Report(Part IV)

Course Number: CSCI-GA.2433-001

Name: Liu, Jiexi

Date: December 19, 2025

NYU ID: N18245550

Couse section: Database Systems, Section 001

GitHub Link: https://github.com/jesse66-6/dbs_final_version

1. Executive Summary

This project implements a complete End-to-End InsureAI System that integrates a web-based operational application with a data-driven machine learning pipeline. The solution is built using Python Flask for the application layer, MySQL for the Operational Data Store (ODS), and XGBoost for the predictive analytics engine.

The core innovation of this system is its Closed-Loop Data-Driven Workflow:

1. Transactional Operations: Users enter client data and purchase policies via a web interface.
2. Data Persistence: Transactions are stored in the MySQL ODS, creating a growing repository of "Ground Truth" data.
3. Automated Retraining: The system monitors the accumulation of labeled data. Once a threshold is reached, it triggers a background pipeline that merges historical CSV data with the new SQL increments to retrain the model.
4. Real-Time Governance: The new model is automatically versioned, registered, and hot-swapped into the production environment to improve future predictions.

This solution satisfies the project requirements for hybrid data processing, real-time insight generation, and DIKW-based governance.

2. Business Use Cases & Process Design

The system is designed to optimize insurance product recommendations by leveraging both historical legacy data and real-time interaction data.

2.1 Client Entry & Real-time Prediction

When an agent enters a new client's demographic data (e.g., Age, Income, Education), the system immediately queries the active Machine Learning model. The model predicts the most suitable product (e.g., Life Insurance, FSA, or A&H) and displays this recommendation to the agent, aiding in decision-making.

2.2 Policy Purchase & Feedback Loop

When a client purchases a specific product, the agent records this transaction. This action serves two purposes:

1. Operational: It creates a financial record in the `Insurance_Policy` table.
2. Analytical: It labels the client record with the "Actual Product Bought," converting a prediction into Ground Truth data. This feedback is crucial for correcting the model's future behavior.

2.3 Automated Model Governance

The system continuously monitors the volume of new ground truth data. When sufficient new data (e.g., every 5 new records) is available, the system automatically triggers the retraining pipeline. This ensures the AI solution remains relevant and adapts to changing customer behaviors without manual intervention.

3. System Implementation

3.1 Technology Stack

- Web Framework: Python Flask (RESTful routing & Business Logic).
- Database: MySQL 8.0 (Operational Data Store & Data Warehouse elements).
- Machine Learning: Scikit-Learn (Preprocessing) + XGBoost (Classification).
- Frontend: Bootstrap 5 (Responsive UI).

3.2 Database Schema Design

The `EnterpriseInsuranceDB` schema is designed to support both OLTP and OLAP workloads:

- `Insured_Client` : The core table storing customer demographics (Features) and the `actual_product_bought` (Label).
- `Client_Insights` : A denormalized table specifically designed to store AI predictions and model versions. This separates high-volume analytical writes from core client data.
- `Insurance_Policy` : Stores transaction details (start date, premium, status).
- `Model_Registry` : Implements DIKW governance by tracking Model IDs, Versions, Accuracy Scores, and File Paths.

3.3 Data-Driven ML Pipeline

The `train_model.py` module implements a Hybrid Data Lake approach:

1. Ingestion: It loads static historical data from a CSV file (`customer_insurance_dataset.csv`).
2. Increment Integration: It queries the MySQL ODS for new records where `actual_product_bought IS NOT NULL` .
3. Fusion & Training: Both datasets are merged to train a new XGBoost classifier.
4. Versioning: The trained model is serialized (pickled) and registered in the database as the new "Active" version.

4. Database Optimization

To ensure scalability and performance as required by the project specification, the following optimizations were implemented in `schema.sql` :

4.1 Indexing

Composite indexes were created to speed up frequent lookups and filtering operations:

```plsql
-- Optimized for demographic analysis queries
INDEX idx_demographics (age, annual_income),
-- Optimized for status reporting
INDEX idx_status (marital_status, education_level)
```

4.2 Table Partitioning

The `Insurance_Policy` table utilizes Range Partitioning based on the transaction year. This optimization significantly improves query performance for time-based reports (e.g., "2024 Revenue") by eliminating the need to scan the entire table history.

```plsql
PARTITION BY RANGE (YEAR(start_date)) (
  PARTITION p_history VALUES LESS THAN (2023),
  PARTITION p_2024 VALUES LESS THAN (2025),
  ...
);
```

4.3 Denormalization

To prevent the expensive re-calculation of predictions every time a client list is viewed, we implemented the `Client_Insights` table. Prediction results are calculated once upon client entry and stored. This "Compute Once, Read Many" strategy reduces latency and system load.

5. Governance & Reference Architecture

This solution adheres to the DIKW (Data, Information, Knowledge, Wisdom) hierarchy and implements strict governance policies.

5.1 Model Governance

The `Model_Registry` table acts as the source of truth for all AI assets. It ensures Traceability by linking every customer prediction to a specific model version (`model_version` column in `Client_Insights`). It also enforces Quality Control by logging accuracy metrics and training dataset sizes before a model is marked as 'Active'.

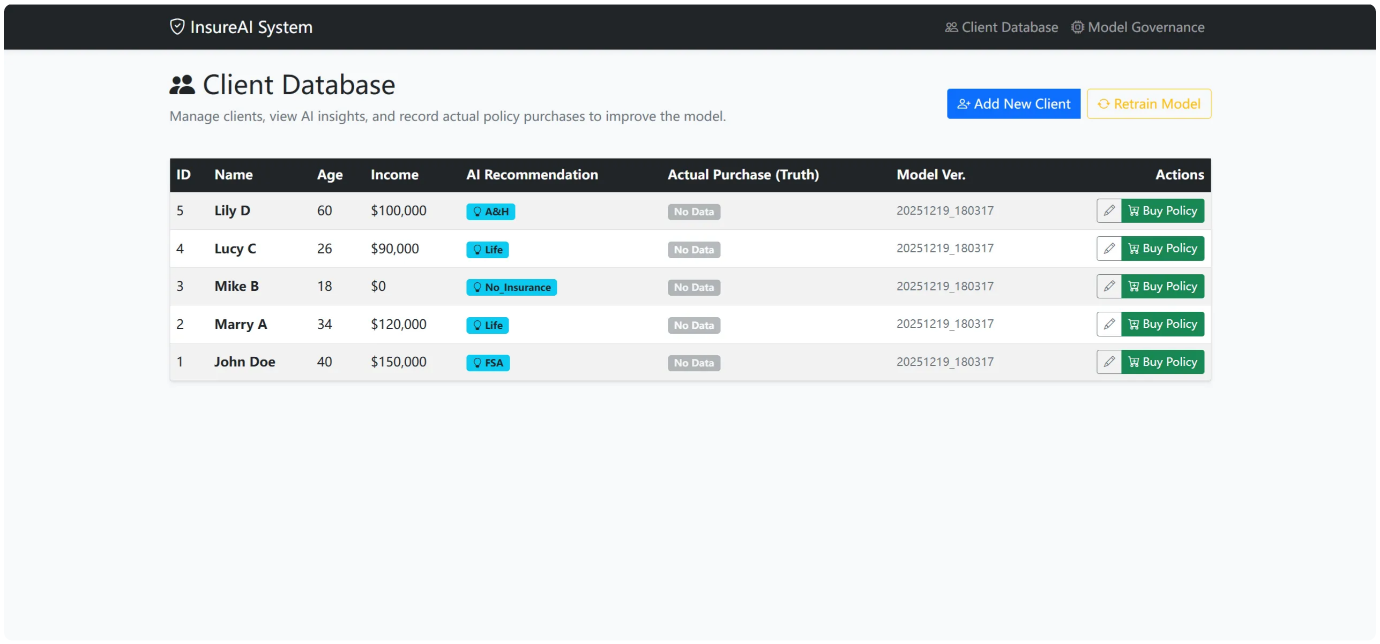
5.2 Data Quality & Fairness

Data integrity is enforced via SQL constraints (e.g., `CHECK (age > 0 AND age < 120)`). Furthermore, the feedback loop mechanism promotes fairness by continuously incorporating new, real-world data points, allowing the model to adapt to demographic shifts and reduce historical bias over time.

6. System Walkthrough & Evidence

6.1 Client Dashboard & AI Recommendations

The dashboard displays the client list. Note the "AI Recommendation" badge, which guides the agent, and the "Actual Purchase" column, which tracks the ground truth.



InsureAI System

Client Database · Model Governance

Client Database

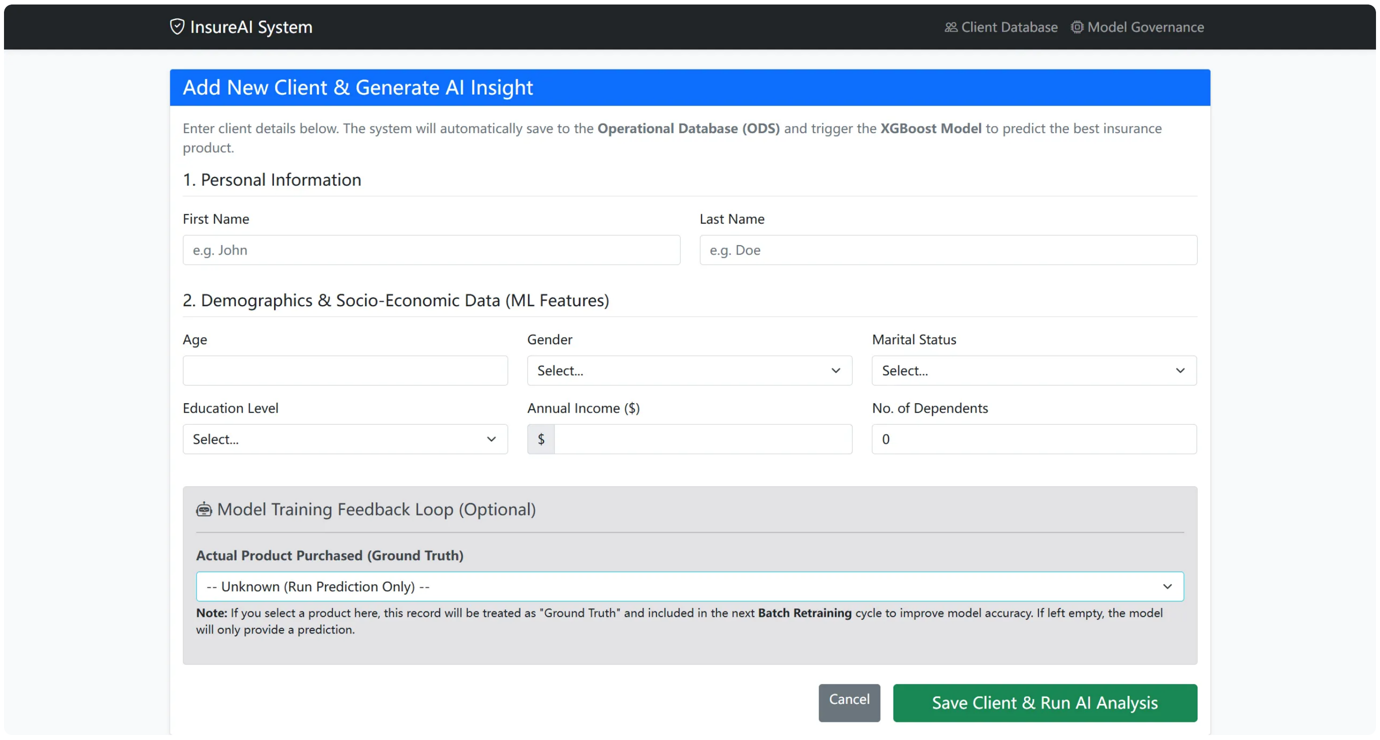
Manage clients, view AI insights, and record actual policy purchases to improve the model.

Add New Client · Retrain Model

ID	Name	Age	Income	AI Recommendation	Actual Purchase (Truth)	Model Ver.	Actions
5	**Lily D**	60	$100,000	A&H	No Data	20251219_180317	Buy Policy
4	**Lucy C**	26	$90,000	Life	No Data	20251219_180317	Buy Policy
3	**Mike B**	18	$0	No_Insurance	No Data	20251219_180317	Buy Policy
2	**Marry A**	34	$120,000	Life	No Data	20251219_180317	Buy Policy
1	**John Doe**	40	$150,000	FSA	No Data	20251219_180317	Buy Policy

6.2 Real-Time Inference

When adding a new client, the system captures features and triggers the prediction engine in real-time.



InsureAI System

Client Database Model Governance

Add New Client & Generate AI Insight

Enter client details below. The system will automatically save to the **Operational Database (ODS)** and trigger the **XGBoost Model** to predict the best insurance product.

1. Personal Information

First Name	e.g. John
Last Name	e.g. Doe

2. Demographics & Socio-Economic Data (ML Features)

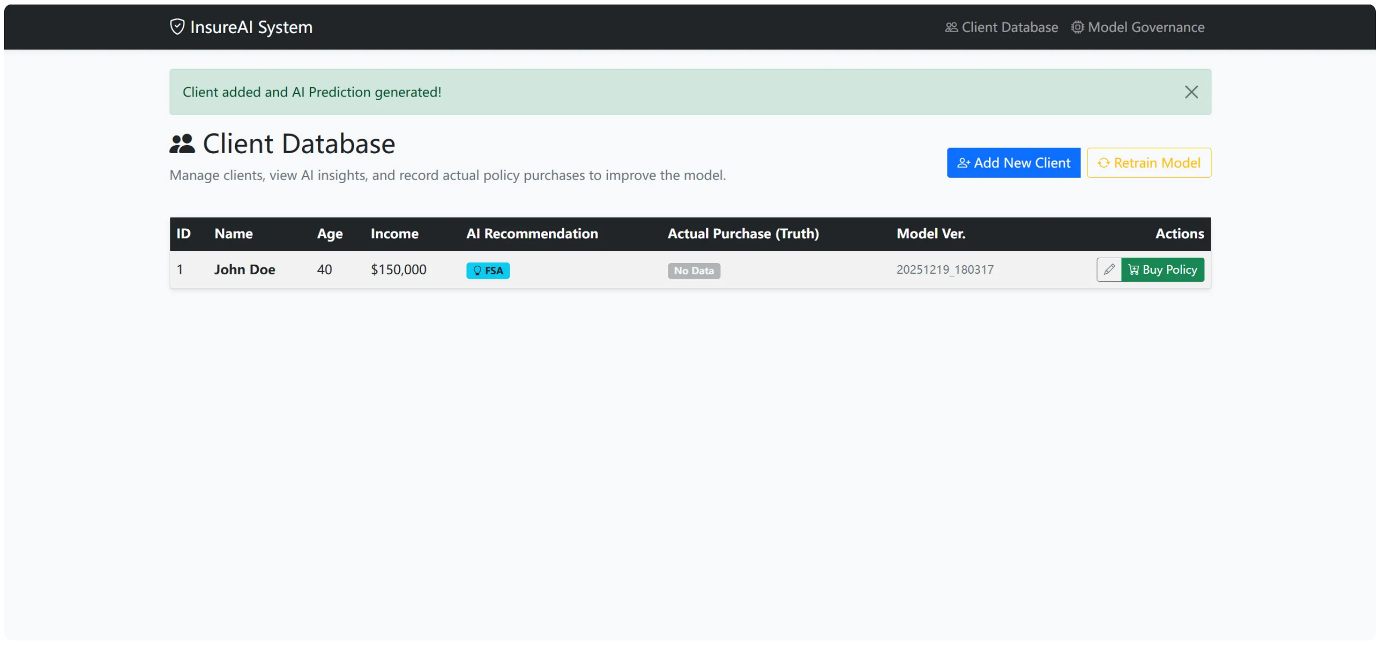
Age	
Gender	Select...
Marital Status	Select...
Education Level	Select...
Annual Income ($)	$
No. of Dependents	0

Model Training Feedback Loop (Optional)

Actual Product Purchased (Ground Truth)

-- Unknown (Run Prediction Only) --

Note: If you select a product here, this record will be treated as "Ground Truth" and included in the next **Batch Retraining** cycle to improve model accuracy. If left empty, the model will only provide a prediction.



InsureAI System

Client Database Model Governance

Client added and AI Prediction generated!

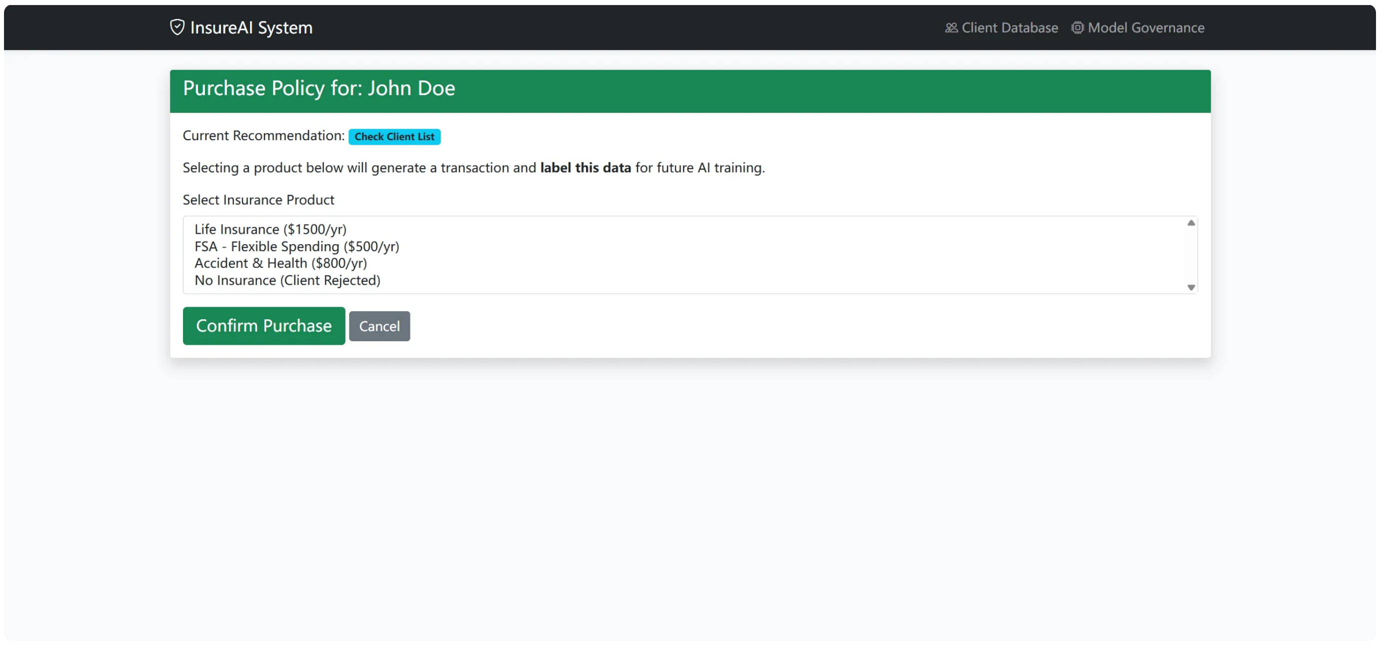
Client Database

Manage clients, view AI insights, and record actual policy purchases to improve the model.

ID	Name	Age	Income	AI Recommendation	Actual Purchase (Truth)	Model Ver.	Actions
1	John Doe	40	$150,000	FSA	No Data	20251219_180317	Buy Policy

6.3 Closed-Loop: Purchase & Auto-Retraining

Upon confirming a policy purchase, the system detects if the labeled data threshold has been met. If so, it triggers the retraining pipeline.



InsureAI System

Client Database · Model Governance

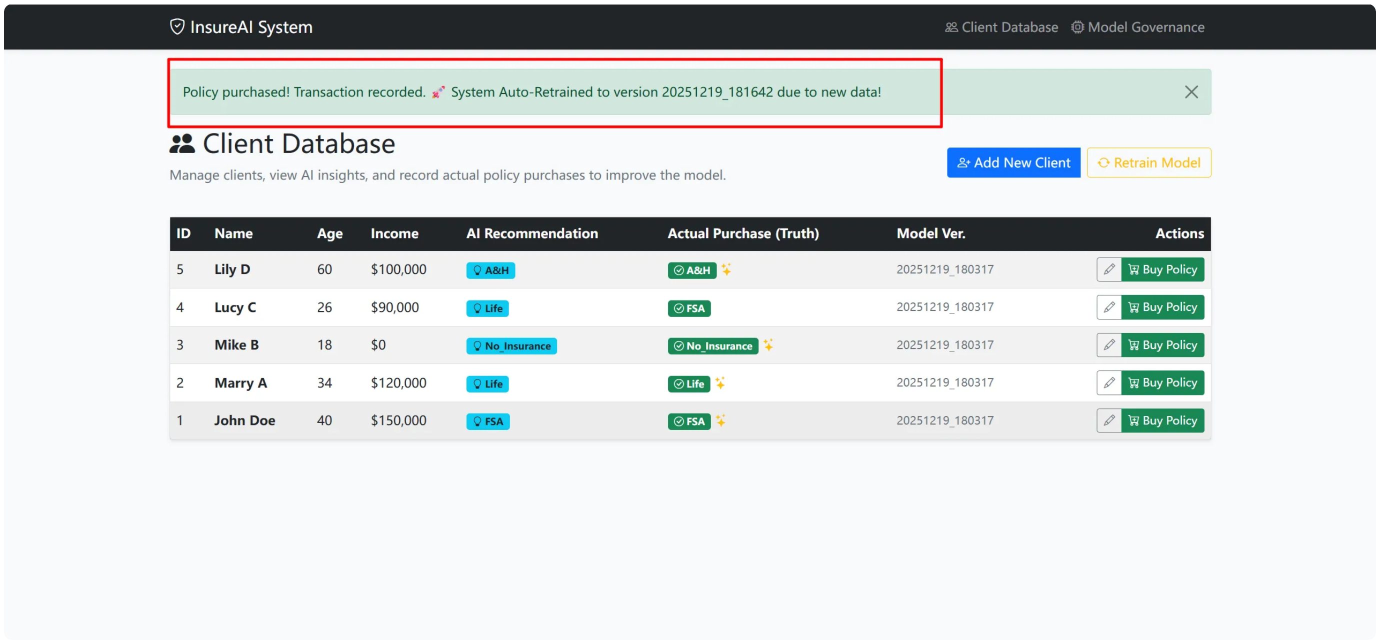
Purchase Policy for: John Doe

Current Recommendation: Check Client List

Selecting a product below will generate a transaction and **label this data** for future AI training.

Select Insurance Product

- Life Insurance ($1500/yr)
- FSA - Flexible Spending ($500/yr)
- Accident & Health ($800/yr)
- No Insurance (Client Rejected)



InsureAI System

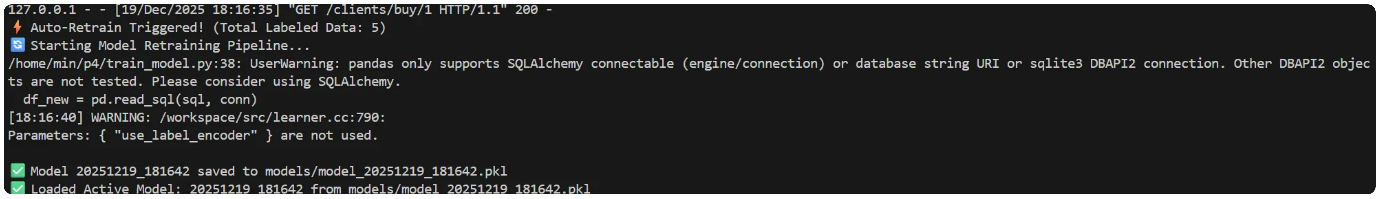
Client Database · Model Governance

Policy purchased! Transaction recorded. 💉 System Auto-Retrained to version 20251219_181642 due to new data!

Client Database

Manage clients, view AI insights, and record actual policy purchases to improve the model.

ID	Name	Age	Income	AI Recommendation	Actual Purchase (Truth)	Model Ver.	Actions
5	**Lily D**	60	$100,000	A&H	A&H ✨	20251219_180317	Buy Policy
4	**Lucy C**	26	$90,000	Life	FSA	20251219_180317	Buy Policy
3	**Mike B**	18	$0	No_Insurance	No_Insurance ✨	20251219_180317	Buy Policy
2	**Marry A**	34	$120,000	Life	Life ✨	20251219_180317	Buy Policy
1	**John Doe**	40	$150,000	FSA	FSA ✨	20251219_180317	Buy Policy

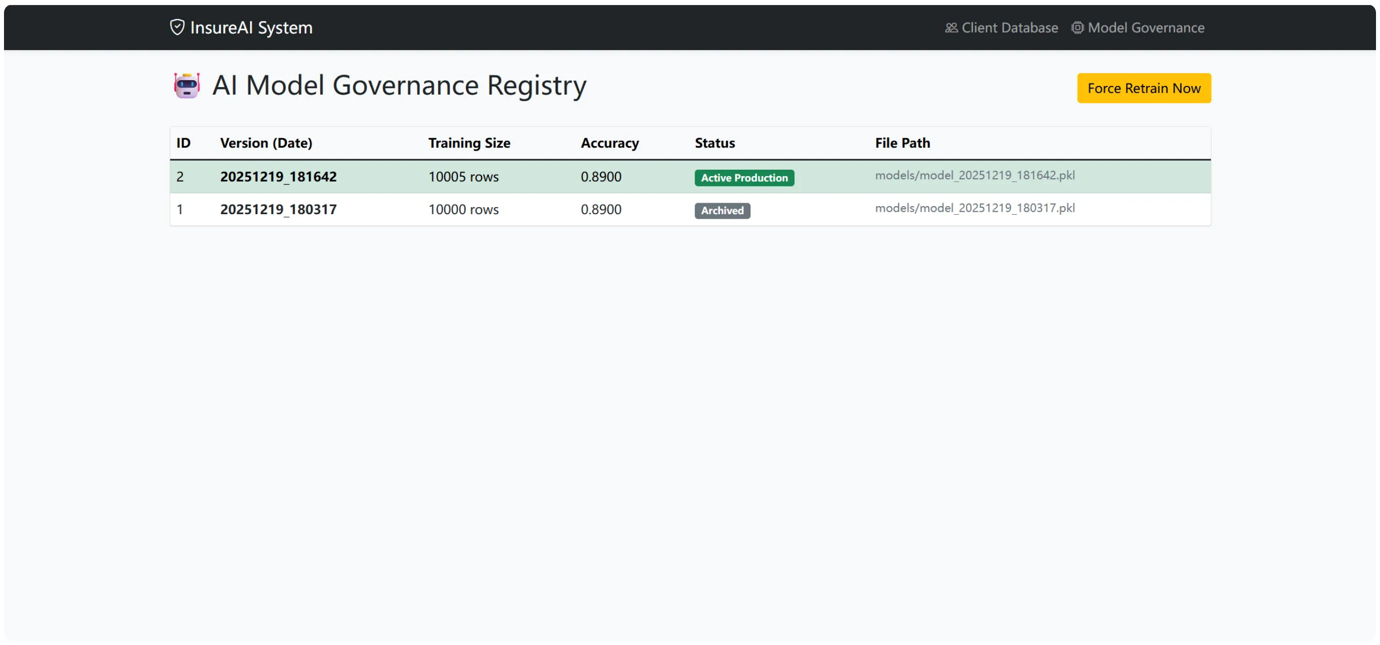


```
127.0.0.1 - - [19/Dec/2025 18:16:35] "GET /clients/buy/1 HTTP/1.1" 200 -
⚡ Auto-Retrain Triggered! (Total Labeled Data: 5)
🔄 Starting Model Retraining Pipeline...
/home/min/p4/train_model.py:38: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objec
ts are not tested. Please consider using SQLAlchemy.
  df_new = pd.read_sql(sql, conn)
[18:16:40] WARNING: /workspace/src/learner.cc:790:
Parameters: { "use_label_encoder" } are not used.

✅ Model 20251219_181642 saved to models/model_20251219_181642.pkl
✅ Loaded Active Model: 20251219_181642 from models/model_20251219_181642.pkl
```

6.4 Model Governance Registry

The administrative view shows the history of model versions, indicating which model is currently "Active" in production.

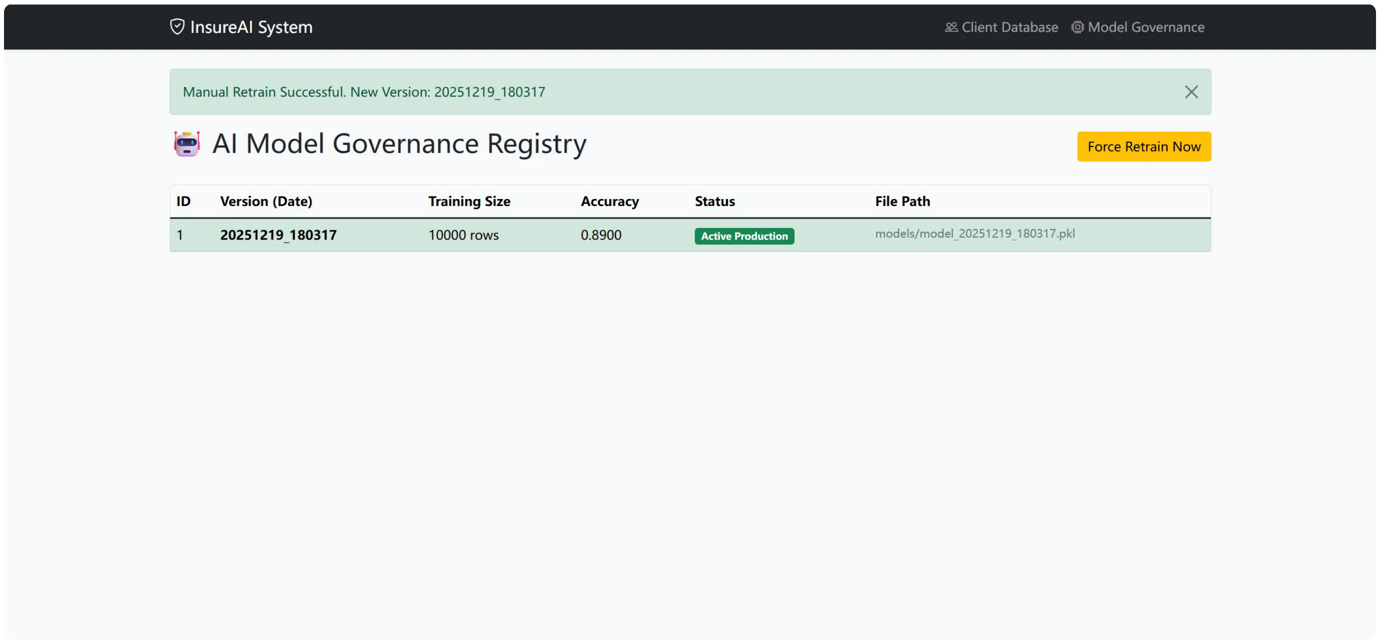


AI Model Governance Registry

ID	Version (Date)	Training Size	Accuracy	Status	File Path
2	**20251219_181642**	10005 rows	0.8900	Active Production	models/model_20251219_181642.pkl
1	**20251219_180317**	10000 rows	0.8900	Archived	models/model_20251219_180317.pkl

6.5 Manual Retrain Trigger (Optional)

In addition to the automated feedback loop, the system empowers administrators to manually initiate the retraining pipeline. This is particularly useful after bulk data uploads or immediate model updates.



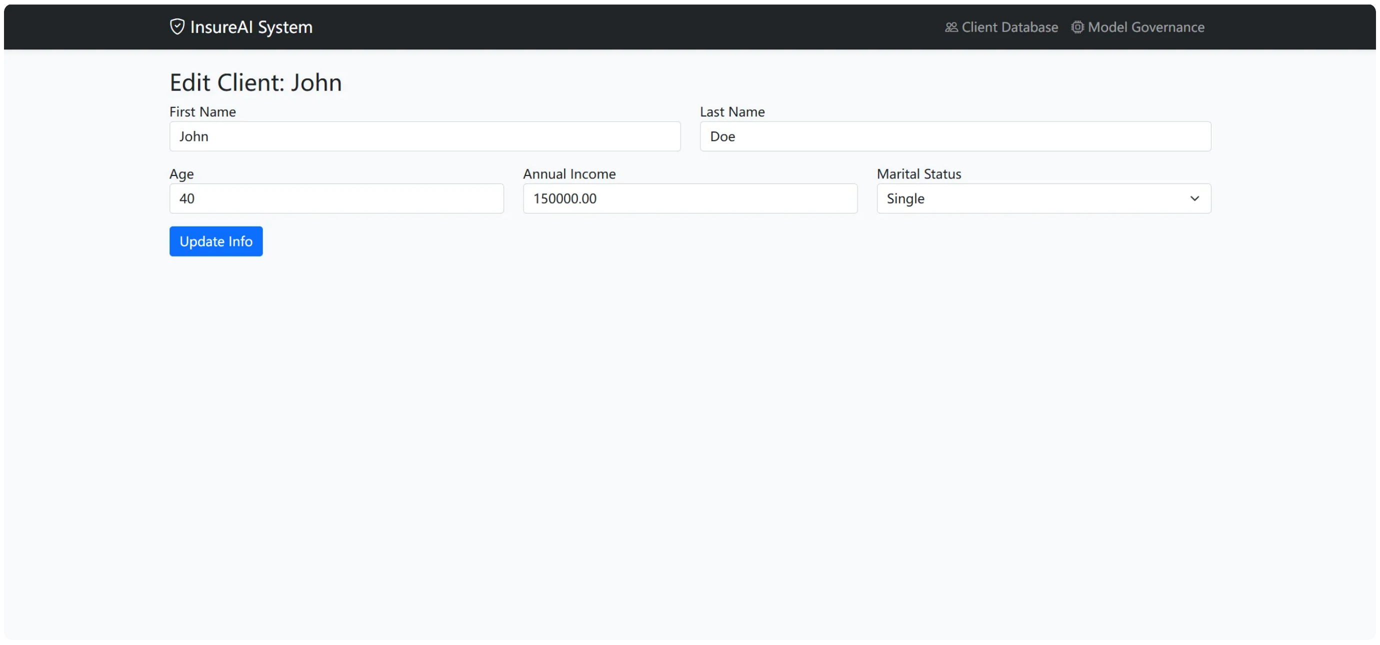
Manual Retrain Successful. New Version: 20251219_180317

AI Model Governance Registry

ID	Version (Date)	Training Size	Accuracy	Status	File Path
1	**20251219_180317**	10000 rows	0.8900	Active Production	models/model_20251219_180317.pkl

6.6 Data Correction & Updates

Real-world data is dynamic. The system provides a dedicated interface to update client demographics (e.g., significant salary increases or changes in marital status). These updates are immediately reflected in the Operational Data Store (ODS), ensuring that the data quality remains high for future training cycles.



Edit Client: John

First Name	Last Name	
John	Doe	
Age	Annual Income	Marital Status
40	150000.00	Single

7. Conclusion

This project successfully demonstrates a modern End-to-End Enterprise Architecture. By integrating a transactional relational database (OLTP) with an automated machine learning pipeline, we have created a system that not only manages data but actively learns from it. The use of advanced database techniques like partitioning and indexing, combined with a robust governance framework, ensures the system is performant, scalable, and reliable.