Final Project Report

YaeJin (Sally) Kang & Maureen Ekwebelem

December 4th, 2025

CISC 5640

Problem Statement & Motivation

Employment scams have surged in recent years, with reports rising by 118% in 2023 alone according to the Identity Theft Resource Center [1]. These scams cause financial harm and create long-term risks; when scammers steal sensitive information such as Social Security numbers or bank account details, job seekers may face identity theft and other lasting consequences. Fraudulent postings also harm the reputation of legitimate companies and erode trust in online recruitment platforms. As job boards continue to expand and scammers produce increasingly sophisticated fake listings, traditional manual review and rule-based filters can no longer keep up with the scale of this problem. To address this gap, we created FraudSpotter, a machine-learning powered job fraud detection system that allows users to enter job details (i.e. the job title, company name, location, employment type, salary range, etc.; see Figure 1 for full data table) and instantly receive a fraud likelihood score. Using the Kaggle "Real or Fake Job Posting Prediction" dataset, we trained a Logistic Regression model to generate interpretable, probability-based predictions of scam-likelihood. These results are then stored in Apache Cassandra, a distributed NoSQL database chosen over traditional relational databases for its scalability, fault tolerance, and ability to support fast, continuous writes. Together, this system supports real-time fraud detection, streamlined data ingestion, and long-term monitoring of emerging patterns, providing job seekers and platforms with a more dependable method for identifying and preventing fraudulent postings.

System Architecture & Data Model

FraudSpotter uses a straightforward end-to-end architecture that begins with data cleaning and class balancing, followed by text preprocessing, tokenization, and TF-IDF vectorization using the Kaggle "Real vs Fake Job Postings" dataset (Fig. 1). A Logistic Regression model is then trained on the transformed text, and both the TF-IDF vectorizer and the trained classifier are serialized using joblib so they can be consistently reloaded across application runs or deployments. FraudSpotter's design builds on Hanif's methodology, which similarly integrates systematic preprocessing, TF-IDF features, and logistic regression for fraud detection [2], providing a strong foundation for comparing preprocessing and class-balancing strategies.

Once trained, these artifacts are loaded into a Streamlit web application that allows users to enter job details and receive an instant fraud prediction. On the backend, a Cassandra database (powered by Astra DB) stores each user submission together with the model's outputs.

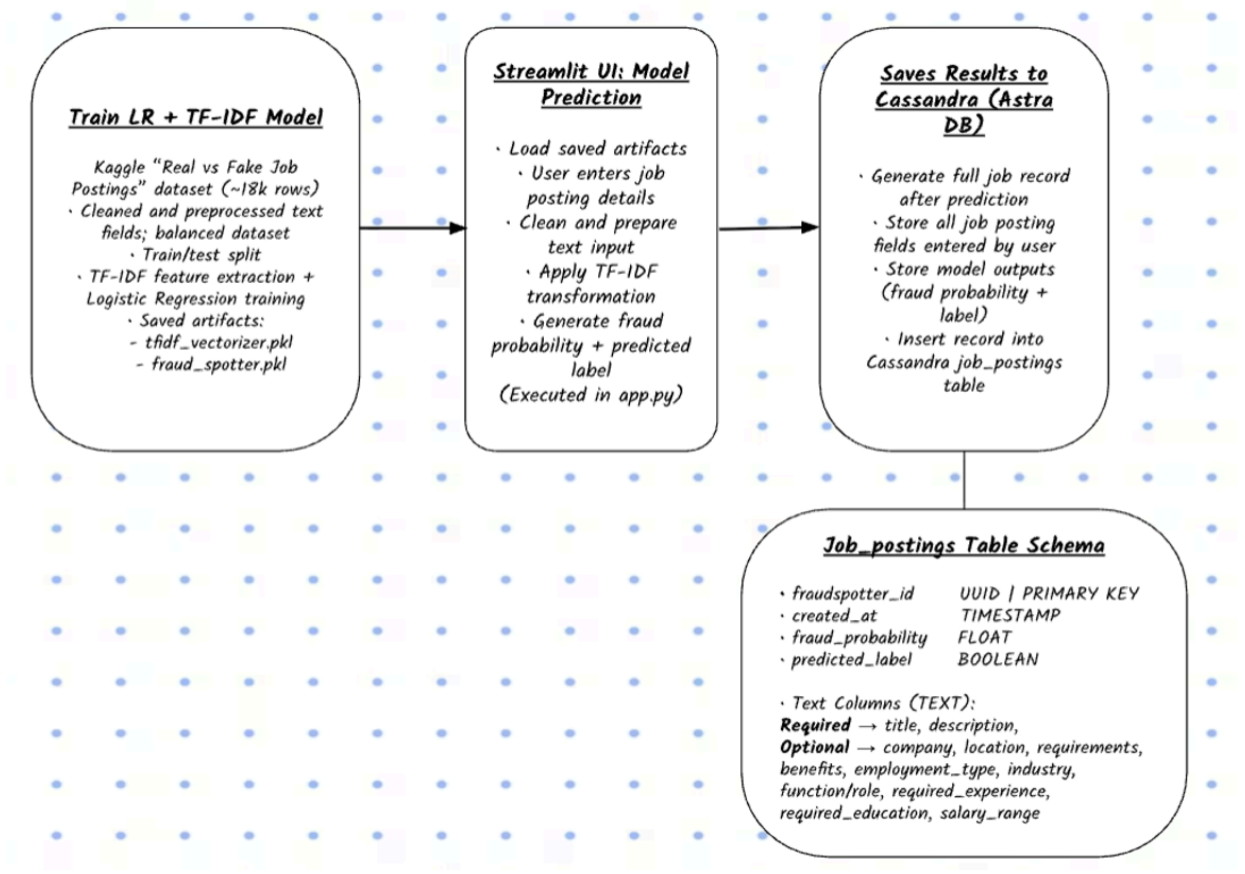

Train LR + TF-IDF Model
Kaggle "Real vs Fake Job Postings" dataset (~18k rows)
· Cleaned and preprocessed text fields; balanced dataset
· Train/test split
· TF-IDF feature extraction + Logistic Regression training
· Saved artifacts:
- tfidf_vectorizer.pkl
- fraud_spotter.pkl
Streamlit UI: Model Prediction
· Load saved artifacts
· User enters job posting details
· Clean and prepare text input
· Apply TF-IDF transformation
· Generate fraud probability + predicted label
(Executed in app.py)
Saves Results to Cassandra (Astra DB)
· Generate full job record after prediction
· Store all job posting fields entered by user
· Store model outputs (fraud probability + label)
· Insert record into Cassandra job_postings table
Job_postings Table Schema
· fraudspotter_id UUID | PRIMARY KEY
· created_at TIMESTAMP
· fraud_probability FLOAT
· predicted_label BOOLEAN
· Text Columns (TEXT):
Required → title, description,
Optional → company, location, requirements, benefits, employment_type, industry, function/role, required_experience, required_education, salary_range

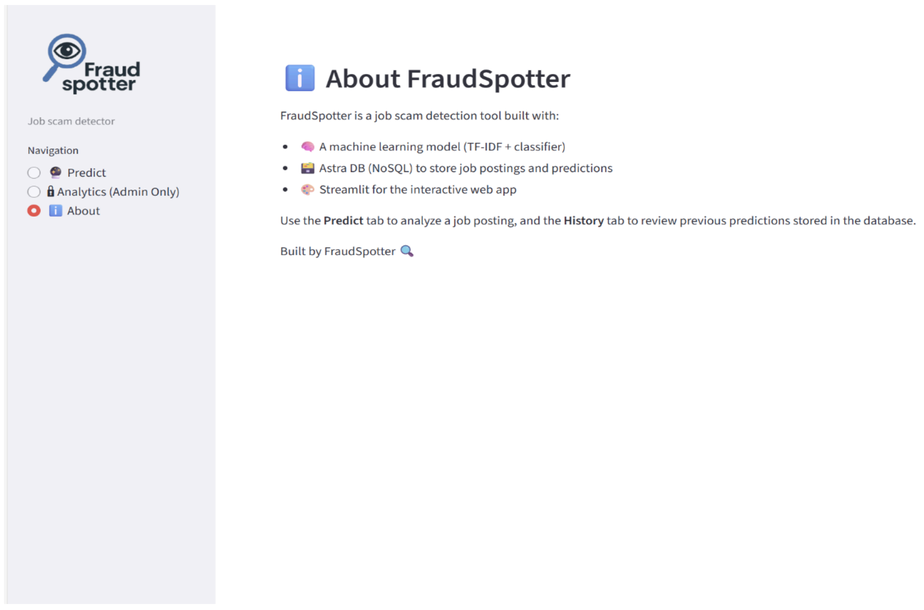
Figure 1. FraudSpotter System Architecture

The data model uses a single consolidated table that stores both the job posting information and the model's prediction results. Each row represents a full job entry and is identified by a UUID partition key to ensure even data distribution across Cassandra nodes. The table includes all relevant posting fields such as the title, description, location, company details, employment type, required skills, benefits, and original metadata, along with a timestamp indicating when the entry was processed. It also stores the model's outputs, including the predicted label, fraud-probability score, and artifact version.

By keeping inputs and outputs in one table, the schema avoids the overhead of maintaining separate tables and eliminates the need for joins, which Cassandra is not optimized for. This design also enables efficient queries, allowing the full context of a prediction to be retrieved with a single read. The UUID-based partitioning strategy further ensures high write throughput and allows the system to scale smoothly as more job postings are scored.

Before selecting a column-family database, we compared several NoSQL options to consider what would work best for FraudSpotter. Document databases like MongoDB offer flexible schemas, but our data does not require nested structures, and MongoDB does not provide the same level of horizontal write scalability we need. Graph databases such as Neo4j are built for analyzing relationships between entities, which isn't part of our workflow, so they would add unnecessary complexity. Key-value stores like Redis are extremely fast, but they aren't designed for storing and querying the historical prediction logs that FraudSpotter relies on for trend analysis and administrative review. We also looked at other column-family systems like HBase, but its reliance on the larger Hadoop ecosystem and its heavier operational requirements made it a poor choice for this project, especially given the time constraints. Cassandra, by comparison, offers high availability, even data distribution, and efficient horizontal scaling making it a strong match for our heavy write activity with consistent log generation requirements. Each prediction generates a new database entry, and Cassandra's distributed architecture can sustain these continuous write operations without performance degradation.

Implementation



About FraudSpotter

FraudSpotter is a job scam detection tool built with:

- A machine learning model (TF-IDF + classifier)
- Astra DB (NoSQL) to store job postings and predictions
- Streamlit for the interactive web app

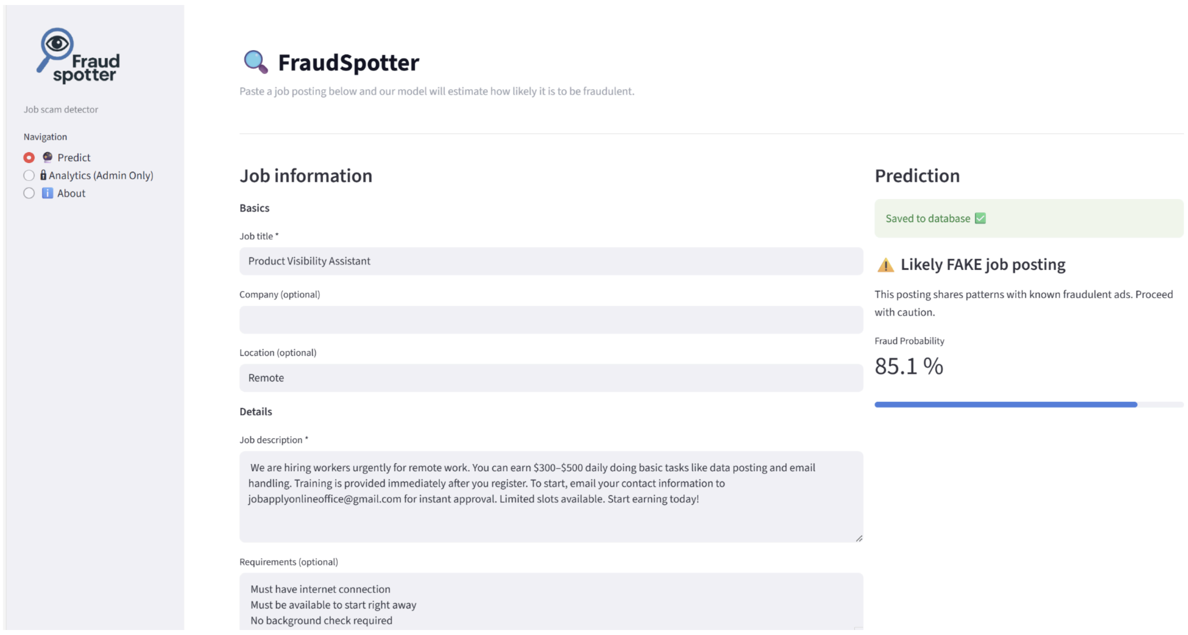
Use the **Predict** tab to analyze a job posting, and the **History** tab to review previous predictions stored in the database.

Built by FraudSpotter

Figure 2. FraudSpotter Page Overview

FraudSpotter is intended for job seekers who want to verify whether a posting is legitimate before sharing any personal information. They navigate to the site, which is hosted in a Streamlit web application that serves as the user-facing interface. Figure 2 displays the About page in FraudSpotter, which offers a brief overview of the application. At runtime, the app loads the saved TF-IDF vectorizer and model artifacts, accepts job descriptions from users, and outputs a predicted fraud label and probability score. If the posting receives a score over the threshold of

50% , it is considered likely to be Fake (Fig. 3). However, if the posting receives a score lower than the threshold of 50%, the posting is deemed likely to be real (Fig. 4).



FraudSpotter

Paste a job posting below and our model will estimate how likely it is to be fraudulent.

Job information

Basics

Job title *

Product Visibility Assistant

Company (optional)

Location (optional)

Remote

Details

Job description *

We are hiring workers urgently for remote work. You can earn $300–$500 daily doing basic tasks like data posting and email handling. Training is provided immediately after you register. To start, email your contact information to jobapplyonlineoffice@gmail.com for instant approval. Limited slots available. Start earning today!

Requirements (optional)

Must have internet connection
Must be available to start right away
No background check required

Prediction

Saved to database ✅

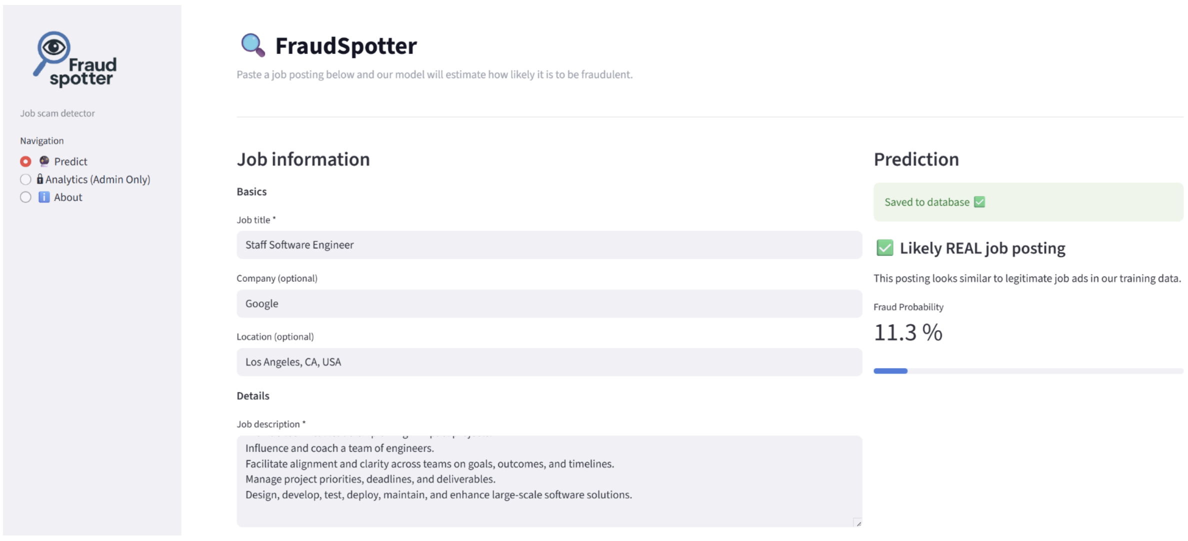
⚠️ Likely FAKE job posting

This posting shares patterns with known fraudulent ads. Proceed with caution.

Fraud Probability

85.1 %

Figure 3. Fraudspotter Implementation Fake Job Example



FraudSpotter

Paste a job posting below and our model will estimate how likely it is to be fraudulent.

Job information

Basics

Job title *

Staff Software Engineer

Company (optional)

Google

Location (optional)

Los Angeles, CA, USA

Details

Job description *

Influence and coach a team of engineers.
Facilitate alignment and clarity across teams on goals, outcomes, and timelines.
Manage project priorities, deadlines, and deliverables.
Design, develop, test, deploy, maintain, and enhance large-scale software solutions.

Prediction

Saved to database ✅

✅ Likely REAL job posting

This posting looks similar to legitimate job ads in our training data.

Fraud Probability

11.3 %

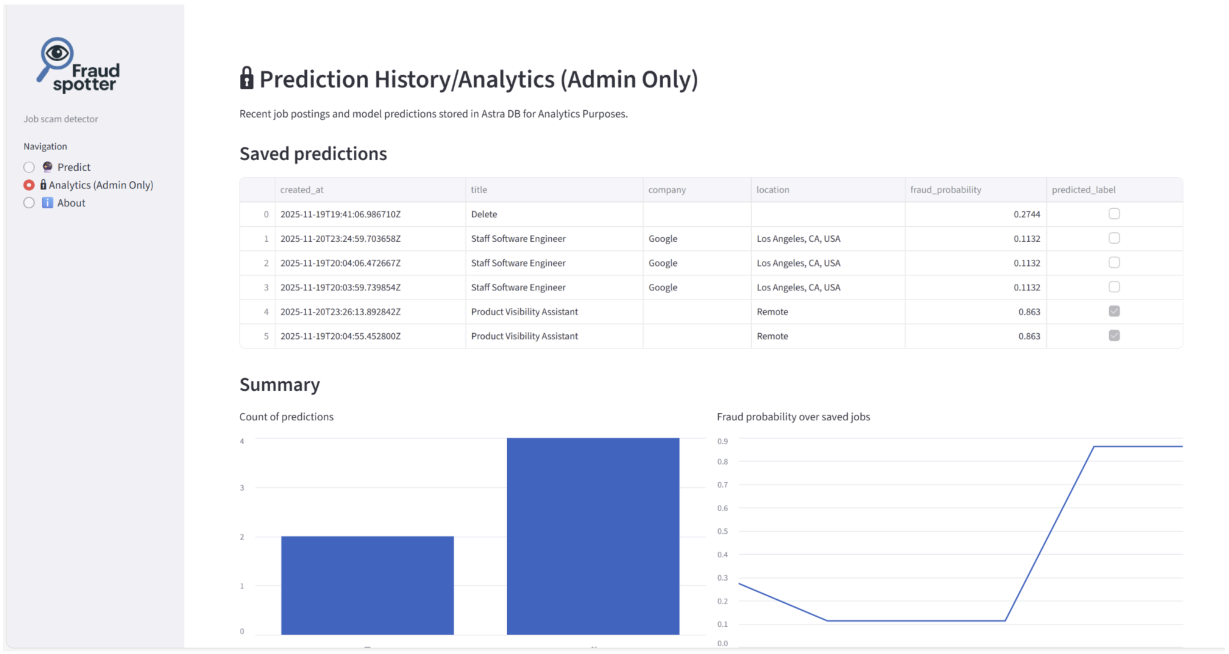
Figure 4. Fraudspotter Implementation Real Job Example

Immediately after generating a prediction, the result along with the original text input, timestamp, and a generated UUID is inserted into Cassandra using the Astra DB Python driver. This supports continuous data ingestion, with each prediction stored as a new fixed entry in the predictions table.

The system also supports retrieval and analysis through meaningful Cassandra queries. Basic read operations allow the application or administrators to fetch all predictions associated with a given UUID, within a specific time range or location. This demonstrates that the prototype not only logs the data but also enables real analytical insight into emerging job scam patterns.

For example, if an administrator wanted to query the number of fake predictions in a certain location, they could use Cassandra and run: SELECT * FROM job_postings WHERE (location = 'New York' OR ‘NY’) AND predicted_label = 'fake';

An administrator could also search the saved predictions and review trends with data visualizations on the Analytics pane, shown in Figure 5.



Prediction History/Analytics (Admin Only)

Recent job postings and model predictions stored in Astra DB for Analytics Purposes.

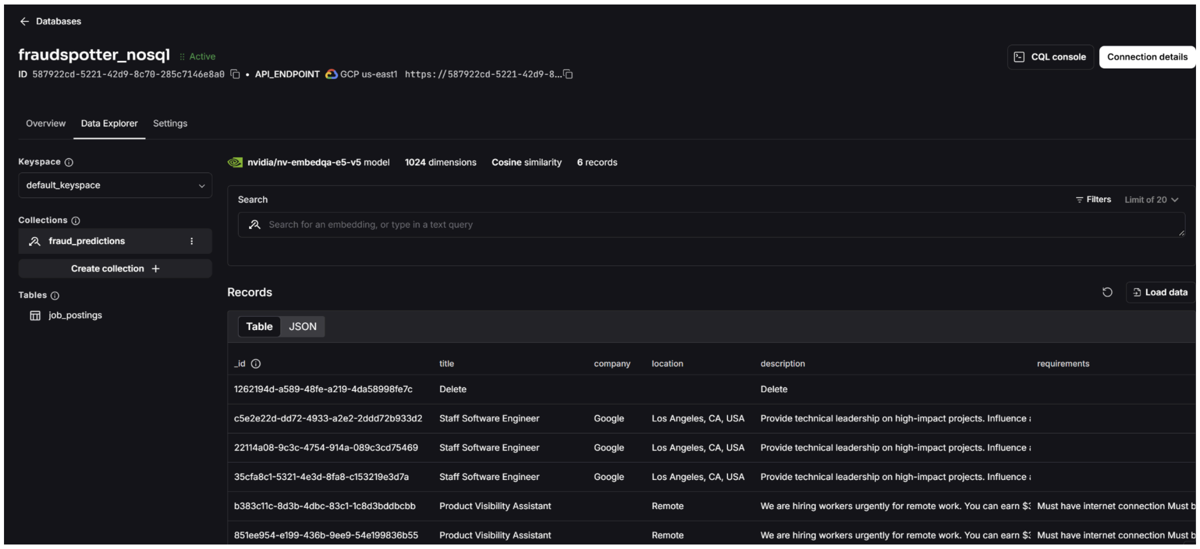
Saved predictions

	created_at	title	company	location	fraud_probability	predicted_label
0	2025-11-19T19:41:06.986710Z	Delete			0.2744	☐
1	2025-11-20T23:24:59.703658Z	Staff Software Engineer	Google	Los Angeles, CA, USA	0.1132	☐
2	2025-11-19T20:04:06.472667Z	Staff Software Engineer	Google	Los Angeles, CA, USA	0.1132	☐
3	2025-11-19T20:03:59.739854Z	Staff Software Engineer	Google	Los Angeles, CA, USA	0.1132	☐
4	2025-11-20T23:26:13.892842Z	Product Visibility Assistant		Remote	0.863	✓
5	2025-11-19T20:04:55.452800Z	Product Visibility Assistant		Remote	0.863	✓

Summary

Figure 5. FraudSpotter Prediction History & Analytics Page

Together, these components implement a complete pipeline: data ingestion through the app, prediction via the machine learning model, storage in Cassandra (Fig. 6), and retrieval through targeted analytical queries, therefore meeting the requirements for a functioning NoSQL-backed prototype application. Because each prediction is stored as a new event identified by a UUID, the system relies on INSERT operations rather than UPDATEs, which satisfies the data-ingestion requirement.



fraudspotter_nosql Active

ID 587922cd-5221-42d9-8c70-285c7146e8a0 • API_ENDPOINT GCP us-east1 https://587922cd-5221-42d9-8...

Keyspace: default_keyspace

Collections: fraud_predictions

Tables: job_postings

nvidia/nv-embedqa-e5-v5 model 1024 dimensions Cosine similarity 6 records

Search for an embedding, or type in a text query

Records

_id	title	company	location	description	requirements
1262194d-a589-48fe-a219-4da58998fe7c	Delete			Delete	
c5e2e22d-dd72-4933-a2e2-2ddd72b933d2	Staff Software Engineer	Google	Los Angeles, CA, USA	Provide technical leadership on high-impact projects. Influence	
22114a08-9c3c-4754-914a-089c3cd75469	Staff Software Engineer	Google	Los Angeles, CA, USA	Provide technical leadership on high-impact projects. Influence	
35cfa8c1-5321-4e3d-8fa8-c153219e3d7a	Staff Software Engineer	Google	Los Angeles, CA, USA	Provide technical leadership on high-impact projects. Influence	
b383c11c-8d3b-4dbc-83c1-1c8d3bddbcbb	Product Visibility Assistant		Remote	We are hiring workers urgently for remote work. You can earn $	Must have internet connection Must b
851ee954-e199-436b-9ee9-54e199836b55	Product Visibility Assistant		Remote	We are hiring workers urgently for remote work. You can earn $	Must have internet connection Must b

Figure 6. Prediction Results Table Stored on Cassandra hosted in AstraDB

Evaluation & Analysis

FraudSpotter successfully meets its core use cases. Users can enter job postings in natural language, receive a fraud probability from our machine-learning model, and have each result stored in a scalable Cassandra database. Cassandra's design favors availability over strict consistency, which works well for our needs: the system can keep ingesting predictions quickly, even under network disruption. Its horizontal scalability and partitioning strategy also support fast writes and low-latency queries as the volume of predictions grows. While this model sacrifices some query flexibility compared to relational systems, the performance benefits and ability to scale seamlessly made it the right choice for FraudSpotter.

Every design decision involved trade-offs. Cassandra favors availability over strict consistency, meaning the system can continue writing even under network disruption. The priority is fast ingestion of data, so Cassandra's horizontal scalability is favorable.The main challenges we ran into were handling scikit-learn version mismatches when loading our pickled artifacts and troubleshooting Astra DB connection bundle errors. We resolved the sklearn issue by updating our environment and clearly noting the version requirement in the project README so the models load correctly on any machine. For the Astra DB issue, we switched from using the connection bundle to using the Astra DB API-based connection method, which made the integration simpler and more reliable.

Reflection

Building FraudSpotter provided us the opportunity to experience designing a system using NoSQL principles rather than using our traditional SQL mindset. Through this project, we found that Cassandra excels at continuous, high-volume write workloads and predictable query patterns, but it also forces careful planning around partition keys and how the data will actually be accessed. Connecting the machine learning workflow with Streamlit showed how a lightweight frontend can interact smoothly with a distributed database.

A scalable and maintainable approach to model updates would involve scheduled or event-driven retraining so the classifier can adapt as scam patterns evolve. FraudSpotter would adopt explicit model-version columns, storing the exact model artifact information alongside each prediction in Cassandra to preserve full traceability. This enables longitudinal evaluation, auditing, and model comparison without losing historical integrity.

As the system grows, we would want additional operational and analytical capabilities layered on top. Trend dashboards could visualize changes in fraud rates over time, segmented by industry, job type, or posting platform. Additionally, we envision FraudSpotter being integrated directly into job-posting platforms such as LinkedIn or Indeed, similar to how fraud-prevention systems partner with Experian to provide fraud services. These enhancements would transform FraudSpotter from a real-time classifier into a fully featured fraud-intelligence and monitoring platform capable of supporting analysts, operational teams, and long-term data-driven decision-making.

References

[1] Identity Theft Resource Center, "2023 trends in identity report: Identity Theft Resource Center sees 118 percent increase in job scams," Jun. 26, 2024. [Online]. Available: https://www.idtheftcenter.org/post/2023-trends-in-identity-report-118-percent-job-scam-increase/

[2] Hanif, A. Agarwal and R. Sharma, "A novel online job scam detection of imbalanced data using machine learning and NLP models," ResearchGate,Preprint,2024.[Online]. Available: https://www.researchgate.net/publication/377447842