Prezentare 1

Concept Description:

Objective:

The goal is to create a system that ingests data from a CSV file into Elasticsearch and allows users to perform searches on indexed data using a RESTful API.

Technologies Considered:

Python: Language of choice for scripting.

Elasticsearch: A distributed, RESTful search engine for storing and retrieving data.

CSV Module: To handle CSV file reading and data parsing.

FastAPI: A modern, fast (high-performance) web framework for building APIs with Python.

HTTP Requests Module: For making HTTP requests to Elasticsearch's REST API.

SSL/TLS Configuration: To ensure secure communication between components.

Workflow Overview:

CSV Data Ingestion:

Implement a Python script that reads data from a CSV file, parses it, and structures it into a suitable format for indexing into Elasticsearch.

Elasticsearch Interaction:

Use Python's requests module to communicate with Elasticsearch's REST API:

Create an index with appropriate mappings to define data types for fields.

Index the parsed CSV data into Elasticsearch in batches for optimized performance.

REST API Design:

Utilize FastAPI to create endpoints for:

Ingesting data from the CSV file into Elasticsearch.

Performing search queries on the indexed data.

Handling SSL/TLS configurations for secure communication.

Error Handling and Security:

Implement robust error handling for file operations, Elasticsearch connectivity, and API requests.

Ensure SSL/TLS encryption for secure data transmission.

REST API Design:

Endpoints:

CSV Data Ingestion:

Endpoint: /api/ingest-csv

Method: POST

Description: Receives the CSV file, parses it, and indexes the data into Elasticsearch.

Search Endpoint:

Endpoint: /api/search

Method: GET

Parameters: query (for search query)

Description: Executes a search query against Elasticsearch and returns relevant results.

Request and Response:

CSV Ingestion:

Request: POST request containing the CSV file.

Response: Success message or error details.

Search:

Request: GET request with a query parameter specifying the search term.

Response: JSON object containing search results or an error message.

Security:

Use SSL/TLS for secure communication between the API server and Elasticsearch.

Implement user authentication and authorization mechanisms if needed.

Conclusion:

Implementing this system would involve developing Python scripts for CSV data ingestion, using FastAPI to create RESTful endpoints for interacting with Elasticsearch, and ensuring security measures for data transmission. The use of appropriate error handling and SSL/TLS configurations would enhance the system's reliability and security.

This conceptual design outlines the overall structure and key components needed to build a system from scratch for CSV data ingestion into Elasticsearch and enabling search functionalities via a RESTful API. Further implementation would involve detailed coding, testing, and deployment of the described components to achieve the desired functionality.