparative study of data integrity and redundancy in distributed databases for web applications. Int J Res Anal Rev 2021;8:383-9.
7. Bakar HA, Razali R, Jambari DI. Legacy systems modernisation for citizen-centric digital government: A conceptual model. Sustainability 2021;13:13112.
8. Gogineni A. Observability driven incident management for cloud-native application reliability. Int J Innov Res Eng Multidiscip Phys Sci 2021;9:1-10.
9. Srinivas M, Ramakrishna G, Rao KR, Suresh Babu E. Analysis of legacy system in software application development: A comparative survey. Int J Electr Comput Eng 2016;6:292-7.
10. Yang G, Jan MA, Rehman AU, Babar M, Aimal MM, Verma S. Interoperability and data storage in internet of multimedia things: Investigating current trends, research challenges and future directions. IEEE Access 2020;8:124382-401.
11. Fahmideh M, Daneshgar F, Rabhi F, Beydoun G. A generic cloud migration process model. Eur J Inf Syst 2019;28:233-55.
12. Seetharaman KM. Internet of things (IoT) applications in SAP: A survey of trends, challenges, and opportunities. Int J Adv Res Sci Commun Technol 2021;3:499-508.
13. Ali M, Hussain S, Ashraf M, Paracha MK. Addressing software related issues on legacy systems - a review. Int J Sci Technol Res 2020;9:3738-42.
14. Murua A, Carrasco E, Agirre A, Susperregi JM, Gãmez J. Upgrading legacy EHR systems to smart EHR systems. In: Smart Innovation, Systems and Technologies. Germany: Springer; 2018.
15. Seetharaman KM. End-to-End SAP implementation in global supply chains: Bridging functional and technical aspects of EDI integration. Int J Res Anal Rev 2021;8:1-7.
16. Thokala VS. Integrating machine learning into web applications for personalized content delivery using python. Int J Curr Eng Technol 2021;11:652-60.
17. Thakran V. Environmental sustainability in piping systems : Exploring the impact of material selection and design optimisation. Int J Curr Eng Technol 2021;11:523-8.
18. Pillai V. A study of database migration: Understanding the user experience. TRACE Tennessee Res Creat Exch 2019.
19. Hussein AA. Data migration need, strategy, challenges, methodology, categories, risks, uses with cloud computing, and improvements in its using with cloud using suggested proposed model (DMig 1). J Inf Secur 2021;12:79-103.
20. Neeli SS. Key challenges and strategies in managing databases for data science and machine learning. Int J Res Publ Rev 2021;2:9.
21. Tupsakhare P. From legacy to modern: Migrating with SQL server migration assistant. J Sci Eng Res 2020;7:304-8.
22. De Haas H. A theory of migration: The aspirations-capabilities framework. Comp Migr Stud 2021;9:8.
23. Dizdarević J, Carpio F, Jukan A, Masip-bruin X. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. 2018;51:1-29.
24. Gupta V, Khera S, Turk N. MQTT protocol employing IOT based home safety system with ABE encryption. Multimed Tools Appl 2021;80:2931-49.
25. Neeli SS. Optimizing database management with devops: Strategies and real-world examples. J Adv Dev Res 2020;11:8.
26. Goyal A. Enhancing engineering project efficiency through cross-functional collaboration and iot integration. Int J Res Anal Rev 2021;8:396-402.
27. Carling J, Collins F. Aspiration, desire and drivers of migration. J Ethn Migr Stud 2018;44:909-26.
28. Pandya S. Predictive analytics in smart grids : Leveraging machine learning for renewable energy sources. Int J Curr Eng Technol 2021;11:677-83.
29. Immadisetty A. Edge analytics vs. Cloud analytics: Tradeoffs in real-time data processing. J Recent Trends Comput Sci Eng 2016;13:42-52.
30. Salvatierra G, Mateos C, Crasso M, Zunino A, Campo M. Legacy system migration approaches. IEEE Lat Am Trans 2013;11:840-51.

31. Immadisetty A, Olusegun J. Real-time data analytics in customer experience management: A framework for digital transformation and business intelligence. Int J Sci Res Comput Sci Eng Inf Technol 2021;10:1280-8.
32. Kolluri V. A comprehensive analysis on explainable and ethical machine: Demystifying advances in artificial intelligence. TIJER Int Res J 2015;2(7):pp. 2349-9249.
33. Abu Bakar HK, Razali R, Jambari DI. A guidance to legacy systems modernization. Int J Adv Sci Eng Inf Technol 2020;10:1042-50.
34. Thokala VS. Utilizing docker containers for reproducible builds and scalable web application deployments. Int J Curr Eng Technol 2021;11:661-8.
35. Wahyudi A, Junirianto E, Franz A. Web and application program interface (API) design "parmon" modern parking application. TEPIAN 2021;2:108-15.
36. Neeli SS. Real-time data management with in-memory databases : A performance- centric approach. J Adv Dev Res 2020;11:1-8.
37. Althani B, Khaddaj S. Systematic review of legacy system migration. In: 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES). United States: IEEE; 2017. p. 154-7.
38. Wijaya YS, Akhmadarman A. A framework for data migration between different datastore of NoSQL database. In: Proceeding - 2018 International Conference on ICT for Smart Society: Innovation Toward Smart Society and Society 5.0, ICISS; 2018.
39. M'Baya A, Laval J, Moalla N. An assessment conceptual framework for the modernization of legacy systems. In: International Conference on Software, Knowledge Information, Industrial Management and Applications. Delaware: SKIMA; 2017.
40. Khan R, McLaughlin K, Kang B, Laverty D, Sezer S. A seamless cloud migration approach to secure distributed legacy industrial SCADA systems. In: 2020 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference. ISGT; 2020.
41. Preti JP, Souza AN, Freiberger EC, De Almeida Lacerda T. Monolithic to microservices migration strategy in public safety secretariat of mato grosso. In: 3rd International Conference on Electrical, Communication and Computer Engineering. ICECCE; 2021.
42. Schnappinger M, Streit J. Efficient platform migration of a mainframe legacy system using custom transpilation. In: Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution. ICSME; 2021.
43. Martens A, Book M, Gruhn V. A data decomposition method for stepwise migration of complex legacy data. In: Proceedings - International Conference on Software Engineering, 2018.