iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.45791

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Microservices Enabled E-Commerce Web Application

Dr. Sujata Terdal[1], Prasad R G[2], Vikas Mahajan[3], Vishal S K[4]
[1, 2, 3, 4]*Department of Computer Science and Engineering, Poojya Doddappa Appa College Of Engineering*

Abstract: ***Microservices are emerging as a new computing paradigm which is a suitable complementation of cloud computing. Microservices decompose traditional monolithic applications into a set of fine-grained services which can be independently developed, tested and deployed. In this work we have developed an e-commerce web application using microservices architecture. The drawbacks of monolithic architecture are overcome by microservices in building the e-commerce web application. The web application is developed using MERN stack technology. The services developed in the application are containerized using Docker containers. The containerized services are then managed using Kubernetes which is an orchestration platform. Skaffold is used in the application to sync local code files with the cold files in Docker. The events emitted by services are managed using an event bus called NATS streaming server.***

Keywords: Microservices, Monolithic, E-commerce, Docker, Kubernetes, MERN stack, Skaffold, NATS streaming server

I. INTRODUCTION

Microservices is an architectural approach born from service orientation. Architecture that emphasizes self-management and lightweightness as a means of improvement of Software agility, scalability, and autonomy.

The buying and selling of goods (or services) over the internet is known as e-commerce (or electronic commerce). To perform online marketing and sales activities, as well as control logistics and fulfilment, most businesses with an e-commerce presence use an e-commerce shop and/or an e-commerce platform.

Microservices for e-commerce can provide code within a service without affecting other parts of the platform or requiring a platform restart. It becomes easier to add and test new site features, and it also empowers developers and brand marketers to think more creatively.

Separate services with their own code bases are included in this design. Serverless events are used to connect these distinct services, and APIs are used to connect them to frontend shopping experiences. Overall, this architecture enables the creation of a world-class e-commerce platform by utilising the greatest services available.

Leading tech giants eBay, Spotify, Netflix, and Walmart have all converted their infrastructures to microservices in order to create flexible, global systems and a new working culture that is both easy to join and motivating for engineers.

Applications used to be created as monolithic pieces of software. Restructuring and upgrading the application's operations, connectivity, and security are necessary in order to add new features. Traditional monolithic apps are seldom altered and have lengthy lifecycles since changes to them affect the entire application.

Microservices Architecture solves these issues, and this is the primary reason for adopting the microservices approach to application development. An attempt is made to address a particular issue using a microservice, such as a data search, logging feature, or web service function. By enabling you to alter the code of a single function without having to completely rewrite or even reinstall the whole architecture, this technique increases flexibility. The entire application architecture is more stable as a result of the failure spots' increased independence from one another.

Microservices architecture assists in the development of fault-tolerant applications. Even if one microservice fails, the others are likely to continue to function. This is an important aspect of the microservices architecture. This method allows for more precise scaling decisions. Scaling decisions can be made at a more granular level with microservice designs, allowing for more efficient system optimization and structure. Earlier an ecommerce web application was built using monolithic architecture. The application faced issues in application uptime, scalability and deployment. The monolithic application even went down completely if any small error occurred in the system. These problems are taken care of in this new application developed using microservices architecture.

Microservices architecture is adopted to develop the e-commerce web application where five services are developed individually with a separate database of their own and are isolated.

II. LITERATURE SURVEY

There are essentially two architectures to take into account when building a web application: monolithic and microservices. The best architecture for our e-commerce web application is chosen by analyzing the numerous research papers and articles that have been published. This comparison is based on a number of factors, including application size, deployment, scalability, and other factors.
Due to the monolithic design of the program, which contains all functionality, its modules cannot be run independently. This design is closely coupled, and a single process houses all of your request handling logic [1].

This architecture has a number of drawbacks that become more apparent as the application and team sizes increase [2][3].

1) Changing the program and understanding it can be challenging. Progress is typically slowed as a result.
2) Continuous deployment is difficult since it requires rebuilding and delivering the entire monolith whenever a small portion of the application is changed.
3) Scaling the application can be challenging (a monolithic architecture can only scale horizontally).
4) A long-term investment in a technology stack is necessary.

New software engineering strategies have emerged as a result of the recent increase in popularity of cloud computing [4], including the release of a minimum viable product and the empowerment of small development teams. In response to the demands of the cloud environment, architectural styles have also evolved, giving rise to the "Microservices" architectural style. Microservices are a type of service-based application development architecture. Under this concept, large applications will be divided into little autonomous service units.
The following characteristics of microservices [2].
Smart endpoints and dumb pipes, as well as decentralised governance, decentralised data management, and infrastructure automation, as well as organisation around business capabilities and design for failure are all examples of componentization.

When creating an application using a microservices architecture, the following are the key considerations[5].

- *Loose Coupling:* When services are connected loosely, changing one should not require changing the other.
- *High Cohesion:* We must identify boundaries within our problem area that serve to group together related behaviour and coordinate with other boundaries as loosely as practical.
- *Modules and Services:* By thinking specifically about what models should be shared and not disclosing our internal representations, we avoid one of the common pitfalls that can lead to tight coupling (the opposite of what we want). It is clear that the organisation needs engineers with advanced degrees to implement microservices architecture. If they lack the essential abilities, implementing microservices architecture will be challenging.

After performing several tests such as load testing, concurrency testing, and other scenarios, a comparison of microservices and monolithic architectures [6] was made. After analysing the results of the test scenarios, they concluded that monolithic architecture is better for small applications with fewer users (about 100).
For the sake of simplicity, a monolithic commerce platform like Salesforce Commerce Cloud or even Shopify Plus could have been a smart place to start. However, additional flexibility, security, and speed are required to evolve trade. This is the benefit of a microservices design.[7].
Like Amazon and other large e-commerce companies, you don't have to construct all of your microservices in-house. While developing your distinguishing services in-house is a wonderful idea, you may also employ third-party core commerce services and APIs.
Scalability and fault tolerance are non-functional requirements that are handled by microservice architectures for high availability [8]. Programs must be designed such that they can tolerate the failure of individual services as a result of the use of microservices. Services need to be able to be detected right away and, if feasible, automatically restored because they can fail at any time. Real-time application monitoring, which includes technical metrics (such the number of queries the database gets per second) and business-relevant metrics, is highly valued by microservice applications (such as how many orders per minute are processed). When anything goes wrong, monitoring can act as an early warning system and motivate development teams to look into it. Elastic capacity management, as offered by cloud infrastructures, should enable scalable systems to automatically adapt to shifting workloads. Microservice architectures allow you to dynamically replicate microservices to cloud infrastructures that are under a lot of stress.

By reducing the distance between consumers and services, microservices implemented in an edge computing environment enhance user perception of service quality [9]. The quantity and Service Level Agreement (SLA) of microservices that can be deployed on a single edge server are constrained due to resource limitations on edge servers. To make sure that the service quality that many users perceive always matches their expectations, a technique using add, delete, adjust, and switch to identify an optimised microservice redeployment solution is being developed. This method takes into account user mobility, or how frequent user location changes can cause a significant decline in user-perceived service quality. Three methods are employed in this, and an experiment redeploying microservice systems using Kubernetes in an actual edge-cloud environment is also conducted.

In a cloud environment, businesses may grow their applications as needed. A fresh method for creating cloud applications is microservice design [10]. An automated solution is created and built to help with microservice deployment and continuous integration after the difficulties of microservice deployment and continuous integration are investigated.

Since containers are lighter and easier to administer than conventional virtual machines, they have recently gained popularity for distributing programmes (VMs). The suggested microservices architecture is deployed and tested using a social networking application as a case study utilising Docker containers. The findings show that a variety of parameters, including response time, throughput, and deployment time, are utilised to assess the effectiveness of monolithic and microservice techniques. The results demonstrate that the time and effort needed for deployment and continuous integration are decreased when an application is developed using a microservices approach and deployed using the suggested design. The results show that microservice-based applications outperform monolithic architectures due to their quick response and high throughput.

A container orchestration system called Kubernetes, often known as K8S, is ideal for automating the deployment, scaling, and management of microservice applications. You can manage hundreds or thousands of containers at production scale with this hugely well-liked framework. It is supported by an ardent open-source community and is generally portable[11].

Microservices are a type of discrete unit used in the application design method known as "cloud-native microservices" by developers. Each microservice can normally run independently of the others, but to enable application functioning, the microservices share data and communicate with one another across a network. Because cloud-native apps are built using microservices architectures, microservices are inherently cloud-native [12].

In the fields of software project management, software schedule management, etc., agile techniques have significant applicability. Agile processes' primary goals are to satisfy the customer and produce products more quickly and with fewer errors. Agile software development is iterative and incremental, and requirements can be changed in response to client needs. It facilitates iterative development, time boxing, and adaptable planning [13]. From the above literature survey, we can conclude that, monolithic architecture-based application is well suited for the small business organisations and show better performance compared to microservices. But for large scale applications like Netflix, Amazon, Walmart, Spotify, etc. monolithic architecture is not well suitable and for such drawbacks of monolithic architecture big companies like these are adapting to microservices, microservices perform better in terms of scalability, flexibility, isolation, deployment and many other such factors compared to monolithic architecture.

III. METHODOLOGY

A methodology is a collection of practises, approaches, processes, and regulations. Methodologies often involve a set of actions and activities at each step of the project's life cycle and are precise, strict, and comprehensive.

Phase-based project management is a feature of the Agile approach. It demands constant growth at every step as well as continuing involvement with stakeholders. Once the task is underway, teams go through a cycle of planning, carrying out, and evaluating. Both team members and project stakeholders need to work together.

The following are the phases of agile methodology (Figure 3.1), after the completion of each step in the cycle, the whole cycle is iterated to make the upcoming changes requested by the client.

Figure 3.1 Phases in Agile Methodology

1) *Requirements:* The requirements for the project are gathered. We needed a strong machine with Docker and Kubernetes setups for the project's development. Five services had to be developed for the product.
2) *Design:* After defining the project, collaborate with stakeholders to establish requirements. Use the user flow diagram to show how new features work and how they will integrate with your current system.
3) *Development:* After the team has established the criteria, work may begin. With the intention of providing a usable product, our team started working on the project. Before it was made available, the product underwent a number of testing and development phases.
4) *Testing:* In this phase, the team tested the services developed, fixed the bugs and then released.
5) *Deployment:* The team released the created product for the user's working environment at this step.
6) *Review:* The product reviews were taken and necessary updates were done.

IV. SYSTEM DESIGN

A required authentication service is needed for the customer-to-customer type of e-commerce application. The application handles online ticket sales and purchases. Therefore, a ticketing service that handles creating and choosing tickets for orders is necessary. It is necessary to have an order management system for both created and placed orders. A payment gateway provider is needed to complete the orders that have been created.

Table 1 shows the activity of each actor in the system.

The interaction between the actors and system is explained using Table 1. A buyer can create and manage account, view list of products(tickets), check the price information and buy them if interested. The system locks the ticket for 15 mins when a buyer clicks on the buy ticket button in the time which he has to complete the payment process. A seller can create and manage account, upload tickets which he wants to sell according to his own prices and manage order information.

Table 1 Result of Functional Analysis

Actor	Activity
Buyer and Seller	Establish a profile and control account data
Buyer	View list of tickets Check price information Buy the ticket
Seller	Upload tickets Manage order information

Table 2 Services provided by the system

Services	Description
Authentication	Everything related to user signup/signin/signout
Tickets	Ticket creation/editing
Orders	Order creation/editing
Expiration	Watches for orders to be created and cancels them after time limit
Payments	Handles credit card payments. Cancels orders if payment fails and completes the order if payment succeeds

The services provided by the application are explained using Table 2. There are five services namely Authentication, Tickets, Orders, Expiration, Payments. Authentication service manages user credentials and authenticates if it is a valid user and provides the service of user signup/signin/signout. The Tickets service allows a user to act as a seller and create tickets or delete them according to his requirement. This service also lists the tickets which are available for buying. The order service manages the orders that have been placed. The Expiration service watches orders that have been booked and cancels the order if it is not completed within certain period of time(i.e 15 minutes in this application). The payments service manages everything that is related to payment process. This service authenticates card details, processes the payment and completes the order, if payment is successful. The order is cancelled in case the payment fails.

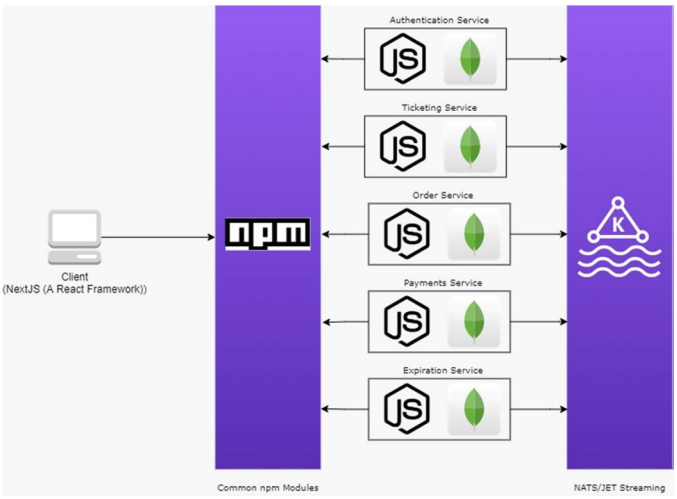


Figure 4.1 Microservices Architecture

The architecture of microservices is shown in Figure 4.1.
Each endpoint has an API Gateway registration. The system was developed utilising HTTP and REST. Every request results in a response that is transmitted as JSON. The communication between services in the system is carried out by NATS streaming server which is based on publisher and subscriber model. Mongo DB is the database that all services utilise.

V. RESULTS AND DISCUSSION

Table 3 The Created Endpoints

Service	Endpoints	Method
Authentication	/api/users/signup /api/users/signin /api/users/signout /api/users/currentuser	POST POST POST GET
Tickets	/api/tickets /api/tickets/:id /api/tickets /api/tickets	GET GET POST PUT
Orders	/api/orders	POST
Payments	/api/payments	POST

The endpoints and their role in the system are shown in Table 3. Each endpoint has an API Gateway registration. REST architecture is used to build web services. There are four HTTP (Hypertext Transfer Protocol) methods included in REST architecture. These are POST, GET, DELETE, and PUT methods. A new resource is created using POST. GET returns a particular resource. A specific resource is deleted with DELETE, and all specific resources are replaced with PUT.

The application is developed and tested. Below are the screenshots of each service functioning as they are intended to.

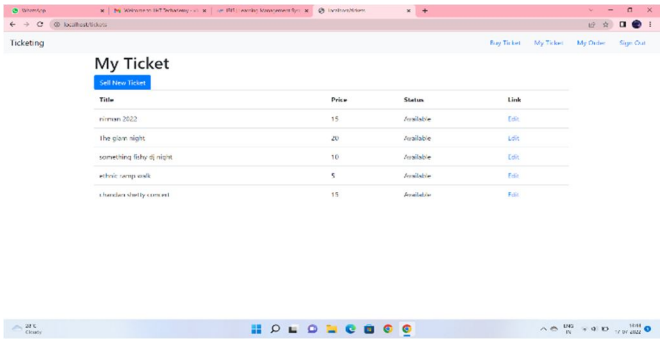


Figure 5.1 Available Tickets

The ticket service allows a registered user to create tickets with prices of his own choice. The created tickets will be made available for other registered users under the buy ticket section. Figure 5.1 shows the tickets available which are created by an user.

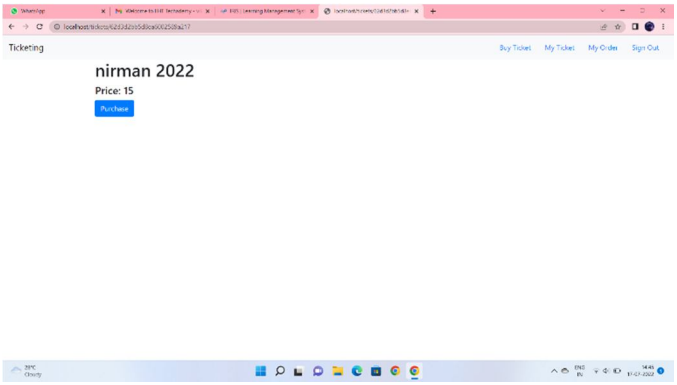


Figure 5.2 Buying a ticket

The customer views various tickets and checks the prices. He then selects a ticket to buy. Figure 5.2 shows the interface of a customer buying a ticket.

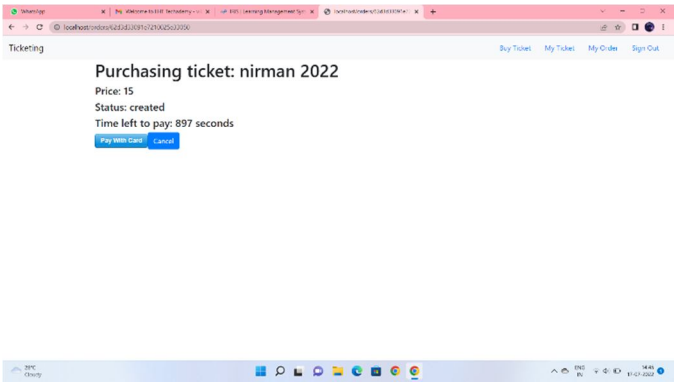


Figure 5.3 Creating an order

Figure 5.3 shows the interface of a user creating an order after clicking the purchase button. He then has to complete the payment process under the limited time period shown.

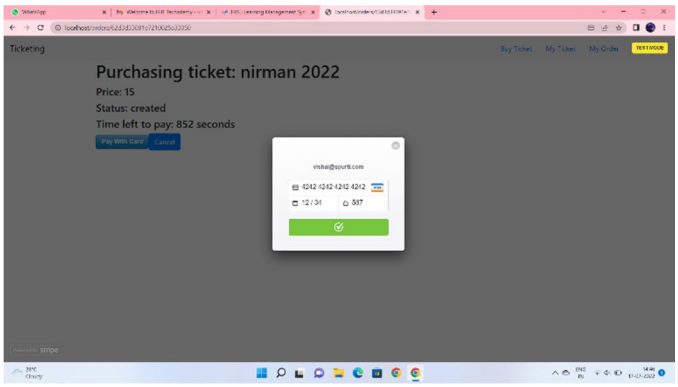


Figure 5.4 Payment process

The payment service comes into action after creating an order. Figure 5.4 shows the payment gateway where user enters the card details and payment is done.

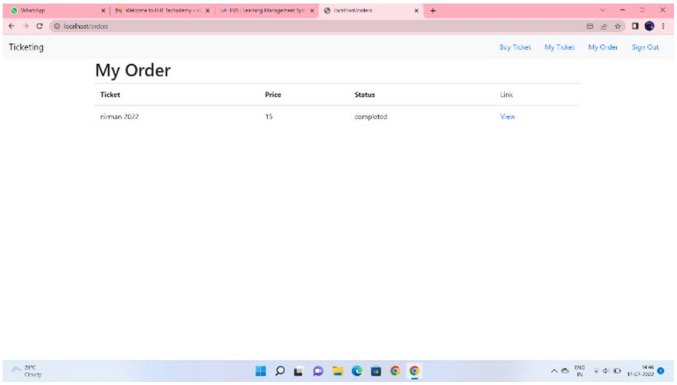


Figure 5. 5 Order successful

On successful payment, one can view the purchased tickets under My Order section as shown in Figure 5.5

VI. CONCLUSION

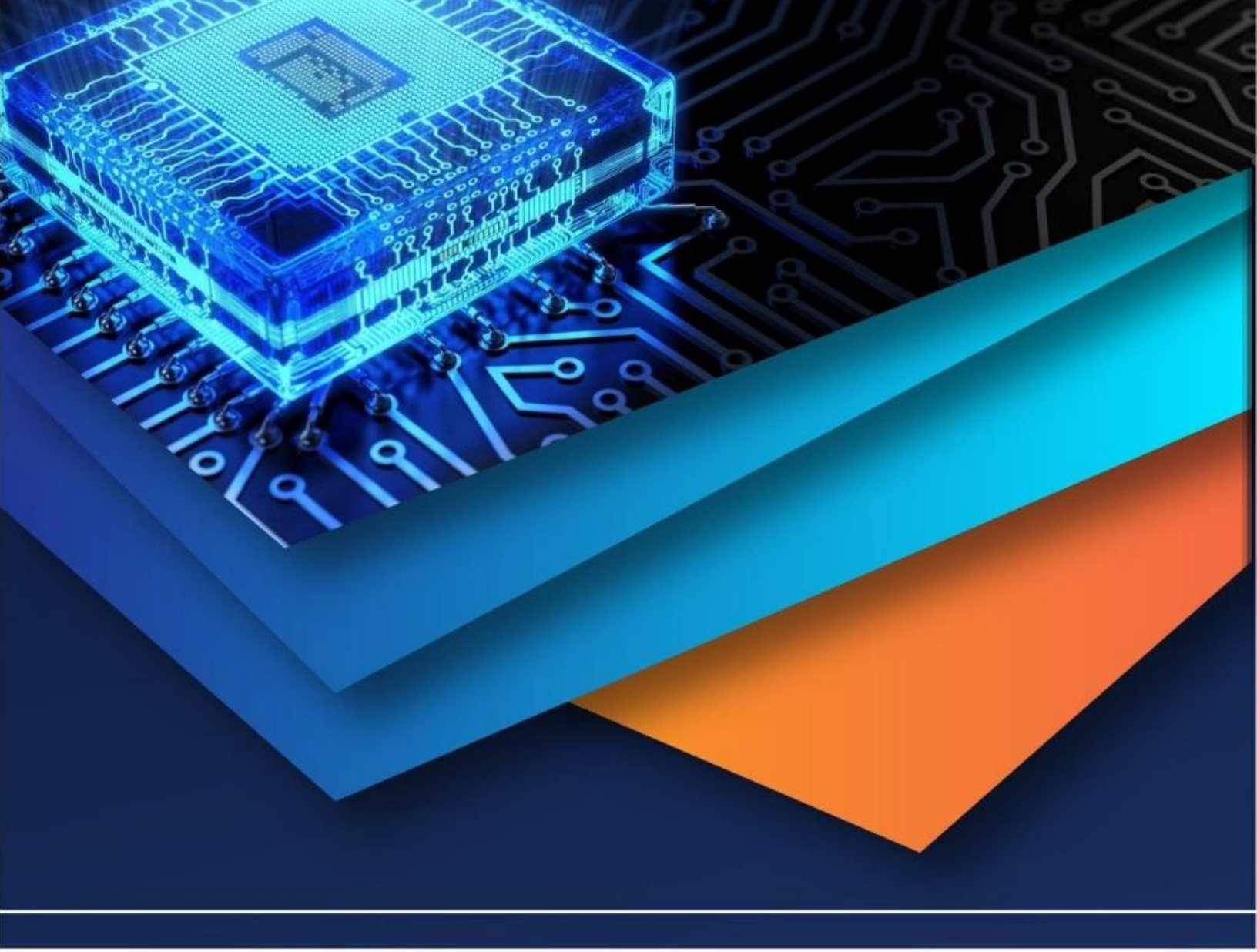
The e-commerce web service can function as predicted based on the findings of the evaluation procedure. System development can be aided by Docker. It supports the development, implementation, and deployment processes for the developers. Docker will create a package including all of the system's dependencies and utilized libraries. It will make the system's deployment and distribution processes simpler. The entire system is divided into numerous smaller services when employing a microservices architecture to build a system. After that, while splitting the service, a number of elements, including service dependencies and service communication, must be taken into account. Microservices need to be carefully planned. The system's development will be challenging in any other case.

Under normal application demand, microservices and monolithic applications can perform similarly. A monolithic application may perform marginally better than a microservices application under a light load of fewer than 100 users. Therefore, monolithic applications are advised for small applications with a small user base. Flexibility and maintainability are two benefits of employing a microservices architecture when creating a system.

REFERENCES

[1] Francisco Ponce, Gastón Márquez , Hernán Astudillo, “Migrating From Monolithic Architecture to Microservices: A Rapid Review”, International Conference of the Chilean Computer Science Society, Sccc 2019.
[2] J. Lewis and M. Fowler, “Microservices. A Definition of This New Architectural Term.” Https://martinfowler.com/articles/microservices.html. [online Accessed]
[3] C. Richardson, “Pattern: Monolithic Architecture.” Https://microservices. Io/patterns/monolithic.html. [online Accessed]
[4] Hulya Vural, Murat Koyuncu , and Sinem Guney, “A Systematic Literature Review on Microservices”, International Conference on Computational Science and Its Applications. Doi:10.1007/978-3-319-62407-5_14
[5] S. Newman, “Building Microservices”, O'reilly Media, 2015

[6] Omar Al-debagy, Peter Martinek, "A Comparative Review of Microservices and Monolithic Architectures", Cinti 2018,18th Ieee International Symposium on Computational Intelligence and Informatics, Budapest, Hungary Doi:10.1109/cinti.2018.8928192, Nov. 21-22, 2018
[7] Https://resources.fabric.inc/blog/answers/ecommerce-microservices-architecture [online Accessed].
[8] Wilhelm Hasselbring, Guido Steinacker, "Microservice Architectures for Scalability, Agility and Reliability in E-commerce", 2017 Ieee International Conference on Software Architecture Workshops, Doi 10.1109/icsaw.2017.11
[9] Zhiying Tu, Xiang He, Xiaofei Xu, Zhongjie Wang, "Re-deploying Microservices in Edge And Cloud Environment for the Optimization of User-perceived Service Quality", International Conference on Service-oriented Computing, Icsoc 2019: Service-oriented Computing Pp 555-560, 22 October 2019
[10] Vindeep Singh, Sateesh K Peddoju, "Container-based Microservice Architecture for Cloud Applications", 2017 International Conference on Computing, Communication and Automation (Iccca), Doi: 10.1109/ccaa.2017.8229914, 5-6 May 2017.
[11] Tesliuk, Anton; Bobkov, Sergey; Ilyin, Viacheslav; Novikov, Alexander Poyda, Alexey Velikhov, Vasily (2019). "Kubernetes Container Orchestration as a Framework for Flexible and Effective Scientific Data Analysis", [IEEE 2019 Ivannikov Ispras Open Conference (ISPRAS) - Moscow, Russia (2019.12.5-2019.12.6)] 2019 Ivannikov Ispras Open Conference(ISPRAS)67–71, doi:10.1109/ispras47671.2019.00016
[12] Vinod Keshaorao Pachghare, "Microservices Architecture for Cloud Computing", Department of Computer Engineering and Information Technology, College of Engineering, Pune, India, Journal of Information Technology and Sciences Volume 2 Issue 1.
[13] Sheetal Sharma, Darothi Sarkar, Divya Gupta, "Agile Processes and Methodologies: A Conceptual Study", International Journal on Computer Science and Engineering (IJCSE). Doi: Vol. 4 No. 05 May 2012



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY