Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Reusability of Legacy Software Using Microservices: An Online Exam System Example

Saad Hussein Abed Hamed

College of Computer Science & Information Technology, Al-Qadisiyah University, Al-Diwaniyah, Iraq. E-mail: shsaadsh2014@gmail.com.

A R T I C L E I N F O

Article history:
Received: 01 /06/2023
Rrevised form: 10 /07/2023
Accepted : 16 /07/2023
Available online: 30 /09/2023

Keywords:

Microservice Architecture , On line exam, Monolithic Architecture, reusability.

A B S T R A C T

A new design approach called microservices-based architecture is quickly emerging as one of the most efficient ways to re-architect aging enterprise systems and reengineer them into new modern systems . Microservice architecture is essential for creating high-quality, scalable, high-performance, and easy-to-maintain software. The newest method of creating new applications or reorganizing existing systems is the microservice-based architecture. It is thought to be more efficient in a number of areas than the more traditional service-oriented design, including: such as maintainability, dependability, scalability, and agility. Software Engineering developed an architecture with independent and autonomous components known as the microservice architecture in response to the requirement to enhance and evolve software architecture design. Online exams are crucial components of education. It minimizes the vast amount of material resources and is both quick and effective. However, monolithic design, which was used to create online exam systems, has more issues and is incompatible with cloud computing, distribution, or new technologies. This paper presents on evolving and reusability a legacy enterprise system (Online Exam) using a microservice architecture. the study's contributions To update a legacy enterprise system, feature-driven microservice-specific transformation rules are adopted. Performance, maintainability, scalability, and testability are prioritized when comparing a historical monolithic architecture to a microservice architecture.

MSC.41A25; 41A35; 41A3

https://doi.org/10.29304/jqcm.2022.14.4.880

1. INTRODUCTION

The most common method of putting an application into action is to create a single executable program that uses the same machine's resources. The resources here could range anywhere from hardware-related resources like memory, databases, etc. to more software-related resources like files and drivers, for instance. This facilitates quick and simple development but creates more problems as the application expands. It will eventually become too much to handle. Any modifications to the

∗ *Corresponding autho :* Saad Hussein

Email addresses: **shsaadsh2014@gmail.com**

Communicated by: Dr. Alaa Taima Albu-Salih application will necessitate the deployment of a new version, which runs the danger of crashing the entire program. The development team will also be limited to using the same language that the program is based on when adding new features. describing a monolithic enterprise program that typically consists of three components: client-side, server-side, and database. The client-side work, which serves as the user interface, is primarily constructed using JavaScript for a browser. The server-side program will take HTTP requests sent from the client-side and carry out the necessary steps to manage a response sent back to the client. The server-side application's queries are finally managed by a database management system. This is also known as a three-tier application, where the client-side is referred to as the presentation tier, the server-side as the application tier, and the database as the data tier. One could refer to a server-side application as a monolith if it only has one logical executable.

Reusability in software engineering refers to the usage of already-existing software components in the development of new software[1] .

Increasing the efficiency of software development teams and lowering associated expenses throughout the lifecycle of a software system are two goals of the long-studied topic of software reusability. Additionally, the primary function of reusability is to enhance the ability of software applications to be maintained and to offer mechanisms for data integration between information systems and interoperability[2].

The vast majority of current educational and academic systems are long-lasting programs that are frequently created using a monolithic architecture. These architectures deteriorate and degenerate as a result of intensive maintenance and outdated technologies. Only updating programming languages, not architecture [3].

Legacy systems impede digital transition, hinder innovation, and consume a lot of resources for maintenance, diverting funds that could be invested in system evolution, such as creating new features[4].

By employing architectures that identify its components through services, concerns with software reusability can be avoided, according to recent research in the field of software engineering[5].

Educational institutions are updating these antiquated, uniform systems in order to remain competitive[6].

As an approach for updating monolithic legacy systems, microservice architectures are currently in vogue[7].

These systems typically start out small but eventually expand to accommodate growing corporate demands. With time and the addition of new functionality, the following issues may start to plague monolithic applications:

- Because the system's components are closely connected, no one component can be scaled separately.
- Tight coupling and hidden dependencies make it difficult to maintain the code.
- Testing gets more difficult, increasing the chance of introducing vulnerabilities.
- Future development and stability could be hampered by these issues. Teams become hesitant to make modifications, particularly when the original developers are no longer engaged in the project and the design documentation is lacking or old.

Academics have become interested in studying software architectural models because it is imperative to produce software systems that are maintainable and reusable[8].

This paper proposes and applies an evolution framework and a set of feature-driven, microservices-oriented evolution rules to modernize legacy enterprise systems, with a focus on analyzing the implications for reusability . In comparison to the previous system, the new one will be faster, more efficient, and more expandable.

In this study, we explore the 'OExam' monolithic system, which we assessed and later transformed into a brand-new microservice-based design. A MS platform was used to fully implement and install the new system. This section gives a brief overview of the monolithic OExam system's history before describing how it transformed into a microservices design. Background information on monolithic systems is presented in the second section, followed by related work in the third, proposed methodologies in the fourth, and implementation in the fifth.

II. Background

We worked on this study in 2011 on electronic test technology because it aims to automate manual tasks like preparing questions, printing them, distributing them to students, and then correcting and monitoring grades. These tasks involve a lot of staff members, and a lot of time and energy are wasted due to errors. We looked at the manual tasks and the supporting documents and relied on data entry in this system. Or remove any inaccurate information or objects that surface when examining reports. the system by Monolithic Architecture.

Monolithic codebases can become complex over time, making it difficult to manage and scale. Vertical scaling, which adds compute resources, can be expensive and limit the application's vertical scaling capabilities. Technology limitations and a single point of failure make it difficult to add or change functionality in monoliths.

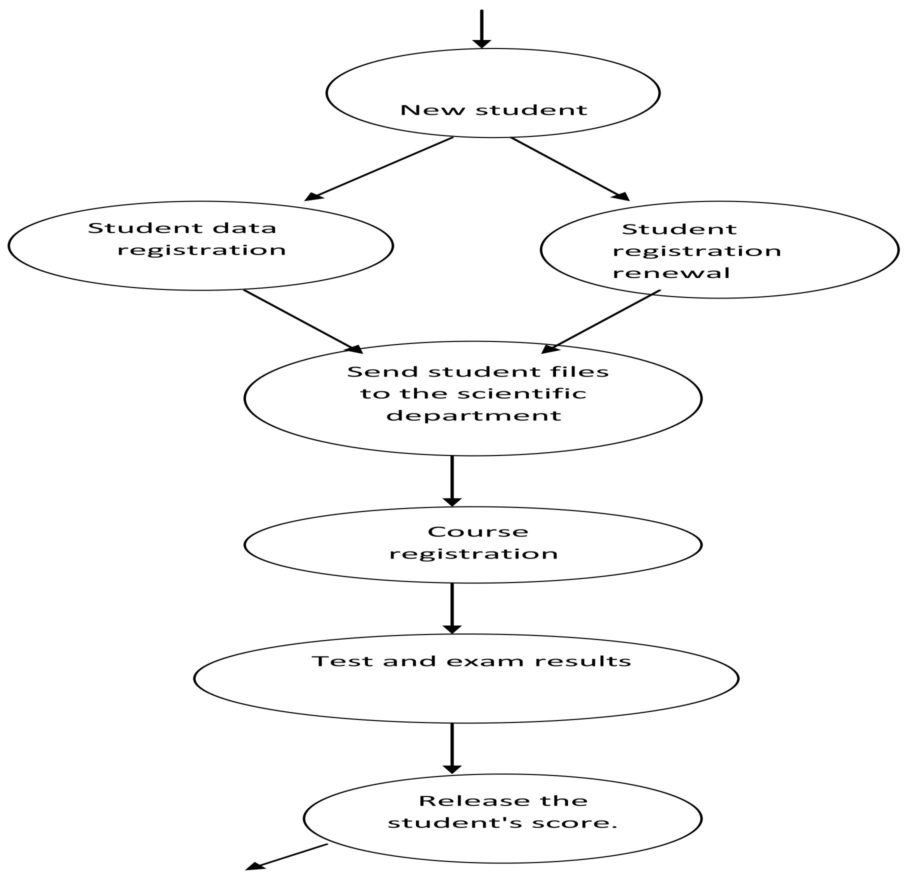
New student
Student data registration
Student registration renewal
Send student files to the scientific department
Course registration
Test and exam results
Release the student's score.

Figure 1, shows the scheme of business activities.

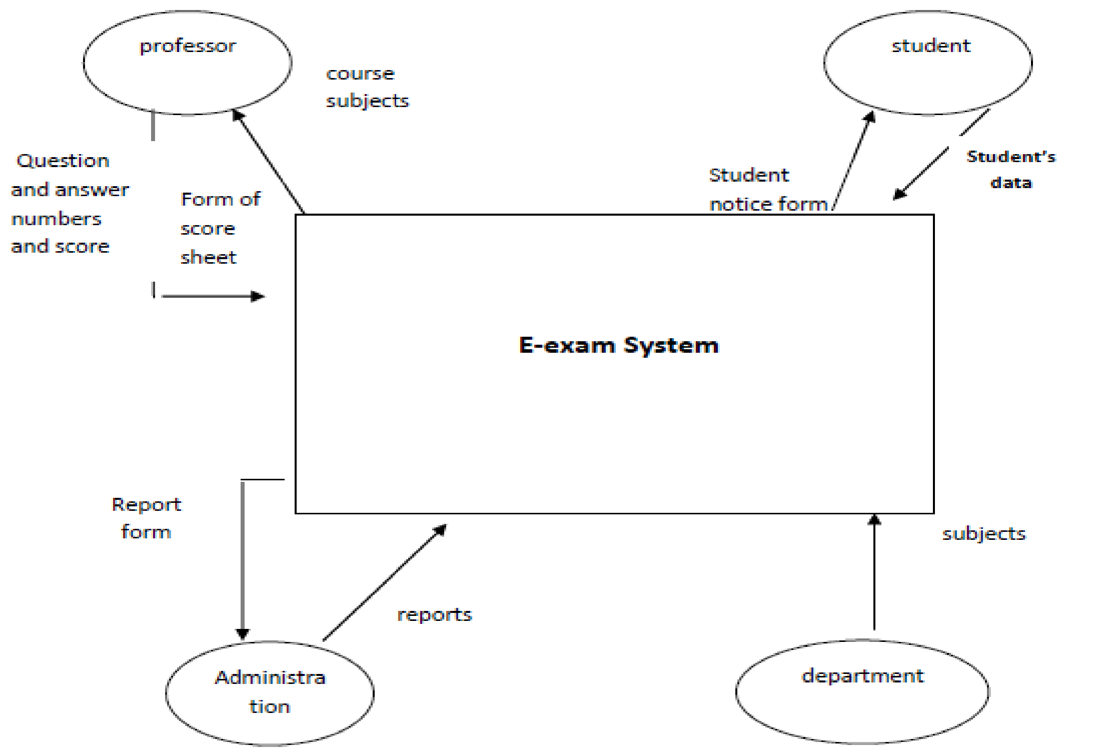

Figure 2 shows the scheme of environmental documents.

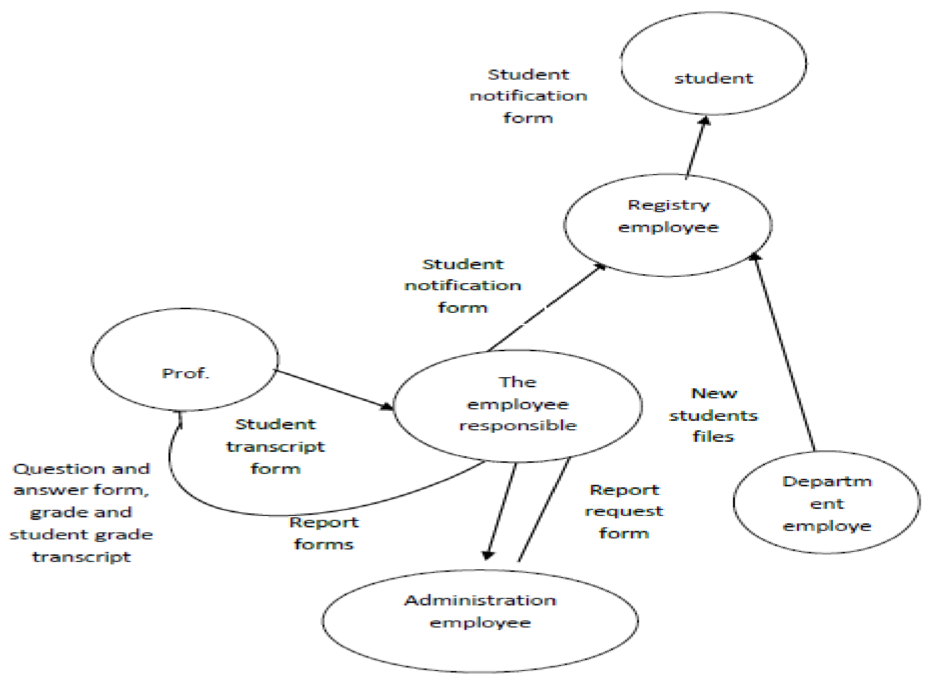

Figure 3 shows the document flow chart

These systems begin simple but eventually expand to accommodate changing business requirements. A monolithic program may eventually start to have the following issues as new features are added:
- Tight coupling and hidden dependencies make it difficult to maintain the code and prevent the system's constituent components from scaling independently.
- Testing gets more difficult, increasing the chance of introducing vulnerabilities.

Future growth and stability may be hampered by these issues. Teams become apprehensive of making modifications, particularly if the original developers are no longer working on the project and the design documentation is lacking or old.

Building, debugging, and reasoning about large monoliths frequently get more difficult over time. This is the time when switching the application over to a microservices design may make sense.

Microservices, in contrast to monoliths, are frequently dispersed, loosely connected execution units.

To prevent failures or overruns, migrating a monolith to a microservice needs a significant amount of effort and money. It's important to comprehend both the advantages and the difficulties that microservices provide if you want to make sure that any move is effective. The advantages include:
- Services can change on their own, depending on user needs.
- Services are scalable on their own to satisfy consumer demand.
- As features are released to the market more quickly, development cycles become shorter over time.
- Services are more resilient to failure and segregated from one another.
- A single service failure won't shut down the program as a whole.
- Testing improves with the use of behavior-driven development.

Microservices offer significant benefits, including scalability, flexibility, and the ability to handle complex, large-scale applications.

Utilizing the single responsibility and independent deployment of each service is the ultimate purpose of implementing a microservices architecture. However, we must carefully consider how to design and construct a microservice in isolation and then test every service simultaneously in the implementation phase before the final release. The size and scope of each microservice must also be carefully considered. The system was thoroughly analyzed in order to comprehend the full architecture before creating a microservices architecture.

The following steps were taken to break down the system into a microservices architecture:
1. Recognize the code and establish the parameters of the system.
2.Created a class diagram for the current system, which will help understand the connections between the classes in the system and will be crucial in determining the service.
3. Identify the issues that the architecture of the microservices should address.
4.Applying a domain-driven design (DDD) will help separate the services from the monolith. A business domain is broken down into smaller functional components using the DDD technique, and the problem's independent steps are described. A bounded context contains information specific to each domain, such as the domain model, data model, and application services.
5. Recognize the shared database schema and deconstruct it so that may extract the tables and save them each in a separate, independent database that will later be used by a microservice.

The use of microservices in software development will not diminish. The following tendencies should be noted:

Rising Adoption: We can anticipate that the use of microservices will continue to grow as more businesses realize their advantages. This is especially true for businesses that must manage large, sophisticated applications or swiftly increase their operations.

Tooling Improvements: As microservices gain popularity, we may anticipate improvements to the tools and technology that support them. Improved monitoring and container orchestration tools are included in this.
Server less designs: Server less designs work well with microservices because they let the cloud provider control the server infrastructure. These two tendencies will likely combine more in the future, as expected.

III. Related work

This section examines academic works that cover the usage of archetypes in developing of Exam applications, including services, architectures, and data persistence.
This study [9] offers a comparison of twelve proctoring systems, a fundamental evaluation methodology for proctoring systems, and many crisis management suggestions for educational institutions. give a fundamental paradigm for proctoring system evaluation, and suggest some things to think about while selecting them. The study had no real-world applications; it was just theoretical.
[10] This study examines how online examinations affect students' anxiety and self-efficacy. Anxiety has a detrimental effect on self-efficacy, according to a cross-sectional survey of 434 university students. The study discovered that emotion-focused coping techniques and adaptive behavioral coping techniques are effective in reducing examination-related anxiety.
[11] The goal of this study is to develop and put into practice an intelligent online proctoring system (IOPS), which is urgently required in online learning environments all over the world, to oversee online exams. The authors tested this approach in a real university online exam as a pilot application, and they verified the proctoring outcome.
IOPS uses the browser/server (B/S) architecture. The server side is constructed using the programming languages C and Python, and it saves information about each examinee's identification as well as the status of any significant behavior changes, such as facial expression, eye and mouth movement, and voice. The examinee writing the online test generates multimodal data, which the browser side collects, processes locally, and sends the server the most crucial behavior status change data. With the use of open-source software, real-time facial and voice recognition is implemented. This study cannot ensure the integrity and equality of all examinees as in conventional classroom tests since it lacks the functionality to stop students from cheating on online exams.
Because a monolithic design would not be able to satisfy these needs, the microservices architecture is chosen, which creates new opportunities. These studies are different from ours in that they concentrate on particular subjects and issues related to Online exams rather than the general manner in which the modernization process is carried out. However, they have different goals than we do. Although these studies offer helpful understandings of the migration process for online exams, they are not concerned with formulating a general roadmap.
This study organized the workflow and process settings for each phase of the conventional re-examination, examined the flaws in the traditional or online software architecture, and made the first attempt to implement the microservices architecture for the online re-examination of students' entrance exams.

IV. Proposed MS OExam

The current mainstream microservice architecture for the Internet has the following two advantages . The modules are highly autonomous, which allows for rapid changes and independent module updates in Internet applications. - The modules have high scalability, which allows for unpredictable users and dynamic resource allocation in Internet applications. As a result, microservices are gradually becoming the dominant architectural model for building Internet applications .[12]

The microservices platform offers the fundamental infrastructure for utilizing and managing distributed services. It also offers the potential for integrating multiple technologies during system construction using a variety of public service components, all while guaranteeing that data standards and specifications are met. Microservices are loosely connected in terms of their operational logic and provide data-level support for one another. They operate independently of one another and can be further integrated to carry out trickier tasks. A multi-tier architecture was used in the design of the online re-examination system for student exams, as depicted in Fig 4. These comprise the presentation layer, application service layer, and data management layer.

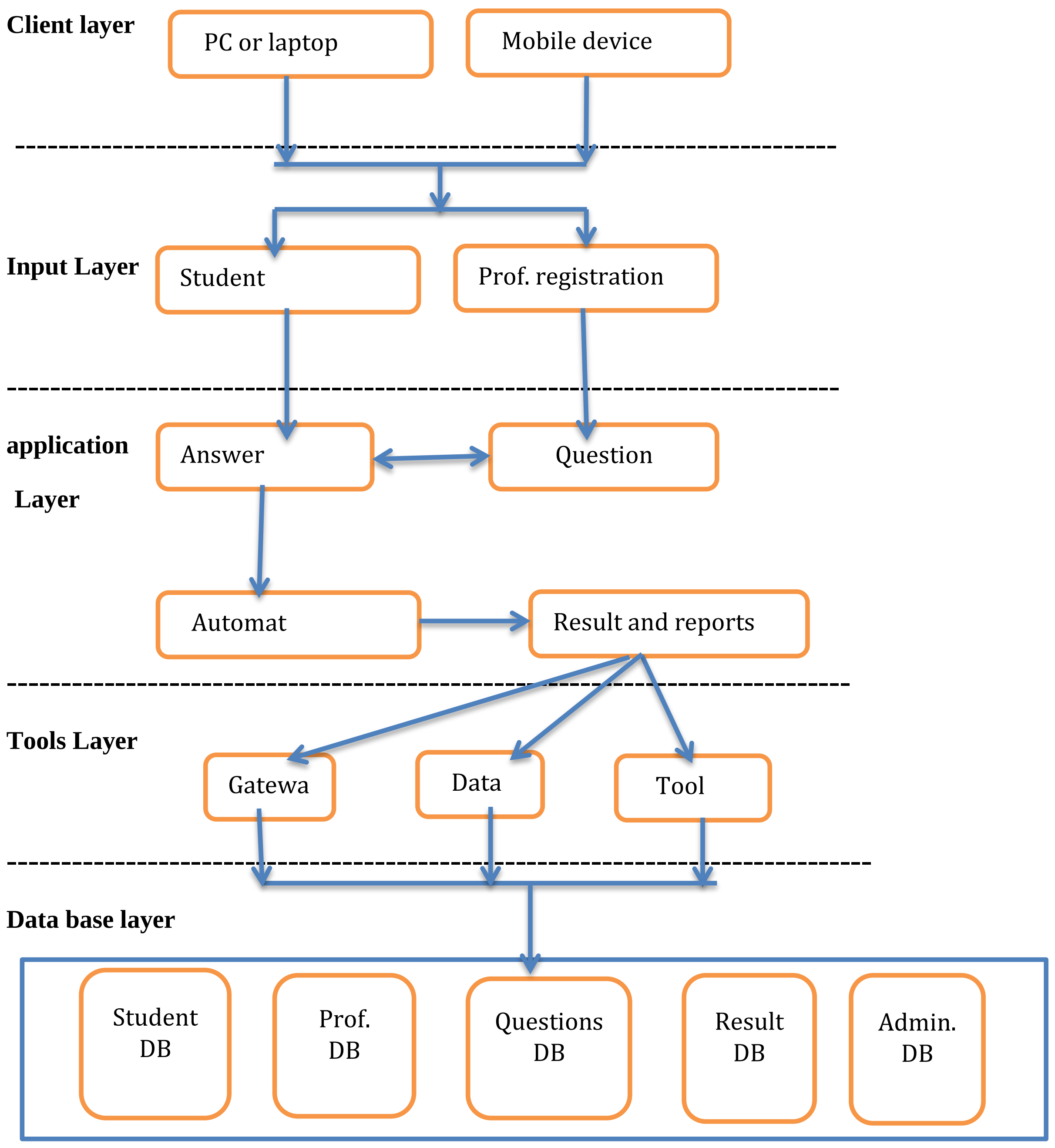

Figure 4, shows the Architecture diagram of OExam system

The display layer, which is also known as the client layer, enables a variety of access techniques, including desktop and mobile browsers, WeChat, and mobile apps. Through a simple communication protocol, the presentation layer can be linked to the application service layer. API Gateway: At the system boundary, the API gateway, which is a crucial component of the microservice architecture, offers centralized powerful control services for the serial APIs.

The API gateway implemented an outer layer and gave the front-end uniform access rules in terms of design patterns. Additionally, the API gateway in this system offers functions like load balancing, service routing, and request filtering that keep the inside and outside of the system separate, assuring the security of back-end services. The application service layer, which makes up the heart of MSA, uses a service-oriented strategy to call and combine "decentralized services" to satisfy the functional needs of various modules. Microservices for video chat, file sending and receiving, payments, data analysis, and messaging were all used in the construction of this layer.

Proposed migration decision model for OExam

1. Investigate the motivating factors.. ***Initiation***

2. Recognize the legacy system. -------------------------------------

3. Break down the old system ***Planning***

4. The microservice architecture should be defined.

5. Implement the modernization.

6. Sync the traditional systems with the microservices **----------------------**

7. Check and confirm the microservices ***Execution***

8. the infrastructure and microservices should be watched. ***Monitoring***

V. IMPLEMENTATION

Flexible application of the aforementioned online re-examination system for student admissions is possible to fulfill the requirements of many operational processes, including interviews, written exams, and admissions. Users can submit information and complete written tests in many categories after logging in. Additionally, the outcomes can be calculated and imported to provide students with answers. The back-end administrator has access to import student lists, troubleshoot written test equipment, speak with specialists, and view student data. The entire system has a microservice design and is split up into numerous application microservices, including qualification reviews.

Each microservice is separately deployed and has the flexibility to be modified to meet the needs of an application that is constantly developing. The system utilizes a microservices design that divides the front- and back-end, increasing development productivity and enhancing code maintainability.

The primary motive for developing and implementing online tests in a large course was to eliminate a lot of paperwork and the time and effort of grading assignments and exams in a large classroom. This system aims to maintain the integrity of exams by providing real-time monitoring.

The first step is to identify both functional and non-functional needs and requirements. List and rank the chosen criteria. needing to Analyze the resources that are at disposal, including the size of the development team, the knowledge and experience of the team members, and the tools and technologies that are at disposal.

Microservices are a suitable fit for the project's requirements for complexity and adaptability as well as high availability and performance. The team possesses the necessary abilities for developing microservices, and their large budget allows them to invest in an infrastructure that can meet their strict reliability and resilience standards.

VI. Conclusion

Any organization that wants to increase business agility must modernize its current systems and services. According to research, services and functions could be moved to take use of cloud computing and Internet of Things (IOT) characteristics. Cloud computing and IOT are challenging to implement in legacy systems. The primary drawbacks of monolithic applications are to the maintenance, upgrading, and scale due to the difficulty of adopting new frameworks or technologies. Compared to monolith (limited architecture), microservices architecture are more dependable. This paper suggests a plan to develop an online re-examination system for student tests in light of the pressing need to reform online assessments within the context of the microservice architecture.

We suggest adopting a common microservice architecture for the online re-examination after looking into the operational mode and workflow settings of each operational procedure of re-examination for student exams and looking into the shortcomings of conventional software architecture. We suggest using a common microservice architecture for the online re-examination of student examinations after examining the operational mode and workflow settings of each operational procedure of re-examination and examining the flaws in traditional software architecture. Every online retest and admission function that has been suggested for students To lessen the coupling between processes, a multi-layer architecture was used in the design and divided into the presentation, application service, and data layers.The adoption of a microservices architecture during development increases maintainability and development efficiency.

References

[1] R. Prieto-Diaz, "Status report: software reusability," in IEEE Software, vol. 10, no. 3, pp. 61-66, May 1993. Doi: 10.1109/52.210605.

[2] N. Padhy, R. Panigrahi, and S. Baboo, "A Systematic Literature Review of an Object Oriented Metric: Reusability," 2015 Int. Conf. on Computational Intelligence and Networks, Bhubaneshwar, 2015, pp. 190-191. DOI: 10.1109/CINE.2015.44.

[3] Robert C. Seacord, Daniel Plakosh, and Grace A. Lewis. 2003. Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices.Addison-Wesley Longman Publishing Co., Inc., USA.

[4] Jason Miller. 2018. Spending on legacy IT continues to grow, but there is light at the end of the tunnel. https://federalnewsnetwork.com/ask-the-cio/2018/08/spendingon-legacy-it-continues-to-grow-but-there-is-light-at-the-end-of-the-tunnel/.

[5] F. Mohr, T. Lettmann, E. Hüllermeier, "Planning with Independent Task Networks, " in Kern-Isberner G., Fürnkranz J., Thimm M. (eds)

[6] KI 2017: Advances in Artificial Intelligence. KI 2017. Lecture Notes in Computer Science, vol 10505. Springer, Cham.

[7] H. Knoche and W. Hasselbring. 2018. Using Microservices for Legacy Software Modernization. IEEE Software 35, 3 (2018), 44–49.

[8] Yingying Wang, Harsha Kadyala, and Julia Rubin. 2020. Promises and Challenges of Microservices: an Exploratory Study. Empirical Software Engineering (2020), 1–45.

[9] M. Agarwal and R. Majumdar," Software Maintainability and Usability in Agile Environment,", Int. J. of Computer Application, Vol.68, No. 4, 2013, pp. 30-36.

[10] Cai, Haiyan." Education Technology for Online Learning in Times of Crisis", IEEE TALE2020 – An International Conference on Engineering, Technology and Education Page 758. Authorized licensed use limited to: University of Canberra. Downloaded on May 23,2021 at 14:19:38 UTC from IEEE Xplore. Restrictions apply.

[11] Arora, S., Chaudhary, P. and Singh, R.K. (2021), "Impact of coronavirus and online exam anxiety on self-efficacy: the moderating role of coping strategy", Interactive Technology and Smart Education, Vol. 18 No. 3, pp. 475-492. https://doi.org/10.1108/ITSE-08-2020-0158.

[12]- Y.W. Luo, J.W. Yao, B. Xu, et al, "Clinical medical teaching management system based on micro-service framework," China Medical Education Technology, vol. 33, no. 6, pp. 738-741, 2019.

[13] Jia, J. and He, Y. (2022), "The design, implementation and pilot application of an intelligent online proctoring system for online exams", Interactive Technology and Smart Education, Vol. 19 No. 1, pp. 112-120. https://doi.org/10.1108/ITSE-12-2020-0246.

[14] J. Zhao, Z.H. Chen, Z.H. Gao, "Design and implementation of integrated scientific research management platform based on microservice architecture," Radio Engineering, vol. 49, no. 05, pp. 436-441, 2019.

[15] S. Prathish, A. N. S and K. Bijlani, "An Intelligent System For Online Exam Monitoring," in 2016 International Conference on Information Science (ICIS)