

GUARDNET

Final Documentation Report

Agentic Fraud Detection Framework for Nepali Digital Banking

Prepared for: GIBL AI/ML Hackathon 2026, Track B

Project: Agentic Fraud Detection Framework for Nepali Digital Banking

Prepared by: Team IGOR

Table of Content

1. Executive Summary.....	4
2. Repository Map.....	5
Key implementation files.....	6
3. Dataset and Track B Rules.....	7
Dataset scale.....	7
Key constraints and data policy.....	7
Embedded fraud patterns used by GuardNet.....	8
4. System Architecture.....	9
High-level flow.....	9
Runtime components.....	10
Startup guarantees.....	10
5. Database Design.....	10
Source-data tables.....	11
Runtime/audit tables.....	11
Ingestion behavior.....	11
6. Transaction Verification API.....	12
Main endpoints.....	12
Transaction context loading.....	12
Audit behavior.....	13
7. Agent Methodology.....	13
7.1 Velocity Agent.....	13
7.2 Geo Agent.....	13
7.3 Graph Agent.....	14
7.4 ML Agent.....	14
7.5 Synthesis Agent.....	16
8. Feature Engineering.....	17
Feature groups.....	17
Missing and noisy data handling.....	18
9. OTP Interlock.....	18
10. Streamlit Frontend.....	19
Demo scenarios.....	19
Hidden-test live stream.....	19
11. Training and Evaluation Methodology.....	20
Training script.....	20
Evaluation scripts.....	20
Threshold policy.....	21
12. Benchmarks and Current Results.....	21
12.2 Honest pre-refit OOT diagnostics.....	21
12.3 Post-refit ROC diagnostic.....	22
12.6 Latency benchmark.....	23

12.7 Feature importance.....	24
12.8 Fraud-type intervention recall.....	25
13. Submission Artifact.....	25
14. Testing.....	26
15. Security, Compliance, and Operational Notes.....	26
Positive controls.....	26
Production hardening items.....	26
16. Known Limitations and Risks.....	27
17. Reproducibility Runbook.....	27
17.1 Start infrastructure.....	27
17.2 Install dependencies.....	27
17.3 Ingest Track B data.....	27
17.4 Train model.....	28
17.5 Run API.....	28
17.6 Run frontend.....	28
17.7 Generate evaluation artifacts.....	28
17.8 Run tests.....	28
18. Conclusion.....	29

1. Executive Summary

GuardNet is a production-style fraud detection prototype for digital banking transactions. It combines rule-aware agentic scoring, supervised machine learning, unsupervised anomaly monitoring, graph analysis, and OTP interlock logic into a FastAPI service backed by PostgreSQL and Redis. The project targets Track B's fraud detection setting: high-volume Nepali banking transactions across eSewa, Khalti, card POS, ATM, RTGS, SWIFT, mobile top-up, and utility bill channels.

The final implementation is organized around four fraud-specialist agents:

Agent	Main responsibility	Core signals
Velocity Agent	Burst, structuring, frequency, and behavioral velocity	txn_count_*, z_score_amount, night_flag, structuring thresholds
Geo Agent	Location, IP, VPN/Tor, impossible travel, device-location mismatch	prev_txn_km, prev_txn_time_delta_min, ip_country, is_vpn, is_tor
ML Agent	XGBoost fraud probability with Isolation Forest anomaly monitoring	262 engineered features from transaction, customer, device, geo, velocity, graph, and baseline tables
Graph Agent	Fraud-seed, mule, reciprocal layering, shared-device, and graph-neighborhood signals	account_graph_nodes, account_graph_edges, runtime device-account graph

The Synthesis Agent combines these outputs with transaction-type-specific weights and applies controlled overrides for high-confidence patterns such as impossible travel, fraud-seed merchants, rooted locale mismatch, dormancy with high z-score, and direct graph/velocity mule evidence. It outputs one of:

Final score band	Decision	Meaning
< 0.40	ALLOW	Proceed without additional friction
0.40 - 0.75	OTP_ONLY	Step-up authentication
>= 0.75	BLOCK_AND_OTP	High-risk block/interlock workflow
SIM-swap escalation	ESCALATED	Manual/SOC escalation path

The project aligns with the research paper in `important_infos/GuardNet.md`, while the final code uses PostgreSQL as the canonical persistent database rather than the earlier SQLite prototype language.

2. Repository Map

```
.
|-- README.md
|-- requirements.txt
|-- docker-compose.yml
|-- FINAL_DOCUMENTATION_REPORT.md
|-- frontend/
|   |-- main.py
|   `-- app.py
|-- src/
|   |-- main.py
|   |-- api/routes.py
|   |-- agents/
|   |   |-- velocity_agent.py
|   |   |-- geo_agent.py
|   |   |-- ml_agent.py
|   |   |-- graph_agent.py
|   |   `-- synthesis_agent.py
|   |-- core/
|   |   |-- config.py
|   |   `-- otp.py
|   |-- db/
|   |   |-- models.py
|   |   `-- session.py
|   |-- ml/
|   |   |-- features.py
|   |   `-- artifact.py
|   `-- schemas/models.py
|-- scripts/
|   |-- ingest_track_b_to_postgres.py
|   |-- train_xgb_model.py
|   |-- eval_classification.py
|   |-- plot_roc_curve.py
|   |-- generate_submission.py
|   `-- benchmark_backend.py
|-- models/
|   `-- feature_columns.json
|-- reports/
|   |-- model_metrics.json
|   |-- roc_metrics.json
|   |-- eval_metrics.json
|   |-- classification_report.txt
|   |-- backend_benchmark_redis.json
|   |-- feature_importance.csv
|   `-- submission_guardnet.csv
|-- tests/
|   `-- test_agents.py
`-- important_infos/
```

```
|-- GuardNet.md
`-- Track B/DATA_DESCRIPTION_Track_B.md
```

Key implementation files

File	Purpose
src/main.py	FastAPI app startup, Redis check, PostgreSQL initialization, NetworkX graph load, model artifact load
src/api/routes.py	Transaction verification, dashboard APIs, hidden-test stream, OTP verify, audit persistence
src/agents/synthesis_agent.py	Async orchestration and final decision policy
src/agents/velocity_agent.py	Redis/historical velocity, z-score, structuring and burst rules
src/agents/geo_agent.py	Geolocation and device context rules
src/agents/ml_agent.py	Model artifact inference and ML agent formatting
src/agents/graph_agent.py	Account graph and runtime device-account graph scoring
src/ml/features.py	Batch and row-level feature engineering contract
src/ml/artifact.py	XGBoost plus Isolation Forest scoring bundle
src/core/otp.py	OTP issuance, hashing, TTL, verification
frontend/app.py	Streamlit wallet, SOC dashboard, and live fraud stream UI
scripts/train_xgb_model.py	Training script for current model bundle
scripts/generate_submission.py	Hidden/evaluation prediction CSV generation
scripts/eval_classification.py	Tiered classification reports
scripts/plot_roc_curve.py	ROC/PR curves and score distribution outputs
scripts/benchmark_backend.py	Backend orchestration latency benchmark

3. Dataset and Track B Rules

The implementation is built around Track B's synthetic banking dataset, documented in [important_infos/Track B/DATA_DESCRIPTION_Track_B.md](#).

Dataset scale

File/table	Rows	Role
transactions_raw.csv	2,000,000	Canonical transaction table
customer_profiles.csv	50,000	KYC, demographic, account profile
device_fingerprints.json	200,000	Device, OS, locale, VPN/Tor/root flags
geo_events.csv	2,000,000	IP, location, travel, datacenter signals
velocity_snapshots.csv	2,000,000	As-of transaction velocity and z-score features
otp_logs.csv	~43,000	OTP challenge events, about 2.2 percent of transactions
account_graph_nodes.csv	50,001	Account graph node metadata and fraud seeds
account_graph_edges.csv	500,000	Directed money-movement graph edges
fraud_labels_train.csv	400,000	Released labels for training/validation
fraud_labels_eval_HIDDEN.csv	1,600,000	Withheld evaluation rows
rule_engine_baseline_predictions.csv	2,000,000	Legacy rule engine baseline
model_verdicts_sample.json	100,000	Reference multi-agent verdict sample

Key constraints and data policy

Rule	GuardNet handling
Hidden evaluation labels must not be used for training	Training query reads only fraud_labels, loaded from fraud_labels_train.csv
fraud_labels_eval_HIDDEN.csv is withheld for final scoring	Frontend live stream uses rows not present in fraud_labels and does not display ground truth
Device, merchant category, and notes fields are sparse	Feature pipeline imputes missing values and includes presence/missing flags
OTP logs are post-decision feedback	Training script removes otp_* features from the train/serve feature contract

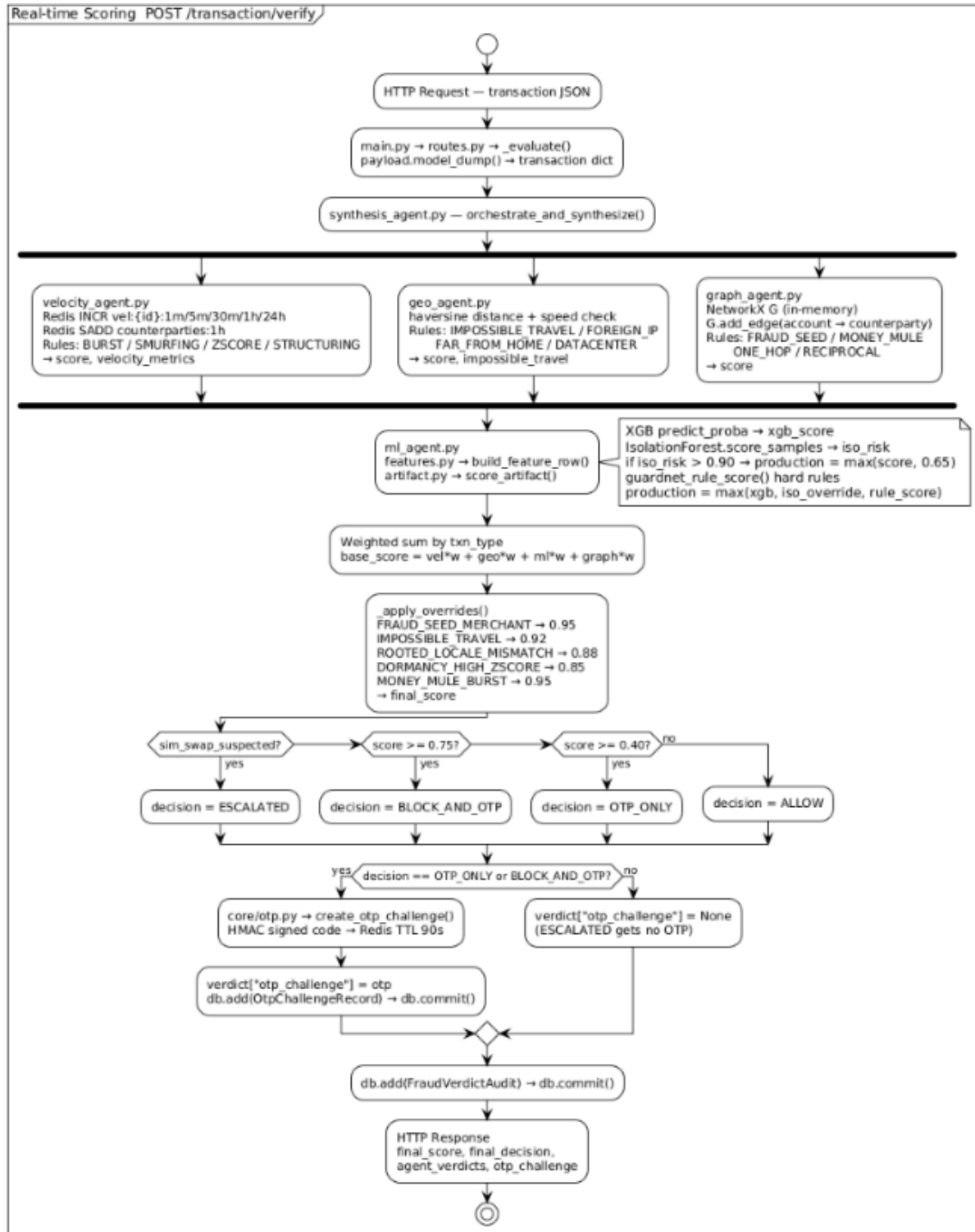
velocity_snapshots are as-of features	Historical replay uses these snapshots instead of repeated Redis increments
Graph edges are directed money movement	Graph agent loads account graph from account_graph_nodes and account_graph_edges

Embedded fraud patterns used by GuardNet

Track B pattern	Implementation response
Structuring near NPR 9,999 / 49,999 / 99,999	Velocity and feature pipeline include threshold bands
Fraud merchant IDs: MERCH-8812, MERCH-9041, MERCH-7712	Graph/synthesis fraud-seed merchant override
Night account-takeover bursts	Velocity agent flags txn_count_1m >= 3 with night_flag
Rooted device plus en_US locale on Nepali IP	Geo and synthesis rooted/locale mismatch rule
Dormancy break plus high z-score	Velocity and synthesis dormancy/high-z override
New counterparty / recent beneficiary pattern	Velocity and feature pipeline include new_counterparty_flag
COMM-042 / graph smurfing community	Graph node/edge features and NetworkX graph scoring

4. System Architecture

High-level flow



Runtime components

Component	Technology	Role
API	FastAPI/Uvicorn	Transaction verification, dashboard APIs, OTP verification
Persistent store	PostgreSQL 16	Canonical Track B dataset, audit rows, OTP challenge records
Runtime state	Redis 7	Velocity counters for real-time transactions and OTP TTL storage
Graph layer	NetworkX	Account money graph and runtime device-account graph
ML layer	XGBoost, scikit-learn IsolationForest, joblib	Fraud probability and anomaly monitoring
Frontend	Streamlit, Plotly	Demo wallet, SOC analytics, hidden-test live stream
Testing	pytest, pytest-asyncio	Agent and feature contract tests

Startup guarantees

src/main.py intentionally fails fast if required production dependencies are absent:

Dependency	Startup behavior
PostgreSQL	init_db() creates schema if needed; data must be ingested
Redis	ping() must succeed
Graph tables	API raises an error if no graph edges are loaded
Model artifact	API raises an error if models/xgb_fraud.pkl is missing or invalid

This prevents the service from silently falling back to hardcoded demo logic.

5. Database Design

The ORM schema in src/db/models.py mirrors Track B's structured tables and adds operational audit tables.n

Source-data tables

Table	Primary key	Description
customer_profiles	account_id	Customer KYC, profile, income, channel, risk tier
device_fingerprints	device_id	Device metadata, locale, root/VPN/Tor/shared-device flags
transactions_raw	txn_id	Canonical transaction record
geo_events	txn_id	Transaction geolocation and travel features
velocity_snapshots	txn_id	As-of velocity features
otp_logs	otp_event_id	Historical OTP challenge outcomes
account_graph_nodes	id	Account node metadata and fraud seeds
account_graph_edges	auto id	Directed money movement edges
fraud_labels	txn_id	Released training labels
rule_engine_baseline_predictions	txn_id	Legacy baseline rule outputs

Runtime/audit tables

Table	Purpose
fraud_verdict_audit	Stores each GuardNet verdict, score, overrides, and serialized agent scores
otp_challenges	Stores OTP challenge metadata, channels, status, and expiry

Ingestion behavior

scripts/ingest_track_b_to_postgres.py:

- Loads all Track B CSV/JSON tables into PostgreSQL.
- Normalizes datetime and boolean columns.
- Deduplicates known duplicate source keys before upsert.
- Applies compatibility fixes for graph-table relationships.
- Creates runtime indexes for transaction, graph, OTP, geo, and velocity lookups.
- Loads only fraud_labels_train.csv into fraud_labels.

6. Transaction Verification API

Main endpoints

Method	Endpoint	Purpose
GET	/health	Check Redis, database, graph, and model status
POST	/transaction/verify	Score a supplied transaction context
POST	/transaction/{txn_id}/verify	Load transaction context from PostgreSQL and score it
POST	/transaction	Acknowledge frontend wallet transaction outcome without inserting synthetic rows
POST	/otp/verify	Verify a pending OTP challenge
GET	/api/dashboard/metrics	Dashboard summary metrics
GET	/api/dashboard/profiles	Streamlit account profile controls
GET	/api/dashboard/transactions	Recent reviewable transactions
GET	/api/dashboard/stream-count	Hidden-test stream count
GET	/api/dashboard/stream-transactions	Hidden-test rows for live stream

Transaction context loading

`load_transaction_context()` builds a complete request object by joining:

- `transactions_raw`
- `velocity_snapshots`
- `geo_events`
- `customer_profiles`
- `device_fingerprints`
- `rule_engine_baseline_predictions`
- `otp_logs`
- `account_graph_nodes`
- `account_graph_edges`
- SQL-derived historical/as-of features

For PostgreSQL-loaded historical rows, `historical_replay=True` is set. This instructs the Velocity Agent to trust stored Track B velocity snapshots rather than increment Redis counters repeatedly during replay/demo scoring.

Audit behavior

Every verified transaction is persisted into fraud_verdict_audit with:

- txn_id
- account_id
- final score
- final decision
- base score before overrides
- override list
- serialized agent verdicts
- timestamp

7. Agent Methodology

7.1 Velocity Agent

File: src/agents/velocity_agent.py

The Velocity Agent detects high-risk temporal and amount-pattern anomalies.

Signal	Rule behavior
txn_count_1m >= 3 and night_flag=True	Critical burst-night score
txn_count_5m >= 6	High burst score
txn_count_30m >= 8	Elevated burst score
txn_count_1h >= 20 or unique_counterparties_1h >= 8	Smurfing activity
txn_count_1h >= 10 or unique_counterparties_1h >= 4	Medium activity warning
Historical daily baseline deviation	Medium warning
z_score_amount >= 3.5	Critical unusual amount
Amount near NPR 9,999 / 49,999 / 99,999	Structuring pattern
new_counterparty_flag	New recipient risk
Dormancy break	Dormant account reactivation risk

Real-time transactions use Redis INCR plus TTL keys across 1-minute, 5-minute, 30-minute, 1-hour, and 24-hour windows. Historical Track B replay uses the stored velocity_snapshots.

7.2 Geo Agent

File: src/agents/geo_agent.py

The Geo Agent handles location and network-context risk:

- Recomputes impossible-travel speed when distance and time delta exist.
- Flags speeds above 900 km/h.
- Compares location against home-district centroid.
- Handles unusable/private/bogus IPs.
- Scores VPN, Tor, foreign IP, domestic anonymized IP, and datacenter IP.
- Detects rooted Android/iOS device with en_US locale on Nepali IP.
- Checks timezone mismatch against Asia/Kathmandu.

7.3 Graph Agent

File: src/agents/graph_agent.py

The Graph Agent loads the account money-movement graph from PostgreSQL:

- Nodes: account_graph_nodes
- Edges: account_graph_edges
- Fraud seeds: is_fraud_seed=True

Runtime checks include:

Graph rule	Purpose
Direct fraud seed counterparty	High risk transfer target
Source account fraud seed	Known compromised or confirmed fraud source
One-hop/two-hop fraud seed	Nearby graph-risk propagation
Money mule ratio	High incoming degree and high in/out ratio
Reciprocal layering	Possible laundering loop
Shared device warning	Multiple accounts on same device
Runtime device bridge	Device-account links seen during live operation
High-risk merchant	Track B fraud merchant list

Important anti-leakage implementation detail: the graph no longer bootstraps device-account links from all 2,000,000 transactions. Device-account links are added as transactions are evaluated, avoiding future/global device leakage in live hidden-test replay.

7.4 ML Agent

File: src/agents/ml_agent.py

The ML Agent loads a joblib artifact containing:

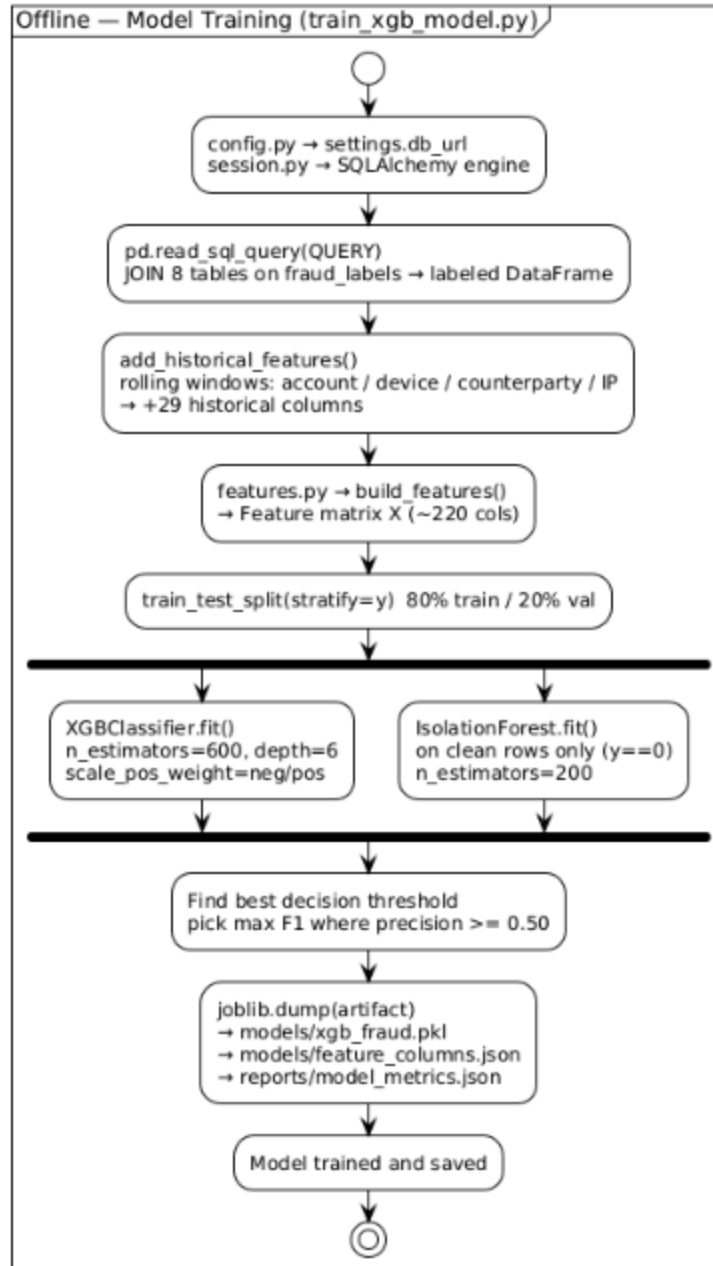
- XGBoost classifier
- Isolation Forest anomaly model
- Feature column contract

- Calibration bounds for isolation risk
- Threshold metadata
- Artifact version and combination description

The production score is computed by `src/ml/artifact.py`:

```
production_probability =  
    (xgb_probability * xgboost_weight + isolation_risk * isolation_weight)  
    / (xgboost_weight + isolation_weight)
```

In the current artifact, XGBoost is the primary model and Isolation Forest is monitored but not weighted in the production probability.



7.5 Synthesis Agent

File: `src/agents/synthesis_agent.py`

The Synthesis Agent runs Velocity, Geo, and Graph concurrently using `asyncio.gather()`, then runs the ML Agent using Velocity Agent metrics. It applies transaction-type-specific weights:

Transaction type	Velocity	Geo	ML	Graph
ESEWA_P2P	0.20	0.35	0.35	0.10
KHALTI_QR	0.35	0.20	0.40	0.05
CARD_POS	0.30	0.25	0.40	0.05
ATM_WITHDRAWAL	0.35	0.30	0.25	0.10
SWIFT_OUTWARD	0.20	0.30	0.30	0.20
RTGS	0.20	0.20	0.30	0.30
MOBILE_TOPUP	0.40	0.15	0.40	0.05
UTILITY_BILL	0.35	0.20	0.40	0.05
Default	0.30	0.25	0.30	0.15

Overrides are limited to high-confidence conditions:

- Fraud-seed merchant
- Impossible travel
- Rooted device plus locale mismatch
- Dormancy plus high z-score
- Direct fraud-seed/mule graph evidence plus sufficient velocity evidence
- Sim-swap escalation

8. Feature Engineering

File: src/ml/features.py

The current feature contract contains 262 features. These features are built consistently in batch training/evaluation and row-level real-time inference.

Feature groups

Feature group	Examples
Amount and velocity	log_amount, z_score_amount, txn_count_1m, txn_count_24h, unique_counterparties_1h
Structuring	near_structuring_threshold, near_structuring_9999, near_structuring_49999, near_structuring_99999
Temporal	hour_sin, hour_cos, late_night_flag, weekend_flag, night_flag
Historical/as-of behavior	hist_account_*, hist_device_*, hist_counterparty_*, hist_pair_*, hist_ip_*
Geo/network	prev_txn_km, prev_txn_time_delta_min, impossible_travel_speed_kmph, is_vpn, is_tor, is_datacenter

Device	is_rooted_or_jailbroken, is_shared_device, locale_mismatch, device_age_days, app version parts
Customer profile	risk_tier, kyc_tier, income band, occupation, district, province, channel preference
Rule baseline	baseline_flag, baseline_confidence, baseline_rule_triggered_*
Graph	source/counterparty degrees, graph ratios, fraud seed flags, reciprocal/first-transfer flags
Composite fraud profiles	account takeover, SIM swap, C2 exfiltration, money mule, synthetic identity, insider threat
One-hot categoricals	transaction type, channel, auth method, response code, MCC, OS, ISP, ASN, district, income band

Missing and noisy data handling

The feature builder converts missing values into controlled defaults and explicit flags:

- Null merchant category -> missing category handling.
- Null device ID -> default device features plus missing/presence signals.
- Empty notes -> notes_present=0.
- Private or bogus IPs -> handled by geo rules.
- Infinite and NaN feature values -> replaced before model scoring.

9. OTP Interlock

File: src/core/otp.py

GuardNet uses Redis-backed OTP state:

- Challenge ID format: OTP-XXXXXX
- Code: 6-digit random value
- Storage: Redis key otp:{challenge_id}
- TTL: GUARDNET_OTP_TTL_SECONDS, default 90 seconds
- Hashing: SHA-256 of challenge ID, code, and GUARDNET_OTP_SECRET
- Attempts: maximum 3
- OTP_ONLY: SMS channel
- BLOCK_AND_OTP: SMS plus EMAIL channel metadata

The current demo can include the OTP code in the API response if

GUARDNET_SHOW_DEMO_OTP_CODE=1. This is useful for local demonstration but must be disabled in production.

10. Streamlit Frontend

Files:

- frontend/main.py
- frontend/app.py

The final frontend is Streamlit-based and calls FastAPI directly. It provides three views:

View	Purpose
Customer Wallet	Simulated banking wallet transfer flow with live GuardNet verification and OTP challenge
SOC Dashboard	API-backed metrics, recent transaction feed, selected transaction inspection
Live Fraud Detection	Hidden-test transaction stream from PostgreSQL, without showing hidden ground truth labels

Demo scenarios

Scenario	Intent
Scenario 1: Standard transfer	Low-risk domestic peer transfer
Scenario 2: Account takeover	Watchlist/student account, web/VPN, higher-risk context
Scenario 3: Mule risk	High-risk recipient, Tor/VPN/rooted context
Custom Sandbox	Manual override controls for transaction, network, device, and baseline telemetry

Hidden-test live stream

The live stream queries:

```
GET /api/dashboard/stream-count
GET /api/dashboard/stream-transactions
POST /transaction/{txn_id}/verify
```

The stream intentionally excludes rows in released fraud_labels and returns label_available=False. This allows a judge-facing demonstration of the final-test stream without displaying hidden ground truth.

11. Training and Evaluation Methodology

Training script

File: scripts/train_xgb_model.py

The training pipeline:

1. Reads released labeled rows from PostgreSQL.
2. Joins transaction, profile, device, geo, velocity, baseline, graph, and OTP tables.
3. Excludes otp_* columns from the deployed feature contract because OTP is post-decision feedback.
4. Builds historical/as-of behavioral features.
5. Splits by time: 70 percent train, 15 percent validation, 15 percent test.
6. Trains XGBoost with class imbalance handling using scale_pos_weight.
7. Trains Isolation Forest as a secondary anomaly monitor.
8. Selects validation-calibrated OTP and block thresholds under operational constraints.
9. Reports pre-refit validation/test metrics.
10. Refits the deployable artifact on all released training labels.
11. Calibrates deployable thresholds on the refit artifact.
12. Saves output artifacts (see below).

Saved outputs:

- models/xgb_fraud.pkl
- models/feature_columns.json
- reports/model_metrics.json
- reports/feature_importance.csv

Evaluation scripts

Script	Purpose
scripts/eval_classification.py	Generates tiered decision reports and confusion matrices
scripts/plot_roc_curve.py	Generates ROC/PR curves, score distributions, and ROC metrics
scripts/generate_submission.py	Generates final evaluation CSV
scripts/benchmark_backend.py	Benchmarks backend transaction scoring latency

Threshold policy

The research paper's conceptual synthesis thresholds are:

```
score < 0.40    -> ALLOW
0.40 to 0.75    -> OTP_ONLY
score >= 0.75   -> BLOCK_AND_OTP
```

The raw ML probability thresholds are calibrated separately because model probabilities are not identical to the synthesis-agent conceptual score. Current model artifact thresholds in reports/model_metrics.json:

```
otp_threshold = 0.497090
block_threshold = 0.599155
```

12. Benchmarks and Current Results

12.2 Honest pre-refit OOT diagnostics

Source: reports/model_metrics.json

These are the most honest currently available released-label diagnostics because they are computed before the final production refit on all released labels.

Split	Rows	Fraud rows	AUROC	AUPRC	Precision	Recall	F1	Recall@1% FPR	Recall@5% FPR
Validation	60,000	1,108	0.574193	0.103585	0.743750	0.107401	0.187697	0.114621	0.146209
Test	60,000	1,073	0.556052	0.096537	0.713376	0.104380	0.182114	0.110904	0.140727

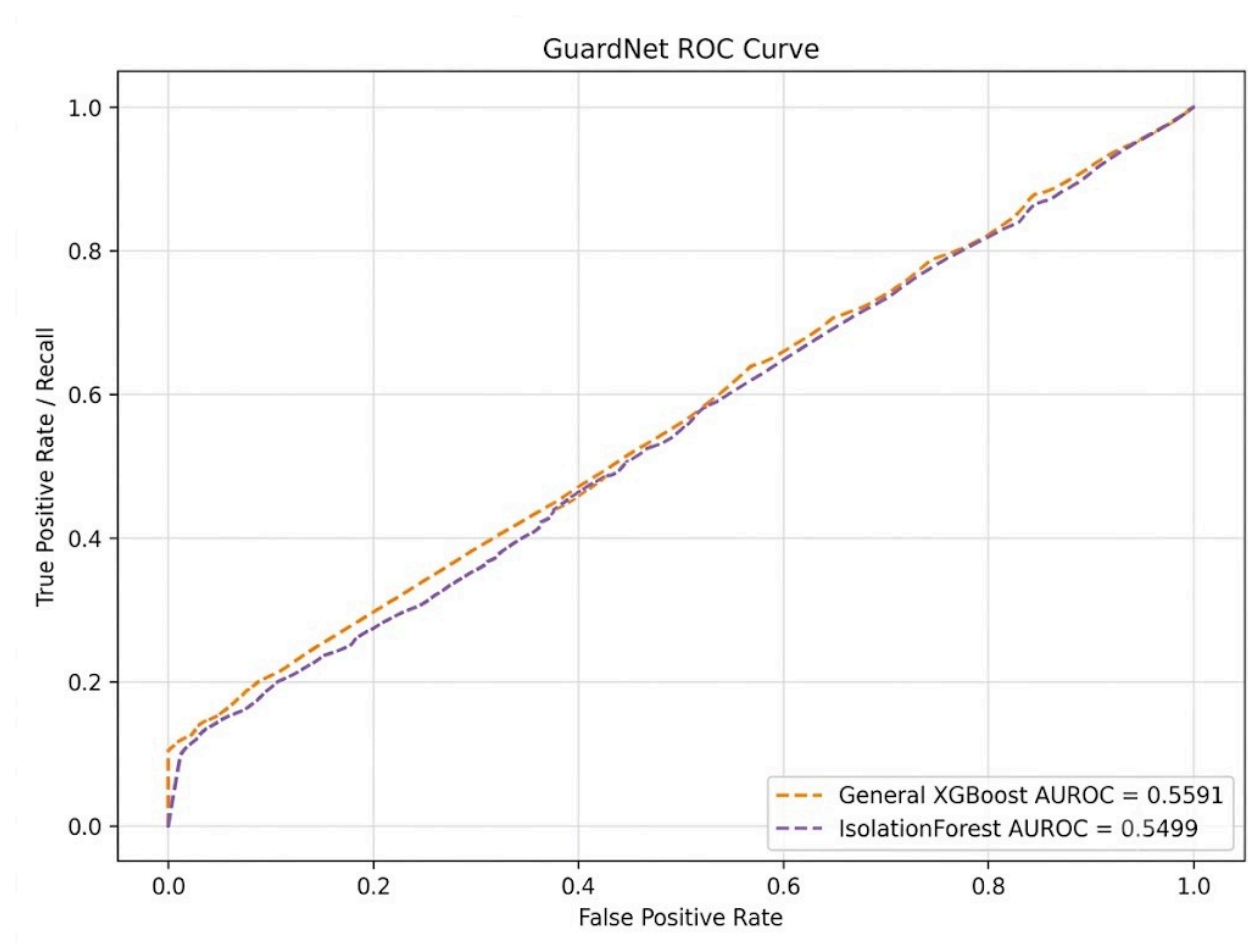
Interpretation: the current model is operationally conservative and has useful precision in a narrow hard-block band, but its honest OOT ranking is below the official legacy-rule-engine AUROC baseline. This is a known limitation and a model-optimization target for later iterations.

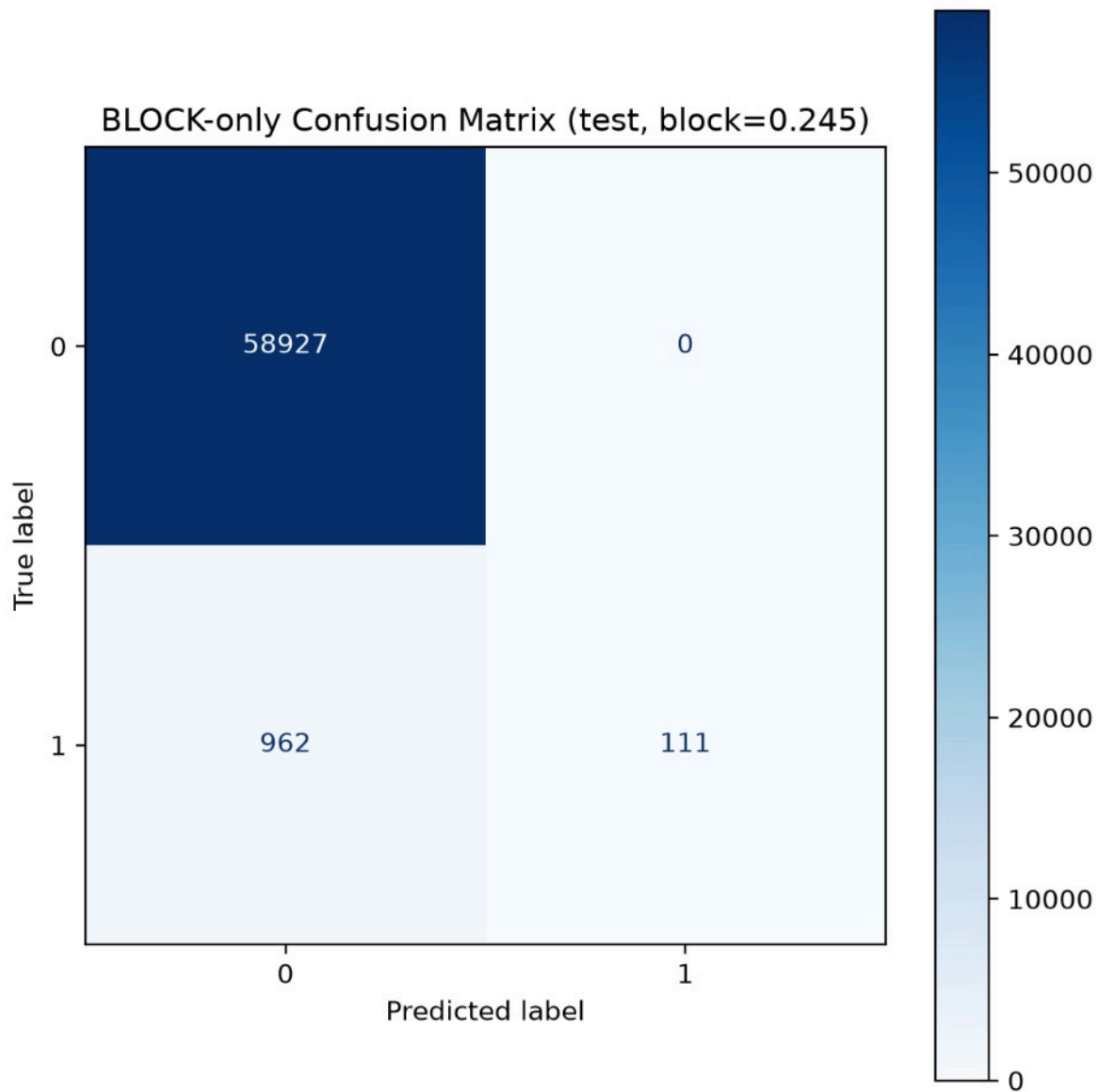
12.3 Post-refit ROC diagnostic

Source: reports/roc_metrics.json

Metric	Value
XGBoost AUROC	0.5591
Isolation Forest AUROC	0.5499

Important caveat: this file evaluates the saved production-refit artifact on a released-label test slice after the artifact was refit on all released labels. It is useful as a fit/score-distribution diagnostic but should not be presented as the honest holdout score.





12.6 Latency benchmark

Source: reports/backend_benchmark_redis.json

Metric	Value
Cache backend	Redis
Measured sample size	150
Warmup size	20
Context load p50	15.2097 ms

Context load p95	17.9416 ms
Pipeline wall p50	11.3563 ms
Pipeline wall p95	13.8743 ms
Reported pipeline p95	13.0 ms
Velocity agent p95	2.0 ms
Geo agent p95	1.0 ms
ML agent p95	11.0 ms
Graph agent p95	1.0 ms

This is well below the Track B latency target of p95 under 800 ms for the benchmarked sample.

12.7 Feature importance

Source: reports/feature_importance.csv

Top current XGBoost features by importance:

Rank	Feature	Importance
1	is_shared_device	18659.1484
2	ip_isp_ROSTELECOM	5469.6929
3	txn_count_1m	4891.6152
4	ip_asn_AS12389	2563.4653
5	notes_GROCERY	328.5955
6	district_LALITPUR	318.6737
7	baseline_decision_FLAG	308.8636
8	mcc_4814	304.9135
9	age_group_45_54	302.0107
10	hist_asn_txn_count_24h	298.5972

12.8 Fraud-type intervention recall

Source: reports/fraud_type_tiered_recall_test.csv

Fraud type	Support	Blocked	OTP only	Intervention recall
C2_EXFILTRATION	96	19	59	0.812500
MONEY_MULE	133	28	76	0.781955
SIM_SWAP	115	18	69	0.756522
INSIDER_THREAT	111	19	64	0.747748
SOCIAL_ENGINEERING	112	16	67	0.741071
SYNTHETIC_IDENTITY	101	14	60	0.732673
CARD_NOT_PRESENT	96	11	58	0.718750
SMURFING	108	19	58	0.712963
ACCOUNT_TAKEOVER	92	21	43	0.695652
FIRST_PARTY_FRAUD	109	13	56	0.633028

13. Submission Artifact

The current submission generator is scripts/generate_submission.py.

Expected output:

```
reports/submission_guardnet.csv
```

Submission columns:

Column	Description
txn_id	Evaluation transaction ID
fraud_probability	Production fraud probability
fraud_decision	ALLOW, OTP_ONLY, or BLOCK
fraud_type_predicted	Currently blank placeholder
latency_ms	Static script value of 12 ms

The generator supports hidden-label metrics only with an explicit flag. By default, it writes unlabeled run metadata and avoids hidden-ground-truth evaluation.

14. Testing

File: tests/test_agents.py

Current test coverage includes:

Test	What it verifies
test_velocity_flags_night_burst_and_zscore	Night burst and z-score velocity rules
test_geo_flags_impossible_travel	Impossible travel detection
test_ml_requires_model_artifact	ML agent fails if no model is loaded
test_ml_scores_with_loaded_model	XGBoost and Isolation Forest scoring contract
test_feature_row_uses_track_b_names	Feature row naming and important Track B feature transforms

Last known test result from current work session:

5 passed

15. Security, Compliance, and Operational Notes

Positive controls

- API startup requires Redis, PostgreSQL, graph tables, and a valid model artifact.
- OTP codes are hashed in Redis, not stored as raw code in the challenge record.
- OTP challenge state expires via Redis TTL.
- Verdicts are persisted for auditability.
- Synthetic wallet demo transactions are not inserted into canonical Track B source tables.
- Hidden-test stream intentionally hides ground truth labels.
- Historical replay mode avoids corrupting velocity counters.

Production hardening items

Area	Current status	Production recommendation
CORS	allow_origins=["*"]	Restrict to approved frontend origins
OTP delivery	Demo/provider-pending metadata	Integrate real SMS/email provider and disable demo code
Secrets	Environment variables supported	Use vault/secret manager
Model explainability	Agent rules and feature importance available	Add per-transaction SHAP/LIME for bonus criteria

Monitoring	Benchmark and audit files exist	Add structured logs, dashboard alerts, and drift monitoring
Data leakage controls	OTP features excluded, historical replay added	Re-audit graph and feature construction before final scoring

16. Known Limitations and Risks

13. Honest pre-refit AUROC is currently below the official rule-engine baseline.

- Current released-label OOT test AUROC: 0.556052.
- Official rule-engine baseline AUROC: 0.710.
- Model improvement is explicitly planned for later iterations.

14. Metric artifacts have mixed provenance and timestamps.

- `model_metrics.json`, `roc_metrics.json`, `classification_report.txt`, and `eval_metrics.json` were not all generated from the same exact artifact timestamp.
- Before final judge submission, regenerate metrics from the final locked model.

15. Post-refit 0.935013 AUROC must not be claimed as honest holdout performance.

- It is a post-refit released-label diagnostic.

16. Fraud-type prediction is not yet implemented in the submission CSV.

- The column is present but blank.

17. Explainability beyond feature importance and agent rule evidence is not yet implemented.

- SHAP/LIME artifact is absent.

18. The Streamlit interface is demonstration-grade.

- It is wired to FastAPI and PostgreSQL, but production SOC workflows would require authentication, role-based access, alert queues, and case management.

17. Reproducibility Runbook

17.1 Start infrastructure

```
docker compose up -d postgres redis
```

17.2 Install dependencies

```
python -m pip install -r requirements.txt
```

17.3 Ingest Track B data

Preflight:

```
python scripts/ingest_track_b_to_postgres.py --preflight
```

Full load:

```
python scripts/ingest_track_b_to_postgres.py --replace
```

17.4 Train model

Do not run this unless intentionally regenerating the model artifact.

```
python scripts/train_xgb_model.py
```

17.5 Run API

```
uvicorn src.main:app --host 0.0.0.0 --port 8000
```

17.6 Run frontend

```
streamlit run frontend/main.py
```

17.7 Generate evaluation artifacts

Classification report:

```
python scripts/eval_classification.py --split test
```

ROC/PR plots:

```
python scripts/plot_roc_curve.py --split test
```

Submission:

```
python scripts/generate_submission.py
```

Backend benchmark:

```
python scripts/benchmark_backend.py --cache redis
```

17.8 Run tests

```
python -m pytest -q
```

18. Conclusion

GuardNet implements a complete fraud detection prototype rather than a standalone notebook model. The system includes:

- PostgreSQL ingestion and normalized schema.
- FastAPI transaction verification service.
- Redis-backed velocity and OTP state.
- NetworkX account graph and runtime device-account graph checks.
- XGBoost-primary ML scoring with Isolation Forest monitoring.
- Transaction-type-aware synthesis logic.
- OTP interlock and audit persistence.
- Streamlit customer, SOC, and hidden-test live-stream interfaces.
- Submission generation and benchmark scripts.
- Unit tests for critical agent and feature behavior.

The current implementation is architecturally complete and operationally wired. Its strongest current engineering qualities are dependency strictness, end-to-end API/frontend integration, low-latency scoring, and honest separation of released labels from hidden-stream demonstration data. Its primary current weakness is model ranking performance on honest out-of-time diagnostics, which remains the main target for future model optimization.