[Nagarro]-[Amcart]

DAR Document

Nagarro Software Pvt. Ltd.

<Shubham>

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| Version | Date | Author/Contributor | Comments |
| 0.1 | 16-12-2024 | Shubham Agrawal | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc154070767)

[1.1 Objective and scope of document 4](#_Toc154070768)

[2 Requirements at a Glance 4](#_Toc154070769)

[3 Available tools 5](#_Toc154070770)

[3.1 RabbitMQ 5](#_Toc154070771)

[3.2 React 5](#_Toc154070772)

[4 Comparison Analysis 6](#_Toc154070773)

[4.1 Weightage Matrix 6](#_Toc154070774)

[4.2 Comparison Table 6](#_Toc154070775)

[5 Recommendation 6](#_Toc154070776)

[6 Assumptions 7](#_Toc154070777)

[7 Risks 8](#_Toc154070778)

[7.1 References 8](#_Toc154070779)

# Introduction

This is an online shopping website, which can be used to purchase the clothes for both men and women.

## Objective and scope of document

The objective is to design online clothes shopping website for women and men.

Design should be technology agnostic.

We need opt a message broker of Async communication between microservices hosted on Cloud(Azure)

# Requirements at a Glance

Various functional modules that will be implemented by the system will be -

• **Login**: Customer log -in into the system by entering valid user id and password for the

shopping.

• **Registration**: Customer needs to be registered to log-in into the application and buy

products

• **Administrator Page**: The administrator should be able to add, update and remove items.

• **Search or choose product**: Customers should be able to search for products.

• **Product Catalog**: This should display a list of products available on site for shopping.

• **Sale**: The products available in this section will have special discount offers.

• **Stock**: Site should be able to check the availability of products.

• **Price**: System should keep check on displayed price.

**• Add to cart**: Direct functionality of adding product to cart should be possible.

• **Payment**: For customer there are many types of secure billing will be direct bank transfer,

check or bank draft. The security will be provided by the third party like Pay-Pal etc.

• **Order Confirmation**: After the payment or surf the product the customer will logged out and

an email will be sent to him.

# Available tools

* **RabbitMQ**
* **Azure Service Bus**

## RabbitMQ

RabbitMQ is an open-source message-broker software that supports the Advanced Message Queuing Protocol (AMQP) and other protocols through a plug-in architecture. It provides a way for applications to communicate by sending and receiving messages, which can be used to decouple applications and services. Key features include message queuing, routing options, reliability through message durability, scalability, security measures, extensibility, and a management plugin for managing the broker. RabbitMQ is widely used in a variety of applications due to its flexibility, reliability, and scalability.

## Azure Service Bus

Azure Service Bus is a fully managed enterprise message broker that enables applications and services to communicate by sending and receiving messages. It is a cloud-based service, meaning you don't need to host your own hardware to use it. Azure Service Bus supports security protocols such as Shared Access Signatures (SAS), Role-Based Access Control (RBAC), and Managed identities for Azure resources. It supports standard Advanced Message Queuing Protocol (AMQP) 1.0 and HTTP/REST protocols. Azure Service Bus fully integrates with many Microsoft and Azure services.

## Kafka

Apache Kafka is an open-source distributed event streaming platform used for building real-time data pipelines and streaming applications. It is designed to handle high-throughput, fault-tolerant, and scalable messaging. Kafka is based on a distributed commit log architecture, where messages are persisted on disk and replicated across multiple brokers for fault tolerance. Key features include high throughput, low latency, horizontal scalability, fault tolerance, durability, and support for stream processing with Kafka Streams.

# Comparison Analysis

Below is the DAR comparison for the architecture type for various factors.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameters | RabbitMQ | Azure Service Bus | Kafka |
| Infrastructure Cost | RabbitMQ is free to use, but you may need to consider the cost of infrastructure if you are running it on-premises or in the cloud | Azure Service Bus has a pay-as-you-go pricing model with no upfront costs. It is more cost-effective if you expect your message volume to fluctuate | Kafka is open-source and free to use. However, you need to consider the cost of infrastructure if you're running it on-premises or in the cloud |
| Scaling | RabbitMQ can handle many messages and queues, but RabbitMQ does not have automatic scalability and disaster recovery built-in | Azure Service Bus is a cloud-based service that can easily scale as your needs change. Azure Service Bus offers benefits of being a cloud service, such as automatic scalability and disaster recovery | Kafka is horizontally scalable and can handle massive message throughput. It supports automatic partitioning and rebalancing, allowing it to scale seamlessly with growing data volumes |
| Open Source | Yes, it is open source and released under the Mozilla Public License | No, it is a proprietary product from Microsoft | Yes, Kafka is open source and maintained by the Apache Software Foundation |
| Ease of Setup and Management | Can be challenging to deploy in a cloud environment, you may need to purchase a support contract from Rabbit Technologies if you require enterprise-level support | Cloud-based, just a few mouse clicks and you're up and running | Kafka can be more complex to set up and manage compared to cloud-based solutions like Azure Service Bus. |
| Message Retention | RabbitMQ does not have automatic message retention. However, it provides features like dead-lettering and message TTL (Time to Live) that can be used to achieve message retention | Azure Service Bus provides automatic message retention. Messages are retained in the queue until they are processed or until the retention period expires | Kafka retains messages for a configurable retention period. Messages are stored durably on disk, allowing consumers to replay them if needed. |
| Failover Handling | RabbitMQ does not have automatic failover handling. However, it provides features like mirrored queues and quorum queues that can be used to achieve failover handling | Azure Service Bus provides automatic failover handling. It uses the concept of Availability Zones to ensure high availability and data durability | Kafka provides built-in fault tolerance through data replication across multiple brokers. It uses leader-follower replication to ensure high availability and durability of messages. |

## Weightage Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Infrastructure Cost | 25 |
| Scaling | 25 |
| Open Source | 25 |
| Ease of Setup and Management | 25 |

## Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Parameters | RabbitMQ | Azure Service Bus | Kafka |
| Infrastructure Cost | 22 | 24 | 23 |
| Scaling | 20 | 24 | 23 |
| Open Source | 23 | 18 | 23 |
| Ease of Setup and Management | 18 | 23 | 20 |
| Message Retention | 20 | 23 | 23 |
| Failover Handling | 20 | 24 | 23 |

# Recommendation

From the comparison we can see that the **Azure Service Bus** will be best suited to our objective.

# Assumptions

* Application will be highly scalable.
* The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs.
* Security is an essential part of any transaction that takes place over the internet, so all data must be secured and encrypted.

# Risks

* Dependency on Azure Ecosystem.

Appendix

## References

<https://cloudinfrastructureservices.co.uk/rabbitmq-vs-azure-service-bus-whats-the-difference/>

<https://www.trustradius.com/compare-products/azure-service-bus-vs-rabbitmq>

https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/rabbitmq-event-bus-development-test-environment