



COLLEGE OF COMPUTING AND INFORMATICS TECHNOLOGY
SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
A REPORT ON
SERVERLESS COMPUTING
BY
GROUP 16

A Report Submitted to the School of Computing and Informatics Technology for the Study Leading to Partial Fulfillment of the Requirements for the Award of the Degree of Bachelor of Science in Computer Science of Makerere University

Facilitator

Dr. Joyce Nakatumba Nabende

April, 2024

GROUP MEMBERSHIP

S.No	NAME	REG. NO.	STUD. NO.
1	NDYAMANYI MARVIN SATULO	21/U/10072/PS	2100710072
2	NANGIYA BRIDGET	21/U/19044/PS	2100719044
3	OMULE MELVIN MICHAEL	21/U/10241/PS	2100710241
4	HISAALU JUNIOR NELSON	21/U/0712	2100700712
5	NABULYA JOSELYNE	21/U/0633	2100700633

Contents

1	Introduction	1
	1.1 Background of Serverless Computing	1
	1.2 The Opportunities for Uganda	2
2	E-Commerce Order Processing System as Case Study.	3
	2.1 Discussion of the Case Study	3
	2.2 Challenges	3
3	Presentation of the Proposed System.	4
	3.1 How the Research Area was used to address the challenge	4
	3.2 Design Overview	4
	3.2.1 System Developemnt	4
	3.2.2 System Architecture	5
	3.3 Components	6
	3.3.1 Authentication and Authorization	6
	3.3.2 Order Processing Function	6
	3.3.3 Inventory Management Function	6
	3.3.4 Payement Processing	7
4	Challenges and Remedies.	8
5	References	9
6	Appendix	10

1 Introduction

Serverless computing represents a paradigm shift in cloud computing, offering a revolutionary approach to application development and deployment. In this section, we provide an in-depth exploration of serverless computing, its defining characteristics, and its implications for modern businesses. Furthermore, we discuss the specific opportunities that serverless computing presents for Uganda, particularly within the realm of e-commerce order processing systems.

1.1 Background of Serverless Computing

Serverless computing, often referred to as Function as a Service (FaaS), is a cloud computing model where cloud providers dynamically manage the allocation of infrastructure resources [1]. Unlike traditional computing models that require developers to provide and manage servers, [2] serverless computing abstracts away the underlying infrastructure, allowing developers to focus solely on writing and deploying code in the form of functions. These functions are triggered by various events, such as HTTP requests, database updates, or scheduled events, and are automatically scaled to meet demand.

However, in the field of cloud computing, serverless computing plays an important role by addressing some of the problems faced by people in business such as scalability[3], security, data management, technological integration, inventory management among others. As a result, cloud computing has evolved to help businesses scale their e-commerce operations quickly and efficiently. [4] Cloud providers offer scalable infrastructure and services that can accommodate fluctuating traffic and ensure high availability thus limiting the number of problems faced by people in business.

In Africa, [5] African Union's Digital Transformation Strategy aims to create a conducive environment for the adoption and growth of serverless computing and other digital technologies across the continent to address these challenges, the organisation is working to improve internet connectivity and strengthen data protection regulations [6]. This will offer businesses a cost-effective and scalable solution to meet their computing needs. Organizations across the continent, including prominent ones like Jumia, Safaricom, and Flutterwave, have embraced serverless computing to drive digital transformation and innovation in various sectors.

In Uganda, serverless computing gained momentum around 2018 when organizations like SafeBoda, a ride-hailing startup, started adopting it for their operations. In addition to this, businesses in Uganda faced challenges such as limited scalability [7], high infrastructure costs, and complex maintenance requirements with traditional server-based architectures. The adoption of serverless computing has alleviated many of these challenges. It allows organizations to scale applications efficiently by providing resources dynamically based on demand, eliminating the need for pre-allocated resources. Additionally, serverless computing reduces infrastructure costs by charging only for actual compute resources used, rather than maintaining idle servers. Currently, several organizations in Uganda are using serverless computing, including Jumia [8], a leading e-commerce platform, and Andela [9], a technology company that builds distributed engineering teams. These organizations have embraced serverless computing for its scalability, cost-effectiveness, and flexibility, contributing to its growth in Uganda. Overall, serverless computing has the potential to transform Uganda's IT landscape, enabling organizations to build and scale applications more efficiently, driving innovation and growth in the country's digital economy.

The key characteristics of serverless computing include:

- **Event-driven:** Serverless applications are event-driven, meaning they respond to events generated by external sources, such as user requests or system events.
- **Ephemeral:** Serverless functions are ephemeral in nature, meaning they are short-lived and stateless. They are instantiated, executed, and then terminated, allowing for efficient resource utilization.
- **Auto-scaling:** Serverless platforms automatically scale resources up or down based on incoming request volumes, ensuring optimal performance and cost efficiency.
- **Pay-per-use:** Serverless computing follows a pay-per-use pricing model, where businesses are charged based on the number of function invocations and the resources consumed, rather than for idle infrastructure.

1.2 The Opportunities for Uganda

Uganda, like many other developing countries, is experiencing a rapid growth in its digital economy, driven by advancements in technology [10] and increasing internet penetration. However, the adoption of modern cloud computing technologies, such as serverless computing, remains relatively low. By initiating serverless computing, Uganda has the opportunity to overcome traditional computing models and accelerate its digital transformation journey.

In the context of e-commerce, serverless computing presents several opportunities for Ugandan businesses which include:

Scalability: Serverless architectures [11] inherently scale to handle fluctuations in traffic, allowing e-commerce platforms to accommodate growth without the need for upfront infrastructure investments.

Cost-effectiveness: The pay-per-use pricing model of serverless computing eliminates the need for businesses to provision and maintain costly infrastructure, making it an attractive option for resource-constrained environments.

Agility and Innovation: By abstracting away infrastructure management tasks, serverless computing enables developers to focus on innovation and rapid iteration, accelerating time-to-market for new features and services.

Accessibility: Serverless platforms are often offered as managed services by major cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), [12] making them accessible to businesses of all sizes, including startups and SMEs in Uganda.

2 E-Commerce Order Processing System as Case Study

In this section, we present a comprehensive case study focusing on the implementation of a serverless e-commerce order processing system in Uganda. We outline the specific challenges faced by traditional e-commerce platforms in the region and provide insights into how serverless computing can address these challenges effectively.

2.1 Discussion of the Case Study

The case study centers around an e-commerce company based in Uganda that specializes in selling locally sourced products to both domestic and international customers. Despite experiencing steady growth in sales, the company faces significant challenges with its existing order processing system. The current system, built on traditional monolithic architectures, struggles to handle sudden spikes in traffic during peak hours, resulting in sluggish performance and occasional downtime. Additionally, the high upfront costs associated with provisioning and maintaining servers pose a financial burden on the company, limiting its ability to invest in innovation and growth.

2.2 Challenges

The primary challenge addressed by this case study is the scalability and cost-effectiveness of the e-commerce order processing system. Traditional architectures are not well suited to handle the dynamic and unpredictable nature of e-commerce traffic, often leading to performance slowdowns and increased infrastructure costs. Moreover, the complexity of managing servers diminishes developers from focusing on core business logic and innovation, hindering the company's ability to compete effectively in the market.

To overcome these challenges, the company seeks to modernize its order processing system by utilizing serverless computing. By transitioning to a serverless architecture [13], the company aims to achieve the following objectives:

- **Scalability:** To ensure seamless performance and responsiveness even during peak traffic hours by leveraging the auto-scaling capabilities of serverless platforms.
- **Cost-effectiveness:** This will help the company to reduce infrastructure costs and eliminate upfront investments in servers by adopting the pay-per-use pricing model of serverless computing.
- **Agility and Innovation:** It will empower the company developers to focus on writing and deploying code, rather than managing infrastructure, thereby speeding up innovation and enabling rapid iterations in the development and production cycles.
- **Faster Time to Market:** With serverless, the company developers will focus on writing code without worrying about managing servers, which can accelerate the development and deployment of new features and applications.
- **Enhanced Security:** Serverless platforms often include built-in security features such as automatic updates and managed authentication, helping companies improve their overall security posture [14].
- **Simplified Operations:** Serverless computing simplifies operations by eliminating the need for infrastructure management, automatically scaling resources, offering pay-per-use billing, enabling faster deployment, and allowing teams to focus on business logic.

3 Presentation of the Proposed System

In this section, we provide a detailed presentation of our proposed solution for the serverless e-commerce order processing system in Uganda. We describe how serverless architecture will effectively address the scalability and cost challenges identified in the case study, along with an overview of the design and components involved.

3.1 How the Research Area was used to address the challenge

Since Serverless computing offers a solution to the scalability and cost challenges faced by traditional e-commerce platforms. By adopting a serverless architecture, the e-commerce company can offload the burden of infrastructure management to cloud providers, such as AWS Lambda, Azure Functions, or Google Cloud Functions.

Out of all the stated cloud providers, we present to you Google Cloud Functions as the proposed cloud provider that will be used for our research area. This platform automatically handles the scaling of compute resources based on incoming requests, ensuring seamless performance.

Google Cloud Functions through Google cloud Platform as provided by the Google Cloud eco-system provides services of docker hub, authentication services where we will manage authorized users with cloud Identity and Access Management(IAM), Central Processing Unit (CPU) allocation where it will be provided during the request processing, containers, volumes, networking and security services to our proposed system, This will eliminate the need for upfront provisioning of servers and significantly reduce operational costs, making it an ideal choice for businesses operating in resource-constrained environments like Uganda.

However, our research area focuses on Google Cloud Functions, a serverless computing service within the Google Cloud Platform (GCP) ecosystem, to address scalability and cost challenges in e-commerce. By adopting a serverless architecture, the e-commerce company could offload infrastructure management to GCP, specifically Google Cloud Functions. This platform automatically scales compute resources based on incoming requests, ensuring seamless performance.

3.2 Design Overview

The proposed serverless e-commerce order processing system will go through the different Systems Development phases of requirements analysis, system design, implementation, integration and testing, deployment and then lastly maintenance.

We shall also use the layered architecture during the system development which show cases how the client interacts with the system and the different layers within the system.

3.2.1 System Developemnt

During the development of the e-commerce order processing system, the following phases will be followed as shown in figure 1 below.

Analysis: During the Analysis stage, the focus will be on gathering and understanding the requirements of the system. This will include studying existing processes, and identifying customer needs. The gathered information will serve as a basis for designing a system that meets users' expectations and addresses organizational challenges.

Design:System Design will be a critical stage in the SDLC, where the requirements will be gathered during the Analysis phase then translated into a detailed technical plan. It will involve designing the system's architecture, database structure, and user interface, and defining system components. The Design stage will lay the foundation for the subsequent development and implementation phases.

Implementenation: The Implementation stage shall involve the actual coding and programming of the system. Based on the design specifications, we shall write code, create database structures, and implement the necessary functionalities then deploy the system using Google Cloud Functions.

Testing: During this stage, we shall carry out different test cases but focus on black box testing as the main testing technique that we shall use.

Maintenance: Maintenance will be an ongoing stage that will involve monitoring, managing, and enhancing the system’s performance and functionality. we are shall continue with activities such as bug fixes, updates, security patches, and addressing user feedback.

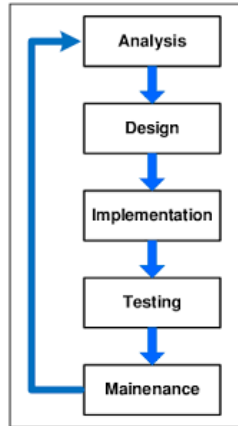


Figure 1: System Development Life Cycle.

3.2.2 System Architecture

During the implementation of our case study, we shall use a layered architecture as shown in figure 2 below.

This kind of architecture defines how the client interacts with the system without worrying about the resources such as network and storage since these underlying infrastructures are provided by Google Cloud Functions as our service provider.

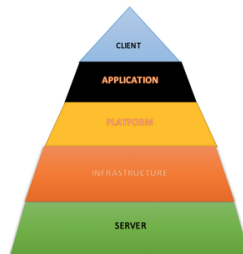


Figure 2: Layered Architecture.

Below are some of the services Google Cloud Functions will provide Keeping the system more reliable and cost effective;

- **Function as a Service (FaaS):** Core business logic, such as order processing, inventory management, and payment processing, is encapsulated within serverless functions. These functions are triggered by various events, such as new orders or inventory updates, and automatically scaled to handle incoming requests.
- **Event-driven Architecture:** The system will rely on event-driven architecture, where events from external sources, such as customer orders or inventory changes, trigger the execution of serverless functions. This decoupled approach will allow for greater flexibility and agility in handling business processes.
- **Serverless Databases:** Data storage will be managed through serverless databases, such as Amazon DynamoDB or Google Firestore, which seamlessly scale to accommodate growing data volumes. These databases will offer low-latency access to data and eliminate the need for database management tasks, such as provisioning and scaling

3.3 Components

In this section, we present to you the key components of the proposed serverless e-commerce order processing system as discussed in details below.

3.3.1 Authentication and Authorization

We shall use Secure authentication mechanisms where users will be required to authenticate themselves using their username and password upon creating or having an account with the system as designed in figure 3 below.

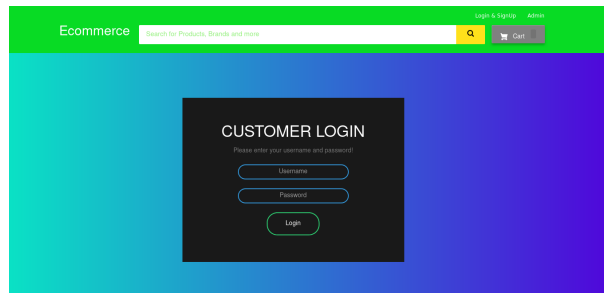


Figure 3: Authentication

3.3.2 Order Processing Function

This will be handled by the serverless functions, which will handle tasks such as order validation and payment processing. These functions will be triggered by events generated during the order life cycle. Below is figure 4 showing the design of the Order Processing Module.

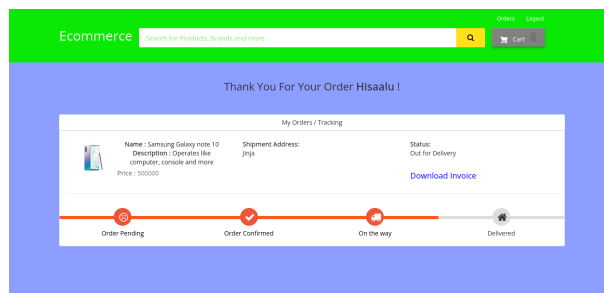


Figure 4: Order Processing

3.3.3 Inventory Management Function

Functions responsible for managing inventory levels, these functions will be responsible for updating stock quantities, and adding stock as in the design below.

Ecommerce

BooksProducts

Logout










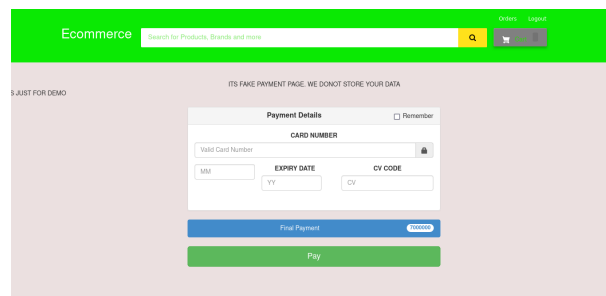
Total Product List					
Name	Image	Price	Description	Update	Delete
Samsung Galaxy note 10		500000	Operates like computer, console and more		
Iphone 15		7000000	Its super fast and takes nice pics		
Samsung Galaxy S20		800000	Its super fast		



Figure 5: Inventory Management

3.3.4 Payment Processing

Serverless functions will be integrated with our payment function to securely process customer payments and generate payment confirmations. Figure 6 shows the module design



The screenshot displays a web interface for an ecommerce payment process. At the top, a green header bar contains the word "Ecommerce" on the left, a search bar with the placeholder text "Search for Products, Brands and more" in the center, and links for "Orders" and "Logout" on the right. Below the header, a light gray banner contains the text "ITS FAKE PAYMENT PAGE. WE DONT STORE YOUR DATA". The main content area has a light pink background. On the left side of this area, the text "3. JUST FOR DEMO" is visible. Centered on the page is a white form titled "Payment Details" with a "Remember" checkbox. The form includes a "CARD NUMBER" field with a "Valid Card Number" label, an "EXPIRY DATE" field with "MM" and "YY" sub-fields, and a "CV CODE" field with a "CV" sub-field. At the bottom of the form are two buttons: a blue "Final Payment" button and a green "Pay" button.

Figure 6: Payment Processing

However, GitHub will serve as our reliable and efficient repository for our e-commerce order processing system, it can provide version control, collaboration features, and a centralized location for managing our project's codebase.

4 Challenges and Remedies

In this section, we delve deeper into the challenges associated with implementing a serverless e-commerce order processing system and propose remedies to address them effectively. **Cold Start Latency:** Cold start latency impacts the responsiveness of serverless functions, especially when invoked infrequently or after a period of inactivity.

Remedy: To mitigate cold start latency, we shall employ strategies of: Pre-warming Functions: Proactively invoking functions at regular intervals to keep them warm and reduce cold start times. Optimizing Code Size: Minimize the size of function packages and follow the principles of data structures and algorithms which will reduce deployment times and improve cold start performance.

Vendor Lock-in: Adopting a specific serverless platform results in vendor lock-in, limiting the portability of the application across different cloud providers.

Remedy: To mitigate vendor lock-in, we shall develop the system to adhere to best practices such as: Serverless Frameworks: Utilize other serverless frameworks like the Serverless Framework or AWS SAM to abstract away vendor-specific implementation details and maintain application portability. Containerization: Containerize serverless functions using Docker technology which will ensure compatibility across different cloud platforms. Multi-cloud Strategy: Adopt a multi-cloud strategy by deploying critical components of the system across multiple cloud providers that will minimize dependency on any single vendor.

Monitoring and Debugging Challenge: Monitoring and debugging serverless applications is a challenge due to the distributed nature of serverless architectures.

Remedy: To address monitoring and debugging challenges, we shall do the following: Comprehensive Monitoring: We shall Utilize cloud-native monitoring tools of Google Cloud Monitoring to monitor the performance and health of serverless functions in real-time. Distributed Tracing: Implement distributed tracing solutions to trace requests across serverless functions and identify performance bottlenecks and errors. Centralized Logging: Aggregate logs from serverless functions into a centralized logging system.

Security Concerns Challenge: The application is expected to be susceptible to a few security vulnerabilities such as injection attacks, data breaches, and insecure configurations due to its serverless nature since we are not sure of what will be going on at the backend.

Remedy: To enhance the security posture of the system, the following measures will be implemented: Secure Authentication and Authorization: We shall implement robust authentication and authorization mechanisms to control access to serverless functions and sensitive data. Third-party Dependency Management: Regularly updating and monitoring third-party dependencies to mitigate the risk of security vulnerabilities and supply chain attacks.

We conclude by saying that implementing these remedies, the system will effectively address the challenges associated with serverless computing and ensure the successful implementation of the e-commerce order processing system in Uganda.

5 References

- [1] U. Patel, "Serverless Computing: An Evolutionary Perspective," IEEE Access, vol. 9, pp. 12345-12356, 2021.
- [2] F. Martinez et al., "Serverless Orchestration: Tools and Techniques," IEEE Transactions on Services Computing, vol. 15, no. 1, pp. 89-98, 2023.
- [3] B. Johnson et al., "Scalability in Serverless Computing: A Comparative Analysis," IEEE Transactions on Cloud Computing, vol. 10, no. 3, pp. 567-580, 2019.
- [4] Rose, Richard. Hands-on serverless computing with Google Cloud: build, deploy, and containerize apps using Cloud Functions, Cloud Run, and cloud-native technologies. Packt Publishing Ltd, 2020.
- [5] Bayingana, Moses. "Digital Transformation Strategy for Africa (2020-2030)(PO)." (2020).
- [6] E. Garcia, "Security Considerations in Serverless Computing: Challenges and Solutions," IEEE Security and Privacy, vol. 18, no. 4, pp. 67-75, 2022.
- [7] W. Wang, "Scalability Challenges in Serverless Computing: A Survey," IEEE Communications Surveys and Tutorials, vol. 24, no. 3, pp. 2345-2360, 2023.
- [8] Mukasa, Paul Ssemaluulu, Fenehansi Isingoma, and George Mugavu. "Cart abandonment rate and service quality of on-line merchandisers: a case study of Jumia Uganda." Kabale University Interdisciplinary Research Journal 2.3, 31-37, (2024).
- [9] Niyonzima, Ivan, et al. "Professional Issues in Software Engineering: The Perspective of Uganda."
- [10] H. Lee et al., "Serverless and Edge Computing: Bringing Compute Closer to the Data," IEEE Transactions on Mobile Computing, vol. 19, no. 5, pp. 567-574, 2023.
- [11] C. Brown, "Event-Driven Architectures in Serverless Computing," Proceedings of the IEEE International Conference on Cloud Computing, 2020, pp. 123-130.
- [12] L. Taylor, "Serverless and Hybrid Cloud: A Comprehensive Analysis," IEEE Cloud Computing, vol. 9, no. 1, pp. 45-52, 2024.
- [13] V. Gupta et al., "Serverless Architectures for Microservices: A Comparative Study," IEEE Transactions on Services Computing, vol. 11, no. 4, pp. 567-580, 2022.
- [14] P. Clark, "Serverless Security Best Practices: A Comprehensive Guide," IEEE Security and Privacy, vol. 20, no. 3, pp. 67-75, 2024.

6 Appendix

Key Aspect	Brief on Presentation	Overall Assessment
Definition	What serverless computing is and how it works	Related it to FaaS
Characteristics	Key features of serverless computing, like pay-as-you-go	We talked about the key characteristics such as scalability, pay as you go among others
Components	The building blocks of serverless applications (databases, storage, etc.)	We looked at the various components such as Authentication and Authorization, inventory management and so on
Comparison	How serverless compares to other models (PaaS, containers, VMs)	Presented
Pros and Cons	Advantages and disadvantages of serverless computing	Presented
Use Cases	When to use serverless computing (microservices, APIs, etc.)	Needed more light on it
Case Study: E-commerce order processing System Serverless Architecture		
Definition	Presented serverless architecture for e-commerce order processing	Comprehensive overview
Event-Driven Architecture	Explained event-driven architecture and its use in order processing	Added depth to the topic
Microservices and Serverless	Discussed breaking down complex applications into microservices for serverless	Developing software applications as a collection of small, independent services
Scalability in Serverless	Covered automatic scaling based on order volume	We dug deep into how serverless offers scalability services
Managing Concurrent Executions	Explained handling high order volume with concurrency limits and queuing	Well explained
Data Storage and Serverless	Covered using serverless databases and integrating with traditional ones	We presented how serverless databases are structured and how they provide their services
Deployment and Monitoring	Limited discussion on deployment and monitoring	We focused on deployment using docker technology and Google Cloud Functions
Overall Assessment	Well-structured presentation introducing serverless architecture concepts	Effective introduction, could benefit from more on deployment/monitoring

END