Tender Recommendation System is a Flask-Based AI-Powered Tender Search Platform Developed by AI

Mr. Salim Khan[1] Rushikesh Kathe[2], Prathmesh Joshi [3],*Anushka Patil*[4]

**1 Assistant Professor, Department of Information Technology, Matoshri College of Engineering & Research Centre, Eklahare, Maharashtra, India*

**2,3,4 Students, Department of Information Technology, Matoshri College of Engineering & Research Centre, Eklahare, Maharashtra, India*

---***---

***Abstract*—** This project presents Tender recommendation System, an AI-driven web application designed to streamline the process of discovering relevant tender opportunities through natural language search and adaptive feedback mechanisms. Built using a Flask-based backend, the system integrates OpenAI's language models for semantic query understanding and employs Elasticsearch for dense vector similarity matching. It supports advanced filtering by country and publication date while incorporating user feedback to iteratively refine the relevance of search results. The application ensures secure access through token-based authentication and features a responsive user interface for an enhanced user experience. By combining natural language processing, vector search, and feedback-driven learning, Tender On Time significantly improves the accuracy and efficiency of tender retrieval, offering a robust solution for both public and private sector procurement needs.

INTRODUCTION

- In recent years, the volume of publicly available tender data has increased significantly, presenting both opportunities and challenges for organizations seeking procurement opportunities. Traditional keyword-based search systems often fail to capture the semantic intent behind user queries, leading to irrelevant or missed results. This shortcoming is particularly critical in high-stakes domains such as government contracts, infrastructure projects, and international development tenders, where precision and relevance are paramount.
- Tender recommendation System addresses these limitations by introducing an AI-powered tender discovery platform that leverages semantic search and user feedback to enhance result relevance. The system is built upon a robust backend using Flask, integrating OpenAI's embedding models to convert user queries into dense vector representations. These embeddings are processed through Elasticsearch's vector search engine, enabling accurate semantic matching with existing tender descriptions.
- Furthermore, the platform incorporates a feedback mechanism, allowing users to indicate positive or negative relevance for retrieved tenders. This feedback is systematically recorded and used to refine future searches, creating a personalized and adaptive search experience over time.
- The web interface, designed with responsiveness and usability in mind, supports filtering by country and date range, while secure token-based authentication ensures controlled access to the application's endpoints.
- By combining state-of-the-art language models with scalable search and feedback architectures, Tender recommendation System demonstrates a practical application of artificial intelligence in the procurement and tendering domain, ultimately aiming to reduce manual effort and increase the discovery rate of relevant business opportunities.

I. LITERATURE SURVEY

- The task of information retrieval in the context of tendering systems has been historically addressed using traditional keyword-based search algorithms. While these approaches provide a basic level of functionality, they are inherently limited in understanding the context or semantic meaning behind user queries
- [1] Recent advances in Natural Language Processing (NLP) and vector-based information retrieval have significantly improved the accuracy and relevance of search systems across various domains.
- Semantic search systems have emerged as a powerful alternative to traditional retrieval methods. Works such as Huang et al.
- [2] demonstrated the effectiveness of dense vector representations in capturing semantic similarity between user queries and document content. The introduction of transformer-based models like BERT and OpenAI's GPT series has further enhanced the ability of machines to interpret natural language in a human-like manner
- [3]. These models are capable of generating contextual embeddings, which serve as the foundation for vector-based semantic matching. The integration of vector search engines such as Elasticsearch with dense vector support or FAISS has become a common practice in modern semantic search platforms

- [4] These systems support approximate nearest neighbor (ANN) algorithms for scalable and efficient similarity computation in high-dimensional spaces, enabling real-time performance even with large datasets.
- [5] and its modern variants adapt the original query vector based on positive and negative examples, enhancing future result relevance. In recent systems, feedback is used not only to refine search results but also to train adaptive models that learn user preferences over time
- [6] In the procurement and tendering domain, few systems have adopted such advanced retrieval techniques. Most platforms still rely on form-based filters and keyword inputs. The Tender recommendation System platform distinguishes itself by combining OpenAI-powered semantic embeddings, Elasticsearch vector search, and dynamic feedback loops in a unified architecture, offering a novel and efficient solution for intelligent tender discovery.

II. METHODOLGY

The Tender recommendation System platform is designed to provide an intelligent, efficient, and user-centric solution for semantic tender discovery. The system architecture follows a modular pipeline that incorporates natural language understanding, vector similarity search, database management, and user feedback learning. The methodology consists of five core components:

3.1 Data Ingestion and Storage

Tender data is preprocessed and stored in a structured format within a MySQL relational database. The system supports importing tenders from multiple sources, and each entry includes fields such as tender ID, title, description, country code, and publication date. This structured data is mirrored into Elasticsearch with vector representations for high-speed semantic querying.

3.2 Semantic Embedding Generation

To convert natural language queries into machine-understandable formats, OpenAI's GPT-based embedding models are employed. When a user submits a query, it is transformed into a dense vector using OpenAI's embedding API. Similarly, tender descriptions are pre-embedded and stored within Elasticsearch's dense vector index.

This allows the system to perform cosine similarity-based search, comparing the query vector against precomputed tender vectors to return the most semantically relevant results.

3.3 Vector-Based Semantic Search

Elasticsearch serves as the semantic search engine, leveraging its knn_vector field type and efficient ANN (Approximate Nearest Neighbor) algorithms. When a user initiates a search, the query vector is matched against all stored tender vectors. The most similar results are retrieved and sorted based on relevance scores. This approach ensures that even if the query wording does not exactly match the tender titles or descriptions, the results returned are contextually appropriate.

3.4 User Feedback Integration

To personalize and improve the quality of future searches, the system incorporates a feedback loop. Users can label retrieved tenders as "positive" (relevant) or "negative" (irrelevant). This feedback is recorded along with the original query context in the database.

Future queries with similar intent can incorporate this feedback to adjust the ranking of results. The system aggregates prior feedback using weighting mechanisms, allowing for query refinement and continuous learning without explicit retraining of ML models.

3.5 API and Security Architecture

The platform exposes multiple RESTful API endpoints for different use cases, including:

- /tenders_search for standard queries
- /customer_feedback for submitting feedback
- /tenders_search_with_feedback and /tenders_search_by_feedback for enhanced search using past user preferences

All endpoints are secured with token-based authentication, ensuring controlled access and user-specific personalization.

Discussion

The integration of OpenAI embeddings with Elasticsearch vector search represents a significant step forward in the domain of tender discovery. Unlike traditional systems limited by keyword matching, this platform is capable of understanding natural language queries and adapting based on user behavior.

The feedback mechanism allows the system to continuously evolve and refine its results, making it ideal for business users who conduct regular procurement searches. Furthermore, by leveraging scalable infrastructure like Elasticsearch and modularizing components into Flask blueprints and utility modules, the platform maintains both flexibility and performance.

This hybrid approach—combining AI, search engine optimization, and real-time feedback—demonstrates the practical application of modern machine learning techniques in the context of enterprise procurements

Proposed System

The proposed system, Tender Recommendation system , is an AI-powered tender discovery platform designed to overcome the limitations of traditional keyword-based search by implementing semantic search, user feedback integration, and intelligent filtering. The system aims to enhance the accuracy, personalization, and efficiency of tender discovery for users across various industries.

◆ Key Features of the Proposed System:

1. Semantic Search with AI Embeddings

- Utilizes OpenAI's embedding models to convert user queries and tender documents into high-dimensional vector representations.
- Performs vector similarity search using Elasticsearch's KNN (k-nearest neighbor) capability to retrieve tenders that are semantically aligned with the user's intent, rather than matching exact keywords.

2. User Feedback Loop

- Integrates a simple feedback mechanism allowing users to mark retrieved tenders as relevant or not relevant.
- This feedback is stored and used to dynamically update future query results using re-ranking algorithms or fine-tuning of embedding relevance, thus enabling a personalized search experience.

3. Filtering Options

- Supports filters based on:
  - Country
  - Date Range
  - Sector or Category (optional enhancement)
- Ensures that users can narrow down results to their specific interests and geographical regions.

4. Web-Based Frontend

- Developed using responsive web design principles (e.g., HTML, CSS, JavaScript, Bootstrap).
- Interfaces with the backend via RESTful APIs to deliver a smooth and interactive user experience.

5. Backend Architecture

- Built with Python (Flask) as the core backend framework.
- Connects to Elasticsearch for semantic retrieval and integrates with OpenAI's API for embedding generation.
- Uses token-based authentication to secure access to protected endpoints and user-specific features.

6. Scalability & Modularity

- Designed to be modular, allowing easy integration of new models (e.g., newer embedding models or LLMs).
- Scalable with cloud infrastructure support for handling increased load and data.

◆ System Workflow:

1. User logs in and submits a search query.
2. The query is converted into an embedding vector using OpenAI's API.
3. Elasticsearch performs a vector search against stored tender embeddings.
4. The system retrieves and ranks relevant tenders.
5. The user provides feedback on the results.
6. Feedback is used to improve future searches through re-ranking or query expansion.

◆ Benefits:

- Improved relevance through AI-based understanding.
- Time savings by reducing manual browsing.
- Adaptivity with a feedback-driven learning system.
- Secure and user-specific access to sensitive or

Algorithm: Semantic Tender Search with Feedback Loop

Input:

- Q: User query (in natural language)
- F: Optional filters (e.g., country, date range)
- U: User ID (for personalized feedback history)

Output:

- R: Ranked list of relevant tenders

Step 1: Preprocessing

1. Clean and normalize the user query Q
2. Apply language correction or tokenization if needed (optional NLP preprocessing)

Step 2: Embedding Generation

3. Use OpenAI Embedding API to generate a dense vector Vq from the user query Q
→ Vq = OpenAI_Embedding(Q)

Step 3: Semantic Vector Search

4. Query the Elasticsearch index using vector Vq and filter F
→ R_base = Elasticsearch.knn_search(Vq, filters=F)

Step 4: Feedback Integration (if enabled)

5. Retrieve prior feedback from user U
→ Feedback_Pos, Feedback_Neg = DB.get_feedback(U)

6. Compute adjusted query vector (optional, if feedback exists):
→ Vq_adjusted = Adjust_Query_Vector(Vq, Feedback_Pos, Feedback_Neg)

7. If feedback exists:
→ R = Elasticsearch.knn_search(Vq_adjusted, filters=F)
Else:
→ R = R_base

Step 5: Ranking and Output

8. Rank results R based on similarity score and feedback relevance
9. Return top-N results to the user

Feedback Collection (Post Search)

10. User marks tenders as "relevant" or "irrelevant"
11. Save feedback as:
→ DB.save_feedback(U, Query=Q, Tender_ID, Label)

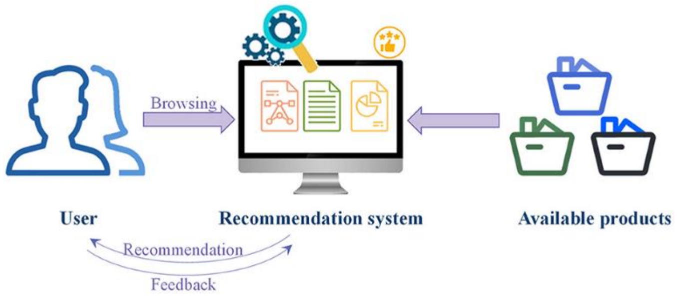
Function: Adjust_Query_Vector()

A heuristic function that combines the original query vector with feedback vectors (positive boosts, negative suppresses).

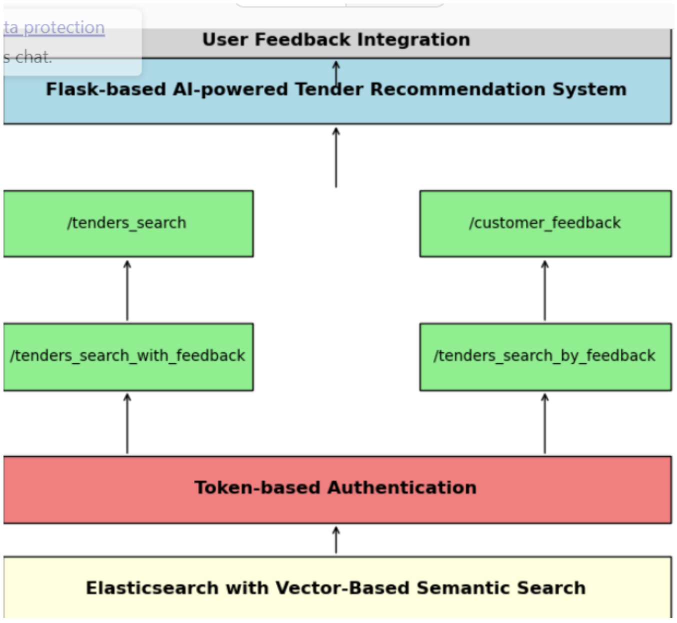
- $Vq\_adjusted = Vq + \alpha * mean(V\_pos) - \beta * mean(V\_neg)$

- α, β = weighting coefficients

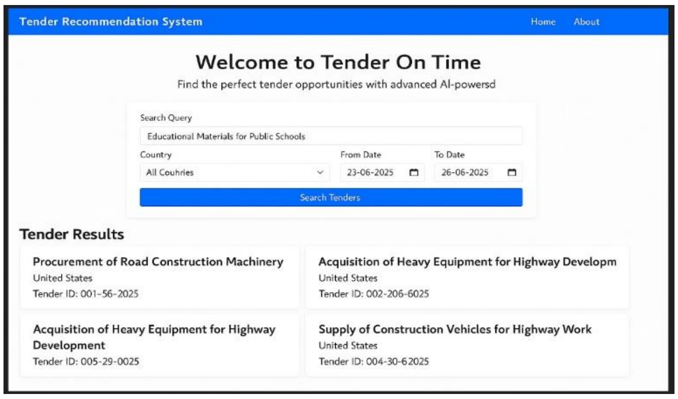
Output



Elasticsearch serves as the semantic search engine, leveraging its knn_vector field type and efficient ANN (Approximate Nearest Neighbor) algorithms. When a user initiates a search, the query vector is matched against all stored tender vectors. The most similar results are retrieved and sorted based on relevance scores. This approach ensures that even if the query wording does not exactly match the tender titles or descriptions, the results returned are contextually appropriate.


ta protection
s chat.
User Feedback Integration
Flask-based AI-powered Tender Recommendation System
/tenders_search
/customer_feedback
/tenders_search_with_feedback
/tenders_search_by_feedback
Token-based Authentication
Elasticsearch with Vector-Based Semantic Search

Screen output:-1 [Tender Search]



Welcome to Tender On Time

Find the perfect tender opportunities with advanced AI-powersd

Search Query: Educational Materials for Public Schools

Country: All Couhries

From Date: 23-06-2025

To Date: 26-06-2025

Tender Results

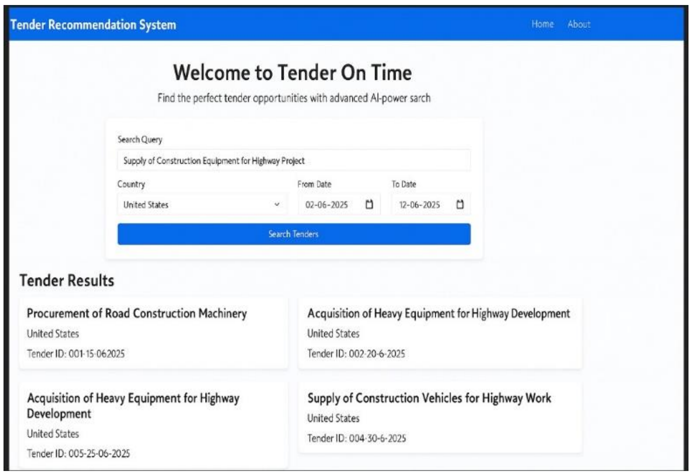
Procurement of Road Construction Machinery
United States
Tender ID: 001-56-2025

Acquisition of Heavy Equipment for Highway Developm
United States
Tender ID: 002-206-6025

Acquisition of Heavy Equipment for Highway Development
Tender ID: 005-29-0025

Supply of Construction Vehicles for Highway Work
United States
Tender ID: 004-30-62025

Screen output:2 [Tender Search]



Welcome to Tender On Time

Find the perfect tender opportunities with advanced AI-power sarch

Search Query: Supply of Construction Equipment for Highway Project

Country: United States

From Date: 02-06-2025

To Date: 12-06-2025

Tender Results

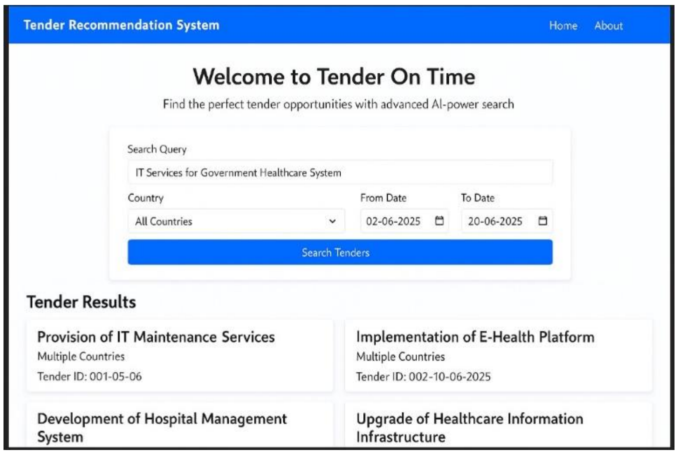
Procurement of Road Construction Machinery
United States
Tender ID: 001-15-062025

Acquisition of Heavy Equipment for Highway Development
United States
Tender ID: 002-20-6-2025

Acquisition of Heavy Equipment for Highway Development
United States
Tender ID: 005-25-06-2025

Supply of Construction Vehicles for Highway Work
United States
Tender ID: 004-30-6-2025

Screen output:-3 [Tender Search]



Welcome to Tender On Time

Find the perfect tender opportunities with advanced AI-power search

Search Query: IT Services for Government Healthcare System

Country: All Countries

From Date: 02-06-2025

To Date: 20-06-2025

Tender Results

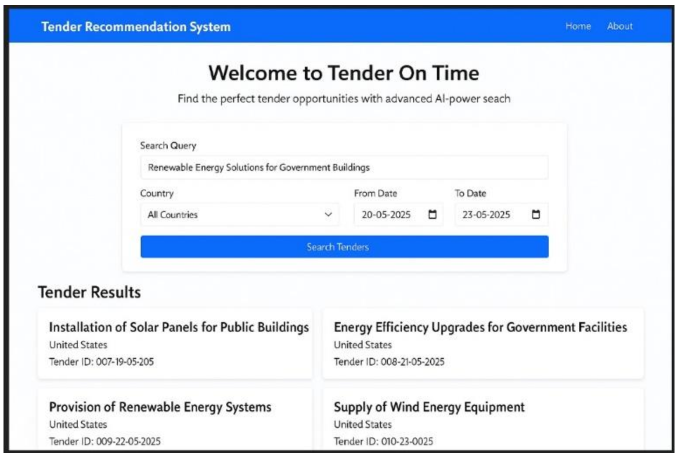
Provision of IT Maintenance Services
Multiple Countries
Tender ID: 001-05-06

Implementation of E-Health Platform
Multiple Countries
Tender ID: 002-10-06-2025

Development of Hospital Management System

Upgrade of Healthcare Information Infrastructure

Screen output:4 [Tender Search]



Tender Recommendation System

Welcome to Tender On Time

Find the perfect tender opportunities with advanced AI-power seach

Search Query: Renewable Energy Solutions for Government Buildings

Country: All Countries

From Date: 20-05-2025

To Date: 23-05-2025

Tender Results

Installation of Solar Panels for Public Buildings
United States
Tender ID: 007-19-05-205

Energy Efficiency Upgrades for Government Facilities
United States
Tender ID: 008-21-05-2025

Provision of Renewable Energy Systems
United States
Tender ID: 009-22-05-2025

Supply of Wind Energy Equipment
United States
Tender ID: 010-23-0025

- User Feedback Integration: Captures relevance feedback to improve future search results.
- Filtering Options: Allows users to refine results by country, date range, and sector.
- Web-Based Frontend: Built with HTML, CSS, JavaScript, and Bootstrap for a responsive UI.
- Backend Architecture: Integrates Flask, OpenAI API,

Conclusion

- Flask-based AI Platform: Hosts the core logic and exposes RESTful API endpoints.
- Token-Based Authentication: Secures access to all endpoints.
- Elasticsearch: Powers semantic search using vector similarity (KNN) and OpenAI embeddings.

REFERENCES

1. Rappaport, T. S. (2002). *Wireless Communications: Principles and Practice*. Prentice Hall.
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781.
3. OpenAI. (2023). *OpenAI API Documentation*. Retrieved from https://platform.openai.com/docs
4. Elasticsearch. (2023). *Elasticsearch: The Definitive Guide*. Retrieved from https://www.elastic.co/guide/
5. Flask. (2023). *Flask Web Development Documentation*. Retrieved from https://flask.palletsprojects.com/
6. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.
7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need.
8. Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *arXiv preprint ar Xiv:1702.08734*. This paper presents FAISS,a library.
9. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., et al. (2020). Language Models Exhibit Few-Shot Learning Capabilities.
10. Gormley, C., & Tong, Z. (2015). Elasticsearch: The Definitive Guide. *O'Reilly Media*. This book provides detailed information on Elasticsearch.
11. Huang, P. S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data.
12. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. *Cambridge University Press*. This book provides information on retrieval systems.