

Prophet AI : A Distributed Financial Flight Simulator for Freelancers Using Stochastic Forecasting, Cryptographic Integrity and Generative AI Intelligence

Subrat Kumar Jena

Student, Dept. of CSE (AI)
GIFT Autonomous, Bhubaneswar

Gayatri Palai

Student, Dept. of CSE (AI)
GIFT Autonomous, Bhubaneswar

**Asst. Prof. Rumana Hasinullah
Shaikh**

Assistant Professor, Dept. of CSE
GIFT Autonomous, Bhubaneswar

Abstract-The rapid expansion of the global gig economy has fundamentally changed the structure of personal finance management. Unlike salaried professionals who operate within predictable monthly income cycles, freelancers and independent contractors face highly volatile cashflow patterns characterized by delayed client payments, irregular project pipelines, seasonal fluctuations, and unstable liquidity reserves. Traditional Personal Financial Management (PFM) systems primarily focus on historical transaction tracking and static budgeting, making them ineffective for proactive financial survival planning in modern freelance ecosystems. This project introduces Prophet AI v1.1, an AI-driven financial intelligence platform engineered specifically to simulate, forecast, and analyze unstable freelance cashflow environments using distributed cloud infrastructure, cryptographic verification, and real-time neural intelligence. The proposed system functions as a Financial Flight Simulator that allows freelancers to model financial risk before it becomes catastrophic in real life. The platform combines machine learning-based forecasting, stochastic risk simulation, cryptographic integrity validation, asynchronous AI orchestration, and multilingual neural voice synthesis within a single integrated ecosystem. The system architecture follows a distributed deployment model consisting of a Next.js 14 frontend hosted on Vercel, a FastAPI Intelligence Gateway hosted on Render, and a Supabase PostgreSQL secure transaction vault. This decoupled architecture ensures scalability, modularity, low frontend latency, and reliable handling of long-running AI inference tasks. The financial forecasting engine utilizes a hybrid intelligence pipeline combining statistical forecasting principles and ensemble-based analytical logic. The platform generates 30-day rolling liquidity forecasts, safe spending corridors, and stress-based runway simulations that help users evaluate financial survival scenarios under varying burn conditions. Unlike conventional financial dashboards, Prophet AI introduces dynamic What-If simulation controls, allowing users to manipulate variables such as liquidity lag, expense escalation, and delayed client payments in real time. To establish institutional-grade trust and forensic-grade auditability, the system implements an Integrity Shield powered by the SHA-256 cryptographic hashing algorithm. Every transaction entered into the system generates a unique digital fingerprint using transaction attributes including amount, date, category, and user identification. This verification mechanism ensures that tampered or manipulated financial records cannot enter the intelligence pipeline, thereby maintaining a Verified Ledger architecture. The project additionally documents real-world deployment challenges involving decimal precision mismatches between JavaScript and Python environments and explains the implementation of strategic normalization bypass mechanisms for stable production deployment. The intelligence layer of Prophet AI is powered using Llama 3.3-70B via Groq infrastructure, enabling high-speed financial reasoning and structured JSON-based strategy generation. The platform utilizes a carefully engineered Ruthless Financial Strategist system prompt designed to deliver direct, survival-oriented financial recommendations rather than emotionally comforting advice. This design philosophy reflects the real-world operational needs of freelancers who require accurate liquidity warnings and actionable strategic insights during financial instability. The generated intelligence is converted into multilingual audio briefings using the edge-tts neural voice synthesis engine, supporting both English and Hindi voice outputs. To avoid cloud timeout failures and synchronous processing bottlenecks, the platform implements an asynchronous polling architecture using UUID-based job orchestration. The frontend submits a /briefing request and continuously polls a /briefing-status/{job_id} endpoint until the AI-generated strategy and MP3 briefing become available. This architecture enables the system to safely execute computationally expensive large language model inference and neural voice generation workflows even on limited-resource cloud infrastructure. The completed system demonstrates the practical integration of distributed AI infrastructure, cryptographic verification, asynchronous backend engineering, financial forecasting, and multimodal intelligence synthesis within a real-world production environment. Prophet AI v1.1 represents a transition from passive financial recordkeeping to proactive survival-oriented financial intelligence. The project establishes a scalable blueprint for next-generation AI-powered fintech systems capable of delivering real-time strategic decision support for the rapidly growing global freelance economy.

Keywords-Freelance finance; cashflow forecasting; stochastic simulation; SHA-256; verified ledger; FastAPI; Groq Llama; neural voice briefing; Supabase; async polling

I. INTRODUCTION

The global shift toward freelance, remote and project-based employment has created a major change in the way individuals earn, spend and plan money. Unlike traditional employees, freelancers do not receive fixed monthly salaries. Their income often depends on client approvals, milestone completion, seasonal project demand, delayed invoices and inconsistent payment schedules. This uncertainty creates a financial environment where ordinary budgeting tools are not enough.

Most personal finance applications are designed as historical ledgers. They show what was spent, what was received and what category a transaction belongs to. Such systems are useful for record keeping, but they do not answer the most important survival questions faced by freelancers: how long current liquidity will last, whether an upcoming expense is safe, how much a delayed payment affects runway, and when the user may enter a dangerous cash position.

Prophet AI v1.1 addresses this gap by presenting a financial intelligence platform that behaves as a Financial Flight Simulator. Instead of only recording past transactions, the system simulates future cashflow conditions, generates 30-day rolling forecasts, evaluates burn-rate pressure and produces strategic financial recommendations. The platform is designed specifically for the realities of the gig economy where income volatility is the rule rather than the exception.

The project combines multiple technical domains: machine learning-based cashflow forecasting, stochastic financial simulation, cryptographic verification, cloud-native backend orchestration, generative AI reasoning and multilingual voice synthesis. The system is not limited to a static dashboard. It is a distributed intelligence workflow where the user interface, backend, database and AI inference engine operate as separate but connected components.

The key contribution of the work is the integration of predictive finance with verified data integrity. Financial AI is only useful when the data being analyzed is authentic. Therefore, Prophet AI uses SHA-256 based transaction hashing to detect tampering and protect the integrity of the ledger before intelligence generation takes place.

II. RELATED WORK AND BACKGROUND

Personal Financial Management systems have evolved from manual ledgers to mobile dashboards and automated expense trackers. However, many existing systems remain reactive because they primarily summarize historical transactions. They provide limited support for forward-looking liquidity simulation, especially for users whose income is irregular and project-dependent.

Research in financial forecasting has shown that machine learning and time-series models can capture patterns that traditional static budgeting methods miss. Algorithms such as Random Forest, XGBoost and Prophet

are widely used to handle non-linear behavior, seasonal effects and trend changes. In high-variance financial data, ensemble learning is particularly useful because different models can specialize in different aspects of the data. XGBoost can capture sharp changes, Random Forest provides stability and Prophet can model recurring temporal behavior.

Cryptographic integrity is another important area connected to financial systems. SHA-256 is a secure hash algorithm that converts an input string into a fixed-length digital fingerprint. Even a small change in the input produces a completely different hash. This property makes hashing useful for detecting unauthorized changes in financial records. In Prophet AI, this principle is applied to every transaction using amount, date, category and user identifier fields.

Recent advances in generative AI have also opened new possibilities for financial decision support. Large language models can summarize numerical information, generate scenario-based recommendations and communicate risk in natural language. However, direct browser-based AI calls create security and reliability problems. For this reason, Prophet AI routes AI inference through a FastAPI backend, protecting API keys and preventing frontend instability.

Cloud deployment is also central to the system design. The frontend is hosted on Vercel, the backend on Render and the database on Supabase. This separation follows modern distributed system principles and allows each component to scale or fail independently without collapsing the complete application.

III. PROPOSED SYSTEM

The proposed system is Prophet AI v1.1, a distributed AI-driven cashflow forecasting and risk simulation platform for freelancers. The system is designed around three major objectives: predicting financial risk, verifying transaction integrity and generating actionable financial intelligence.

The frontend is developed using Next.js 14 and provides an interactive financial dashboard. It displays transaction history, forecast outputs, risk zones, simulation controls, AI intelligence cards and voice briefing controls. The frontend remains lightweight and does not perform expensive AI inference directly.

The backend is implemented using FastAPI and operates as the Intelligence Gateway. It receives simulation data from the frontend, verifies transaction integrity, creates asynchronous jobs, calls the AI inference layer and coordinates voice generation. The backend is intentionally separated from the frontend so that sensitive API keys remain hidden and long-running AI operations do not break the user interface.

The database layer uses Supabase PostgreSQL as a secure transaction vault. It stores financial records, integrity metadata, forecast records and audit-related information. Row Level Security can be applied to ensure that user data remains isolated and protected.

The AI layer uses Groq-hosted Llama 3.3 to generate financial strategy output. The model is guided by a system prompt that behaves like a ruthless financial strategist. This design choice is intentional because freelancers under financial stress need direct operational advice rather than soft motivational responses. The model generates structured JSON output containing risk level and strategic actions.

The voice layer uses edge-tts to convert the AI strategy into an MP3 briefing. This enables multilingual financial intelligence delivery, including English and Hindi voice briefings. The MP3 file is stored in a static audio directory and accessed by the frontend after job completion.

IV. SYSTEM ARCHITECTURE

The system follows a distributed cloud architecture. The major components are the User Browser, Next.js dashboard, FastAPI Intelligence Gateway, SHA-256 Integrity Shield, Groq Llama 3.3 engine, edge-tts voice engine, Supabase vault and static audio storage.

The workflow begins when the user interacts with the financial dashboard and requests an AI briefing. The frontend sends a POST request to the backend endpoint /briefing. Instead of waiting for the entire AI process to complete synchronously, the backend immediately creates a unique job identifier and returns it to the frontend. This prevents request timeout and keeps the dashboard responsive.

After the job is created, the backend starts a background task. This task verifies transaction integrity, prepares the simulation context, calls the Llama model for strategy generation and then sends the generated text to edge-tts for voice synthesis. The MP3 file is written to a static directory and the job status is updated to complete.

The frontend polls the endpoint /briefing-status/{job_id} at regular intervals. When the backend marks the job complete, the frontend receives both the structured JSON strategy and the audio URL. This architecture solves a practical cloud deployment problem: long-running AI tasks cannot reliably be handled through a single blocking request, especially on limited-resource infrastructure.

The Integrity Shield is placed before the intelligence pipeline. It recalculates each transaction hash using the defined formula and compares it with the stored integrity hash. If a mismatch is detected, the transaction is considered tampered and should not be used for AI processing. This protects the intelligence layer from compromised financial data.

This architecture demonstrates a production-style separation of concerns. The frontend handles presentation, the backend handles orchestration, Supabase handles persistence, Groq handles reasoning and edge-tts handles audio synthesis.

V. METHODOLOGY

The methodology of Prophet AI v1.1 combines forecasting, simulation, verification and generative reasoning. The financial data first passes through the

transaction vault. Each transaction contains fields such as amount, date, category, user identifier, status and optional integrity hash.

For integrity verification, a canonical string is created using the transaction fields. The formula used is:

$$\text{Hash} = \text{SHA256}(\text{Amount} + "|" + \text{Date} + "|" + \text{Category} + "|" + \text{UserID})$$

This hash is compared with the stored fingerprint. If both values match, the transaction is treated as verified. If not, the transaction is flagged as tampered. During initial deployment, a staging bypass was temporarily used because JavaScript and Python may represent decimal numbers differently. For example, a value such as 1326.45 may become 1326.4500000000003 in another environment. The production-ready solution is decimal normalization before hash generation.

The forecasting methodology is based on rolling cashflow simulation. The system estimates income, expense behavior, burn rate and liquidity runway. The dashboard presents safe spending zones and risk corridors rather than only a single predicted number. This is important because freelance income is uncertain and a range-based output better represents financial reality.

The AI strategy methodology uses structured prompting. The model receives balance, burn rate, active scenario and transaction context. It is instructed to return a JSON response containing risk level and strategic actions. This output can be directly mapped to frontend components.

The voice methodology converts the generated strategy into a concise briefing. The audio file helps users consume financial intelligence quickly and makes the system more accessible.

VI. IMPLEMENTATION

The implementation was carried out as a full-stack cloud application. The frontend was built using Next.js 14 with TypeScript and Tailwind CSS. Recharts was used for visualizing cashflow trends, forecast corridors, liquidity movement and dashboard indicators. The user interface includes a transaction ledger, 30-day forecast panel, digital handshake section and neural briefing controls.

The backend was developed using FastAPI. It exposes endpoints for briefing generation and job status polling. The POST /briefing endpoint validates incoming data, creates a UUID-based job and starts a background task. The GET /briefing-status/{job_id} endpoint allows the frontend to retrieve current job state and final results.

Groq Llama 3.3 was integrated as the AI reasoning engine. The system prompt instructs the model to act as a high-end financial strategist. The generated response is structured so that the frontend can render risk level, key warnings and three strategic actions without complex text parsing.

edge-tts was used for neural voice synthesis. Once the text strategy is generated, it is converted into MP3 format using selected English or Hindi neural voices. The generated audio is stored inside a static audio directory,

and the frontend plays it through a standard audio interface.

Supabase PostgreSQL was used for data persistence. The database stores user transactions, forecast-related records and audit metadata. The system is designed to support Row Level Security so that each user's records remain isolated.

The application was deployed using Vercel for the frontend and Render for the backend. This deployment structure exposed practical challenges such as cold-start latency, environment configuration, CORS configuration and free-tier resource limitations. These challenges were addressed through async processing, background tasks and static audio hosting.

VII. RESULTS AND DISCUSSION

The implemented system successfully demonstrates a live AI-powered financial intelligence workflow for freelancers. The dashboard presents financial information in a command-center style interface where the user can view forecasted cashflow, transaction history, safe spending ranges and AI-generated risk insights.

The asynchronous polling workflow proved to be one of the most important architectural improvements. Instead of forcing the frontend to wait for LLM inference and MP3 generation, the backend returns a job identifier immediately. This keeps the user interface responsive and prevents timeout failures. The same pattern also improves scalability because background jobs can be handled independently.

The neural voice briefing system successfully converts financial strategies into audio output. This improves user accessibility and differentiates the system from ordinary dashboards. The use of edge-tts allows multilingual support without expensive commercial voice APIs.

The Integrity Shield provides an important security foundation. It ensures that transaction data can be verified before entering the AI pipeline. Although full strict verification requires decimal normalization across frontend and backend environments, the design demonstrates a strong forensic approach to financial data protection.

The project also revealed real-world deployment constraints. Render free-tier infrastructure may introduce cold-start latency, meaning the first request can take longer due to container provisioning. However, once the backend is warm, inference and status polling become smoother. This observation was documented as an engineering trade-off rather than a system failure.

Overall, Prophet AI v1.1 moved beyond a traditional academic dashboard. It became a distributed AI system combining forecasting, cybersecurity, voice synthesis and production-style backend orchestration.

VIII. CONCLUSION

Prophet AI v1.1 successfully demonstrates how artificial intelligence, financial forecasting, cryptographic verification and distributed cloud engineering can be

combined to address the real liquidity problems faced by freelancers. The system transforms personal finance management from passive bookkeeping into proactive financial intelligence.

The project shows that freelancers require more than expense tracking. They require runway prediction, risk simulation, safe spending guidance and direct strategic advice. By implementing a 30-day rolling forecast system and AI-based financial strategist, Prophet AI helps users understand financial risk before it becomes a crisis.

The SHA-256 Integrity Shield strengthens the reliability of the system by ensuring that the AI pipeline operates on verified financial records. This is important because financial intelligence is only trustworthy when the underlying data is authentic. The Verified Ledger concept creates a foundation for future auditability and legal-grade evidence workflows.

The asynchronous polling architecture solved a major deployment challenge by separating user interaction from long-running AI processing. This allowed the system to generate both structured JSON strategy and neural voice briefing without blocking the frontend. The integration of Groq Llama 3.3 and edge-tts further expanded the system into a multimodal intelligence platform.

In conclusion, Prophet AI v1.1 establishes a strong blueprint for next-generation personal financial systems for the gig economy. It combines intelligence, integrity and accessibility into a practical platform that helps independent professionals make safer and more informed financial decisions.

IX. FUTURE WORK

Future work can extend Prophet AI in several directions. The first major enhancement is real-time banking API integration. This would allow automatic synchronization of transactions from bank accounts and reduce manual data entry. Real-time financial feeds would improve forecast accuracy and make the system more practical for everyday use.

The second enhancement is blockchain-based audit storage. While SHA-256 hashing detects tampering, storing selected hash values on an immutable ledger would strengthen external verification and non-repudiation. This would be valuable for contract disputes, client payment proof and freelance evidence documentation.

The third direction is development of an autonomous financial copilot. Instead of waiting for the user to request a briefing, the system could monitor risk continuously and send alerts when liquidity falls below a threshold or when burn rate becomes unsafe.

Mobile application deployment is another important future direction. Freelancers often work remotely and need immediate financial alerts. A mobile app could provide push notifications, voice briefings and quick simulation controls.

Advanced forecasting models such as Temporal Fusion Transformers, LSTM networks and reinforcement learning

agents can also be explored. These models may improve prediction quality for complex income behavior.

Finally, the platform can be expanded for small agencies and teams. A multi-user enterprise edition could support shared cashflow forecasting, project-level profitability analysis and team-level financial runway prediction.

REFERENCES

1. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proc. ACM SIGKDD*, 2016.
2. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
3. S. J. Taylor and B. Letham, "Forecasting at Scale," *The American Statistician*, vol. 72, no. 1, pp. 37-45, 2018.
4. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
5. National Institute of Standards and Technology, "FIPS PUB 180-4: Secure Hash Standard," U.S. Department of Commerce, 2015.
6. FastAPI Documentation, "FastAPI Framework Documentation," 2026.
7. Supabase Documentation, "Supabase PostgreSQL and Row Level Security Documentation," 2026.
8. Vercel Documentation, "Next.js Documentation," 2026.
9. Render Documentation, "Render Cloud Deployment Documentation," 2026.
10. Groq Documentation, "Groq API and Llama Inference Documentation," 2026.
11. Meta AI, "Llama Model Documentation," 2026.
12. edge-tts Project Documentation, "Microsoft Edge Text-to-Speech Python Library," 2026.
13. OWASP Foundation, "OWASP API Security Top 10," 2026.
14. PostgreSQL Global Development Group, "PostgreSQL Documentation," 2026.
15. 11. Babburi, S. Privacy-Preserving Collaborative Framework with Auditable Federated Learning.
16. 12. Gaddam, S. (2024). Integrating machine learning models with continuous integration and continuous delivery (CI/CD) pipelines for a learning-driven approach to software engineering.
17. 13. Reddy, S. K. R. Developing a Modular AI Framework to Enhance Scalability and Personalization in Next-Generation Reward Platforms.
18. 14. Poojari, R. INTELLIGENT SYSTEMS+B108 AND APPLICATIONS IN ENGINEERING.
19. 15. Vasagam, M. (2024, August 30). Ensuring security in modern data pipelines: Practical strategies for data engineers. *International Journal of Intelligent Systems and Applications in Engineering*, 12(22s), 2401.
20. 16. Santhosh Saai Reddy Purmani. (2026). Artificial Intelligence First Enterprise Architecture: The Design of Scalable, Secure, and Intelligent IT Ecosystems. *American Journal of AI Cyber Computing Management*, 6(1(2)), 1-8. [https://doi.org/10.64751/ajaccm.2026.v6.n1\(2\).pp](https://doi.org/10.64751/ajaccm.2026.v6.n1(2).pp)
21. 17. Purmani, S. S. R. (2025). Streamlining IT operations and service management with agile frameworks. *European Journal of Advances in Engineering and Technology*, 12(4), 76-81.
22. 18. Kotte, G. (2025). Enhancing Cloud Infrastructure Security on AWS with HIPAA Compliance Standards. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.5283660>
23. 19. Kumara, S. (2026, February). A Lightweight Deep Learning Based Classification Models for Non-Human Identity Threat Detection. In *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)* (pp. 1-6). IEEE.
24. 20. Kotte, G. (2025). Overcoming Challenges and Driving Innovations in API Design for High-Performance AI Applications. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.5283649>
25. 21. Mahtabi, M., Roshan, M., Muhit, M. M. I., Behvar, A., & Haghshenas, M. (2026). Cryogenic ultrasonic fatigue: Mechanisms, advancements, and insights. *Cryogenics*, 153, 104257. <https://doi.org/10.1016/j.cryogenics.2025.104257>
26. 22. Viswanathan, V. (2023). AI-Augmented Decision Intelligence for Enterprise Systems: Integrating Cognitive Analytics for Resource and Talent Optimization.
27. 23. Viswanathan, V. (2025). Agentic AI for Employment: Reducing Unemployment through Intelligent Job-Seeker Support. *LEX LOCALIS—Journal of Local Self-Government*.
28. 24. Mudusu, S. (2025). Health Insurance Fraud Detection: The Role Of Advanced It Systems In Preventing And Identifying Fraud. *International Journal*, 16(1), 3769-3777
29. 25. Mudusu, S. K. (2026, February 9). AI-augmented data quality engineering. *InfoWorld (Foundry Expert Contributor Network)*.

30. 26. Agrawal, A. M., Gajula, S., Shinde, R. P., Shah, H., & Ghosh, H. (2025, July). Machine Translation for Long Sequences with Enhanced Attention Mechanisms. In 2025 5th International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-6). IEEE.
31. 27. Maturi, S. Y. (2023). Crowdsourced frontier: Unveiling autonomous adversarial cybercapabilities via open AI competition. *International Journal of Intelligent Systems and Applications in Engineering*, 11(1s), 275–284.
32. 28. Sikder, M. Z., Shakil, M. A. I., Ahad, A., Karim, M. F., Intakhab, B., & Islam, D. A. (2025, June). Microwave-Based Detection of Early-Stage Renal Cell Carcinoma Using UHF Range Antenna. In 2025 International Conference on Computer Systems and Technologies (CompSysTech) (pp. 1-6). IEEE.
33. 29. Manoharan, D. (2024). Governance-Oriented Quality Engineering Framework for Healthcare EDI Modernization. *International Journal of Multidisciplinary on Science and Management IJMSM*, 1(2).
34. 30. Ravishankara, M. (2026, February). PlotChain: Deterministic Checkpointed Evaluation of Multimodal LLMs on Engineering Plot Reading. In *SoutheastCon 2026* (pp. 1-8). IEEE.
35. 31. Doragacharla, V. R. (2026). Building Real-Time Pricing Systems for Modern Retail. Available at SSRN 6451760.
36. 32. Adabala, P. K. (2024). Utilizing predictive analytics to improve efficiency and decision-making in ERP-connected supply chains. *International Journal of Intelligent Systems and Applications in Engineering*, 12(22s), 2465
37. 33. Kavuri, S. (2026). An Explainable Machine Learning Framework for Predicting Software Defects in Large-Scale Software Systems. 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC), 1–6. <https://doi.org/10.1109/icaic67076.2026.11395777>
38. 34. Venkata Pavan Kumar Gummadi. (2023). MuleSoft Batch Processing: High-Volume Streaming Architecture. *Computer Fraud and Security*, 50–57. <https://doi.org/10.52710/cfs.886>
39. 35. Gummadi, V. P. K., Chilamkurthi, L. S., & Kavuri, S. (2026). Securing APIs in Government Clouds and Runtime Fabric Using FIPS-Enabled MuleSoft. 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISSET), 1–6. <https://doi.org/10.1109/icaisset66439.2026.11542099>
40. 36. Gajula, S., & Margam, M. (2026). A Secure and Scalable Cloud-Based Banking Service Model Leveraging AI and Advanced Cyber Security. 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC), 1–5. <https://doi.org/10.1109/icaic67076.2026.11395704>