NAGP-Ecommerce Store

DAR Document

Nagarro Software Pvt. Ltd.

Prasad Bhalerao

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| Version | Date | Author/Contributor | Comments |
| 0.0.1 | 26/01/2025 | Prasad Bhalerao | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc190015958)

[1.1 Objective and scope of document 4](#_Toc190015959)

[2 Requirements at a Glance 5](#_Toc190015960)

[3 Available tools 6](#_Toc190015961)

[3.1 Cosmos DB 6](#_Toc190015962)

[3.1.1 Features 6](#_Toc190015963)

[3.1.2 Pricing 6](#_Toc190015964)

[3.2 Elasticsearch 6](#_Toc190015965)

[3.2.1 Features 6](#_Toc190015966)

[3.2.2 Pricing 7](#_Toc190015967)

[4 Comparison Analysis 7](#_Toc190015968)

[4.1 Weightage Matrix 8](#_Toc190015969)

[4.2 Comparison 8](#_Toc190015970)

[5 Recommendation 9](#_Toc190015971)

[6 Assumptions 10](#_Toc190015972)

[7 Risks 11](#_Toc190015973)

[8 Appendix 12](#_Toc190015974)

[8.1 References 12](#_Toc190015975)

# Introduction

Decision Analysis and Resolution (DAR) is critical to the development of any complex software system. DAR helps developers make informed decisions based on available options, constraints, risks, and trade-offs, ensuring the best possible course of action is taken.

## Objective and scope of document

* Objective:

The purpose of this document is to outline the importance of DAR in developing an ecommerce web application. It describes the key decisions that need to be made and the data that needs to be analyzed to inform decision-making.

* Scope:

This document is relevant to developers, project managers, and stakeholders involved in the development of the ecommerce web application.

* Key Decisions:

Key decisions that need to be made include:

* + Programming languages, frameworks, and libraries to use
  + Hosting infrastructure selection
  + Payment processing and security mechanisms to implement
  + Data Analysis:
* Data analysis is critical to inform decision-making. Relevant data to be analyzed includes:
  + User feedback
  + Market trends
  + Industry standards

# Requirements at a Glance

* Functional Requirements
* Product Catalog: Customers can browse and search products, view details, and add to cart.
* Shopping Cart: Customers can view cart contents, update quantities, and checkout.
* Checkout: Customers can enter shipping/payment information, review order, and complete purchase.
* Order Management: Customers can view order history, track status, and cancel/return items.
* Customer Account Management: Customers can create/manage account, update contact information, and change password.
* Product Management: Merchants can manage products, such as adding, updating, and managing inventory.
* Payment Gateway Integration: Web app is integrated with popular payment gateways to accept multiple forms of payment.
* Non Functional Requirments:
  + Performance: Web app handles high volume of requests with fast response times for customers.
  + Scalability: Web app able to handle increasing traffic and load as customer and order numbers grow.
  + High Availability: Web app always available to customers with minimal downtime.
  + Security: Web app protects customer data and transactions from unauthorized access and hacking attempts.
  + Compliance: Web app complies with relevant industry regulations and standards, e.g. PCI DSS for online payments.
  + Monitoring and Logging: Web app has monitoring and logging capabilities for tracking and troubleshooting.
  + Cost-efficiency: Web app is cost-efficient, utilizing cloud services and optimized resource usage.
  + Mobile Responsiveness: Web app is mobile responsive and accessible from any device.
  + User Experience: Web app has user-friendly interface, easy navigation, and provides a good user experience.

# Available tools

* Cosmos DB
* Elasticsearch

## Cosmos DB

Cosmos DB is a globally distributed, multi-model database service from Microsoft that allows you to elastically and independently scale throughput and storage across any number of geographical regions.

### Features

* Multi-model database service: supports document, key-value, graph, and column-family data models.
* Global distribution: allows you to elastically and independently scale throughput and storage across any number of geographical regions.
* High availability: offers industry-leading SLAs for read and write availability, latency, and consistency.
* Automatic indexing: automatically indexes all data to deliver millisecond query responses, with no schema or index management required.
* Advanced security: provides built-in support for Azure Active Directory, encryption at rest and in transit, and role-based access control (RBAC).
* Real-time analytics: integrates with Azure Stream Analytics, Azure Functions, and other Azure services for real-time data processing and analytics.

### Pricing

Cosmos DB pricing is based on four main factors:

* Request Units (RUs): RUs measure the resources consumed by database operations such as reads, writes, and queries. The more RUs you provision, the higher the cost.
* Data storage: the amount of data stored in Cosmos DB is charged based on the GBs stored per month.
* Data transfer: charges apply for data transfer between regions and for outbound data transfer.
* Other features: charges apply for features such as backups, global distribution, and indexing.

## Elasticsearch

Elasticsearch is a free, open-source search engine and analytics tool that combines a scalable data store with search engine capabilities. It is offered as IaaS offering as well as OnPrem offering.

### Features

* Scalable: Elasticsearch is highly scalable and can store and analyze large amounts of data.
* Fast: Elasticsearch can perform near real-time monitoring and search.
* Flexible: Elasticsearch can be used for a wide variety of search types, including full-text, geo, and scripted queries.

### Pricing

Elasticsearch pricing is based on 3 main factors:

* Subscription Types: Elasticsearch is provided as IaaS offering on multiple clouds providers. It offers 4 Subscription types – Standard, Gold, Platinum and Enterprise.
* Hardware Profile: Based on different workloads, there are 4 Hardware Profiles – Storage optimized, CPU optimized, Vector Search optimized and General purpose.
* Size and Availability zones: The underlying VM is pre configured and user can choose the Hardware as per his needs from 35GB storage, 1GB RAM, Upto 8.5vCPU onwards. User can also choose from 1-3 availability zones.

# Comparison Analysis

While both Elasticsearch and Azure Cosmos DB are NoSQL databases, Elasticsearch is primarily designed for powerful text search and analysis, making it ideal for log analysis and complex queries, while Cosmos DB excels at globally distributed, low-latency data access with flexible data models, suited for applications requiring fast reads and writes across different data types and geographies; essentially, Elasticsearch is better for deep data exploration and analysis, while Cosmos DB is better for real-time, globally distributed data access with high availability.

Key Differences:

1. **Focus**:

* Elasticsearch is primarily a search engine, optimized for full-text search and complex query analysis, whereas Cosmos DB is a multi-model database, supporting various data types like documents, graphs, and key-value pairs with a focus on low latency and global distribution.

1. **Data Model**:

* Elasticsearch uses a document-oriented data model with flexible schema, while Cosmos DB allows for more diverse data models depending on our needs.

1. **Scalability:**

* Both systems can scale horizontally, but Cosmos DB automatically scales based on workload demands, while Elasticsearch requires more manual management for scaling.

1. **Consistency Models:**

* Cosmos DB offers multiple consistency models, allowing you to choose the level of data consistency based on our application needs, whereas Elasticsearch primarily focuses on eventual consistency.

1. **Pricing**:

* Cosmos DB is primarily priced based on "Request Units" (RU/s) which measure the combined read/write operations per second, while Elasticsearch pricing is mostly based on the amount of storage used, with additional charges for features like reporting depending on the plan you choose.
* Generally, Cosmos DB can be more expensive for high-throughput workloads due to its provisioned throughput model, while Elasticsearch may be cheaper for large data storage needs with less intensive query patterns.

1. **Deployment and Maintainability**:

* Cosmos DB is available as a Managed SaaS offering by Azure, whereas for Elasticsearch, it can be self-managed or as a SaaS offering.
* Elasticsearch is generally not considered a suitable primary database and should be used alongside a separate, traditional database as its primary data store, as it is primarily designed for fast search and analysis rather than transactional operations and data integrity that a primary database provides; meaning you typically need a separate database to store our core data before feeding it into Elasticsearch for search purposes.

When to choose Elasticsearch:

* Complex search queries with extensive text analysis needs
* Log analysis and monitoring applications
* Applications requiring advanced data aggregation and filtering capabilities

When to choose Cosmos DB:

* Globally distributed applications requiring low latency reads and writes across different regions
* Applications with diverse data types needing flexible data models
* Scenarios where automatic scaling and high availability are crucial

## Weightage Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Focus | 5 |
| Global Scalability | 5 |
| Consistency | 5 |
| Deployment and Maintainability | 5 |
| Pricing | 5 |
| Data Model | 2 |
| Search Performance | 5 |

## Comparison

|  |  |  |
| --- | --- | --- |
| Feature | Cosmos DB | Elasticsearch |
| Focus | **5** | **2** |
| Global Scalability | **5** | **3** |
| Consistency | **5** | **3** |
| Deployment and Maintainability | **5** | **3** |
| Pricing | **3** | **5** |
| Data Model | **2** | **1** |
| Search Performance | **2** | **5** |

# Recommendation

For our e-commerce application, we recommend:

**Cosmos DB**: As the primary database for product information due to its global distribution, multi-model support, strong consistency, and seamless Azure integration (including search and AI capabilities). Its scalability and replication features are ideal for high availability and low latency.

# Assumptions

1. Product Data: It is assumed that the application will have around 1 Lakh product data, including product images, descriptions, and specifications. This data would be in semi-structured format with different fields varying as per the product categories.
2. Performance: It is assumed that the application will need to support high levels of traffic and user activity. The database will need to be designed to support high throughput and low latency to ensure fast and responsive user experiences.
3. Scalability: It is assumed that the application will need to scale to accommodate growth in user base and data volume. The database will need to support horizontal scalability, allowing for easy expansion to meet increasing demands.

# Risks

**Cosmos DB Risks:**

* **Cost:** Cosmos DB can be expensive, especially if not carefully configured and managed. Its pricing model is based on Request Units (RUs), and inefficient queries or data modelling can lead to unexpectedly high costs. Over-provisioning RUs or not optimizing queries can significantly impact budget.
* **Vendor Lock-in:** While Cosmos DB supports multiple APIs, migrating away from it could still be complex and costly due to data structure and dependencies.
* **Complexity:** Cosmos DB offers many features, which can be complex to understand and configure. Mastering the nuances of indexing, partitioning, and consistency levels requires expertise. Incorrect configuration can lead to performance issues or unexpected behaviour.
* **Performance Tuning:** While generally performant, complex queries or large datasets can require careful performance tuning in Cosmos DB. This often involves optimizing queries, indexing strategies, and partitioning. Not doing so can result in slow response times.

# Appendix

## References

1. Microsoft Cosmos DB documentation: https://docs.microsoft.com/en-us/azure/cosmos-db/
2. Getting started with Cosmos DB: <https://docs.microsoft.com/en-us/azure/cosmos-db/create-sql-api-dotnet>
3. Elasticsearch Search Applications: https://www.elastic.co/enterprise-search/search-applications
4. Data modeling in Cosmos DB: https://docs.microsoft.com/en-us/azure/cosmos-db/modeling-data
5. Querying data in Cosmos DB: https://docs.microsoft.com/en-us/azure/cosmos-db/sql-query-getting-started
6. Performance tuning in Cosmos DB: https://docs.microsoft.com/en-us/azure/cosmos-db/performance-tips
7. Elasticsearch as a primary database: <https://dev.to/er_dward/elasticsearch-as-a-primary-database-15a5#:~:text=Elasticsearch%20is%20not%20a%20transactional,It%20a%20good%20AP%20sytem>.
8. Elasticsearch vs. Microsoft Azure Cosmos DB: <https://db-engines.com/en/system/Elasticsearch%3BMicrosoft+Azure+Cosmos+DB>
9. https://learn.microsoft.com/en-us/azure/cosmos-db/use-cases#:~:text=Azure%20Cosmos%20DB%20offers%2099.99,and%20all%20multi%2Dregion%20accounts