Azure AI Search Case Study

**Prerequisites**

* **Resource Group**: A logical container in Azure that helps manage resources collectively. It supports lifecycle management, role-based access control (RBAC), and cost tracking.
* **Cosmos DB (NoSQL)**: A fully managed, globally distributed NoSQL database service that supports key-value, relational, and vector data models.

**Loading Data into Cosmos DB**

1. **Manual Loading**: Use the Data Explorer in the Azure portal to add new items manually.
2. **Script/API Loading**: Load data programmatically using REST APIs with the database URL and access keys.

**Azure AI Search Service Overview**

Azure AI Search is a standalone service that performs two main functions:

1. **Indexing (Transforming Data into Searchable Format)**

Data Sources: Supported sources include Azure Cosmos DB, Azure Blob Storage, Azure SQL Database, etc., connected via connection strings and credentials.

Index Schema: Define fields with attributes such as:

Key: Unique identifier

Searchable: Enables full-text search

Filterable: Allows filtering

Sortable: Enables sorting

Facetable: Supports faceted navigation

Retrievable: Controls visibility in search resultsSearching (user query -> ranked results)

A screenshot of a computer

AI-generated content may be incorrect.

Indexers (Optional): Automate data ingestion from the source to the index.

Text Processing (Optional): Apply language analyzers and tokenizers.

Vectors (Optional): Enable vector search for semantic and similarity-based queries.

1. **Searching (User Queries → Ranked Results)**

Keyword Search: Use search=\* to match all documents for free-text search.

Filters: Apply OData filters, e.g., experienceYears gt 2.

Facets: Provide quick pivots, e.g., facet=location with counts.

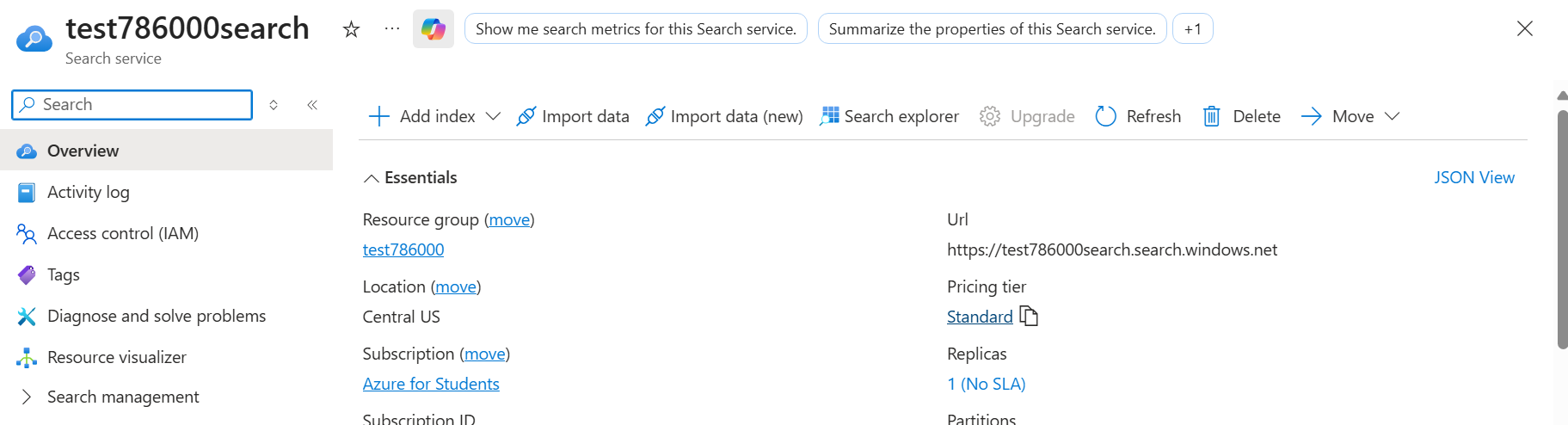
Sorting & Paging: Use parameters like orderby, top, and skip.

Suggestions & Autocomplete: Enable suggesters on fields like name or title.

Semantic & Vector Search: Use semantic rankers and hybrid search (combining keyword and vector search).

1. **Field Access controls:** Implement field-level access controls in your application to ensure sensitive data is excluded from search results and only accessible to authorized users.
2. Implementation:
   1. Set Up Prerequisites: Create necessary Azure resources and decide whether to pull data using indexers or push data manually.
   2. Prepare Data: Ensure data is available and structured in Cosmos DB.
   3. Design the Index: Define the schema based on search requirements.
   4. Create Data Source & Indexer: Connect the data source and configure the indexer.
   5. Test Queries: Use Search Explorer or application integration to validate search functionality.

Screenshots



Azure AI Search (Deployed) Dashboard

Use the URL to communicate or integrate with the application (frontend) or Use search Explorer for testing.

A screenshot of a computer

AI-generated content may be incorrect.

**Costing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tier** | **Storage** | **Max Indexes** | **Price per SU** |
| **Free** | 50 | 3 | 0 |