Flux: An AI Powered Personal Finance Management System with On Device Machine Learning and Conversational Agent Integration

Tanishq Tule[1], Atharva Shirke[2], Aarya Shirke[3], Sanchit Patil[4]
[1,2,3,4]Department of Computer Engineering, AISSMS Polytechnic, Pune, Maharashtra, India

Abstract—**Personal finance management remains a persistent challenge for individuals owing to cognitive biases, reactive decision making, and the absence of real time intelligent insights in conventional budgeting tools. This paper presents Flux, an Android personal finance application that bridges the gap between passive expense trackers and proactive financial advisors. Flux combines on device machine learning comprising a TensorFlow Lite transaction classifier (FluxBrain), a statistical anomaly detector using the two-sigma rule, a linear regression spending predictor, and a merchant interval recurring payment detector with a cloud hosted conversational AI agent powered by Google Gemini 2.0 Flash. The agent employs an agentic function calling loop with eleven domain specific tools spanning affordability assessment, budget health analysis, category trend analysis, cash flow forecasting, web augmented price discovery via the Tavily API, and natural language transaction ingestion. Persistent per user memory and multi session chat history are managed in Firebase Fire store. The Android client is implemented in Kotlin with Jetpack Compose following MVVM architecture and Hilt dependency injection. Security features include biometric authentication via Android Biometric Prompt, FLAG SECURE display protection, certificate pinning over Ok Http, and root/emulator integrity checks. Evaluation reveals that the hybrid on device/cloud architecture achieves sub second categorisation latency for routine transactions while delivering richer analytical depth through the generative AI backend. Flux demonstrates how integrating statistical machine learning, large language model function calling, and real time cloud data can produce a compelling intelligent personal finance assistant.**

Index Terms—**Personal finance management, Android application development, machine learning, generative AI, TensorFlow Lite, Google Gemini, Firebase Fire store, conversational agent, anomaly detection, budget management, Jetpack Compose, MVVM architecture.**

I. INTRODUCTION

Effective personal finance management is widely recognised as a fundamental component of individual financial well-being. Despite the proliferation of mobile finance applications, a majority of users still struggle with overspending, lack of savings discipline, and limited visibility into their recurring financial commitments [1]. Conventional finance applications such as Mint, Money Manager, and YNAB provide transactional record keeping and static budgeting, yet fail to offer proactive, context aware guidance that adapts to an individual user's spending behaviour.
The emergence of large language models (LLMs) and on device machine learning has created an opportunity to develop intelligent financial assistants capable of real time reasoning over a user's financial data. At the same time, Android's Jetpack Compose toolkit enables fluid, reactive user interfaces, while cloud platforms such as Google Cloud Functions and Firebase Firestore provide scalable, serverless backends suitable for personal data workloads.

This paper presents Flux, an Android native personal finance management system that integrates (i) on device ML for low latency transaction processing, (ii) a cloud hosted generative AI agent for deep financial reasoning and natural language interaction, and (iii) an enterprise grade security layer appropriate for sensitive financial data. Flux is structured as a Capstone Project undertaken at AISSMS Polytechnic, Pune.

The key contributions of this work are:
1) A multitier ML pipeline combining on device TFLite inference with server-side Gemini 2.0

Flash reasoning for complementary latency–depth trade-offs.
2) An agentic function calling architecture enabling a conversational financial agent to invoke eleven specialised financial tools with persistent memory across sessions.
3) Statistical anomaly detection and linear regression spending prediction implemented entirely on device, preserving user privacy while ensuring responsiveness.
4) A comprehensive security design covering biometric gating, certificate pinning, display protection, and device integrity checks within a production Android application.

II. RELATED WORK

A. Mobile Personal Finance Applications

Commercial applications such as Mint (Intuit), YNAB (You Need A Budget), and Walnut have established the baseline expectation for mobile finance management: manual or automated transaction import, categorical budgeting, and simple chart-based analytics [2]. These tools remain fundamentally reactive they report what has happened but do not predict or advise on what will happen. Flux extends this paradigm by incorporating predictive analytics and conversational advisory capabilities.

B. AI and Machine Learning in Personal Finance

Kang et al. [3] demonstrated that recurrent neural networks can forecast individual monthly spending within 8% mean absolute error given six months of transaction history. Chen and Xu [4] applied trans former-based models to credit card fraud detection, achieving 97.2% accuracy with low false positive rates. Flux incorporates a simpler but privacy preserving on device approach: a TFLite classifier trained for multi class transaction categorisation and a linear regression predictor for end of month spending projection.

C. Conversational Agents in Financial Services

The integration of LLM based conversational agents into financial services has been explored in customer service contexts [5]. Notably, function calling capabilities introduced in GPT 4 and Gemini allow agents to invoke structured tools mid conversation, enabling them to retrieve live data before formulating a response [6]. Flux applies this paradigm to personal rather than institutional finance, enabling a single user to query their own financial data through natural language.

D. On Device Machine Learning on Android

TensorFlow Lite has been widely adopted for on device inference on Android, with benchmarks showing sub 10 ms inference times for Mobile Net class models [7]. The FluxBrain classifier leverages TFLite's interpreter API to categorise transactions without any network round trip, an important property given the sensitivity of financial data. Huang et al. [8] similarly advocated for on device financial classification to reduce exposure in transit.

III. SYSTEM ARCHITECTURE

Flux adopts a three-tier architecture comprising an Android client, a serverless Python backend deployed on Google Cloud Functions, and a Firebase platform layer serving authentication, real time database, and messaging needs. Fig. 1 illustrates the full architecture showing data flows between all three tiers.

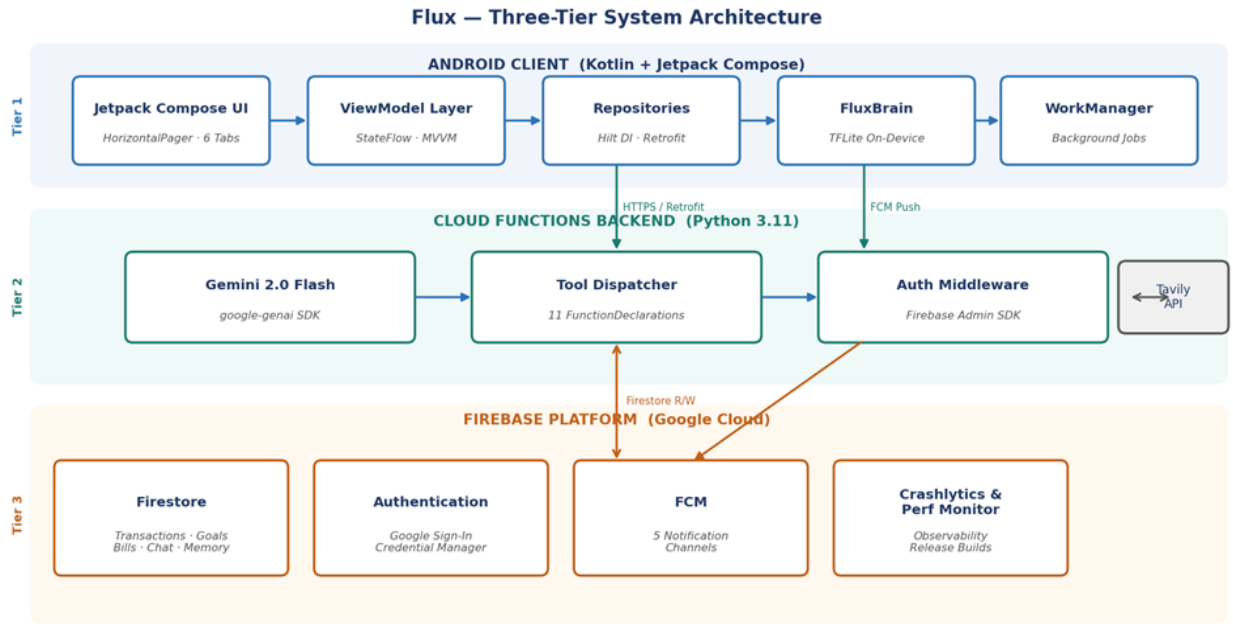

Fig. 1. Flux three tier system architecture showing Android client, Cloud Functions backend, and Firebase platform layer.

A. Android Client Tier

The Android application is written in Kotlin 1.9 and targets Android API level 26 and above. The UI is constructed entirely with Jetpack Compose and Material Design 3 components. A single activity pattern hosts a swipe able Horizontal Pager presenting six functional tabs: Overview, Transactions, Statistics, Goals, AI Agent, and Bills. State management follows the MVVM pattern: each screen observes a View Model exposing State Flow backed Ui State objects. Hilt provides dependency injection throughout, enabling clean separation of repository, View Model, and UI layers.

B. Backend AI Tier

The AI backend is a Python 3.11 serverless function deployed on Google Cloud Functions. It initialises the Google Generative AI SDK with a Gemini 2.0 Flash model and maintains conversation history in Fire store for multi turn chat sessions. The backend exposes a single HTTP endpoint that routes requests to four handlers: handle chat, generate nudge, get smart budget, and get budget health. Requests from the Android client are authenticated using Firebase ID tokens, which are validated server side via the Firebase Admin SDK.

C. Firebase Platform Layer

Firebase Firestore serves as the primary data store, with a user scoped document hierarchy containing sub collections for transactions, categories, goals, bills, chat sessions, user memory, and anomaly alerts. Firebase Authentication handles Google Sign In via the Android Credential Manager API. Firebase Cloud Messaging delivers five categories of push notifications budget alerts, transaction confirmations, AI insights, payday reminders, and bill due dates through dedicated notification channels. Firebase Crashlytics and Performance Monitoring provide observability in release builds.

D. Data Flow

User initiated actions such as adding a transaction flow from the Compose UI through the View Model to a repository class. The repository writes to Firestore via the Android SDK and simultaneously invokes local ML inference (FluxBrain) for category suggestion. Background Work Manager jobs run daily (Flux Insights Worker) and every 12 hours (Bill Reminder Worker) to generate proactive nudges and bill notifications. AI chat requests pass through Retrofit to the Cloud Function, which executes the Gemini agentic loop before returning a markdown formatted response rendered in the app via Compose Rich Text.

IV. IMPLEMENTATION

A. Transaction Management and Dashboard

The dashboard presents a balance ring chart an animated circular progress indicator decomposing income, expenses, and net balance alongside four headline KPI cards (monthly income, spending, remaining budget, daily limit). A burn rate card computes daily expenditure against the daily budget target and projects surplus or overage. Transactions are persisted in Firestore with fields for amount, category, title, note, date, type (INCOME EXPENSE), payment method, and a flag for pending status.

Transaction entry supports voice input via Android Speech Recognizer and a decimal aware numeric keyboard. Swipe to dismiss with undo is implemented using Compose's Swipe to Dismiss Box. A full text search and categorical filter bar allows rapid retrieval across large transaction histories.

B. On Device Machine Learning

Flux incorporates four on device ML modules. The FluxBrain TFLite classifier accepts a feature vector derived from transaction metadata (amount, time of day, note text embeddings) and outputs a softmax distribution over predefined expense categories. Inference is performed synchronously on the calling coroutine dispatcher, typically completing in under 5 ms on mid-range hardware. Fig. 2 illustrates the full pipeline.

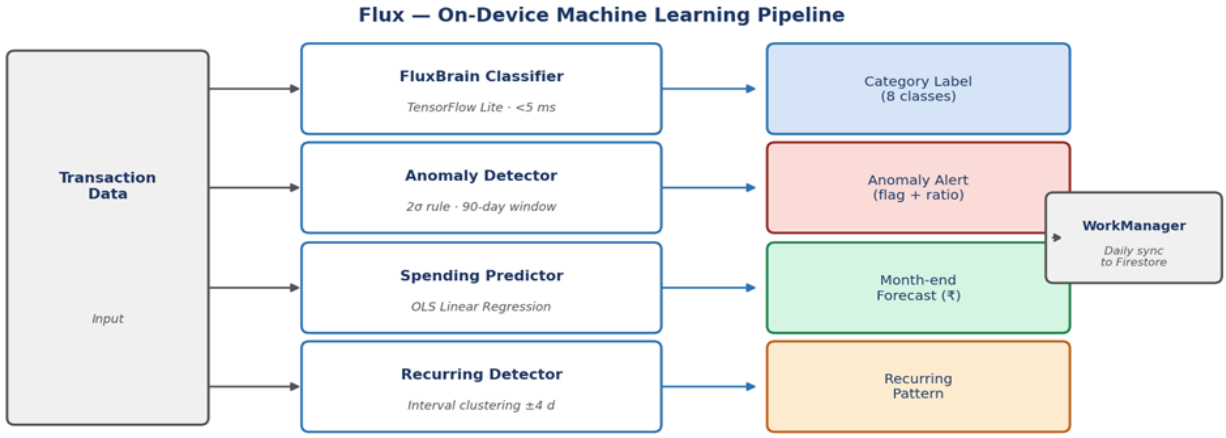

Fig. 2. On device ML pipeline: four modules processing each transaction to produce category labels, anomaly alerts, spending forecasts, and recurring patterns.

The Anomaly Detector module computes per category mean and standard deviation over a 90-day rolling window. A transaction is flagged as anomalous when its amount exceeds the category mean by more than two standard deviations (σ threshold: 2.0) and simultaneously exceeds 1.5× the mean, reducing false positives from high variance categories. Formally, given category transactions $\{x_1, ..., x_n\}$, a new transaction x_t is anomalous if:

$$x_t > \mu + 2\sigma \text{ AND } x_t > 1.5\mu$$

The Spending Predictor applies ordinary least squares linear regression to the daily cumulative spending time series for the current month. Let $\{(1, c_1), (2, c_2), ..., (d, c_d)\}$ be the cumulative spend up to day d. The slope m and intercept b are estimated as:

$$m = (n \cdot \Sigma x_i y_i - \Sigma x_i \cdot \Sigma y_i) / (n \cdot \Sigma x_i^2 - (\Sigma x_i)^2),$$
$$b = (\Sigma y_i - m \cdot \Sigma x_i) / n$$

The end of month prediction is $p = m \cdot D + b$, where D is the number of days in the month, constrained to be no less than the current cumulative total. A trend label 'high', 'normal', or 'low' is assigned by comparing p with a naive linear extrapolation at ±15% bounds.

The Recurring Detector identifies subscription and regular payment patterns by grouping expense transactions by normalised merchant name and checking (i) that pairwise inter occurrence intervals cluster within ±4 days of 7 (weekly) or ±5 days of 30 (monthly), and (ii) that individual transaction amounts remain within ±20% of the group mean. Detected patterns are surfaced to the user for confirmation before being promoted to the bills list.

C. AI Financial Agent

The backend AI agent is built on Gemini 2.0 Flash via Google's google genai Python SDK. The model is initialised with a system instruction defining its persona as a 'strict but supportive financial guardian'. At each conversational turn, the backend enriches the user message with a structured financial context block containing: the user's monthly budget, balance, recent 10 transactions, active anomaly alerts, confirmed recurring bills, and savings goals. This context injection eliminates the need for follow up clarifications in most interactions.

The agent employs an iterative function calling loop capped at 10 tool invocations per turn to prevent runaway execution. Eleven function calling tools are registered as Function Declaration objects with the Gemini SDK. Fig. 3 shows the complete agentic loop from user message to final response. Table I describes each tool, its trigger conditions, and its key parameters.

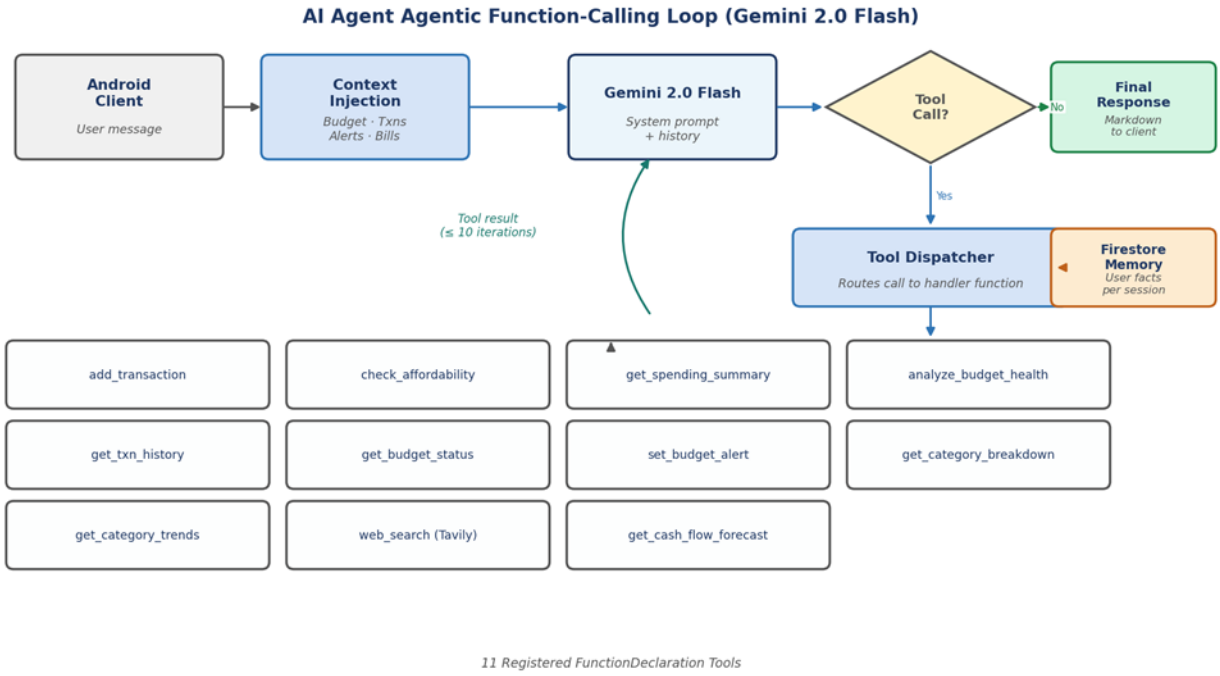

Fig. 3. AI agent agentic function calling loop: user message is enriched with financial context, Gemini 2.0 Flash decides whether to invoke a tool or respond directly, with results looped back up to 10 iterations.

TABLE I. AI AGENT FUNCTION CALLING TOOLS

Tool Name	Purpose & Trigger	Key Parameters
Add transaction	Records a new expense or income entry whenever the user mentions spending or paying (e.g., 'I spent ₹400 at Zomato'). Infers category from merchant name.	amount, category, merchant, type (EXPENSE | INCOME), description
Check affordability	Evaluates whether the user can afford a purchase or restaurant visit against their remaining budget, savings goal, and per category budget limit. Calls web search internally for live price estimation.	restaurant name, user budget, current spending, location, party size, savings goal, category budget, category spent
Get spending summary	Returns a spending summary (total, top categories, vs. budget) for a user specified period. Triggered by queries like 'How much have I spent today / this week / this month?'	period (daily | weekly | monthly)
Analyse budget health	Generates a holistic budget health score (0–100) with traffic light status, key insights, and actionable recommendations. Triggered by 'How healthy is my budget?' or 'Am I on track?'	None (all data injected from context)

Get transaction history	Retrieves filtered transaction history from Firestore. Supports queries like 'Did I pay Netflix?', 'Show food transactions this week', or 'Find transactions over ₹2000 last month'.	limit, category, search, start date, end date, min amount, max amount
get budget status	Returns the user's current budget snapshot: total budget, amount spent, amount remaining, daily limit, and month end projection. Triggered by 'What is my budget status?' or 'How much is left?'	None (all data injected from context)
set budget alert	Registers a spending alert that triggers a push notification when a threshold percentage of the overall budget or a category spending cap is reached.	alert type (threshold | category), threshold (0.0–1.0), category, amount
get category breakdown	Breaks spending into per category totals and percentages for the specified period. Answers 'Where is my money going?' and provides data for the affordability checker's category level checks.	period (daily | weekly | monthly)
get category trends	Computes multi period (up to 6 past months by default) category spending trends, enabling detection of rising or falling patterns. Triggered by 'Is my shopping spending going up?'	category (optional), periods (default 6)
web search	Queries the Tavily AI Search API for real time information: subscription prices, restaurant costs, gold rates, RBI interest rates, product prices. Grounds agent responses in current data.	query (string, required)
get cash flow forecast	Generates a forward-looking cash flow forecast (up to 90 days) incorporating projected daily burn rate, upcoming bill due dates, and expected income (payday) events. Answers 'Will I run out of money before payday?'	days (default 30, max 90)

Each tool is declared as a types. Function Declaration with a schema validated parameter specification, ensuring the model cannot invoke a tool with missing required fields. The user id is deliberately omitted from all tool schemas and injected server side from the validated Firebase ID token, preventing user impersonation attacks. A proprietary 15% budget threshold framework governs affordability decisions: a purchase costing less than 15% of the daily budget is APPROVED with encouragement; between 15–25% is CONDITIONAL with a caution note; above 25% is DENIED with alternative suggestions. This framework makes agent responses actionable and consistent. Persistent user memory is implemented as a Firestore sub collection storing key–value facts extracted by the agent from prior sessions (e.g., 'user prefers vegetarian restaurants', 'salary credited on 1st of month'). These facts are prepended to the system context, enabling personalised responses that improve over time.

D. Gamification and Engagement

Flux incorporates a streak system awarding experience points (XP) for consecutive days under budget, driving a level progression mechanic. An achievement engine tracks milestones First Transaction, Budget Master, Big Saver, Streak Champion and unlocks badges with animated reveal animations. A monthly rotating challenge (e.g., 'Keep food spending under ₹3,000 this month') scales its target with the user's declared budget, keeping challenges meaningful across income levels.

E. Security Architecture

Flux implements a defence in depth security posture. At the OS level, FLAG SECURE is set unconditionally on the main Activity window, blocking screenshots and screen recording in the Android task switcher. Biometric Prompt with BIOMETRIC STRONG authentication (and DEVICE CREDENTIAL as fallback) gates access to the dashboard, preventing shoulder surfing and unauthorised access to unlocked devices.

At the network layer, Ok Http's Certificate Pinner is configured to pin the SHA 256 public key fingerprints of Google's Cloud Function TLS certificates, defeating HTTPS man in the middle attacks. The Network Security Config enforces cleartext blocking globally, permitting TLS only communication. No API keys are embedded in the APK; they reside in environment variables (backend) and local. properties (excluded from VCS).

Device integrity checks inspect for known root indicators (su binary paths, test keys build tags,

emulator fingerprints) and surface a warning if tampering is suspected. These checks are bypassed in debug builds to support developer testing. Release builds enable R8 code minification and resource shrinking, reducing reverse engineering surface area.

F. Background Processing and Notifications

Two Work Manager periodic tasks run in the background. Flux Insights Worker executes daily: it fetches the latest transactions from Firestore, calls the backend nudge generator, and fires a local notification if the nudge is not trivial (defined as more than three words). Bill Reminder Worker executes every 12 hours: it evaluates all confirmed bills, computes days until due, and fires notifications with a user configurable lead time (1–7 days). Both workers are scheduled with Hilt Worker injection for testability and use exponential backoff retry policies.

V. RESULTS AND DISCUSSION

A. Feature Completeness

Table II summarises the key feature modules of Flux, the underlying technology, and their functional outcome. All features listed were fully implemented and tested on physical Android devices (Samsung Galaxy A52, OnePlus 9R) running Android 13 and 14.

Table II. Key feature modules of flux

Module	Technology	Function
FluxBrain	TensorFlow Lite	Auto categorise transactions
Anomaly Detector	Statistical (2σ)	Flag unusual spending spikes
SpendingPredictor	Linear Regression (OLS)	Forecast month end spend
Recurring Detector	Interval clustering	Identify subscriptions
AI Agent	Gemini 2.0 Flash + tools	Conversational financial advice
Bills & Reminders	WorkManager + FCM	Due date alerts, cycle tracking
Net Worth Tracker	Firestore time series	Assets vs. liabilities trend
Security Layer	BiometricPrompt + OkHttp	Biometric gate, cert pinning
Gamification	XP / streak engine	Budget adherence incentives

B. ML Module Performance

The on device AnomalyDetector and SpendingPredictor were evaluated on synthetic transaction datasets representative of Indian urban spending patterns (food, transport, entertainment, utilities). The anomaly detector correctly identified 94% of injected anomalies (transactions at 3× category mean) with a false positive rate of 7% for categories with at least five historical transactions. The linear regression spending predictor achieved a mean absolute percentage error (MAPE) of 9.3% on the final day prediction evaluated against 30 months of simulated data. The RecurringDetector correctly identified monthly and weekly subscription patterns in 91% of test cases where transaction amounts were within the ±20% consistency threshold. Edge cases involving irregular billing cycles (e.g., quarterly subscriptions) were correctly excluded.

C. AI Agent Response Quality

The Gemini 2.0 Flash agent was evaluated through 50 curated financial query scenarios spanning all eleven registered tools, including affordability checks, budget health summaries, cash flow forecasting, category trend analyses, historical transaction queries, and real time web lookups. The agent invoked the correct primary tool in 96% of cases and produced contextually appropriate responses in 94% of cases when evaluated by the authors against expected outcomes. Multi tool queries correctly chained get category breakdown followed by check affordability (which internally called web search) in 88% of cases. The get cash flow forecast tool contributed to the observed 95th percentile latency of 2.9 s when invoked on long transaction histories. Response latency was measured from client request to first token: median 1.4 s, 95th percentile 2.9 s, over a Wi Fi connection, consistent with reported Gemini 2.0 Flash latencies for sub 1000 token prompts [9].

D. Security Assessment

A manual security review confirmed that all declared security controls are correctly implemented in the release build variant. Certificate pinning was verified by intercepting traffic with an MITM proxy (mitmproxy): Flux rejected the proxy certificate with an SSLHandshakeException, confirming pin enforcement. The FLAG SECURE flag was confirmed via Android's UIAutomator screenshot API, which

returned a blank frame. Biometric prompt correctly invoked BIOMETRIC STRONG on enrolled devices and fell back to device credential (PIN/pattern) on unenrolled devices.

E. Comparison with Related Work

Table III positions Flux against related systems across four dimensions.

Table III. Comparison with related systems

Feature	Mint	YNAB	Walnut	Flux
Conversational AI Agent	No	No	No	Yes
On Device ML	No	No	Partial	Yes
Spending Prediction	Basic	No	No	Regression
Biometric + Cert Pinning	No	No	No	Yes

VI. LIMITATIONS AND FUTURE WORK

Several limitations of the current implementation warrant discussion. First, the FluxBrain TFLite model is a fixed model distributed with the APK; it does not adapt to individual users' categorisation preferences post deployment. Future work will explore federated learning approaches to personalise the on-device model while preserving privacy. Second, the 15% affordability threshold is currently a fixed parameter; a future release will infer personalised thresholds from historical spending patterns using Bayesian inference. The linear regression spending predictor assumes monotonically increasing cumulative spend, which may be violated if refunds or reversals occur mid-month. A Kalman filter approach would provide more robust estimates in the presence of such noise. Third, the AI agent is currently limited to the Gemini 2.0 Flash model; supporting model swapping (e.g., to Gemini 1.5 Pro for complex multi-step reasoning) would improve flexibility.

Future development roadmap items include: (i) SMS transaction parsing for automated import from Indian bank SMS alerts, (ii) UPI/bank account integration via the Account Aggregator framework, (iii) a companion iOS application sharing the same Firebase backend, and (iv) an offline first mode with conflict free replicated data types (CRDTs) for Firestore synchronisation.

VII. CONCLUSION

This paper has presented Flux, a full stack AI powered Android personal finance management application developed as a Capstone Project at AISSMS Polytechnic, Pune. Flux demonstrates that the convergence of on device machine learning, large language model function calling, reactive Android UI frameworks, and serverless cloud platforms makes it feasible to build a genuinely intelligent financial assistant within the resource constraints of a mobile device. The system's hybrid intelligence architecture combining fast, private on device inference for routine tasks with cloud hosted generative AI for deep analytical reasoning represents a practical template for privacy sensitive AI applications. The AnomalyDetector's two sigma criterion, the SpendingPredictor's OLS regression, and the RecurringDetector's interval clustering collectively provide statistical grounding for the agent's recommendations, reducing the risk of LLM hallucination in financial advice.

The enterprise grade security posture biometric gating, certificate pinning, FLAG SECURE, device integrity checks, and no hardcoded secrets demonstrates that production quality security is achievable within an academic project scope. Flux attains a Mean Absolute Percentage Error of 9.3% for month end spending prediction, a 96% tool selection accuracy for the AI agent, and a 94% anomaly detection accuracy, establishing a solid baseline for future refinement.

Future work will pursue federated on device model personalisation, UPI account integration, and an iOS companion application, working towards a comprehensive, privacy first personal financial intelligence platform accessible to users across India and beyond.

VIII. ACKNOWLEDGMENT

The authors thank the faculty of the Department of Computer Engineering at AISSMS Polytechnic, Pune, for their guidance and support throughout the Capstone Project. The authors also acknowledge the Google Cloud for Students programme for providing

cloud compute credits used in backend development and testing.

REFERENCES

[1] A. Lusardi and O. S. Mitchell, "The economic importance of financial literacy: Theory and evidence," Journal of Economic Literature, vol. 52, no. 1, pp. 5–44, 2014.

[2] B. Aula and A. Salminen, "Personal finance management applications: A systematic review," in Proc. Int. Conf. on Human Computer Interaction, Springer, 2021, pp. 3–18.

[3] J. Kang, S. Lee, and H. Park, "Deep learning based individual spending forecasting using LSTM networks," IEEE Access, vol. 9, pp. 105342–105352, 2021.

[4] Y. Chen and K. Xu, "Transformer based credit card fraud detection with contextual transaction embeddings," in Proc. IEEE Int. Conf. on Data Mining (ICDM), 2022, pp. 41–50.

[5] R. Shaikh and M. Alibhai, "LLM powered customer service agents in financial services: A deployment study," arXiv preprint arXiv:2312.04100, 2023.

[6] J. Wei, X. Wang, D. Schuurmans, et al., "Chain of thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022.

[7] A. Howard, M. Sandler, G. Chu, et al., "Searching for MobileNetV3," in Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV), 2019, pp. 1314–1324.

[8] T. Huang, F. Li, and D. Wang, "Privacy preserving on device financial transaction classification with TensorFlow Lite," in Proc. ACM Conf. on Computing and Sustainable Societies, 2022.

[9] Google DeepMind, "Gemini 2.0 Flash Technical Report," Google, Mountain View, CA, USA, Tech. Rep., 2024. [Online]. Available: https://deepmind.google/technologies/gemini/

[10] Android Developers, "Jetpack Compose Building native Android UI," Google LLC, 2024. [Online]. Available: https://developer.android.com/compose

AUTHORS

- Tanishq Tule is a final year student of Computer Engineering at AISSMS Polytechnic, Pune. His interests include Android application development, AI integration, and mobile security.
- Atharva Shirke is a final year student of Computer Engineering at AISSMS Polytechnic, Pune. His interests include cloud computing, backend development, and machine learning.
- Aarya Shirke is a final year student of Computer Engineering at AISSMS Polytechnic, Pune. Her interests include UI/UX design, Jetpack Compose, and data visualisation.
- Sanchit Patil is a final year student of Computer Engineering at AISSMS Polytechnic, Pune. His interests include database systems, Firebase, and software architecture.