VIRGINIA TECH

Final Presentation
Team 2 - Search and Recommendation

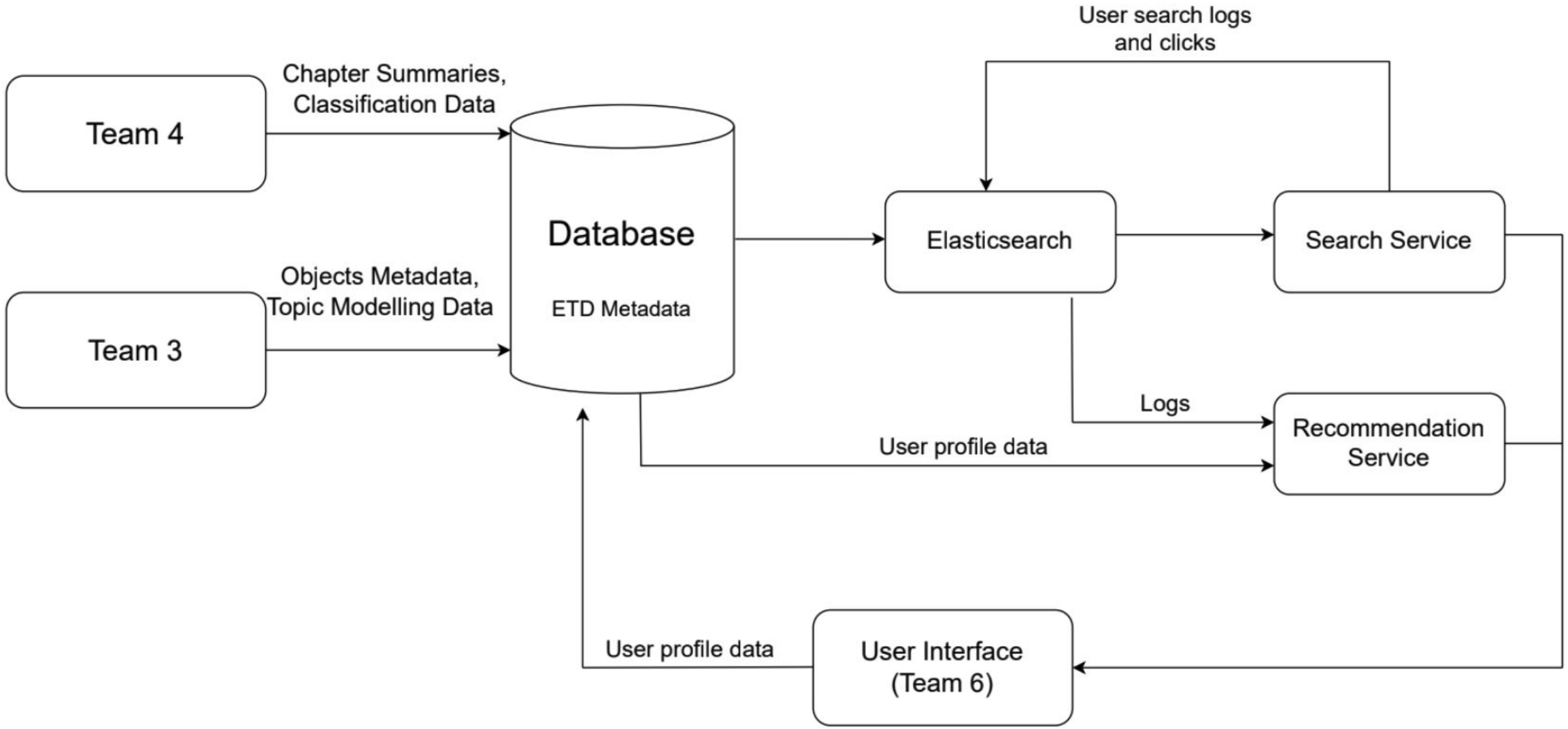
CS5604: Information Storage and Retrieval

Instructor: Edward A. Fox, SME: Satvik Chekuri

Team: Jason Banuelos, Mahesh Maddhuru, Ujjwal Maheshwari, Nikhil Ram, Aseem Khandelwal, Harsha Bhamidipati

November 30, 2023

Virginia Tech, Blacksburg, VA - 24061



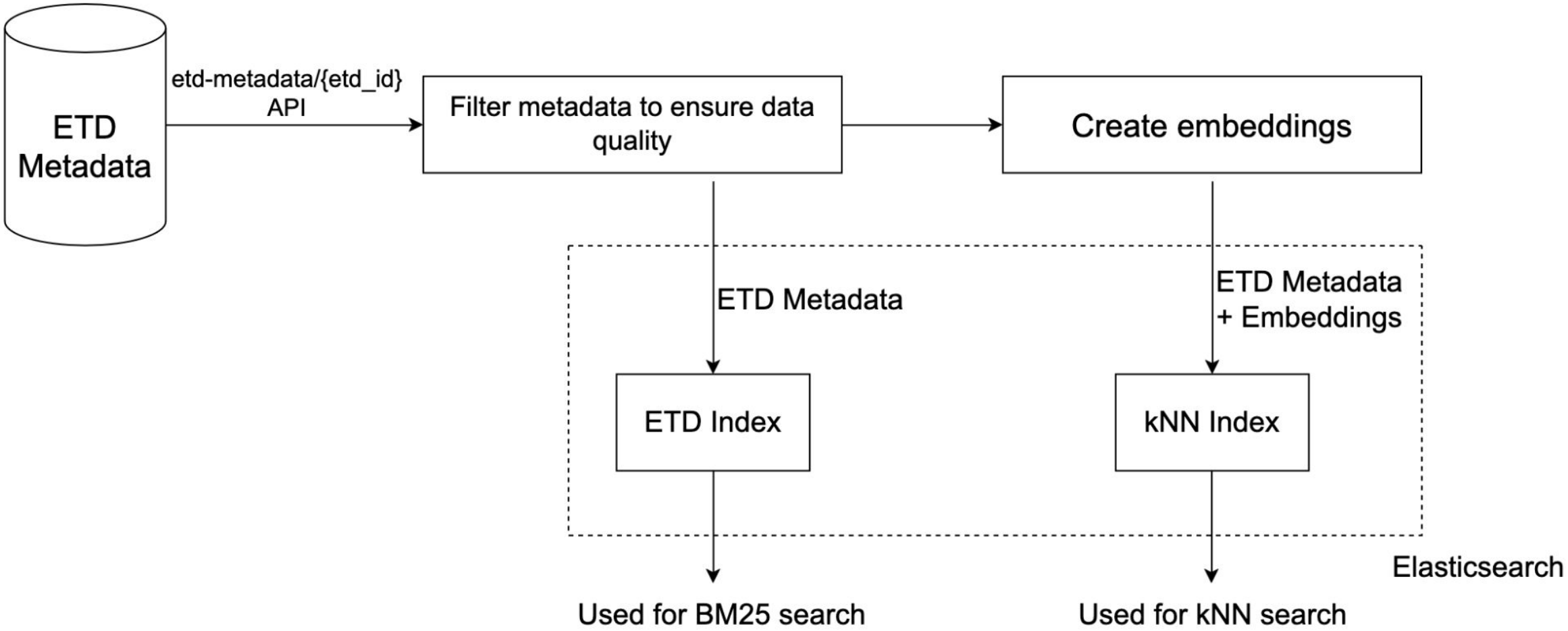
The flow of data between teams.

Search

Summary of search tasks

- Reviewed and changed the index schema for ETD metadata and chapter summaries.
- Fixed the search service APIs to serve frontend interface.
- Implemented code to support creation of embeddings which are used in kNN search.
- Wrote new indexing script for new APIs and database schema.
- Processed 500K ETDs and indexed the filtered metadata.
- Improved search relevance by implementing Reciprocal Rank Fusion (RRF).

ETD indexing flow



New indexing script and embeddings model

- Processed 500K ETDs. 182,689 failed title/abstract validation.
- The entire pipeline takes 45 hours and 50 minutes.
- Network calls and creation of embeddings using E5-large-v2 model are the time consuming processes.
- Tried multithreading the code but didn’t work.
- Solution: Run multiple processes in parallel with different offsets and sizes.

Security incident with Elasticsearch

- The Elasticsearch instance in endeavour was a victim of a ransomware attack. The attack took place twice.
- The attackers deleted all of our existing data and demanded crypto currency in exchange for restoring our data.
- With the help of Team 5, we were able to enable security for the instance.
- Kibana and Elasticsearch APIs are now secure and cannot be accessed without authentication.
- We haven’t noticed any unusual activity since enabling security.

Search Algorithms

- Best Matching 25 (BM25)
- k-Nearest Neighbors (kNN)
- Combined Method - Reciprocal Rank Fusion (RRF)

Best Matching 25 (BM25)

- Default search method for Elasticsearch.
- Utilizes term weighting to assign different weights to terms based on their frequency in an ETD.
- **Scaling to 500K ETDs:**
  - minimal performance impact, response time increased from average 300ms to 3-4 seconds.

k-Nearest Neighbors (kNN)

- kNN is used for semantic-based retrieval.
- ETDs are represented as vectors, and similarity is measured using distance metrics (e.g., cosine similarity).
- **Scaling to 500K ETDs:**
  - Timeout Handling: Extended ElasticSearch timeout from 30 to 180 seconds to prevent timeouts.
  - Adjusted k from 100 to 10, reducing response time significantly from an average of 1.5 minutes to about 16 seconds.
  - This came with a potential trade-off with accuracy, though not thoroughly assessed.

Combined Method - Reciprocal Rank Fusion

- RRF combines rankings from different algorithms by considering the reciprocal ranks of documents.
- Utilized Reciprocal Rank Fusion to integrate BM25 and kNN results.
- Combining different methods helps diversify search results and amplify the ETDs that are consistently ranked higher by all methods.
- **Scaling to 500K ETDs:**
  - Average response time = ~kNN response time

Experimenter

What are experiments?

Experiments in our context are specialized search trials designed to enhance and evaluate the effectiveness of information retrieval from Electronic Theses and Dissertations (ETDs) for an user with *Experimenter* role.

- Search experiments allow experimenters to index custom vectors for each ETD, enabling a hybrid search approach that combines k-Nearest Neighbors (kNN) with traditional keyword search.
- The goal is to analyze and improve the relevance and accuracy of search results by leveraging advanced text embedding models.
- An Experimenter can Create, Run and Delete the experiments based on his liking.
- Search on Collection: A method enabling hybrid searches on indexed ETDs to create and experiment with specific collections, refining retrieval techniques.

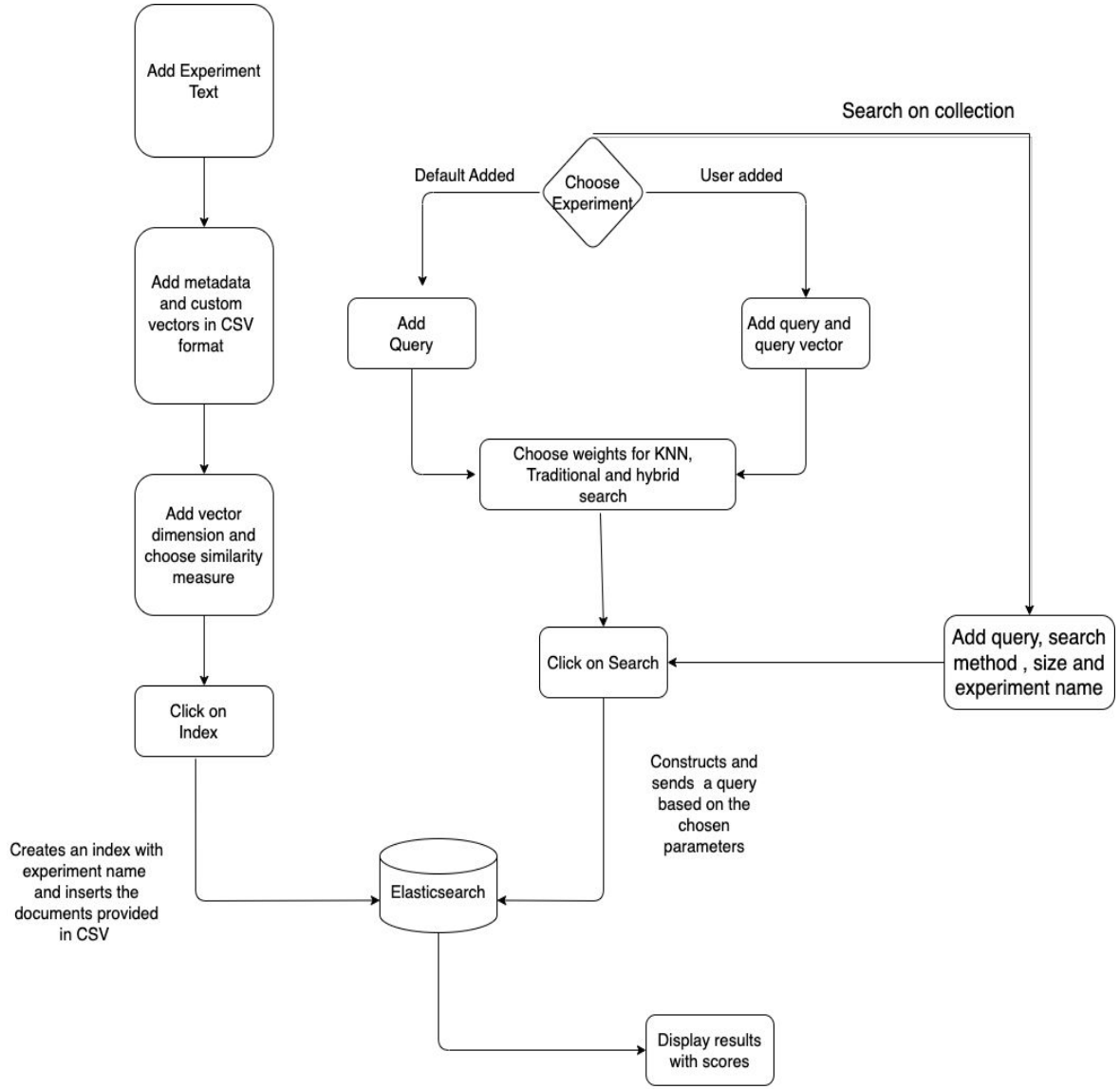
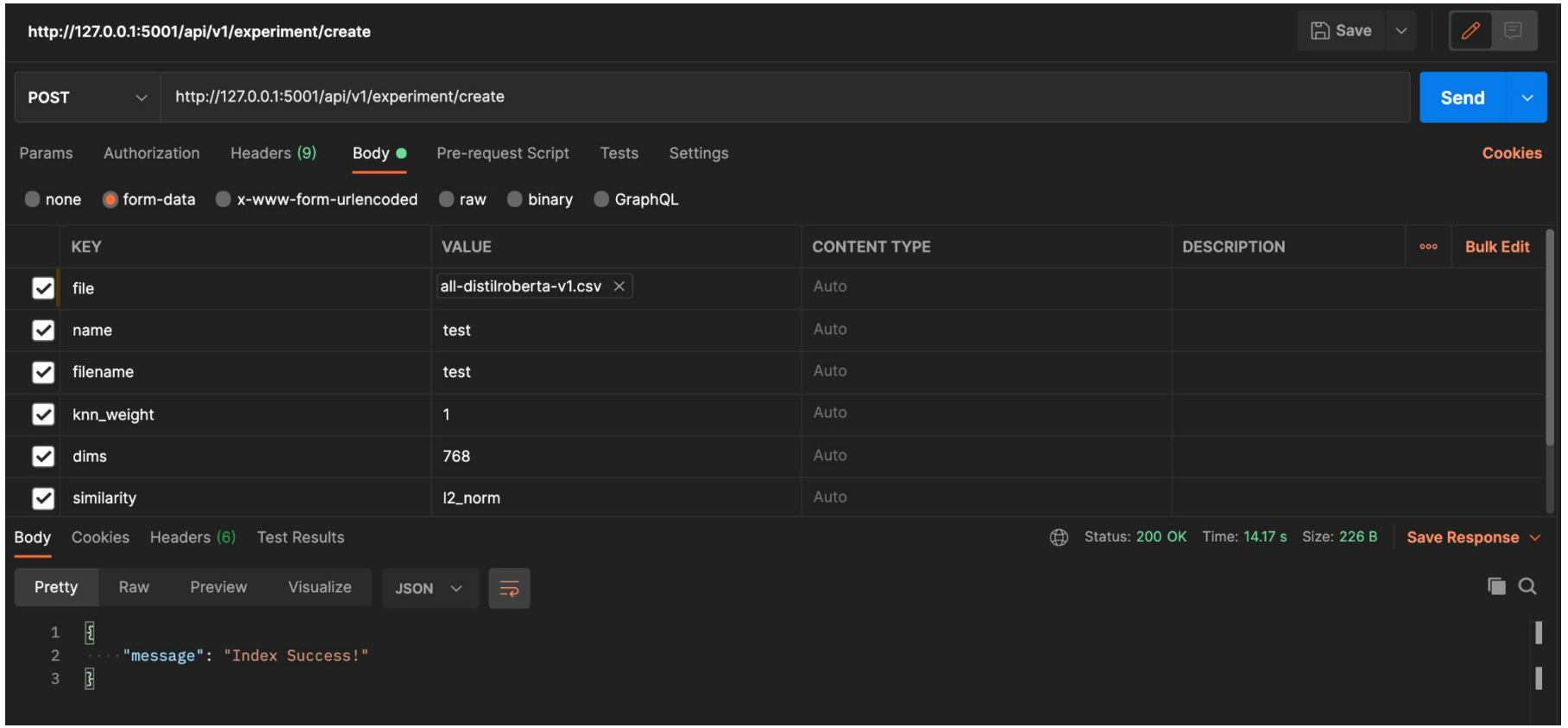


Fig. Experiments Flow

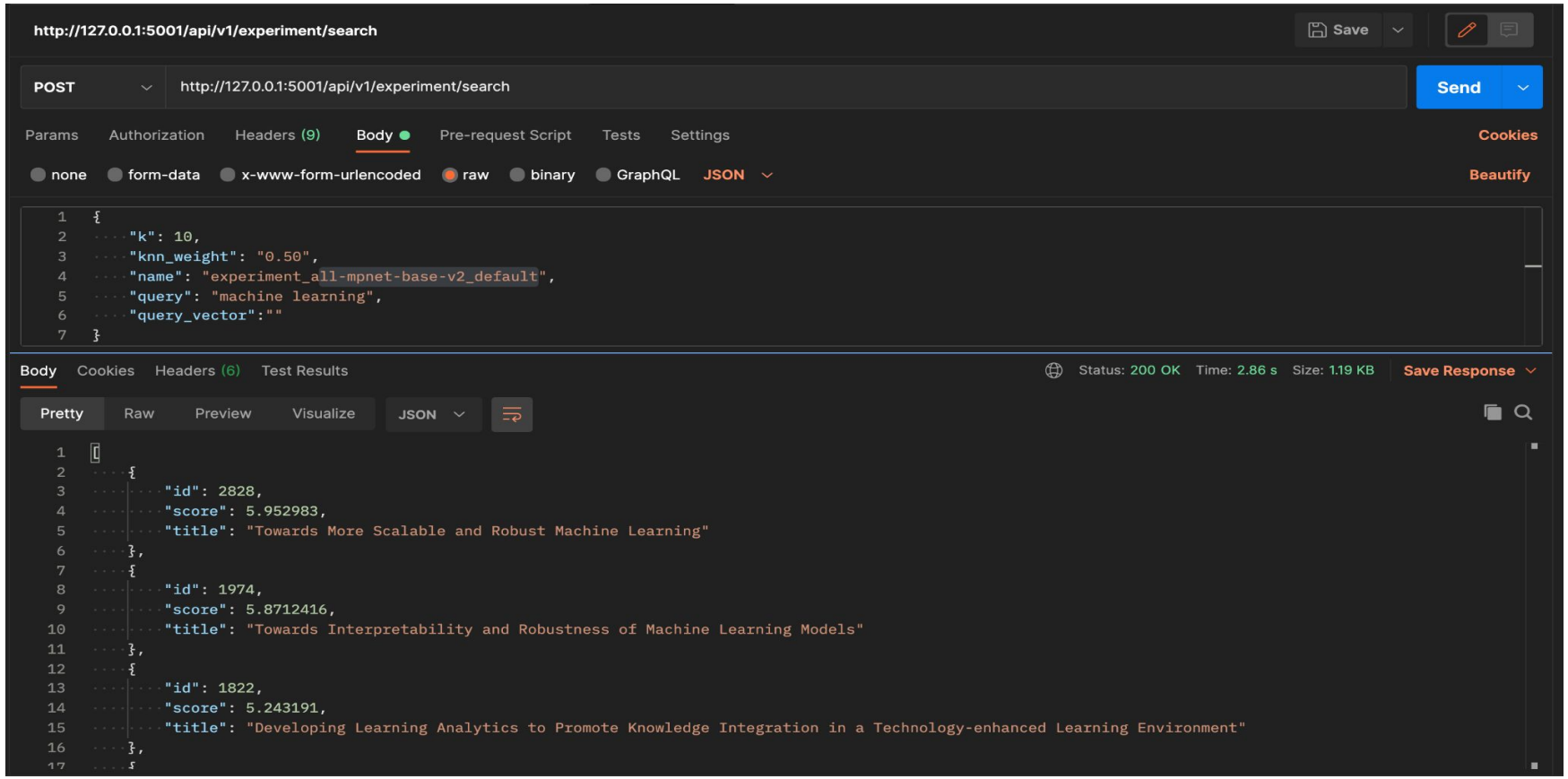
VIRGINIA TECH

- Creating an experiment



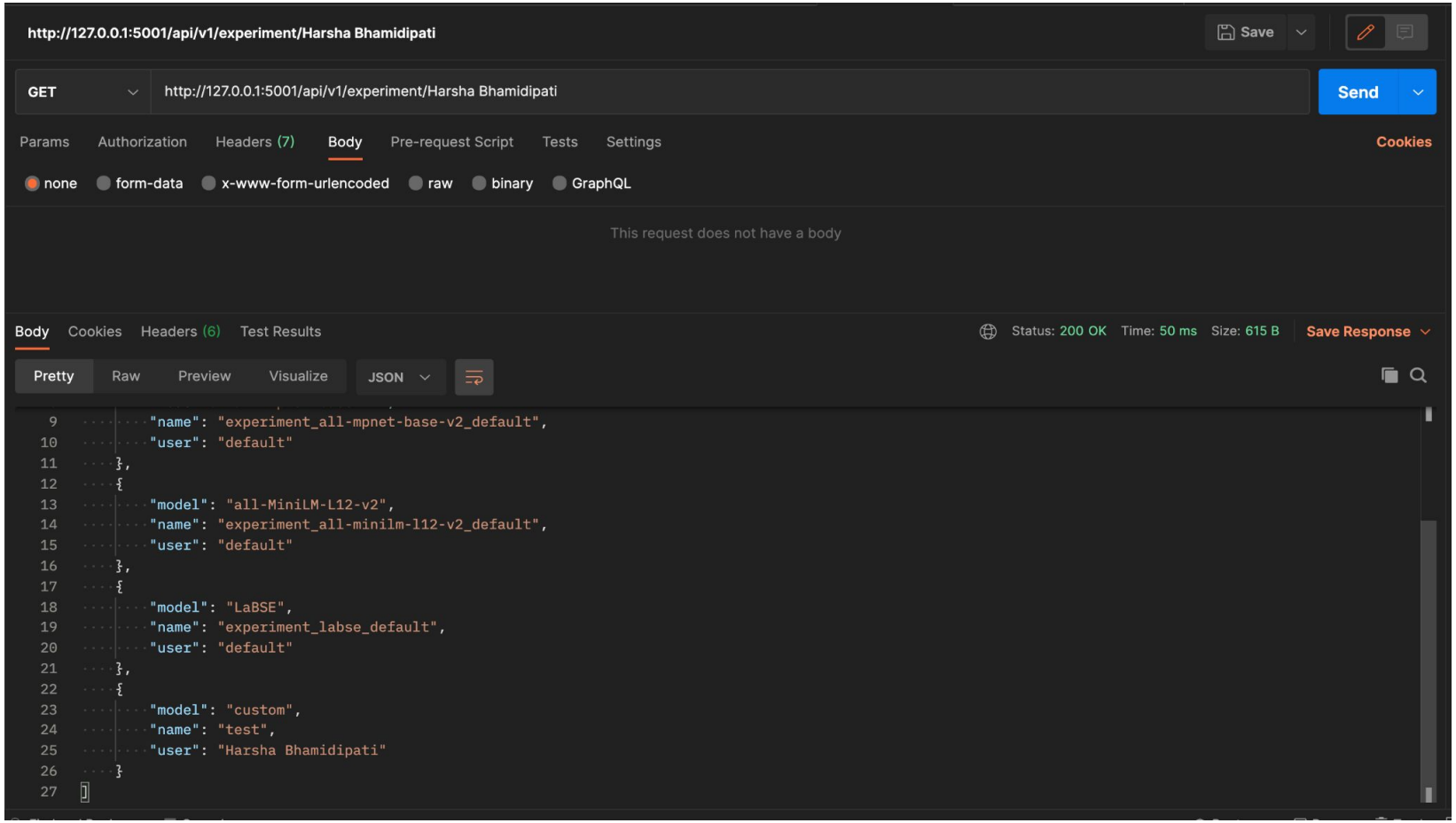
VIRGINIA TECH

- Running a search experiment.



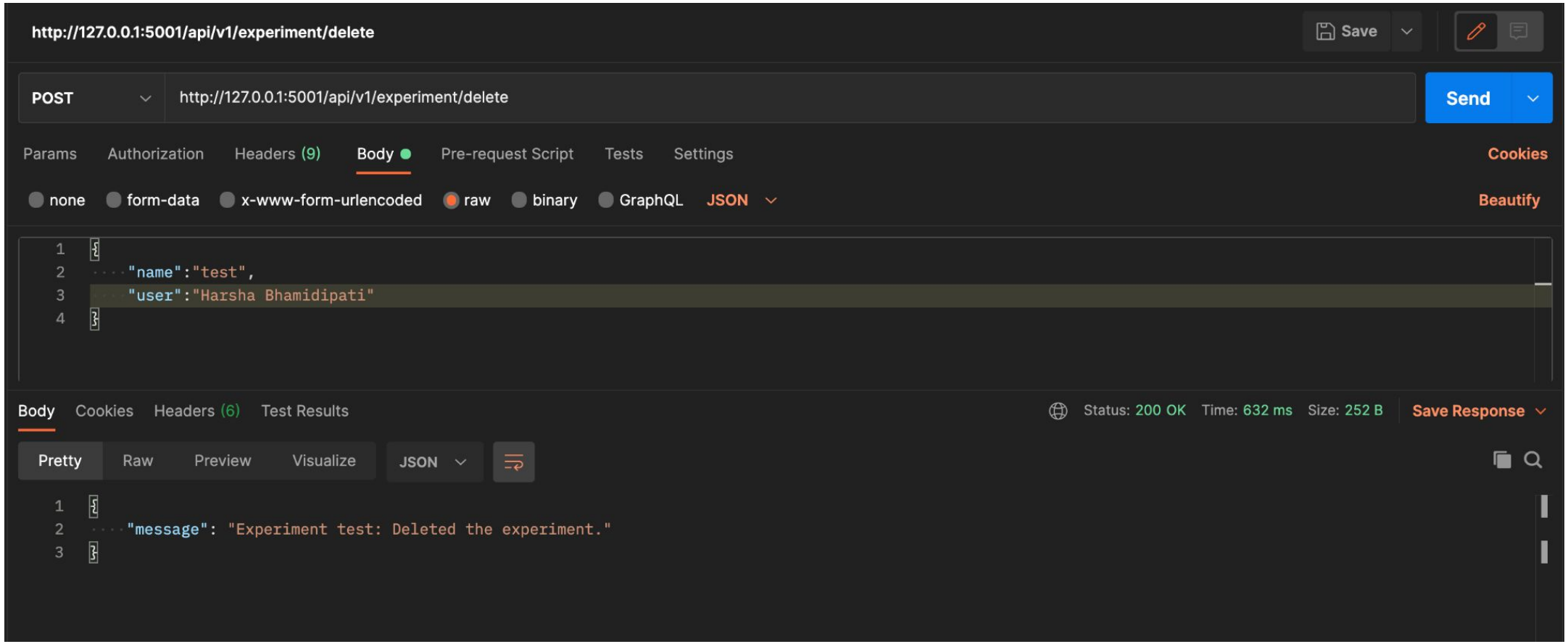
VIRGINIA TECH

- Listing the user created experiments



VIRGINIA TECH

- Deleting the user created experiment



Logging

Why did we implement logging?

- **Capturing User Engagement**: To capture user actions, preferences, and interactions with the system.
- **Personalized Recommendations**: By analyzing logs of user interactions, recommendation systems can tailor suggestions to individual preferences, enhancing user satisfaction and engagement.
- **Streamlined Error Debugging**: Facilitate service level error debugging.
- **Enhanced Performance Monitoring**: Logging facilitates the monitoring of system performance and the identification of potential issues or anomalies.

How did we implement logging?

- **Index creation:** Established an Elasticsearch index structure, featuring a versatile "meta" field for dynamic log data inclusion.
- **Logging service creation:** Deployed a dedicated logging service ensuring systematic log handling functions.
- **API Integration for Logging:** Integrated logging mechanisms into APIs.
- **Log-Specific Kafka Topic Implementation:** Established a Kafka topic named 'logs' to automatically store bulk log data from authorized sources.

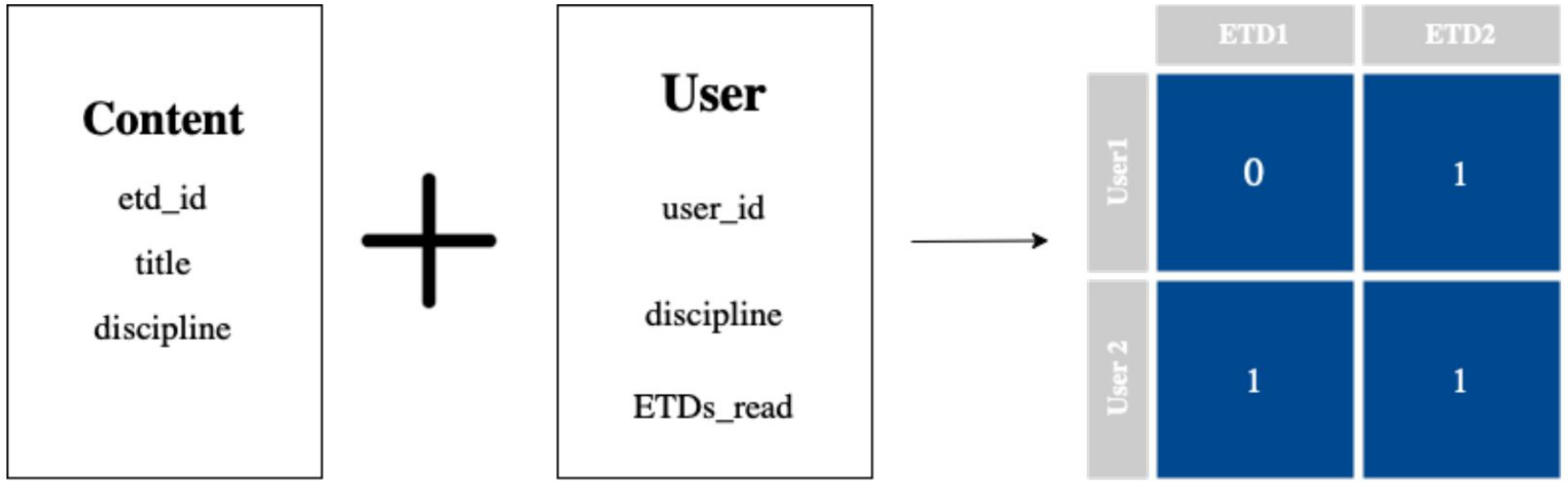
Integrated endpoints

API description	ID - uuid	api_endpoint	type	user_id	user activity
Search with a query	f47ac10b-58cc-4372-a567-0e02b2c3d479	/search	INFO, ERROR	USER	FETCH
Fetch a document based on ETD ID	f12ac10b-58cc-4372-a567-0e02b2c3d479	/documents/<etd_id>	INFO, ERROR	USER	FETCH
Autocomplete API	f12ac10b-58cc-4372-a567-0e02b2c3d480	/autocomplete	INFO, ERROR	USER	FETCH
Insert new ETD in the index	f12ac10b-58cc-4372-a567-0e02b2c3d479	/index	INFO, ERROR	USER	UPDATE
Fetch user logs by user id (ETD_ID and Chapters)	f12ac10b-58cc-4372-a567-0e02b2c3d473	/logs/<user_id>	INFO, ERROR	USER	FETCH
Search for a chapter	f12ac10b-58cc-4372-a567-0e02b2c3d477	/chapters/search	INFO, ERROR	USER	FETCH
fetch details of a chapter	f12ac10b-58cc-4372-a567-0e02b2c3d478	/chapters/<chapter_id>	INFO, ERROR	USER	FETCH
Create Experiment index if not present	f47ac10b-58cc-4372-a567-0e02b2c3d472	/experiment/create	INFO, ERROR	USER	CREATE
Update Experiment index with new details	f47ac10b-58cc-4372-a567-0e02b2c3d480	/experiment/create	INFO, ERROR	USER	UPDATE
Deleting experiments	f47ac10b-58cc-4372-a567-0e02b2c3d481	/experiment/delete	INFO, ERROR	USER	DELETE
Get all experiments by User	f47ac10b-58cc-4372-a567-0e02b2c3d482	/experiment/<user>	INFO, ERROR	USER	FETCH
Search experiments	f47ac10b-58cc-4372-a567-0e02b2c3d483	/experiment/search	INFO, ERROR	USER	FETCH
Generic Log Save	q39ac10b-58cc-4372-a567-0e02b2c3d4423	/application/log	INFO, ERROR,[	SYSTEM_ID	GENERIC

Recommendation

Recommendation system

- Started out trying to improve the original system, which was a single user-item model.
- Model was an autoencoder that had a one-hot user-item matrix as its input and output.



Raw Data

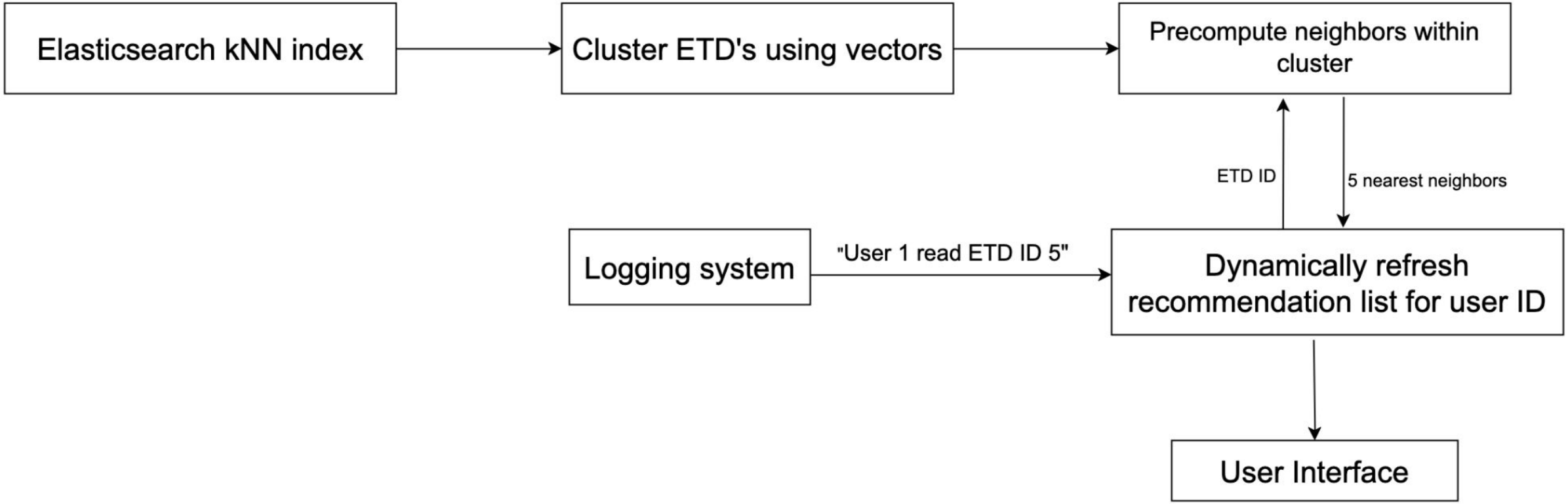
Synthetic Data

User-Item Matrix

Recommendation system

- Autoencoder was a good model choice, but their architecture wasn't really scalable because of fixed model I/O vs. non-fixed number of users and ETDs.
- Use of a *one-hot* matrix was also questionable since the number of times a user clicked on an ETD could potentially inform the user's level of interest in the content.
- Since they had no real user data, they had to use synthetic data for initial model training.

New Recommendation system



New Recommendation system

- Initial setup
  - Embeddings for all ETDs are pulled from the kNN Elasticsearch index.
  - Use K-means clustering to create clusters.
  - Inside each cluster, compute cosine similarity.
  - Initially, the categories and topics selected by users will be used to generate recommendations.
- When user clicks on an ETD
  - The ETD ID and user ID is sent to the recommendation server.
  - The 5 nearest neighbors of the ETD ID will be obtained from the cluster.
  - The existing recommendations will be pushed down by 5 places. The newly obtained neighbors will become the top 5 recommendations.

Future Work

- Improve kNN search performance.
- Shard Elasticsearch indexes for better search performance.
- Reduce time taken to index data into Elasticsearch.
- Add more options for search experiments.
- Create test routines for search and recommendation APIs.
- As user data becomes available, cluster it to form User-User relations.
- Implement a User-Item model using LightFM.

Thank you!
Any Questions?

VIRGINIA TECH