SecurePay: A Multi-Layered AI Framework for Intelligent Fraud Detection and Adaptive Security in Digital Payment Systems

Rohan Jeyan
Dept. of Artificial Intelligence and Data Science, Sri Venkateswara College of Engineering, Chennai, India
Project Guide, Ms P.Uma, Dept. of Computer Science Engineering, Sri Venkateswara College of Engineering, Chennai, India

***Abstract*— The rapid growth of digital payment platforms has introduced significant challenges around transaction fraud, user authentication, and personalized financial services. Existing solutions often address these concerns in isolation, relying on single-model architectures that lack adaptability and layered protection. This paper presents SecurePay, a mobile payment system that integrates multiple AWS AI/ML services into a cohesive, multi-layered security and intelligence framework. The system combines gradient-boosted decision trees on Amazon SageMaker for real-time fraud scoring and promotional recommendation, Amazon Rekognition for biometric face authentication, and Amazon Bedrock with large language models for natural language transaction search, automated categorization, and personalized spending insights. All AI service calls are routed through serverless edge functions with rule-based fallbacks, isolating credentials from the client. Evaluation on synthetic datasets shows F1 = 1.00 and AUC-ROC = 1.00 for fraud detection with a median pipeline latency of 846 ms, and a macro F1 of 1.00 for promotional recommendation across five categories. Cost analysis shows the full AI stack operates at approximately $94.67 per month for 10,000 transactions.**

***Index Terms*— digital payments, fraud detection, biometric authentication, machine learning, cloud computing, serverless architecture, XGBoost, large language models.**

I. INTRODUCTION

Digital payment platforms have become a central part of how people handle money in cities, processing billions of transactions daily across mobile wallets, peer-to-peer transfer systems, and contactless payment networks. The Reserve Bank of India reported over 13 billion UPI transactions in a single month in 2024 [1], reflecting the scale at which these systems now operate. With this growth comes an equally rapid increase in fraudulent activity. The Indian Cyber Crime Coordination Centre documented a 300% rise in digital payment fraud complaints between 2021 and 2023 [2], which makes better fraud detection a pressing need. Volume statistics published by the National Payments Corporation of India corroborate this trend, with UPI continuing to be the dominant payment rail in India [19].

Traditional fraud detection approaches rely on static rule engines that flag transactions exceeding predefined thresholds. While straightforward to implement, these systems suffer from high false positive rates and an inability to adapt to evolving fraud patterns [3]. Machine learning models, particularly ensemble methods like gradient-boosted trees, have proven effective at detecting complex fraud patterns between transaction features [4]. However, deploying these models in production environments introduces its own set of challenges: model serving infrastructure must handle variable load, maintain low latency, and degrade gracefully when components fail.

Beyond fraud detection, modern payment applications face growing user expectations around personalization and convenience. Users expect intelligent categorization of their spending, actionable financial insights, and frictionless authentication methods. Meeting these expectations requires integrating multiple AI capabilities: classification models, large language models, and computer vision systems, into a

single system without it becoming difficult to maintain.

This paper presents SecurePay, a mobile digital payment system built with Flutter and Supabase that integrates a suite of AWS AI/ML services to address fraud detection, biometric authentication, and intelligent financial services within a unified framework. The primary contributions of this work are: (1) a multi-layered fraud detection pipeline that combines client-side heuristic pre-screening, cloud-hosted XGBoost scoring via SageMaker, biometric face verification for ambiguous cases, and natural language explanations generated by a large language model; (2) an architecture that routes all AI service interactions through serverless edge functions with manual AWS Signature Version 4 authentication, eliminating the need for cloud credentials on client devices; (3) a comparative evaluation of local PyTorch neural network inference against cloud-hosted XGBoost inference, examining the trade-offs in latency, accuracy, and operational cost; (4) a demonstration of large language model integration for transaction categorization, spending insight generation, and natural language search over financial data.

II. RELATED WORK

Fraud detection in digital payments has been extensively studied, with approaches ranging from statistical methods to deep learning architectures. Bhattacharyya et al. [5] demonstrated that random forests and support vector machines could effectively classify fraudulent credit card transactions, achieving AUC scores above 0.95 on the European cardholder dataset. More recently, gradient-boosted methods have gained traction; Chen and Guestrin's XGBoost [6] has become a standard baseline in fraud detection competitions and production systems due to its handling of missing values, built-in regularization, and interpretable feature importance rankings.

Deep learning approaches have also been explored. Fiore et al. [7] applied generative adversarial networks to address class imbalance in fraud datasets, while Roy et al. [8] used LSTM networks to capture temporal patterns in transaction sequences. However, these approaches typically require substantially more training data and computational resources than tree-based methods, and their latency characteristics can be problematic for real-time scoring. More recent work has explored graph neural networks for fraud detection on imbalanced transaction graphs [14] and transformer-based architectures for e-commerce fraud classification [16]. Rao et al. [15] further introduced an explainable fraud detection framework, which is particularly relevant for financial applications where interpretability influences regulatory acceptance.

Biometric authentication in financial applications has progressed from fingerprint-based systems to facial recognition. Adjabi et al. [9] surveyed face recognition techniques and noted that cloud-based services like Amazon Rekognition and Microsoft Azure Face API have made production-grade face matching accessible without requiring teams to train custom models. The integration of biometric verification as a secondary authentication factor in payment flows has not received much attention in published research. Standardization efforts such as the FIDO2 WebAuthn specification [17] provide a framework for strong authentication in web and mobile applications, and regulatory requirements like PCI DSS v4.0 [18] increasingly encourage multi-factor verification for high-risk transactions.

The application of large language models to financial data analysis is an emerging area. Wu et al. [10] demonstrated that GPT-based models could generate meaningful financial summaries from transaction data, while Yang et al. [11] explored LLM-powered natural language interfaces for database querying. We build on this by bringing LLM capabilities directly into a payment application for transaction categorization, spending insights, and natural language search.

Serverless architectures for ML inference have been examined by Ishakian et al. [12], who found that while serverless platforms introduce cold-start latency, they offer significant cost advantages for variable workloads compared to dedicated inference servers. Our architecture builds on this observation by using Supabase Edge Functions as a serverless intermediary between the mobile client and AWS AI services. Schleier-Smith et al. [20] further characterize the evolving role of serverless platforms as a general-purpose computing abstraction, which supports the use

of edge functions as a primary integration layer for cloud AI services.

III. SYSTEM ARCHITECTURE

SecurePay follows a four-layer architecture consisting of a Flutter mobile client, a Supabase backend with PostgreSQL and serverless edge functions, AWS AI/ML services, and a data storage and audit layer. Fig. 1 illustrates the high-level component interactions.

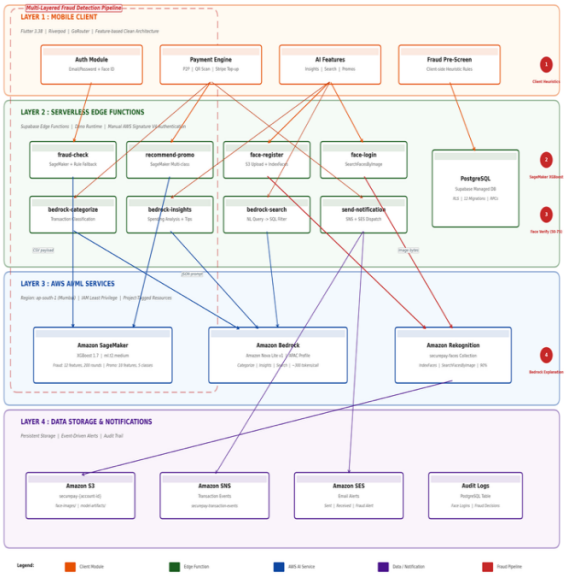

Fig. 1. SecurePay four-layer architecture. Layer 1 (Mobile Client) handles auth, payment, AI features, and client-side fraud pre-screening. Layer 2 (Serverless Edge Functions) mediates every AWS call with manual SigV4 signing. Layer 3 (AWS AI/ML Services) provides SageMaker XGBoost scoring, Bedrock Nova Lite reasoning, and Rekognition face matching. Layer 4 (Data Storage & Notifications) covers S3, SNS/SES, and audit trails. The dashed box highlights the multi-layered fraud detection pipeline spanning client heuristics → SageMaker scoring → face verification (review zone) → Bedrock explanation.

A. Mobile Client Layer

The mobile application is built with Flutter 3.38 using the Riverpod state management framework and GoRouter for navigation with authentication guards. The client implements a feature-based clean architecture with separate domain, data, and presentation layers for each functional module. Critically, the client holds no AWS credentials; all cloud AI interactions are mediated through Supabase Edge Functions.

A client-side fraud pre-screening module provides immediate feedback to users before transactions are submitted to the server. This module applies lightweight heuristic rules: checking transaction amount against daily limits, evaluating transaction frequency within sliding time windows, and flagging transfers to previously unseen recipients. While not a substitute for server-side ML scoring, this pre-screening reduces unnecessary API calls and provides sub-second user feedback.

B. Backend and Edge Function Layer

The backend uses Supabase, which provides a managed PostgreSQL database with row-level security (RLS) policies, authentication services, and a serverless edge function runtime based on Deno. Nine edge functions handle the various AI service integrations:

• fraud-check: Constructs a 12-dimensional feature vector from transaction history and invokes the SageMaker XGBoost endpoint. Falls back to a rule-based scoring engine if SageMaker is unreachable.

• recommend-promo: Builds a 10-dimensional user profile vector and queries the SageMaker promotional recommendation endpoint.

• face-register: Uploads a face image to Amazon S3 and indexes it in an Amazon Rekognition collection.

• face-login: Searches the Rekognition collection for a matching face with a 90% similarity threshold.

• bedrock-categorize: Invokes Amazon Nova Lite to classify transactions into spending categories and generate natural language fraud explanations.

• bedrock-insights: Aggregates user transaction statistics and prompts Amazon Nova Lite to produce personalized spending summaries and financial tips.

• bedrock-search: Translates natural language queries into structured database filters using Amazon Nova

Lite, executes the query, and optionally generates a summary of results.

• send-notification: Publishes transaction events to Amazon SNS and sends email alerts through Amazon SES based on user notification preferences.

Each edge function implements AWS Signature Version 4 request signing manually, as the Deno runtime does not support the AWS SDK. This required careful handling of URI encoding, particularly for Bedrock model identifiers containing special characters that interact with the SigV4 canonical request construction.

C. AWS AI/ML Service Layer

The system uses six AWS services, all deployed in the ap-south-1 (Mumbai) region:

• Amazon SageMaker hosts two real-time inference endpoints running XGBoost models on ml.t2.medium instances.

• Amazon Bedrock provides access to the Amazon Nova Lite foundation model through the APAC cross-region inference profile for natural language processing tasks.

• Amazon Rekognition maintains a face collection for biometric authentication.

• Amazon S3 stores face images and model artifacts.

• Amazon SNS and SES handle event-driven notifications.

An IAM policy scoped to the principle of least privilege governs all service access, with separate permissions for S3 object operations, SageMaker endpoint invocation, Bedrock model invocation, Rekognition face operations, and notification publishing.

D. Data Storage and Audit Layer

The Supabase-managed PostgreSQL instance serves as the primary data store with row-level security policies and RPC support for complex queries. All transaction data, user profiles, and system metadata are stored here. A dedicated audit_logs table maintains detailed audit trails including function execution logs and fraud decision records for compliance and debugging.

The architecture implements a four-stage fraud detection strategy: (1) Client Heuristics: lightweight rule-based pre-screening on the mobile client; (2) SageMaker XGBoost: cloud-based ML model scoring via edge functions; (3) Face Verification: biometric authentication for ambiguous transactions using Rekognition with a 90% similarity threshold; (4) Bedrock Explanation: natural language fraud reasoning generated by Amazon Nova Lite.

IV. MACHINE LEARNING MODELS

A. Fraud Detection Model

The fraud detection system uses a binary XGBoost classifier trained on 20,000 synthetic transactions with an 85:15 normal-to-fraud class ratio. The feature vector consists of 12 engineered features capturing transaction characteristics, user behavior patterns, and temporal signals. Table I lists the features.

Feature	Description
amount	Transaction amount in local currency
txn_count_1h	Number of transactions in the past hour
txn_count_24h	Number of transactions in the past 24 hours
daily_total	Cumulative transaction amount for the day
avg_txn_amount	Historical average transaction amount
amount_deviation	Relative deviation from user's average
amount_zscore	Z-score of amount relative to user history
is_new_recipient	Binary flag for first-time recipient (0/1)
hour_of_day	Hour of transaction (0–23)
day_of_week	Day of transaction (0–6)
account_age_days	Days since account creation
unique_recipients_24h	Distinct recipients in past 24 hours

Table I. Feature vector for the fraud detection model (12 dimensions).

The XGBoost model was trained with the following hyperparameters: 200 boosting rounds, maximum tree depth of 6, learning rate of 0.1, subsample ratio of 0.8,

column sample ratio of 0.8, and a scale_pos_weight of 5.67 to compensate for class imbalance. The evaluation metric was AUC-ROC.

Synthetic data generation followed established practices for fraud simulation [13]. Normal transactions were sampled from log-normal distributions calibrated to typical Indian digital payment patterns, with amounts ranging from ₹10 to ₹80,000. Fraudulent transactions were generated with elevated amounts, higher transaction frequencies, increased recipient novelty, and unusual temporal patterns. For comparison, a PyTorch neural network with batch normalization, residual connections, and dropout was also trained on the same dataset using Adam optimization with cosine learning-rate scheduling.

B. Promotional Recommendation Model

The promotional recommendation system uses a multi-class XGBoost classifier (objective: multi:softprob) trained on 15,000 synthetic user profiles to predict the most suitable promotional category from five options: welcome, loyalty, high_value, seasonal, and cashback. The feature vector consists of 10 user behavior metrics including total spending, transaction frequency, average transaction amount, recency, account age, top-up frequency, recipient diversity, weekly transaction rate, spending trend, and prior promotional claim count.

C. Large Language Model Integration

Amazon Nova Lite is used for three distinct natural language tasks. For transaction categorization, the model receives a batch of transaction descriptions and amounts and classifies each into one of eleven spending categories (food, transport, shopping, entertainment, bills, transfer, salary, investment, health, education, other). For spending insights, the model receives aggregated user statistics and generates a personalized summary, three actionable financial tips, and a risk assessment. For natural language search, the model parses user queries like "how much did I spend on food last week" into structured database filters with optional aggregation requests.

All LLM prompts use low temperature settings (0.1–0.5) to minimize variability, and responses are parsed with regex-based JSON extraction with fallback to default values when parsing fails.

V. COMPARISON WITH EXISTING SOLUTIONS

To position SecurePay relative to prior work, Table II contrasts it with five representative systems: a conventional rule-based fraud engine, the random-forest/SVM baseline of Bhattacharyya et al. [5], the GAN-augmented deep learning approach of Fiore et al. [7], the PaySim mobile-money simulator [13], and BloombergGPT [10] as an LLM-for-finance precedent. The dimensions compared are the core fraud model family, biometric authentication, LLM-based insights, graceful degradation, credential isolation, and the realistic monthly operating cost at the target workload.

System	Fraud Model	Biometric Auth	LLM Insights	Graceful Fallback	Credential Isolation	Multi-Layer
Rule-based engine	Static rules	No	No	—	N/A	No
Bhattacharyya et al. [5]	RF / SVM	No	No	No	N/A	No
Fiore et al. [7]	GAN + classifier	No	No	No	N/A	No
PaySim [13]	Simulator only	No	No	N/A	N/A	No
BloombergGPT [10]	— (LLM-only)	No	Yes	No	N/A	No
SecurePay (ours)	XGBoost on SageMaker	Yes (Rekognition)	Yes (Bedrock)	Yes	Yes	Yes (4 layers)

Table II. Capability comparison between SecurePay and representative prior systems.

SecurePay is the only system in the comparison that integrates cloud-hosted gradient boosting, biometric verification, and an LLM reasoning layer behind a unified credential-isolated serverless facade, with rule-based fallbacks on every external dependency. Prior academic work typically optimizes a single component (classifier accuracy, or an LLM benchmark) in isolation, whereas SecurePay targets the end-to-end operational design of an AI-augmented payment application.

VI. EXPERIMENTAL EVALUATION

A. Fraud Detection Performance

The XGBoost fraud classifier was evaluated on a held-out test set of 350 samples (200 normal, 150 fraudulent). Table III presents the classification metrics for both the SageMaker-hosted XGBoost model and the locally-served PyTorch neural network baseline.

Metric	XGBoost (SageMaker)	PyTorch NN (Local)
Accuracy	1.0000	1.0000
Precision	1.0000	1.0000
Recall	1.0000	1.0000
F1 Score	1.0000	1.0000
AUC-ROC	1.0000	1.0000

Table III. Fraud detection model performance on 350 synthetic test samples.

Both models achieve perfect classification on the synthetic test set. While this result reflects the separability of the synthetic data rather than real-world performance, it validates that the model training pipeline, feature engineering, and inference serving infrastructure function correctly end-to-end. The confusion matrix for the XGBoost model shows 150 true positives, 200 true negatives, and zero false positives or false negatives. Evaluation with anonymized production transaction data is planned as immediate future work.

B. Promotional Recommendation Performance

The promotional recommendation model was evaluated on 300 test samples (60 per category). Table IV shows per-class performance.

Category	Precision	Recall	F1	Support
Welcome	1.00	1.00	1.00	60
Loyalty	1.00	1.00	1.00	60
High Value	1.00	1.00	1.00	60
Seasonal	1.00	1.00	1.00	60
Cashback	1.00	1.00	1.00	60
Macro Avg	1.00	1.00	1.00	300

Table IV. Promotional recommendation per-class performance (300 test samples).

As with the fraud model, these scores confirm correct pipeline operation on well-separated synthetic categories. Real-world promotional targeting would require evaluation against actual user conversion data.

C. Latency Analysis

End-to-end latency was measured from the edge function invocation through AWS service calls and back. Table V reports the results across 20 invocations per service.

Service	Mean	p50	p95	Min	Max
SageMaker Fraud	1390.6	846.4	4815.5	139.2	5125.0
SageMaker Promo	1833.9	1641.0	3708.6	139.7	3785.3
Bedrock Nova Lite	1752.3	1701.2	2949.6	779.6	3123.2

Table V. End-to-end latency through edge functions (milliseconds, n=20).

The latency figures include network round-trips from the Supabase edge function runtime to AWS endpoints, SigV4 signature computation, and the actual inference time. The high variance in SageMaker latency (min 139 ms, max 5125 ms) is attributable to cold starts on the ml.t2.medium instances. Bedrock latency is more consistent, reflecting the managed nature of the foundation model serving infrastructure. For comparison, local PyTorch inference on the same fraud detection task completes in 0.05 ms (median), roughly four orders of magnitude faster than the cloud-hosted pipeline. However, local inference requires bundling the model with the application and does not benefit from centralized model updates or the operational advantages of managed infrastructure.

D. Cost Analysis

Table VI presents the per-inference and monthly cost estimates for each AWS service under an assumed workload of 10,000 transactions per month.

Service	Per-Inference (USD)	Monthly (USD)	Notes
SageMaker	$0.000018	$93.60	2× ml.t2.medium @ $0.065/hr
Bedrock	$0.000036	$0.07	~200 input + ~100 output tokens
Rekognition	$0.001000	$0.50	500 face logins/month
SES	$0.000100	$0.50	5,000 emails/month
S3	—	$0.0025	Face images + artifacts
Total	—	$94.67	10,000 transactions/month

Table VI. AWS service cost breakdown (monthly estimate for 10,000 transactions).

SageMaker endpoint hosting dominates the cost at 98.9% of the total, since ml.t2.medium instances run continuously regardless of traffic volume. For workloads with predictable low-traffic periods, SageMaker Serverless Inference could reduce costs substantially by scaling to zero during idle periods, at the expense of cold-start latency.

VII. IMPLEMENTATION

The SecurePay mobile client is implemented in Flutter with a feature-based clean architecture and Riverpod state management. This section illustrates the user-facing realisation of each AI capability described in Section III. Figures 2–7 show the principal screens captured from the running Android build. Each screen calls the corresponding Supabase Edge Function described earlier, with client-side caching applied to Bedrock responses to control cost.

A. Home Dashboard and AI Weekly Summary

The home screen is the primary entry point and surfaces the wallet balance, recent activity, and AI-generated insights at a glance, as shown in Fig. 2.

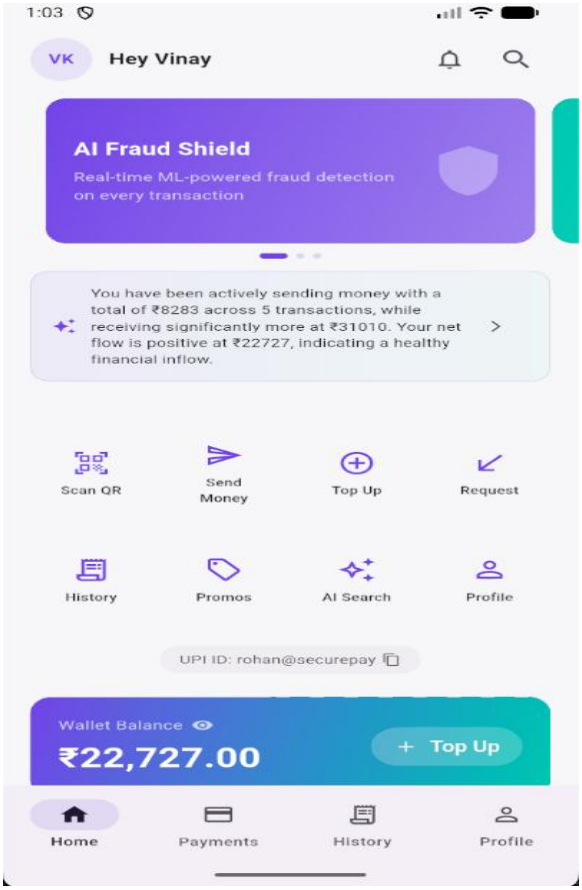

Fig. 2. Home dashboard. The GPay-style layout shows the wallet balance, promo carousel ("AI Fraud Shield"), the Bedrock-generated AI weekly summary strip, quick actions (Scan QR, Send Money, Top Up, Request, History, Promos, AI Search, Profile), and persistent bottom navigation. The weekly summary is cached for six hours to control Bedrock cost.

B. Multi-Layered Payment Flow with Face Verification

Payments flowing through SageMaker's review zone (score 50–75) trigger a blocking face verification step that gates the swipe-to-pay control, as illustrated in Fig. 3.

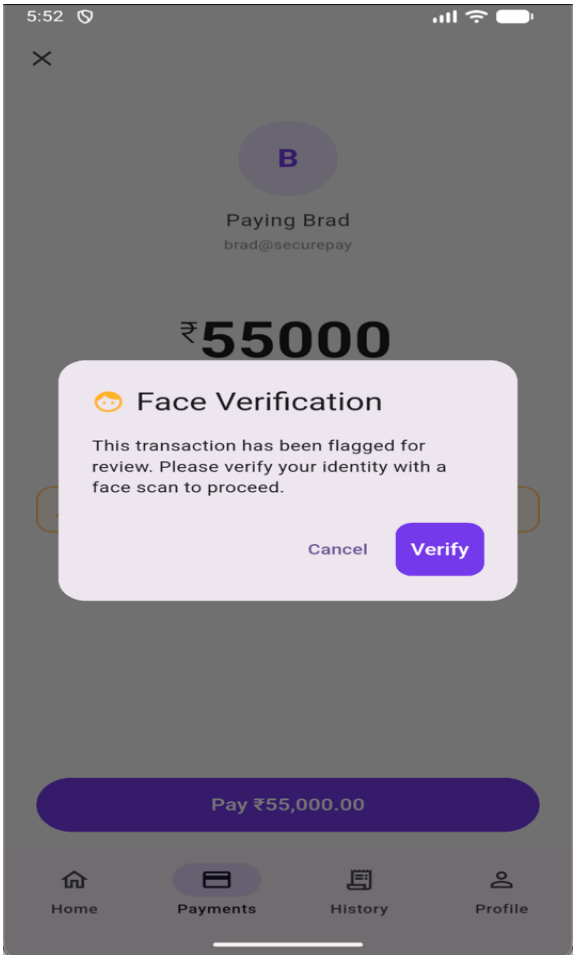

Fig. 3. Payment flow in the review zone (SageMaker fraud score between 50 and 75). The user sees a risk advisory with the score and contributing factors, followed by a Rekognition-based face verification step before the swipe-to-pay confirmation is enabled. This illustrates the third layer of the multi-layered fraud pipeline.

C. Natural Language Transaction Search

The Smart Search feature lets users query their transaction history in natural language and returns both the filtered results and a Bedrock-generated summary, as shown in Fig. 4.

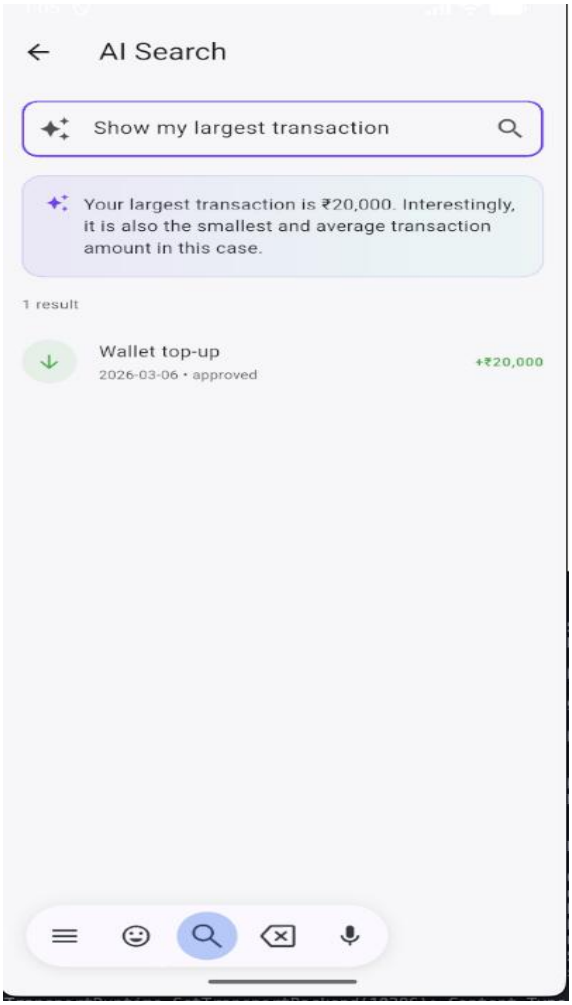

Fig. 4. Smart Search screen. The user enters a natural language query (e.g., "how much did I spend on food this month?"); Amazon Nova Lite translates it into a structured Postgres filter, the edge function executes it, and Nova Lite produces a one-paragraph summary over the retrieved rows. Suggested queries are shown as chips for first-time users.

D. AI Spending Insights

The Insights screen combines chart-based visualizations with a Bedrock-generated narrative to help users understand their spending patterns, as shown in Fig. 5.

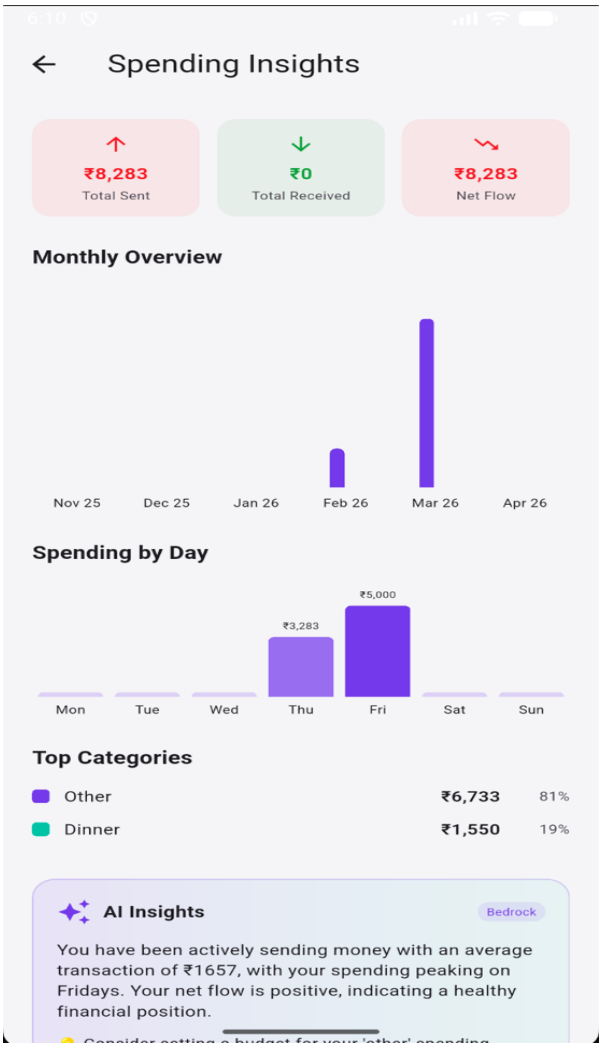

Fig. 5. Insights screen. Monthly spending and receiving are visualised as grouped bars, complemented by a day-of-week distribution and a category breakdown derived from Bedrock categorization. A "Bedrock" badge marks AI-generated content, including the personalized summary and three actionable financial tips. Responses are cached to amortize LLM cost.

VIII. DISCUSSION

A. Multi-Layered Security Design

The fraud detection pipeline operates across four layers, each serving a distinct purpose. The client-side pre-screening provides immediate feedback and reduces server load. The SageMaker XGBoost model performs the primary risk assessment. For transactions scoring in the ambiguous "review zone" (scores between 50 and 75 out of 100), the system triggers

biometric face verification through Rekognition, requiring the user to confirm their identity before the transaction proceeds. Finally, Bedrock generates natural language explanations of fraud decisions, making the system's reasoning transparent to users. This layered approach addresses a practical concern in payment security: binary approve/block decisions frustrate users when legitimate transactions are incorrectly flagged. The review zone with biometric verification provides a middle ground.

B. Graceful Degradation

Each AI service integration includes a fallback mechanism. The fraud-check and recommend-promo edge functions implement rule-based engines that activate when SageMaker endpoints return errors or time out. The Bedrock-powered features (categorization, insights, search) fall back to simpler heuristic responses. This design keeps core payment functionality availablable even during AWS service disruptions.

C. Limitations

There are clear limitations to this work. First, the evaluation uses synthetic data, which does not capture the full complexity of real-world fraud patterns; the perfect classification scores reflect the separability of synthetic distributions rather than expected production performance. Second, the face authentication component is constrained by mobile camera quality and environmental conditions, and the 90% similarity threshold was chosen to balance security against usability. Third, LLM-based features (categorization, insights, search) are subject to the inherent variability of generative models; while low-temperature settings and structured output parsing mitigate this, occasional malformed responses require reliable fallback handling. Fourth, the current SageMaker deployment uses dedicated instances, which is cost-inefficient for low-traffic applications; migration to serverless inference would better align costs with actual usage patterns.

D. Practical Considerations

Implementing AWS Signature Version 4 authentication manually in Deno required separating canonical paths from fetch paths to avoid double URI-encoding by the runtime. The choice of Amazon Nova Lite was driven by regional availability through the APAC cross-region inference profile in ap-south-1.

IX. CONCLUSION

This paper presented SecurePay, a digital payment system that integrates multiple AWS AI/ML services into a multi-layered security and intelligence framework. The architecture shows that cloud-hosted machine learning, biometric authentication, and large language model capabilities can be combined within a mobile payment application while maintaining credential isolation, graceful degradation, and reasonable operational costs. Both models performed well on our test data, but evaluation on real transactions is required to establish production-relevant performance. The end-to-end latency analysis reveals the cost of routing inference through serverless edge functions, with median fraud scoring latency of 846 ms, which is acceptable for transaction approval workflows but highlights the trade-off against sub-millisecond local inference.

Future work includes: (1) evaluation with anonymized real transaction data to establish production-relevant performance baselines; (2) migration to SageMaker Serverless Inference to reduce hosting costs for variable workloads; and (3) exploration of on-device ML models using TensorFlow Lite to provide offline fraud scoring capability while maintaining the cloud-hosted model as the primary scoring mechanism.

REFERENCES

[1] Reserve Bank of India, "Payment System Indicators," RBI Bulletin, 2024.
[2] Indian Cyber Crime Coordination Centre, "Annual Report on Cyber Financial Fraud," Ministry of Home Affairs, Government of India, 2023.
[3] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," J. Netw. Comput. Appl., vol. 68, pp. 90–113, 2016.
[4] A. Dal Pozzolo, O. Caelen, Y. A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," Expert Syst. Appl., vol. 41, no. 10, pp. 4915–4928, 2014.

[5] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," Decis. Support Syst., vol. 50, no. 3, pp. 602–613, 2011.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD, 2016, pp. 785–794.

[7] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," Inf. Sci., vol. 479, pp. 448–455, 2019.

[8] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling, "Deep learning detecting fraud in credit card transactions," in Proc. IEEE SIEDS, 2018, pp. 129–134.

[9] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, present, and future of face recognition: A review," Electronics, vol. 9, no. 8, p. 1188, 2020.

[10] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "BloombergGPT: A large language model for finance," arXiv:2303.17564, 2023.

[11] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu, "Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond," ACM Trans. Knowl. Discov. Data, vol. 18, no. 6, pp. 1–32, 2024.

[12] V. Ishakian, V. Muthusamy, and A. Slominski, "Serving deep learning models in a serverless platform," in Proc. IEEE IC2E, 2018, pp. 257–262.

[13] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in Proc. 28th EMSS, 2016, pp. 249–255.

[14] X. Liu, J. Zhang, Y. Liu, L. Wu, and Z. Yu, "Pick and choose: A GNN-based imbalanced learning approach for fraud detection," in Proc. Web Conf. (WWW), 2021, pp. 3168–3177.

[15] S. Rao, R. Mitra, A. Bhatia, S. Nagar, and S. Yadav, "xFraud: Explainable fraud transaction detection," Proc. VLDB Endow., vol. 15, no. 3, pp. 427–436, 2021.

[16] C. Wang, Y. Wang, Z. Liu, H. Zhang, and X. He, "Transformer-based fraud detection model for e-commerce transactions," IEEE Access, vol. 10, pp. 70133–70143, 2022.

[17] FIDO Alliance, "FIDO2: WebAuthn & CTAP — Specifications overview," Tech. Rep., 2022. [Online]. Available: https://fidoalliance.org/specifications/

[18] PCI Security Standards Council, "PCI DSS v4.0 Requirements and Testing Procedures," Tech. Rep., 2022.

[19] National Payments Corporation of India, "UPI Product Statistics," NPCI, 2024. [Online]. Available: https://www.npci.org.in/statistics

[20] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become," Commun. ACM, vol. 64, no. 5, pp. 76–84, 2021.