AZURE AI INTELLIGENT SEARCH APPLICATION

TCS INDUSTRY PROJECT REPORT

Abstract:

This project implements an intelligent enterprise search system using Azure AI Search to retrieve company policies and knowledge documents efficiently. It enables users to search through large volumes of HR, finance, IT, and company policy data using natural language queries. The application provides fast, accurate, and user-friendly search results through a modern web interface. A secure backend architecture is used to protect sensitive credentials and ensure safe communication with Azure services. This system improves information accessibility and reduces time spent searching for organizational policies.

Problem Statement:

In traditional organizations, company policies and documents are stored in multiple files and folders, making information retrieval slow and inefficient. Employees often struggle to find the exact policy they need, resulting in wasted time and reduced productivity. Keyword-based search systems lack intelligence and do not provide relevant, structured results. Hence, there is a need for an intelligent, centralized, and searchable knowledge system.

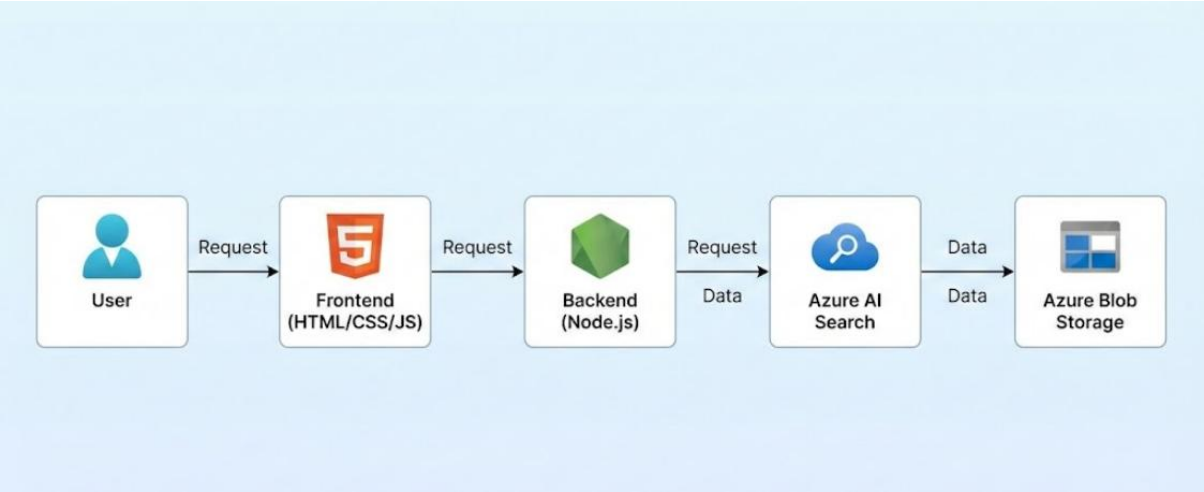
Objectives:

- To build an **intelligent search application** for company policies using Azure AI Search.
- To ensure a **secure backend** where API keys are protected and not exposed to the frontend.
- To provide **fast and accurate data retrieval** with a user-friendly interface.

System Architecture:

The system follows a three-tier architecture:

User → Frontend (HTML/CSS/JS) → Backend (Node.js) → Azure AI Search → Azure Blob Storage



Explanation:

- The user interacts with the web UI.
- The frontend sends search queries to the backend.
- The backend securely communicates with Azure AI Search.
- Azure AI Search retrieves indexed data from Azure Blob Storage and returns results.

Implementation:

1. Frontend

- Built using HTML, CSS, and JavaScript.
- Provides a search bar and displays search results in a structured card format.
- Includes features such as:
    - Title
    - Category
    - Short description
    - “Read Full Policy” toggle
- Communicates only with the backend, not directly with Azure.

2. Backend

- Developed using Node.js and Express.js.
- Stores Azure Search API keys in environment variables for security.
- Handles search requests from the frontend and forwards them to Azure AI Search.

- Returns filtered and formatted results to the frontend.

3. Azure AI Search

- Used to index company documents stored in Blob Storage.
- Provides full-text search capabilities.
- Supports fast and scalable retrieval of policy documents.

4. Azure Blob Storage

- Acts as a data repository for all policy documents.
- Stores structured files such as .txt, .json, and .csv.
- Connected to Azure AI Search using an indexer.

Testing:

Testing was performed to validate the functionality and reliability of the system using:

- **Test_Scenarios.xlsx**
    - Defines overall use cases such as searching policies, viewing full content, and API validation.
- **Test_Cases.xlsx**
    - Contains individual test cases with inputs, expected outputs, and results.

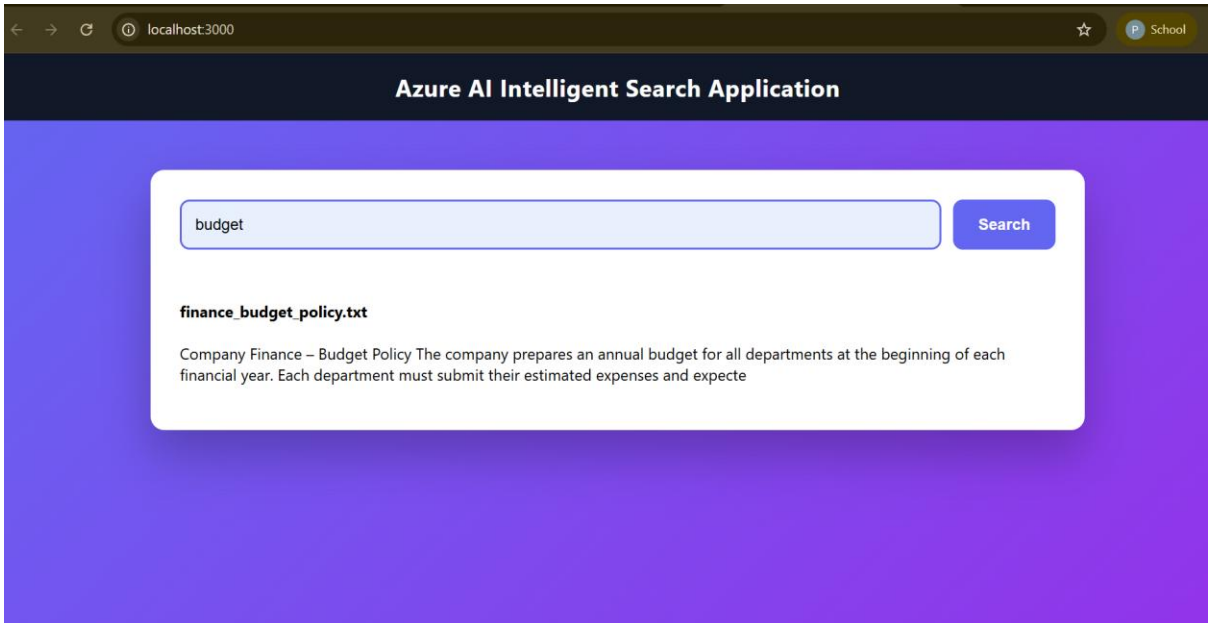
These documents ensure:

- Correct search results
- Proper UI rendering
- Secure backend communication
- Error-free data retrieval

Results:

The application successfully retrieves relevant policy documents based on user queries.
Features verified:

- Accurate keyword matching
- Fast response time
- Structured display of policies
- Secure handling of API keys



Conclusion:

The Azure AI Intelligent Search Application successfully fulfills its objective of creating an efficient and intelligent knowledge retrieval system. It simplifies access to organizational policies and documents, improves productivity, and demonstrates a real-world implementation of Azure AI Search in enterprise applications. The system is secure, scalable, and user-friendly.

Future Enhancements:

1. **Semantic Ranking**
   Improve search relevance by understanding the meaning of queries rather than relying only on keywords.
2. **Authentication**
   Implement user login using Azure AD or JWT to restrict access to authorized employees.
3. **AI Chatbot Integration**
   Add a chatbot that answers user questions directly using Azure OpenAI + Azure AI Search.