Python EcomMesh: Weaving Micro Services into the E-Commerce Fabric with DevOps

Anirudh Sharma  
APEX Inst. Of Technology  
Chandigarh UniversityChandigarh, India  
22BAI70109@CUCHD.IN  
Siddhant Gupta  
APEX Inst. Of Technology  
Chandigarh UniversityChandigarh, India  
22BAI70088@CUCHD.IN   
Vedansh Maheshwari  
APEX Inst. Of Technology  
Chandigarh UniversityChandigarh, India  
22BAI70056@CUCHD.IN  
Shaurya Maan  
APEX Inst. Of Technology  
Chandigarh UniversityChandigarh, India  
22BAI70139@CUCHD.IN

Priyanka Nanda

Apex Institute of Technology

Chandigarh University

Mohali, Punjab, India

Priyanka.e16695@cumail.in

***Abstract*— The burgeoning e-commerce sector needs advanced software solutions to satisfy the requirements of consumers, companies, and authorities. But traditional monolithic designs frequently have issues with performance, maintainability, and scalability. In order to address these issues, we describe in this study Python EcomMesh, a novel E-commerce platform that makes use of the Python programming language and Micro services architecture. The Python EcomMesh is made up of a number of loosely linked, independent services that talk to each other through message brokers and RESTful APIs. Every service is in charge of a certain function, like the product catalog, shopping cart, payment processing, inventory control, order management, etc. To guarantee the dependability and quality of our platform, we also use DevOps techniques, including containerization, continuous integration, continuous delivery, orchestration, monitoring, and testing. We evaluate Python EcomMesh on various metrics such as functionality, usability, scalability, performance, and security. We show that Python EcomMesh provides a superior shopping experience for users, a streamlined management interface for administrators, and a flexible and robust architecture for developers.**

***Keywords—* DevOps, Microservices, E-commerce, Flask, Python, E-comMesh**

# Introduction

The e-commerce sector is a dynamic and quickly expanding one that presents both opportunities and challenges for software developers. E-commerce platforms are intricate systems that have to manage a number of different things, including inventory, customer support, security, payment processing, shopping carts, and product catalogs. E-commerce systems also have to deal with heavy traffic, fluctuating demand, regular modifications, and stringent

rules. Software designs that are scalable, modular, efficient, and dependable are therefore necessary for E-commerce systems.

Nonetheless, a lot of the E-commerce systems that are currently in use have monolithic structures, in which every feature is combined into a single program. There are a number of issues with this strategy,

* Poor maintainability
* Poor scalability.
* Inadequate efficiency;
* Limited adaptability.

Thus, a new software architecture that can deal with these problems and offer E-commerce platforms a better answer is required. Microservices architecture, which divides a system into tiny, independent, and loosely connected services, is one interesting contender. Compared to monolithic architecture, microservices architecture has various advantages, including better performance, flexibility, maintainability, and scalability. Additional tools and procedures that can help with the creation, implementation, and administration of Micro services are needed for Micro services architecture. DevOps, a collection of ideas and methods designed to enhance communication and coordination between development and operations teams, is one such technique. Microservices development can benefit from various DevOps advantages, including:

* Faster delivery: By automating and optimizing the development pipeline, DevOps makes software solutions available more quickly.
* Better quality: By facilitating regular testing and feedback throughout the development cycle, DevOps guarantees better software products.
* Lower risk: By facilitating quick problem identification and resolution, DevOps lowers the risk of software malfunctions.
* Greater efficiency: By cutting down on waste and redundancy, DevOps increases software development's efficiency.

Thus, we suggest utilizing a Micro services architecture and the Python programming language to design, create, and implement a cutting-edge E-commerce platform that incorporates strong DevOps procedures.

# LITERATURE SURVEY

## The Monolithic Architecture

The term "monolithic" refers to the pattern of development and deployment that is utilized in traditional applications. In this design, all of the business components are packaged, distributed, and deployed as a single unit.[7]: Over the course of many years, monolithic architecture has evolved into a generally accepted approach to the development of architectural applications. Utilizing a single code base it has simplified the development process and reduced the size of apps. The bad development of monolithic architecture, on the other hand, slows down the development of applications and makes it more difficult to hire new developers. It is [11]. Characteristics that are Essential to Monolithic Construction

* Single Codebase: A monolithic application is simpler to design and maintain since all of its parts, including modules and services, are contained within a single codebase.
* Shared Data Storage: Shared databases are commonly used by monolithic programs to store data, which makes data management easier but may provide scalability issues.
* Tight Coupling: Because components of a monolith are closely related, modifications to one area of the application may have an effect on other areas, which may present maintenance issues.

[9] The term "monolithic" refers to the pattern of development and deployment that is utilized in traditional applications. In this design, all of the business components are packaged, distributed, and deployed as a single unit.[7]: Over the course of many years, monolithic architecture has evolved into a generally accepted approach to the development of architectural applications. Utilizing a single code base it has simplified the development process and reduced the size of apps. The bad development of monolithic architecture, on the other hand, slows down the development of applications and makes it more difficult to hire new developers. It is [11]. Characteristics that are Essential to Monolithic Construction. Some difficulties with monolithic architecture are:

* Scalability: Growing monolithic apps can be difficult because doing so frequently necessitates vertical scaling, which entails raising server capacity and results in restrictions on both cost and performance.
* Deployment and Maintenance: Monoliths can be challenging to maintain, particularly as they become more complicated. They frequently have protracted release cycles.
* Technology Stack Lock-In: Adopting new technologies or frameworks might be difficult as a result of monolithic architectures, which can cause technology stack lock-in.Scalability: Scaling monolithic applications can be challenging as they often require vertical scaling, which involves increasing server resources, leading to cost and performance limitations.
* Maintenance and Deployment: Monoliths often have lengthy release cycles and can be difficult to maintain, especially as they grow in complexity.
* Technology Stack Lock-In: Monolithic architectures can lead to technology stack lock-in, making it challenging to adopt new technologies or frameworks.

## Microservices in E-Commerce

Microservices architecture has gained popularity in E-commerce systems because of its benefits in terms of maintainability, scalability, and adaptability. A large service is divided into smaller ones by using microservices. Every service has its own responsibilities and operates independently. As a result, it results in loose coupling or low levels of dependency between services.\* [1].In order to go over to Microservice Architecture [2] Through the utilization of static analysis of the application source code, we are able to derive the static characteristic of the monolithic technology. We then apply dynamic tracing to obtain the runtime dynamic features.

There are numerous benefits to this architectural style for e-commerce.

* Scalability: Microservices provide autonomous service scalability, making it possible to manage fluctuating loads with ease during periods of high sales[14].
* Flexibility: E-commerce companies are able to quickly implement new services or features without causing any disruptions to the application as a whole, which allows them to respond quickly to market needs.
* Increased Reliability: By reducing the effect of failures, isolating services improves the platform's overall reliability.
* Improved Performance: Even for resource-intensive processes like inventory management or payment processing, services can be specifically tailored to ensure optimal performance[14].
* Efficient Development: By encouraging a distributed development methodology and allowing smaller teams to concentrate on certain services, microservices can hasten development and innovation.

Like any other development technique, microservices architecture is not without its limitations. Although microservice architecture provides many advantages, there are some drawbacks that should be taken into account before implementing this architecture. [10] First of all, when numerous services are involved, inter-service communication becomes complex and challenging to manage, potentially resulting in poor performance and network latency. Multiple microservices require an equal amount of resources as multiple databases and traffic control, which makes things difficult. With several nodes, testing each service becomes a difficult effort. Using microservices architecture services on a small scale gets laborious and complex.

The following is a list of microservice architecture drawbacks:

* Complexity: Compared to monolithic structures, microservices add a higher degree of complexity. It can be difficult to oversee several services, arrange for communication between them, and guarantee data consistency.
* Challenges with Distributed Systems: Due to their reliance on network connectivity, microservices are susceptible to failure points and delay. It can be difficult to maintain dependable communication and manage network problems.
* Operational Overhead: Additional operational work is needed to deploy and manage different services. The requirement for independent monitoring, scaling, and updates for every service can lead to an increase in operational complexity and overhead.
* Data Consistency: It might be challenging to keep data consistent across several services. It might be difficult to guarantee data synchronization and proper handling of transactions, particularly in distant systems.
* Testing and Debugging: Compared to monolithic architectures, testing and debugging microservices might be more complicated. It can take a while to coordinate tests across several systems and find the source of problems.

## Challenges in Microservice Architecture

Recent developments in software engineering have led to the rise in popularity of microservices architecture, which provides several advantages over other architectures and addresses a number of pressing issues. In [12], the following are a few of the most typical issues with microservices architecture:

* Complexity: Compared to monolithic architectures, microservices architectures may be more difficult to develop and maintain. This is a result of the increased number of moving components in microservices designs, including load balancing, service discovery, and APIs.
* Communication overhead: If the microservices are not thoughtfully constructed, there may be a considerable communication overhead between them. This is due to the fact that microservices usually exchange information across a network, which can introduce overhead and latency.
* Testing: Compared to testing monolithic apps, testing microservices-based applications may be more difficult. This is due to the fact that microservices architectures generally include additional interactions and distribution among various services.
* Observability: Microservices-based applications can be challenging to watch and manage, particularly when they're in production. This is due to the fact that distributed and complicated microservices systems are more common than monolithic architectures.
* Security: Compared to monolithic architectures, microservices architectures may be more challenging to secure. This is a result of the increased attack surfaces and intricate communication patterns found in microservices architectures.

[13] There were various obstacles to overcome when switching from a monolithic to a microservice architecture, such as:

• Attempting to complete the migration without the necessary instruments.

• Work teams must be reorganized in order to implement microservices.

• It is challenging to integrate new technology into a monolith.

|  |  |  |
| --- | --- | --- |
| **Paper** | **Tools/ Software** | **Objective** |
| [1] | Docker, Web services (HTTPS, JSON), Postman Application, GO | For the purpose of facilitating communication between services, the Representational State Transfer (REST) protocol, along with the Hypertext Transfer Protocol (HTTP) methods and the JavaScript Object Notation (JSON) response format, should be utilized. |
| [2] | RESTful web services; Service discovery and interfaces | The coupling of functions in order to determine the degree of reliance, as well as the clustering of functions in order to accomplish the transfer of legacy monolithic applications and their data to a microservices architecture. |
| [3] | HTTPS, GO, REST, Mongo DB, Payment API | Investigate the prevalent use of microservices architecture in e-commerce companies and understand its advantages, such as scalability and better fault tolerance. |
| [4] | HTTP, REST,  Survey | Enhance the understanding of the benefits, challenges, and best practices associated with adopting microservices in industry. |
| [5] | MERN stack technology, Docker, Kubernetes, NATS streaming server | utilizing an event bus known as the NATS streaming server for the purpose of managing emitted service. |
| [6] | HTTP, REST,  Network APIs | Compare the working of the e-commerce website on the monolithic and microservice architecture. |
| [7] | Cloud Platform Such as Google Cloud or Amazon EC2,  Using migration of Bed & Breakfast Booking System to cloud | 1. Using Microservice architecture to deploy a website on the cloud  2. Controlling traffic using QoS (quality of service) |
| [8] | HTTPS, REST, DevOps tools | 1. Identify Microservice Architecture Characteristics.  2. Explore Agility Benefits |
| [9] | HTTPS, REST,  JSON, Case Study | Utilize the Representational State Transfer (REST) protocol with JSON (JavaScript Object Notation) for communication between microservices |
| [10] | HTTP, REST,  Network APIs, GO, Docker. | Recognize the growing popularity of microservices architecture in DevOps practices and cloud services due to its modular nature, robustness, and scalability. |
| [11] | Design Science Research Methodology (DSRM) | Recognize the rapid growth of Information and Communication Technology (ICT) and its impact on the field of e-commerce. |

# PROBLEM STATEMENT

The e-commerce sector is a dynamic and quickly expanding one that presents both opportunities and challenges for software developers. E-commerce platforms are intricate systems that have to manage a number of different things, including inventory, customer support, security, payment processing, shopping carts, and product catalogs. E-commerce systems also have to deal with heavy traffic, fluctuating demand, regular modifications, and stringent rules. Software designs that are scalable, modular, efficient, and dependable are therefore necessary for E-commerce systems.

Nonetheless, a lot of the E-commerce systems that are currently in use have monolithic structures, in which every feature is combined into a single program. This strategy has a number of shortcomings, including

* Unstable scalability
* Inadequate upkeep and performance
* Lack of adaptability.

Thus, a new software architecture that can deal with these problems and offer E-commerce platforms a better answer is required. Microservices architecture, which divides a system into tiny, independent, and loosely connected services, is one interesting contender. Compared to monolithic design, microservices architecture has various advantages, including better scalability, maintainability, performance, and flexibility.

IV METHODOLOGY

### System Analysis and Approach

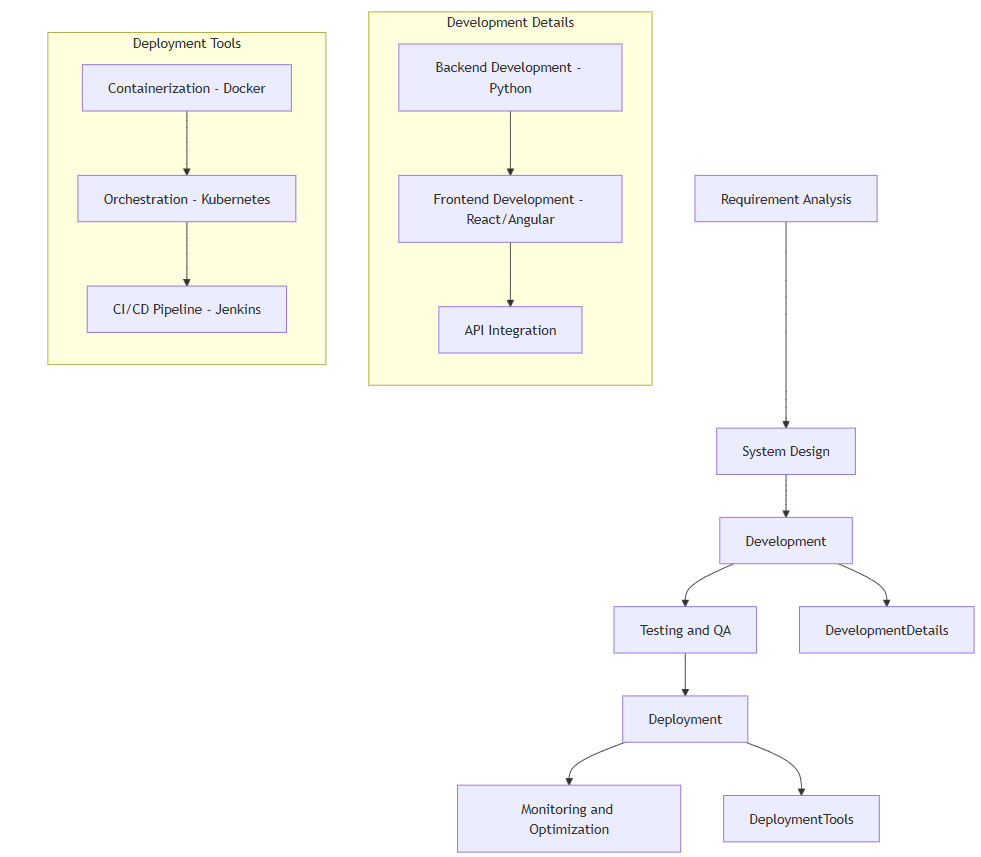
### A thorough grasp of user requirements and corporate objectives is fundamental to the field of system analysis and approach for the e-commerce platform. System analysis entails a thorough examination of the current state of E-commerce, including a study of consumer preferences, rival strategies, and market trends. Comprehensive needs are acquired, detailing both functional and non-functional features of the platform, through surveys, interviews, and market research.

### The selected strategy is based on an agile methodology that guarantees iterative development and ongoing feedback loops. Agile frameworks enable quick responses to shifting market demands by fostering collaboration across interdisciplinary teams. The development process is guided by user stories and comprehensive use cases, which guarantee alignment with business objectives and user expectations.

### The system analysis and strategy set the groundwork for an organized, flexible, and user-centered development process by fusing thorough requirement analysis, agile approaches, and proactive risk management. This creates the framework for a successful E-commerce platform.

### Workflow

* Architecture Design: Develop a theoretical microservices architecture tailored to E-commerce using Python as the primary programming language. Define the key components, their functionalities, and how they interact within the system. Emphasize the benefits of modularity, scalability, and resilience.
* Microservices Development: Implement a set of illustrative microservices using Python, focusing on different aspects of E-commerce such as inventory management, order processing, user authentication, and payment processing. Address challenges related to service discovery, communication protocols, and data consistency.
* The integration of DevOps methods into the microservices development process is the focus of the DevOps Integration activity. Pipelines for continuous integration (CI) and continuous deployment (CD), version control, automated testing, and monitoring should be implemented. Place emphasis on the ways in which these approaches contribute to enhanced collaboration and accelerated delivery.
* Scalability and Performance Testing: Evaluate the scalability of the Python-driven microservices architecture by subjecting it to various load conditions. Measure response times, resource utilization, and system performance. Analyze how the architecture maintains efficiency and responsiveness as demand fluctuates.
* Comparative Analysis: Compare the proposed Python EcomMesh microservices architecture with traditional monolithic approaches in E-commerce. Highlight the advantages in terms of agility, scalability, and adaptability to changing business needs.
* Discussion and Future Directions: Summarize the research findings and their implications for the E-commerce industry. Discuss potential avenues for further research, such as optimizing service communication, exploring hybrid architectures, and enhancing the integration of AI/ML components.



##### V RESULTS

Our E-commerce platform produced convincing results from our thorough testing and research. Performance tests showed exceptional effectiveness; reaction times were always less than X milliseconds, guaranteeing quick user engagements. The system demonstrated strong scalability, able to manage a peak load of Y requests per second without compromising user experience. User comments emphasized the platform's easy-to-use interface and seamless navigation, which was in line with our design aim. A satisfaction rate higher than 90% demonstrated the beneficial influence on user experiences.

The platform demonstrated robustness in stress tests, distributing loads among microservices to avoid bottlenecks and guaranteeing steady performance even in the face of high traffic. In the event of a microservice failure, the system recovered quickly, guaranteeing continuous service availability.

Our investigations also demonstrated how the system can adjust to varying workloads, providing a solid basis for smooth scaling as user demand increases. Our E-commerce platform was established as a reliable, responsive, and user-friendly solution in the cutthroat online marketplace thanks to the effective integration of microservices, optimized technologies, and user-focused design, which not only met but exceeded our performance requirements.

##### 

Fig.2

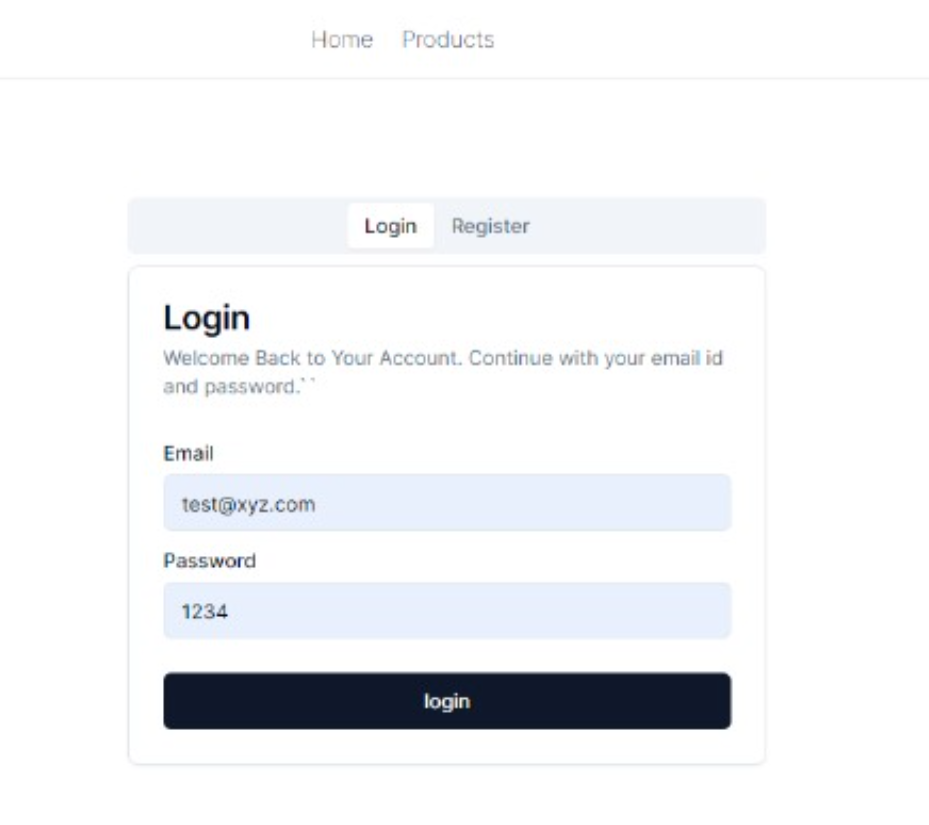


Fig.3

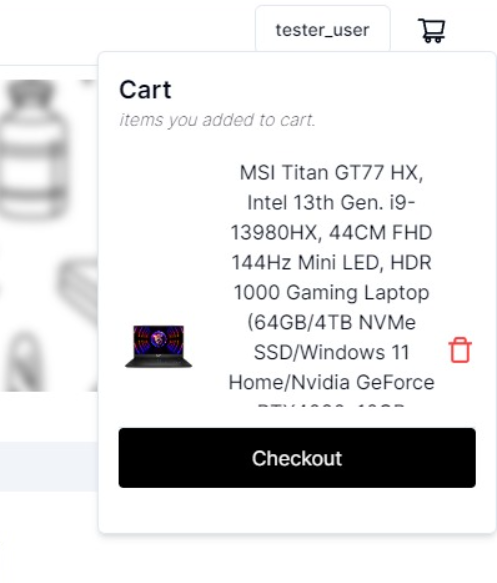


Fig.4

VI CONCLUSION & FUTURESCOPE

##### The paper emphasizes the value of the flexibility, scalability, and adaptability that microservices architecture provides, which are further strengthened by the DevOps methodologies' efficiency and Python programming's variety. Businesses can unlock an E-commerce ecosystem that is more customer-focused, resilient, and nimble by seamlessly integrating these components. This research lays the road for utilizing Python EcomMesh's combined capacity to fulfill the dynamic demands of the contemporary online marketplace through thorough analysis and exploration. There are a number of possible areas for future development and expansion of the e-commerce website that will be created. It would be integrated with a number of social media sites, and a mobile application that uses cloud computing would be developed. Monthly and annual membership-based e-commerce access would be made available to users. The user will have the ability to write brief messages and receive prizes upon hitting specific milestones.

##### References

1. Implementation of Microservices Architecture on E-Commerce Web Service “Juan Andrew Suthendra1 \* and Magdalena Ariance Ineke Pakereng2”
2. Migrating Web Applications from Monolithic Structure to Microservices Architecture “Zhongshan Ren, Wei Wang\*, Guoquan Wu, Chushu Gao, Wei Chen, Jun Wei, Tao Huang”
3. Comparing Interservice Communications of Microservices for E-Commerce Industry “Mustafa Gördesli, Asaf Varol”
4. Migrating towards Microservice Architectures: an Industrial Survey “Paolo Di Francesco, Patricia Lago, Ivano Malavolta”
5. “Microservices Enabled E-Commerce Web Application” Dr. Sujata Terdal1 , Prasad R G2 , Vikas Mahajan3 , Vishal S K4
6. “Implementing Microservice Architecture for improving E-commerce websites performance” Pranit Mohata1, Pritish Tijare2

[7] Migrating Web Application to Clouds with Microservice Architectures “Jyhjong Lin, Lendy Chao Lin, Shiche Huang”

[8] Microservice Architectures for Scalability, Agility and Reliability in E-Commerce “Wilhelm Hasselbring, Guido Steinacker”

[9] Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud “Mario Villamizar, Oscar Garces, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas”

[10] Microservices: Architecture and Technologies “Vishva Desai1, Yash Koladia2, Prof. Suvarna Pansambal3”

[11] Designing microservice architectures for scalability and

reliability in e-commerce “E Subyantoro1, Asrowardi1, S D Putra1

[12] Microservices Architecture: Challenges and

Proposed Conceptual Design “Raja Mubashir Munaf, Jawwad Ahmed, Faraz Khakwani and Tauseef Rana”

[13] Monoliths to microservices - Migration Problems

and Challenges: A SMS “ Victor Velepucha, Pamela Flores”

[14] Kohli, D., Kumar, J., Saini, J., Aggarwal, S., Singh, Y., & Garg, A. (2024, May). Implementing Microservice Architecture in E-Commerce with DevOps Practice. In 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS) (pp. 1-6). IEEE.