International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

The Architectural Reengineering of Existing Web-based System

Enkhtuul Bukhsuren[1], Adiyatumur Tsogtkhuu[2*], Baatarbileg Altangerel[3], Oyun-Erdene Namsrai[1], Munkhtsetseg Namsraidorj[1], Amirlan Enkhtur[4]

Submitted: 03/05/2024 **Revised**: 16/06/2024 **Accepted**: 23/06/2024

Abstract: In this article, we discuss web-based system development, including its trends, stages of software architecture development, comparison of service-oriented, microservices, and serverless architectures, and research on scalability. We also delve into software re-engineering techniques, methods for migrating old web system architectures to new ones, and the new architecture's structure and components. Our research and methodology aimed to improve the architecture of a web access system in need of change, and we implemented a test using a case study on the SiSi system of the National University of Mongolia.

Keywords: *forward engineering, microservices, monolithic, serverless system re-engineering, reverse engineering, system architecture, web-based software development*

1. Introduction

Any software is the best possible system existing at that moment, built on good structure and business process based on the latest platform and technology. However, as user requirements and technologies are constantly changing and evolving, there is always a need for maintenance and modification in any system. Examples of these modifications are the improvement of the system architecture, making the system more flexible and more accessible to develop, researching for ways to improve the procedure and structure of offering services, the age/years using the service being modified, and efficiency. When user access to a web system increases, the inability to process the high request load can cause system security loss, inability to access the system, data loss, user frustration, and system instability.

Based on these reasons, there is an urgent need to separate existing systems into functional and system modules, classify system users, and adopt a microservices architecture consisting of integration of web applications separated by each business activity that meets modern development trends and security standards being deployed in existing systems [1].

2. Web-based system development

Most systems that assist people in their daily activities are web-based and are widely used in research and development in computer science, information systems, web technology, and other fields. Computer-based systems, especially web platforms, are being used in many industries. For example, information processing systems [2-3], supporting systems for research and

analysis, training and educating [4], decision-making [5], computerized medical, and knowledge management [6].

Web application architecture describes the software components that comprise the system (client-side, middleware, server-side, database, etc.) and how they interact. For system monolithic, service-oriented, and microservice architecture, HTTP is used to transmit requests and responses. In contrast, message queue architecture shows how to communicate between the "producer-producer" or client and the "consumer-consumer" or server using AMQP protocol. In addition, it ensures that all user requests contain valid data. It creates and manages accounts while providing permission-based access and authentication. Choosing the exemplary system architecture that meets business requirements determines the organization's growth, reliability, interoperability, and future IT needs. For this reason, the components that make up the software architecture of a web-based system should be carefully studied, and the architecture that suits the organization should be chosen.

The software architecture of a web-based system consists of 3 main components:

- **Web browser:** The user-side component/front-end/ is the main component that interacts with the user, receives input, controls the user's interaction with the application, and contains the user logic of the system.
- **Web Server:** The server-side component/backend/ processes user requests by handling business logic, routing requests to the suitable component, and managing

[1]*Department of Information and Computer Sciences, School of Information Technology and Electronics, National University of Mongolia. Email:enkhtuul@seas.num.edu.mn, oyunerdene@seas.num.edu.mn, munkhtsetseg@seas.num.edu.mn*
[2]*Unimedia Solutions LLC, Mongolia. Email:adiyatumur@gmail.com*
[3]*Office for Digital Transformation Policy, National University of Mongolia. Email: a_bbileg@num.edu.mn*
[4]*Student of Mongol Aspiration International School, Mongolia Email:amirlan@gmail.com*
**Corresponding Author Email: adiyatumur@gmail.com*

the entire application process. Here, a variety of user requests can be run and tracked.

- **Database:** A database server handles the data required by an application and the tasks associated with that data. In a multi-tier architecture, business logic can be managed by stored procedures that execute database services.

2.1. Development trends

In a traditional 2-tier architecture, there are two components: the user-side system, or user interface, and the database server. The business logic is executed through the user interface and the results are processed in the database server. The downside of the 2-tier architecture is that performance degrades as the number of users increases. In addition, the direct interaction between the database and the user's device creates certain security issues. The next step in the development phase is the 3-tier architecture, with the following layers [7]:

• Presentation layer/Client Layer

Enables users to interact with the server and server services through a browser. It receives requests and provides the necessary information to the user, as the necessary code resides in the browser.

• Application Layer/Business Layer

It is a key component of a web application architecture that receives user requests, executes business logic, and delivers required data to front-end systems. This includes servers, databases, web services, etc.

• Data Layer

A core component for storing and managing data in a 3-tier architecture of web-based system software. This layer can use SQL, a relational database, and NoSQL, a non-relational unstructured database.

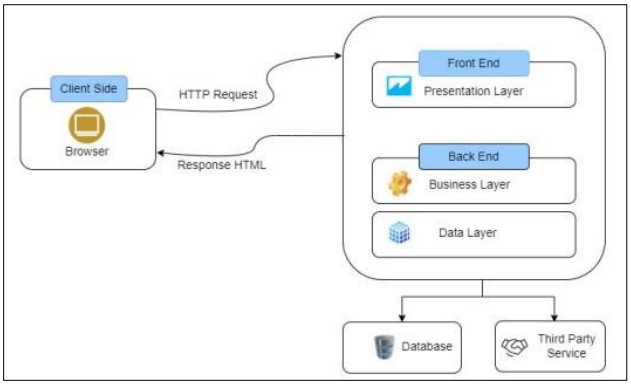

Fig. 1. General architecture of a three-tier web application.

In Figure 1, intermediate servers receive user requests and process them by coordinating with subordinate servers using business logic. The communication between the client and the database is managed by a middle application layer, which allows the client to access databases with many different configurations. A 3-tier architecture is more secure because the user does not directly access the data. Hosting application servers on multiple machines allows for greater scalability, better performance, and reusability. In addition, it is horizontally expandable in each direction. It can abstractly differentiate and separate the core operations from the database server for efficient load balancing. Data integrity means that all data passes through an application server that decides how and who can access the data. This modular design allows for changes to one hierarchy without affecting other components. Based on this theory, the following new Web-based architectures have emerged.

3. Software architecture of Web-based systems

Several web application architectures are based on software development and deployment models.

Monolithic architecture

A monolithic architecture is a traditional software development model where the entire software is developed using a standard waterfall model. This means that all the components are interdependent and interconnected; for one component to run, the whole program must run. To reprogram a function, it is necessary to recompile the entire code, even if it is to change one part of the program. Monolithic architecture [8] treats the whole code as a single application, so creating a new project, selecting frameworks, scripts, templates, testing, and deploying is simple. It is more suitable for small-scale projects and for organizations that want to save on the cost of developing applications. There are many tools available to simplify the development. All operations are performed with a single control, which makes it easy to manage what goes in and to where. Monolith saves a lot of time as developers can make changes and updates simultaneously instead of separately.

Compared to applications with microservices architecture, the performance is faster due to the small size of the app. For example, in a microservice architecture, there are at least 40 services, each with a user interface thus, the performance is slower than monolithic architecture. The downside is that the larger the code, the more difficult it becomes to manage and understand the entire project code, even for small changes.

Since each element is interdependent, scaling the application is not easy. Furthermore, just one small error can make the entire program unworkable.

Service-based architecture

Service-oriented architecture (SOA) is a style of software architecture in which software agents are loosely connected and separated by their role to perform the required functionality.s

SOA has two leading roles: service provider and service consumer. A software agent can perform both of these roles. The application organizes its modules to integrate them

together in a way that makes them easy to reuse. Due to the independent and loosely connected nature of the functional components of service-oriented applications, these components can be reused across multiple applications without affecting other services. As each software service is an independent unit in service-based architecture, it is easy to update and maintain without damaging other services. For example, large enterprise applications can be more easily managed when broken down into services. A service is more accessible to debug and test than a monolithic one, because the code is fragmented. A service-oriented architecture is layered, so parallelism is encouraged during development. Individual services can be developed and completed simultaneously.

Microservice architecture

Microservices are a type of service-based software architecture that consists of several independent components that make up an application. Unlike a monolithic architecture where an application is built as a single indivisible unit, microservices architecture has many small separate components, each with its own APIs.

Microservices architecture is important because it adds unique value by simplifying complexity in a system. Decomposing your system or application into many smaller parts reduces duplication, increases coherence, and shows the connections between parts, making the overall parts easier to understand, extensible, and modify.

Many companies have changed their architecture from the monolithic approach and opted for a Microservices architecture. The most popular companies are Netflix, Amazon, Twitter, eBay, and PayPal.

Figure 2 shows a comparison of the above three architectures.

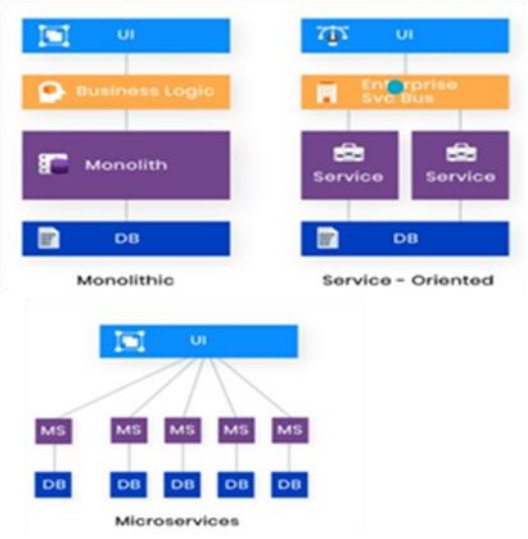

Fig. 2. Monolithic, service-based, and microservice architectures [9].

Serverless architecture

This architecture is a way to build, provide, and operate systems without managing the infrastructure. Although the application still runs on the server, the infrastructure provider handles all server management. Software companies can run their applications, databases, and system resources without hosting or maintaining servers. The advantages of this architecture are paying only for your usage, focusing only on application development, scaling automatically depending on application access and load, high-cost process calculations, automatic deployment, and fast loading of applications by end users. /redirects to a server near the user's location/ etc. On the downside, it is difficult to test and detect errors and data location problems when backend applications are deployed, are not designed for long-term processes, and may cause technical issues when changing hosting. Serverless computing is not a serverless solution but rather a solution in which third-party cloud computing providers provide the infrastructure without the need for developers to worry about server infrastructure.

Amazon Web Services (AWS), Google Cloud, and Microsoft Azure are the three major cloud computing providers.

In serverless architecture, users can obtain two types of services to produce a system architecture [10].

It includes:

• FaaS-Function as a Service – A cloud computing model that allows developers to host parts of functions in the cloud and execute these parts independently

• BaaS-BackEnd as a Service (Server-side service) - A cloud computing model that allows the backend of the system to be managed as a service, such as database management, cloud storage, hosting, user authentication, etc.

Fig. 3. Serverless architecture services.

Figure 3 shows the system's scalability during high user demand for Monolithic, Microservices, and Serverless architectures. The monolithic architecture has an infrastructure solution that allows the system to be cloned to another server when the demand increases. At the same time, microservices can be placed on a cloned server in some parts, and in the case of serverless architecture, they can be freely expanded when the load increases. Unlike traditional apps, it is capable of handling multiple requests.

4. Software Reengineering

Any software process has potential problems, such as outdated designs and technologies, malfunctions, defects, and development stalls. These require reengineering. Reengineering is reengineering an existing design or technology to be more efficient and durable [11].

Reasons for reengineering: Improve financial performance; reduce costs/expenses; reduce pressure from external competition; decrease market share; respond to emerging market opportunities; improve customer satisfaction; improve product and service quality, etc.

Reengineering has two levels:

1. Business Level: Done by business analysts;

2. Software level: Done by PC engineers;

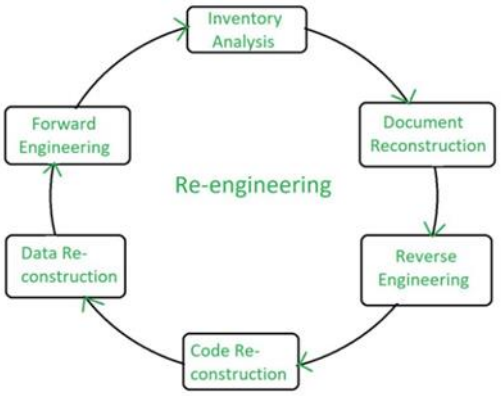

Fig. 5. Frequency of software reengineering.

When the National University of Mongolia (NUM) first moved from the traditional education management system to the online system, it introduced the SISI system to the "School of Information Technology" in the fall semester of 2009 and implemented it in other departments. It was a significant transition in terms of operations, and it was quite an enormous task to enter previous information into the system and educate the users [12].

Around ten years later, in 2021, another major system conversion started again. Two thousand twenty research and testing activities were carried out, and in 2021, access to the student section began. In spring 2021, the old and new systems were run side-by-side, followed by a complete transition to the new system in fall 2022. This migration on the learner web was carried out using “Cascading” and “Parallel” conversion methods. More specifically, the old and new systems will run on separate servers in parallel, with a single database in between. For the old system, modules other than the course selection module will work, but for the new system, the ready modules and course selection module will work, and the modules that are not ready and soon to be added will not work.

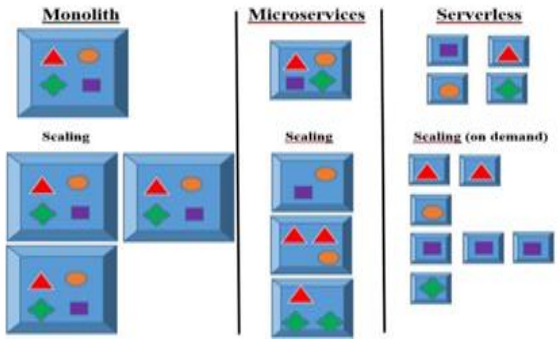

Fig. 4. Scaling for Software architecture.

In this way, there are no differences in data, giving users time to adapt and compare the correct operation of parallel modules.

5. Experimental Results

5.1. Reengineering design on SiSi system

The SiSi system was first developed in 2009 with a 3-tier web system architecture. In 2022, the system was divided into modular subsystems, classifying students, teachers, and teaching departments/others according to the main stakeholders. At the architectural level, the single unified structure of the SiSi system was divided into two parts based on whether there were user interaction and business operations, and the API for data output was transferred to a microservice structure by dividing it into many small services.

5.2. Architectural solutions

To transform the previous monolithic structure into a service-based structure, it is necessary to divide the web application into a backend and a user interface. Along with this, it's essential to design the architecture for more incredible speed and importance Figure 6.

The components of the web application are presented in five parts, each of which can be hosted separately. When the user accesses the system, the front-end application (1) that will be pulled into the user's browser, the cache server (2), and the backend application (3) that will receive the data of the application, the data storage area (4), and the notification application (5) that will send notifications to the front-end application.

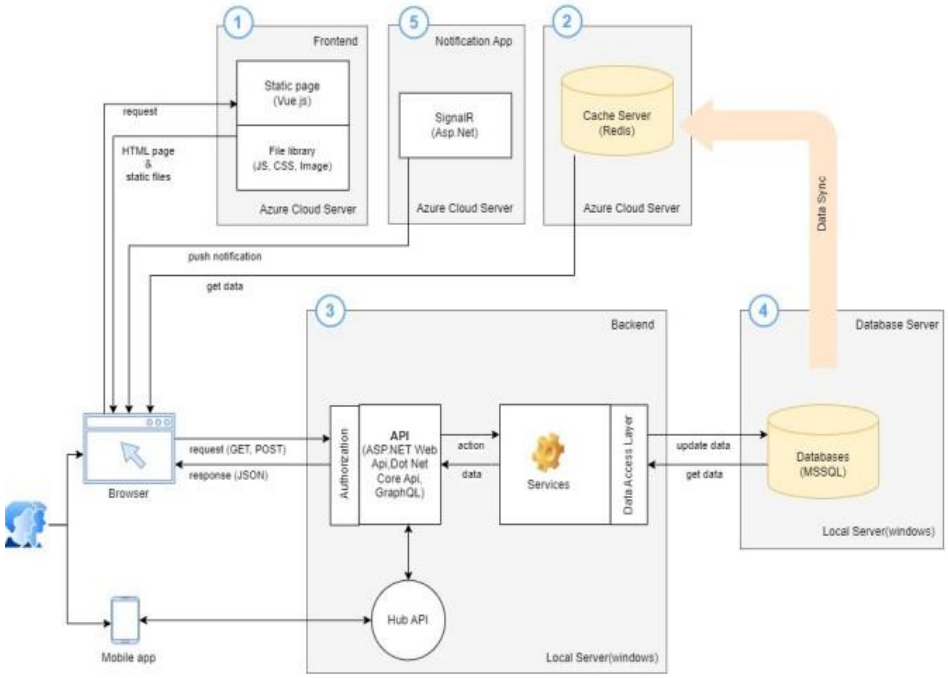

Fig. 6. Web architecture of system.

Following the law of Mongolia, the deployment of applications is shown in such a way that the main data storage and data release APIs are located on local servers, following the provision of personal secret storage and hosting of core systems in the territory of Mongolia [13].

- **User interaction application**

There are static HTML pages and pages containing JS, CSS, and images (Single Page Application). This application can be hosted on-premises or on a cloud server and have multiple copies.

- **Cache server**

A cache server is used to reduce database access. If the user changes the defined data, the data is designed to be updated from the main database or to work without data discrepancies with the main database. The data structure of the cache server originates only from the main database, and the data in the cache server is used only for display to the user.

- **Backend Application Authorization**

There is a function to check if the user is logged in and has access to that API. The user's JWT in the request is checked to identify the user and whether the token is valid to determine whether to access the next-level functions.

API: There are data transfer APIs for user interface applications. After getting user requests, it executes only the necessary services, enters the data into a structure, and returns data in a JSON structure.

Services: By carefully analyzing the business process, it will be possible to define the services appropriately and create a structure that can be used again. Optimizing services will reduce API usage and data overhead and reduce the rate of database access.

Hub API: Hub APIs are for that front-end application only. A hub API is a common bridge between these APIs and other systems. In the example shown in the Figure 7, when the system introduces a mobile application, there will be the addition of APIs in Hub API that are redefined for mobile.

- **Database**

It is advisable to reorganize the database decentralized depending on the application and user. Being separate has the advantage of being located separately on different servers and improving data security.

- **Notification application**

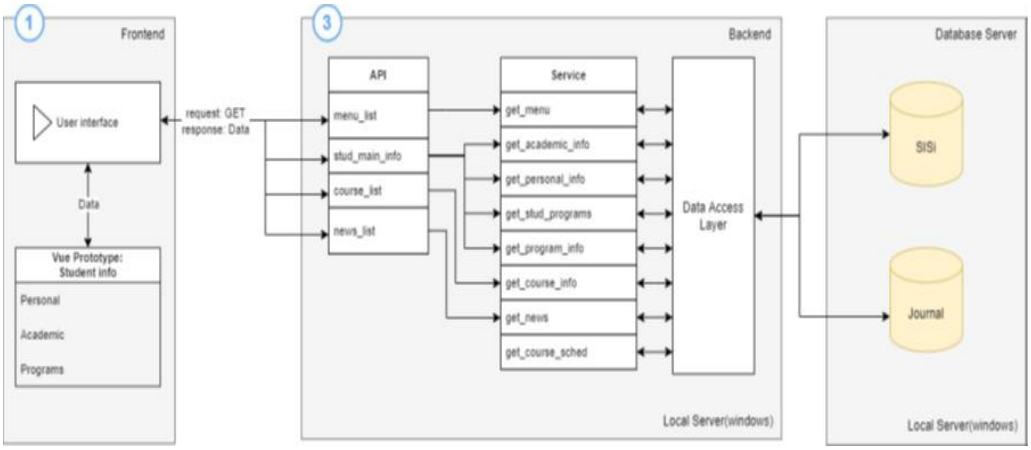

Fig. 7. Web architecture example-1 (first access or home page).

A real-time notification application that sends notifications to front-end applications without the user having to send a request.

The student information API processes the services related to the student's personal, official, and program information and returns it in JSON format. Let's explain the architecture

API - getstudprograms: Study programs/plans/

API - getprograminfo: Program info

API - getcourseinfo: Course info

API - getnews: News and announcements

API - getcoursesched: Course schedule

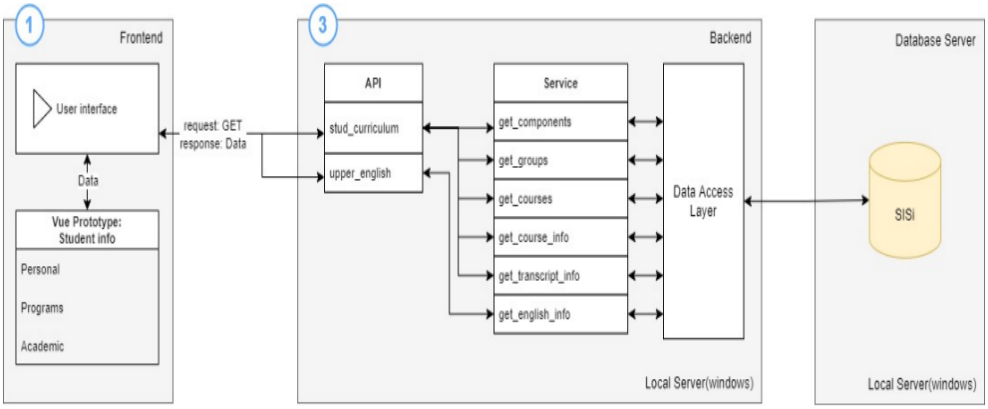

Fig. 8. Web architecture example-2 (education plan).

in Figure 8 with an example.

User interface: Graphical user interface

Vue Prototype: A global variable accessible to all components of the Vue framework. The example shows how to store student data that can be used in most components.

API - menulist: Get user interface menus

API - studmaininfo: Get student information

API - courselist: Get courses being studied in the current term

API - newslist: Get news and announcements

API - getacademicinfo: Official information

API - getpersonalinfo: Personal information

In the student profile view, the student's information is retrieved from the global variable with the help of APIs related to study programs and plans.

API - studcurriculum: Get a student's curriculum

API - upperenglish: Get English-level information

API - getcomponents: Get plan packages

API - getgroups: Get subgroups

API - getcourses: Get the unit courses

API - getcourseinfo: Get course information

API - gettranscriptinfo: Get transcript information

API - getenglishinfo: Get English level information

Comparison of previous and developed systems

Tables 1 and 2 compare system user requirements, further development and changes, service availability, technological improvements, security, user interface, and user-friendliness.

System evaluation

The developed system is relatively better than the previous system in terms of adaptability, architecture, and development environment. The evaluation highlights technical and development progress and advantages.

Technical specifications

For web applications, the speed of loading in the user's browser or response from the server (not including network speed) is one of the most important indicators, so the loading speed of the previous system and the developed system is compared. On average, loading speed is 2-4 times faster, depending on the content, file size, and contents on the page.

Development indicator

Features related to development: Fast and straightforward to implement changes. New modules can be developed regardless of technology. Services are ready for data exchange with external systems.

Unit testing and error detection are easy because they are done by unit services. User interface and business operations are developed separately.

Multiple front-ends can be used regardless of language and framework. Data acquisition services are ready to create a mobile version of the system. A small library has been designed to develop a UI with a unified theme.

The dependency on the developer has lowered. The downside is that maintaining many projects, writing complex services, and doing integrated testing has become complicated and time-consuming.

6. Conclusion

This article examines web-based software architecture's structure, development, and future trends. It explores ways to create new systems by improving existing systems with good serviceability and easy-to-develop architecture.

As part of the testing and implementation work, the "Student Information System" module of the NUM education management system was selected and completed. Based on the study of the system, the current situation of the system was analyzed, and the business process of the system was researched and designed to convert it to the new architecture. Also, the user interface and business operations sections, which contain the main modules that should be in the student's information system (packaged in 9 categories according to the actions to be performed), have separately developed applications that can manage the access load.

Table 1. Comparison of systems developed by converting 3-tier architecture to microservices architecture -1

Comparison	*Old system*	*Developed system*
Architecture	3-tier architecture, Monolithic	Many small services and SPAs based on microservices architecture.
Technology	ASPX, CSS	Asp.Net (DotNeCore, SignalR), Vue(SPA), Bootstrap.
Programming language	C#	C#, Linq, Javascript
Stability	Even if there is a tiny little error, the whole program may not work because of it	Errors will be indicated only in that part, other parts will work normally.
Structure of development	One whole.	Many modules and small services are separated from Backend and Frontend
Loading status	A response is sent to the user after the entire code is ready /build/	The front end is a static web, so it pulls directly into the browser and gets the data from the API. Faster

Table 2. Comparison of systems developed by converting 3-tier architecture to microservices architecture-2

Comparison	*Old system*	*Developed system*
Database status	MSSQL; Uses a lot of functions and procedures to perform calculations and operations on the database; Data selection operations are /high cost query/ linking multiple tables with large amount of data;	MSSQL; Redis cache server /as addition/; Many small services: Reduced the use of functions and procedures on the database as well as the expensive data access operations for data selection;
Data transfer	XML and direct data access	JSON
User interface	Some modules are not suitable for mobile and small screened devices	Fully flexible /responsive/; Improved appearance, component placement, and color coordination;
User login	SiSi System Session	Another system; Unified Access Environment.
Deployment	Internal server	A combination of internal and cloud

It is concluded that the system created as a result of this work is easy and flexible for further development and modification in terms of basic parameters such as architecture, functions, interfaces, development platform, and data storage.

References

[1] Ian Sommerville, Software Engineering (10th ed). Pearson Education, 2016.

[2] J.T. Yao and Y.Y. Yao, “Web-based information retrieval support systems: building research tools for scientists in the new information age,” Proceedings of the IEEE/WIC International Conference on Web Intelligence, pp. 570-573, 2003.

[3] Y.Y. Yao, “Information retrieval support systems,” Proceedings of FUZZ-IEEE’02, pp. 773-778, 2002.

[4] Samuel O. Adejumo Chinedu E. Mbonu, Samuel M. Alade,Wole M. Olatokun, “Re-engineering a web-based Student Information management System: Development Perspective for the Offic of International Programmes, University of Ibadan”, International Research Journal of Computer Science (IRJCS), vol 8, pp. 226-236, 2021.

[5] D.J. Power and S. Kaparthi, “Building Web-based decision support systems,” Studies in Informatics and Control, 11, pp. 291-302, 2002.

[6] Yao, Jingtao, “Design of Web-based Support Systems,” Springer London, 2012.

[7] Roger S. Pressman, Software Engineering A Practitioner’s Approach (7th ed)., 2010.

[8] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 10th Computing Colombian Conference (10CCC), Bogota, Colombia, 2015, pp. 583-590, doi: 10.1109/ColumbianCC.2015.7333476.

[9] https://rubygarage.org/blog/monolith-soa-microservices-serverless

[10] Sanjeev Sharma and Bernie Coyne, DevOps For Dummies (2nd ed). John Wiley Sons, 2015.

[11] Manar Majthoub, Mahmoud H. Qutqut and Yousra Odeh, “Software Re-engineering: An Overview”, 8th International Conference on Computer Science and Information Technology (CSIT), 11-12 July 2018, Amman.

[12] Ouynzul.D, Study research for architecture of Sisi system, 2021.

[13] Law on the protection of personal information of Mongolia.