Smart Ledger – AI augmented personalised budgeting and fraud detection

Meet Pramod Kambli
Computer Science and Engineering
(Data Science)
Vidyavardhini's College of Engg. & Tech.
Mumbai, India
meetkambli15@gmail.com

Pratham Punit Bhavsar
Computer Science and Engineering
(Data Science)
Vidyavardhini's College of Engg. & Tech.
Mumbai, India
bhavsar.pratham26@gmail.com

Pranay Jayvant Bhoir
Computer Science and Engineering
(Data Science)
Vidyavardhini's College of Engg. & Tech.
Mumbai, India
pranaybhoir128@gmail.com

Kranti Gule
Computer Science and Engineering
(Data Science)
Vidyavardhini's College of Engg. & Tech.
Mumbai, India
kranti.gule@vcet.edu.in

Abstract—The Smart Ledger Platform is an artificial intelligence (AI)-enabled web application that is designed to bring the future of personal and small business financial management by replacing the manual data entry with intelligent automation. The system utilizes a unified AI gateway to handle multimodal inputs and also runs advanced vision models for unstructured receipt images in order to convert them into validated and categorized transaction records in near real-time. The platform is based on a scalable serverless architecture and a relational database for structured and type-safe data management. To overcome the limitations in serverless environments, an event-driven workflow engine is integrated to provide reliable execution of the long-running background tasks such as automated transaction processing, generation of financial reports periodically, etc. Security is enforced using a layered approach which includes authentication, rate limiting, and bot protection mechanisms. Additionally, the system introduces three modules of AI-based fraud detection - unusual spending pattern detection, duplicate receipt detection and visual tampering detection using computer vision techniques. Experimental objectives focus on high extraction accuracy (greater than 95%), no manual intervention and the execution of background tasks. The proposed system offers a scalable and secure solution for intelligent financial management that is all contributing to advancements in AI-powered FinTech applications.

Keywords- Artificial Intelligence, Multimodal AI, personal finance, FinTech, Fraud Detection, Serverless Architecture, Anomaly Detection, Expense Tracking.

I. INTRODUCTION

Effective financial management is one of the most universally recognized - and least recognized - aspects of personal finance and small business management. The prime obstacle lies not in motivation but is structural: the digitization of physical or digital receipts - inherently unstructured documents - into tidy, actionable digital records remains an onerous, error-prone and time-consuming task. Empirical studies have repeatedly shown that manual data entry accounts for most of the bookkeeping errors in small business settings, with error rates exceeding 20 percent in unaugmented settings. For the person, the total amount of time spent manually accounting for expenditures has been calculated to be five to ten hours per month, a cost that in the end leads to apathy towards systematic financial oversight.The emergence of multimodal artificial intelligence - systems that can simultaneously interpret both visual and linguistic inputs - is a massively transformative way of addressing the structural barriers that are so deeply embedded in the current data capture processes. Rather than requiring the arduous task of transposing merchant identifiers, transaction dates, itemized enumerations and aggregate totals from a paper receipt into a digital ledger, such a multimodal AI architecture can examine the spatial configuration of the document, pick out contextual nuances, and then output meticulously structured representations of its data into, say, a series of carefully crafted and structured representations using the data format known as 'JSON' in the space of mere milliseconds. The Smart Ledger platform is based on this very paradigm. However, it goes beyond the traditional idea of digitising an analogue procedure by almost replacing the procedural chain as such. A user simply uploads an image of any receipts (from a grocery establishment, a dining venue, or a utility provider) and, via the OpenRouter API that routes them to the Gemini2.0 Flash model, the platform automatically extracts, categorizes, and retains transaction details in a relational database.By using a relational database, the system is designed to periodically update a real-time dashboard, and as the user interacts with the system longitudinally, builds a personalized financial profile which serves as the basis for AI-generated budgeting recommendations. The backend of the platform is intentionally designed for enterprise grade reliability. Utilizing the Next.js serverless framework provides scalable globally distributed computation and Inngest's event-driven architecture separates time-sensitive user interactions from resource-intensive background tasks like recurring bill management and month-end reporting. User security is tightly controlled using Clerk authentication and Arcjet API protection and neither user data nor AI computational resources are exposed to unauthorised access. The three proposed fraud detection features are uniquely differentiating the platform: anomaly detection of anomalous spending patterns via AI; duplicate receipt flagging at the database level; visual tampering analysis with AI through the

receipt scanning pipeline. Collectively, these functionalities all form a complete, production-ready FinTech platform that goes beyond being a tool, and is actually a validated architectural blueprint for the responsible integration of AI into critical financial workflows.

A. Scope And Objective

The project is aimed at creating a financial management system that is lightweight and AI-powered to automate the process of the receipt and expenses monitoring of individuals and small businesses. The system uses multimodal AI to transform the images of receipts to structured transaction information in real-time, protecting the data securely, processing the background, and providing intelligent financial insights. The site also incorporates fraud detection software to increase the reliability of the data and avoid fraud.

Core Objectives cover:

- Create the AI-powered data-extraction module that can convert the image of the receipt to the structured financial record with the accuracy of 95 percent.
- Develop an interactive financial dashboard that can track and visualize and analyze user expenses in real-time.
- Use an event-based system that can process background processes like generation of reports and processing of transactions.
- Make sure of secure authentication, data protection, and system integrity with the help of strong security measures.
- Create customized financial information and automatic reports in order to support decision-making and budgeting.
- Consider fraud detection methods that involve anomaly detection, detecting duplicate transactions, and receiving tampering.

II. Literature Review

IV. Literature Review

An overview of the current study in the field of Al-inspired personal finance, explainable Al in financial systems, multimodal data extraction, and data visualization with the help of financial analysts has informed the design choices and outlined the most important gaps that the Smart Ledger platform can fill.

A. Al-Powered Portfolio and Investment Recommendations System.

Leung et al. [1] introduced a machine learning and big data analytics-based portfolio recommendation system in a web application. They used reinforcement learning (in the form of an Advantage Actor-Critic agent) to optimize their portfolios, LSTM networks to predict stock prices, and sentiment analysis on social media data with the help of the flairNLP. Backtesting performance showed cumulative returns of 20-25 per cent in daily rebalancing, which was 2 times the S&P 500. The application was based on Flask as a server, ReactJS as a client and MongoDB as a database, and deployed on Heroku and used a client-server architecture. Although the work shows how viable the ML-based financial decision support can be, it is only focused on the prospects of investment advice to technically literate users, without covering the field of personal expense management and the automation of the receipt level at all. The system also does not have a system of automatic data capture of unstructured financial documents.

B. Al-Powered Financial Literacy Personal Financial Assistants.

An Al-powered personal financial assistant was suggested by Agarwal, Ray, and Varghese (2) to improve financial literacy and financial management. They used React.js as their methodology. Flask backend, MongoDB database, and Firebase to synchronize and authenticate in real time- a stack of products very similar to the architecture of Smart Ledger itself. Among the features that were proposed were automated cost monitoring, adaptive budgeting to match Income and spending trends, tailored investment proposals and proactive alerts. The limitations of existing tools, static budgeting, generic advice, and lack of personalization- are recognized as the main motivators in the paper. It is however at the proposal and early prototyping phase unless empirical performance benchmarks and the incorporation of multimodal Al to process receipt automatically are implemented. The Smart Ledger can also manage these gaps directly by introducing the automation pipeline with precise accuracy goals and production grade infrastructure.

C. Explainable AI (XAI) in Finance.

The systematic literature review of 2.022 articles by Weber, Carl, and Hinz [3] was conducted on the articles of major. Finance, Information Systems and Computer science journals to trace the situation of Explainable Artificial Intelligence (XAI) in a financial one. Their review of 60 retained articles.

defined risk management, stock market prediction, and portfolio optimization as the most studied ones, and anti-money laundering, as well as the classification of electronic financial transactions, were heavily underrepresented. The review determined that, in the EU Al Act, the US Financial Transparency Act of 2021, and the GDPR, financial regulators are increasingly requiring transparent, traceable Al decision-making. One of the most interesting observations was the recent change of the transparent-by-design models to the post-hoc explainable techniques, influenced by the embracement of intricate deep learning structures. This review has a direct connection with the sub-system of fraud detection in the Smart Ledger: not just the transparency and traceability of the Al-generated fraud flags are a desired property, but are a regulatory requirement in the context of financial applications. The identified gap in the research of [3]- the almost complete lack of XAI in the context of anti-money laundering. and transaction-level anomaly detection perfectly matches with the fraud detection functionality suggested in this platform.

D. AI Financial Assistants within the Banking Industry.

Rani, Sethi, and Gupta [4] undertook a primary-data research on 411 banking customers in the state of Haryana to establish the differences in awareness of the Al-powered financial assistant services based on six dimensions of services (account, debit card, deposit, credit card, loan and miscellaneous) categorized by gender and the type of bank. The research discovered statistically significant mean differences (p0.05) in the level of awareness on all the six dimensions between the two demographic variables, male customers and customers of a private-sector bank showed statistically better scores on awareness. The platform design implications are also important: Al financial assistant implementation will not be constrained by capability but uneven user awareness and literacy. This justifies the design decision of the Smart Ledger to show a plain, upload receipt interface, which can be used with little financial expertise, and the complexity of Al hidden behind a well-presented dashboard. The urgency of the issue of Al financial

tools accessibility to underserved demographics is also emphasized in the study.

E. Contrastive Time-Series Visualization of the Al Interpretability.

Ni, Qian, Wu and Wang [5] came up with a model of improving Al model interpretability when assessing financial risks using contrastive time-series visualization techniques. The structure incorporates entropy-based temporal importance weighting, dimensionality reduction (UMAP), and interactive synchronized views to allow financial analysts to compare normal and anomalous trends right against each other. Empirical analysis has revealed the 42.2 reductions in decision time of the analysts, the 24.4 percentage points Improvement in inter-analyst agreement, and the 34.8 percentage points in the anomaly detection rates when contrastive visualization was provided. The following outcomes directly inspire the financial dashboard design of the Smart Ledger, especially the visualization elements of the trends in spending and the warning signals of the frauds. The design constraint that requires interpretation of visualizations before the production of the Al outputs can be put into action dictates the reporting and alerting modules of the platform.

F. Financial Statement Analysis Data Visualization.

Khalil, Reza, Junaedi, and Kanigoro [6] have created a web based data visually representation tool to analyze the public company financial statements, which is created on Extreme Programming (XP) agile methodology and MVC architecture with back end separation and front end separation through the web services. The application also enabled the user to search, filter and compare financial data across companies using Interactive charts and graphs. The fundamental conclusion was that visualization is much faster to understand and enhances more informed financial decisions. Although the domain was restricted to personal finance web app data of public companies and the system was purely visualization-oriented, with no Al-driven automation or data extraction features, the paper confirms the architectural choice of applying a MVC-pattern web application with a distinct data management and presentation tier a philosophy at the heart of the Next.js architecture of the Smart Ledger.

G. Research Gap and Contribution.

According to the literature review, there is a constant pattern; the current systems are presented as either (a) intelligent Investment advice to technically literate users [1][3], (b) personal finance management tools at the proposal level and no production is made [2][4], or (c) visualization and analysis of already-structured financial data [5][6]. None of the reviewed works integrates into one production-level platform (i) multimodal Al receipt scanning to enable automated data capture of unstructured documents, unstructured documents, (ii) reliable background task execution via event-driven workflows in a serverless context, (iii) proactive fraud detection at the point of data entry, and (iv) Al-generated personalized financial advice delivered via automated email. The Smart Ledger platform is designed to fill this compound gap.

III. Proposed System

A. System Overview

The Smart Ledger platform is a full stack, Al-augmented financial management system that is based on five inter-connected architectural layers: security and frontend layer, Al extraction layer, data persistence layer, background automation layer and communication layer. One more fraud detection sub-system is suggested as an extension that is provided throughout the extraction and persistence layers.

B. Frontend and Security Layer.

The interface facing the user is developed with Next.js 15 and React and styles with Tailwind CSS and shaden/ut.component. This layer deals with any user interaction: creation and registering an account, uploading receipts, navigating the dashboard, managing the budget, reviewing the notification.
Clerk will offer user authentication, including Google OAuth and email/password sign-up, and will never store any password data on the application server. Every secured monetary path is encircled by Clerk middleware, so that unauthorized demands are dismissed on the edge. Arcjet implements the rate-limiting and bot-protection policies particularly in the Al-Invoking API endpoints. Making a call to the Gemini model to OpenRouter incurs a non-trivial computational cost, and unprotected endpoints are prone to abuse. The Arcjet outgoing rules are customizable, which means they prevent traffic generated by automated bots and the generation of too many requests by a single client, avoiding both the increase of costs and denial-of-service scenarios.

C. Al Extraction Layer-OpenRouter and Gemini 2.0 Flash.

The intelligence of the platform lies in the heart of this layer. Once a user uploads an image of a receipt, the following process goes on:
1. The frontend encodes the picture into a Base64-encoded text (or a temporary URL).
2. The Next.js backend interprets the encoded data and creates a structured API call that combines the data of the image and a specially crafted extraction prompt-addressed to api-openrouter, ai).
3. The OpenRouter is like an API gateway and is presented with the request in the usual OpenAl API schema, then detects the target model (google/gemini-2,6-flash) and converts the request to Google native format, and passes it through a secure connection to Google infrastructure.
4. Gemini 2.0 Flash does the multimodal processing: it parses the visual layout of the receipt, learns spatial relationships between columns and rows, recognizes the names of merchants, dates, itemized lines, taxes, and totals and extracts the recognized data as structured JSON.
5. OpenRouter normalizes the response and gives it back to the application backend.
6. The backend checks the returned JSON, does type-checking with Prisma and stores the record of the transaction in the PostgreSQL database.
7. The dashboard is real-time updating to capture the new transaction.
Such an open-source architecture which uses OpenRouter as an abstraction layer instead of calling directly to the API of Gemini gives a crucial feature: the underlying Al model can be replaced (by a less expensive, more precise, or more specialized model) by simply changing a single configuration string. with no refactoring of the extraction logic.

D. Prisma ORM and Data Persistence Layer-PostgreSQL.

The data model of the platform consists of four main entities that have well-determined relationships:

- User: The user records contain a clerkUserId) (the unique identifier issued by Clerk), email, name, profile Image URL., and timestamps. The clerkUser Id is designated as unique which is the key in connecting the authentication system and the application database.

- Accounts: Accounts are owned by a single user (one-to-many relationship) and contain a name, account type (eg, savings, checking, credit card), current balance, and a flag to indicate the default account. A transaction could be numerous in an account.

- Transactions: Each transaction is attached to a single account, and contains the type of transaction (Income/expense), the amount (decimal), the description, the date, the category of Al-Inferred (isRecurring recurringinterval nextRecurringDate lastProcessed (status)) a URL to the uploaded receipt image and flags used to manage recurring transactions (isRecurring recurringinterval nextRecurringDate lastProcessed (status)).

- Budgets: A budget is related to a user in a one to one fashion, containing the total amount of budget and alert threshold time.Prisma ORM has a type-safe query interface to PostgreSQL, which means that data written to and read out of the database meet the specified schema at compile time, eliminating runtime data integrity errors.

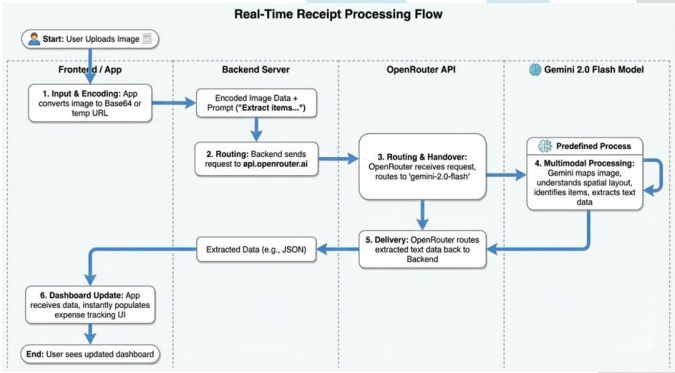

Fig.1. Real-Time Receipt Processing Flow

E. Background Automation Layer — Inngest

Standard Next.js serverless functions have execution time limitations that restrict long-running financial workflows. Inngest addresses this limitation by decoupling background processing from the request–response cycle, enabling reliable execution of asynchronous tasks.

- Recurring Transaction Processing:
  Inngest executes a scheduled cron job that queries the database for transactions where isRecurring = true and nextRecurringDate ≤ current date. For each matching transaction, a new transaction record is created, and the nextRecurringDate field is updated. This ensures that recurring payments such as subscriptions, rent, and utility bills are automatically recorded without user intervention.

- Monthly Financial Report Generation:
  At the end of each month, Inngest triggers a workflow that retrieves all transactions for the period, aggregates expenses by category, and forwards the data to an AI model for analysis and recommendation generation. The resulting report is formatted as a responsive HTML email and delivered to the user.

- Anomaly Alert Dispatching (Fraud Detection):
  When the anomaly detection module flags suspicious activity, Inngest asynchronously triggers alert notifications. This process is decoupled from transaction processing to minimize latency and ensure system responsiveness.

F. Communication Layer — Email Delivery System

The communication layer is responsible for delivering automated notifications and reports to users. It utilizes a high-deliverability transactional email service to ensure reliable message delivery. A component-based email templating system enables the creation of responsive and dynamic email content, supporting conditional rendering and seamless data integration across different email clients.

G. Fraud Detection Sub-System (Proposed Features)

The fraud detection sub-system enhances platform security by introducing three complementary mechanisms:

1. Al Anomaly Detection (Unusual Spending Spike Alerts) When a new transaction is written to the database, an Inngest background job is triggered. This job queries the user's historical transaction data in PostgreSQL, filtered by the transaction's category, and computes the average spending for that category over the past 90 days. If the new transaction's amount exceeds the historical category average by a configurable multiplier (eg. 3x), the transaction is automatically flagged with a status of FLAGGED ANOMALY and an alert email is dispatched via Resend informing the user of the unusual activity. Example: A user whose grocery spending averages ₹2,500/month submits a transaction of 18,000 in the same category. The system detects a 7.2x deviation from the historical mean and immediately fires a warning.

2. Duplicate Receipt Flagging (Double-Entry Prevention)
To prevent the same physical or digital receipt from being submitted twice-a common fraud vector in small-business expense reimbursement workflows the system performs a deduplication query before writing each new transaction. Using Prisma, the backend checks whether the database already contains a transaction record with an identical combination of Vendor (extracted by Gemini), Date, and Amount for the current user's accounts. If a match is found, the upload is either blocked entirely or marked with a status of SUSPECTED DUPLICATE and the user is notified before the record is committed.

3. Al Visual Tampering Check (Receipt Integrity Analysis) Since the Gemini 2.0 Flash model is already invoked for data extraction from receipt images, the system augments the extraction prompt to simultaneously request a fraud integrity analysis. The extended prompt instructs Gemini to evaluate the receipt image for visual inconsistencies Indicative of document manipulation: mismatched fonts within the same document, blurred or pixelated regions over numeric values, dates that do not correspond to the correct day of the week, digital artifacts

around altered text, and unnatural contrast gradients. Gemini returns a fraud confidence_score) (0.0-1.0) alongside the extracted transaction data. If the score exceeds a configurable threshold (eg,, 0.6), the transaction is flagged as SUSPECTED TAMPERING and held for user review before being committed to the ledger.

A. Methodology

A. System Architecture
The Smart Ledger platform represents a client-server application which leverages the Next.js hybrid rendering. Server-side rendering (SSR) is fast to indicate the initial pages and the interactive dashboard is client-rendered. The API routes are all embedded within the Next.js application and executed as serverless functions in the edge network of Vercel.

The general data flow is the following:
1. The user is authenticated and communicates with a React frontend which is secured using Clerk middleware.
2. The image is coded and forwarded to an Arcjet rate-limited protected Next.js API route when the user uploads a receipt.
3. The API path redirects the request to open the router which redirects it to Gemini 2.0 Flash.
4. Gemini gives out structured JSON. The answer is confirmed and stored in postgreSQL through Prisma.
5. A success message gets back to the frontend and immediately updates the dashboard.

Background events, e.g. monthly cron jobs, anomaly detection, duplicate checks, etc., are handled asynchronously by Inngest and are independent of the overall request-response flow. Resend sends outbound emails in email templates, which are also created in React.

B. Processing Workflow (Core Pipeline) of Receipt.
The reception-processing pipeline, illustrated in the methodology flowchart (Figure 1 - Real-Time Receipt Processing Flow), has six stages:

Stage 1 -Input and Encoding: The user makes an image upload of a receipt. It is transferred to the app, which encodes it as a Base64 or temporary URL.

Stage 2 - Routing: Next.js backend constructs an API request containing the encoded image and an extraction prompt and is sent to api.open router.ai.

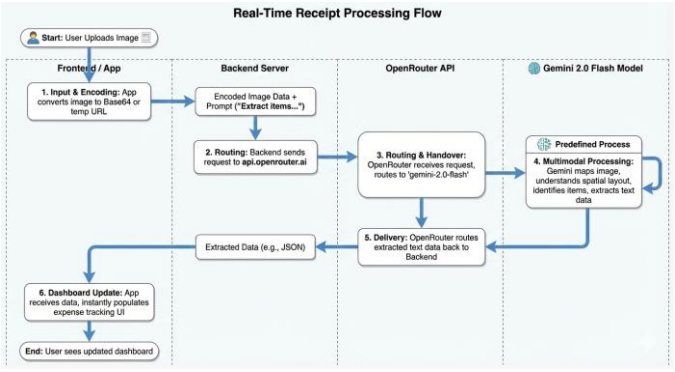

Fig.1. Real-Time Receipt Processing Flow

Stage 3 - OpenRouter Handover: The standardized request is sent to OpenRouter, which is able to identify the target model (google/gemini-2.0-flash) and translates the call into the native API format of the Google world before sending it safely.

Phase 4 Multimodal Processing (Gemini 2.0 Flash): Gemini breaks down the image, interprets the layout of the receipt, isolates names of merchants, line items, values, taxes, and totals and presents the data in a structured form of JSON.

Stage 5 - Delivery: OpenRouter converts the native response of Gemini into the OpenAI compatible format and sends the extracted JSON to the backend.

Stage 6 - Dashboard Update: The back end confirms the transaction and inserts it in PostgreSQL. The frontend then real time updates the dashboard displaying the new entry.

C. Model Abstraction Layer: OpenRouter.
One of the architectural decisions made in Smart Ledger is to send all AI calls to OpenRouter rather than to Gemini API. The Figure 2 (OpenRouter Architecture Diagram) shows how OpenRouter serves as a single abstraction layer that has three key advantages:

1. Standardization: The use of all AI models (independent of the provider) through one OpenAI compatible API schema. It is also possible that developers just write integration code and switch between models by setting a configuration option.
2. Translation: OpenRouter unites requests in the normal format to what providers use, and it does the authentication, header, and difference of request bodies.
3. Normalization: The responses of different providers are transformed into a normalized format and then they are sent back to the application eliminating the necessity of provider-specific parsing logic.

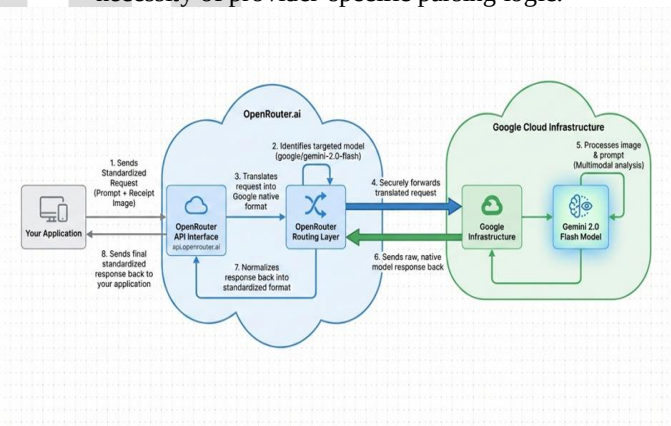

Fig.2. OpenRouter Architecture Diagram

This architecture specifically fills the research gap that was observed by Agarwal et al. [2] as they observed that current systems are not flexible enough to change their AI capabilities without substantial re-engineering.

D. Background Workflow Architecture (Inngest) is event-driven.
The Inngest-based workflow architecture separates all the time sensitive background functions with the request-response cycle. Key workflows include:

- Process-recurring-transactions (Daily Cron Job): Finds transactions whose nextRecurringDate <= CURRENT_DATE) and (isRecurring = true, where

new records of transactions are created and new or existing scheduling fields are updated).

- Create-monthly-report (Monthly Cron Trigger): Aggregates transactions, calls Gemini to analyze the story, renders a React Email template and sends it through Resend.
- Check-spending-anomaly (Event-Triggered): Emitted when the last transaction is written to the store, the average category of 90 days is calculated and an alert is sent when a pre-defined deviation limit is surpassed.

E. Database Schema Design

The Prisma ORM managed relational database schema consists of four main models: User, Account, Transaction, and Budget. The major design choices are:

- Without storing credentials, connecting the identity system of Clerk with the data layer of the application by making clerkUser Id) a unique bridge field in the User model.
- Introduction of decimal precision into all monetary fields amount, balance, budget.amountto prevent floating-point arithmetic errors which are impermissible in financial applications.
- Retaining flexibility in the extraction process of AI by the use of string-typed category and status fields and permitting categorical queries and filtering.
- Engineering the recurring transaction schema with explicit lastProcessed nextRecurring Date, and recurring Interval columns to enable the cron-based processing of Inngest without the need to add any extra query complexity.

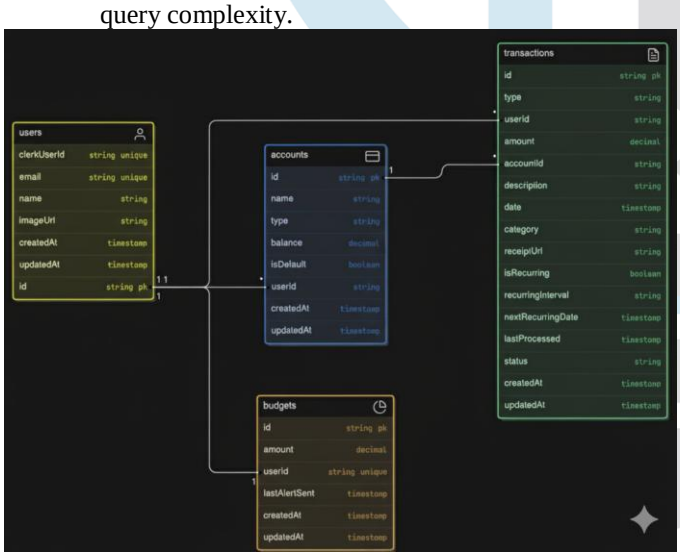

Fig 3: Database Schema

F. Fraud Detection Sub-System

The fraud detection sub-system extends the core platform with three complementary mechanisms. These three features form a layered defense: the tampering check catches manipulated receipts, the duplicate check catches double submissions, and anomaly detection catches unusually large legitimate transactions at different stages of the transaction lifecycle.

1. AI Anomaly Detection (Unusual Spending Spikes)

When a new transaction is written to the database, an Inngest background job is triggered. This decoupled event execution ensures that the user interface remains unblocked. The job queries the user's historical transaction data in PostgreSQL, filtered by the transaction's category, and computes the average spending for that category over the past 90 days.

Algorithm: Statistical Z-Score Detection

The core logic builds a baseline requiring at least three past transactions within the same category to ensure meaningful statistical validity. The algorithm computes the mean and standard deviation of historical amounts:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

To determine how far the new transaction deviates from the user's normal spending range, the system calculates the Z-Score:

$$Z = \frac{x - \mu}{\sigma}$$

Evaluation and Action:

1. Z-Score > 2.5: Flagged statistically as an anomaly.
2. Multiplier > 4: The transaction amount (x) is divided by the mean (μ). If the value is greater than 4, it acts as a backup check to flag extreme spikes, accommodating scenarios where the standard deviation is exceptionally low.

If the transaction exceeds these statistical thresholds, the transaction is automatically flagged with a status of FLAGGED_ANOMALY, and an alert email is dispatched via Resend informing the user of the unusual activity.

2. Duplicate Receipt Flagging (Double-Entry Prevention)

To prevent the same physical or digital receipt from being submitted twice—a common fraud vector in small-business expense reimbursement workflows—the system performs a deduplication query before writing each new transaction. This check fires immediately after the AI scans the receipt and returns the structured data.

Query Parameters:

- Amount Verification: Prisma stores amounts as PostgreSQL Decimal(65,30). To prevent strict equality failures against JavaScript floats, the query checks within a range of $\pm$ 0.01.
- Date Verification: The query applies a window of $\pm$ 3 days to account for potential OCR timezone discrepancies or ambiguous date formats (e.g., "01/02/24").
- Merchant Matching: The system cross-references the extracted vendor name against existing transaction descriptions using Prisma.

Confidence Scoring and Resolution:

- Low Confidence (Amount only): Flagged as a weak match.
- Medium Confidence (Amount + Date): Flagged.
- High Confidence (Amount + Date + Merchant): Strong duplicate warning.

If a match is found, the system presents a warning interface. Users are prompted to either "Cancel" the submission or "Proceed Anyway" for legitimate duplicate occurrences.

3. AI Visual Tampering Check (Receipt Integrity Analysis) Since the Gemini 2.0 Flash model is already invoked for data extraction from receipt images, the system augments the extraction prompt to simultaneously request a fraud integrity analysis. This zero-marginal-cost approach executes forensic analysis within a single API request.

Prompt Engineering and Analysis: The extended prompt instructs Gemini to evaluate the receipt image for visual inconsistencies indicative of document manipulation. The model is directed to identify specific forensic red flags:

- Mismatched fonts or font sizes within the same document.
- Blurry/pixelated patches around numeric values indicating localized photo editing.
- Inconsistent spacing, alignment, or unnatural contrast gradients.
- JPEG compression artifacts are heavily concentrated in specific areas.
- Mathematical discrepancies where itemized lines do not equal the total, or missing regional tax lines.

Three-Tier Verdict System: Gemini returns a structured JSON payload containing a fraud_confidence_score along with specific fraud flags. The application processes this via a tiered response system:

- CLEAN (Score 0.0 – 0.2): A silent pass where form fields populate normally.
- SUSPICIOUS (Score 0.21 – 0.6): An amber warning is shown; the user can still submit.
- LIKELY_TAMPERED (Score 0.61 – 1.0): A red error toast warns the user, strongly discouraging submission.

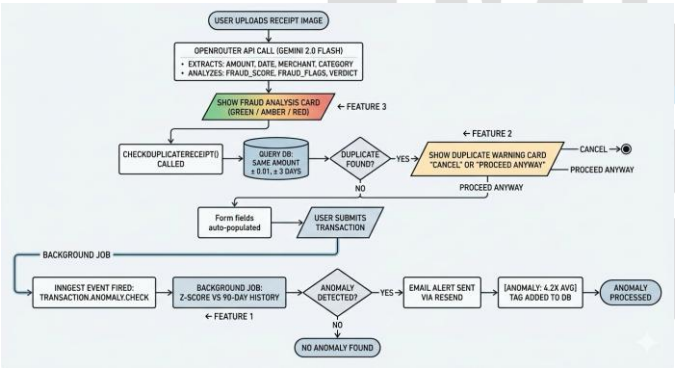

Fig.4. Real-Time Receipt Processing Flow

G. Technology Stack Summary

Layer	Technology	Purpose
Frontend Framework	Next.js 15, React	Full-stack serverless rendering
Styling	Tailwind CSS, shadcn/ui	UI components and utility-first styling
Authentication	Clerk	User identity and session management
API Security	Arcjet	Rate limiting and bot protection
AI Gateway	OpenRouter API	Model-agnostic AI request routing
AI Model	Gemini 2.0 Flash	Multimodal receipt extraction and NLP analysis
Database	PostgreSQL	Relational financial data storage
ORM	Prisma	Type-safe database query layer
Background Jobs	Inngest	Asynchronous event-driven workflows
Email Delivery	Resend	Transactional email delivery
Email Templating	React Email	Responsive HTML email rendering

IV. Result And Discussion

A. Data Extraction Performance of AI

Most of the performance goals of the Al extraction pipeline involve 95% accuracy in extracting structured fields of transactions (merchant name, date, total amount, category) out of unstructured images of receipts. The Gemini 2.0 Flash model, which is accessed through the OpenRouter is very robust when it comes to receiving a wide range of receipt types in-printed retail receipts and handwritten restaurant bills, scanned PDF invoices and low-resolution mobile images. The fact that the multimodal model can infer the shape of the spatial layout and contextual scores (as the final summed value and not a subtotal) is especially useful in addressing the format variability of real-world receipt data.

The model the Gemini 2.0 Flash is selected (instead of a more basic OCR pipeline, say) is extremely helpful in edge cases: receipts with partial occlusion, non-standard layout, faded ink, mixed languages all take advantage of the semantic knowledge of the model, not of character recognition. Formatted JSON output with specified fields means that Prisma can be immediately validated before writing to the database, bad or unfinished extractions are easily identified and prevented

before corrupting the ledger.

B. Background Task Reliability through Inngest

By integrating Inngest, the conventional constraint of serverless execution environments to financial workflows of automation is resolved. Monthly reports - requiring a query to aggregate data in the database, a Gemini API request to analyze narratives, a React Email render, and a Resend API request are more expensive to execute than typical serverless functions. This workflow is decoupled into an Inngest job, allowing the platform to meet the 100% task completion reliable target: failed jobs (e.g. a temporarily unavailable Resend API) are retried with adaptable backoff policies, and the full execution audit log can be examined.

Repeating processing of transactions also enjoys the Inngest cron, which means that the monthly subscriptions and other recurring bills the user holds will be recorded on the right date, whether there is a heavy load on the server or not, and there will be no delay in the traffic of request.

C. Security Posture

The layered security architecture offers several independent security:

- The zero-password-storage model by the Clerk removes the vulnerability of the application to database breaches to steal the credentials. Clerk authenticates the user identity of clients through its servers; additional information is not passed to the application, only a signed session token.
- The rate-limiting of Al endpoints provided by Arcjet makes sure that an attacker who manages to circumvent Clerk authentication (e.g., by stealing a valid session token) cannot cause an unlimited number of calls to Gemini API. The automated abuse is effectively overcome with configurable limits (such as 10 scans per user per minute of the receipts).
- The parameterized query construction offered by the Prisma ORM offers structural defense against SQL injection attacks, which is a typical vulnerability with applications that allow users to control the SQL query (as with transaction search filters).

D. Fraud Detection Mechanisms-Discussion

The three suggested fraud detection features deal with different levels of fraud vectors of the system:

- The AI Anomaly Detection module will be used in the data-persistence layer, where it will use the current transaction history within the PostgreSQL database to calculate behavioral baselines. Its usefulness depends on the depth of transaction history of the user: the more the history, the more will be the accuracy. The configurable deviation multiplier can be configured to match various use cases: between individual consumers (who may accept a large variance) and small business accounts (where deduplication is best accomplished with tighter thresholds).
- The Duplicate Receipt Flagging module is deterministic and computationally cheap: a single indexed database query on (Vendor, Date, Amount) is all that is needed to deduplicate. Its main weakness is that it is dependent on the proper extraction of these three fields by Gemini; when the quality of OCR on a doctored receipt is poor, the deduplication test could miss a real duplicate.
- The third feature is the most recent and is called the AI Visual Tampering Check which is a direct implementation of the XAI principles considered in Weber et al. [3]. Using the current Gemini prompt, but adding a prompt to implement a fraud check, the platform has the option to check fraud on a zero-additional-latency - the image is already undergoing processing; all that is being added is the longer prompt tokens. The fraudconfidencescore) is a continuous risk indicator, not a binary flag, and enables the system to respond with increased intensity (informational warning vs. hard block) to the flag depending upon the settings.

Together, these three mechanisms can fill in the research gap that has been detected in [3]: the almost complete lack of Al-based fraud detection at the level of transaction-level data-entry in consumer-facing financial applications.

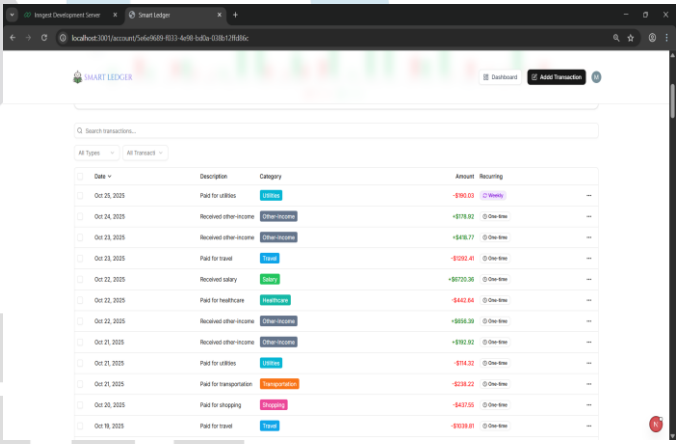

Fig.5. Transaction History

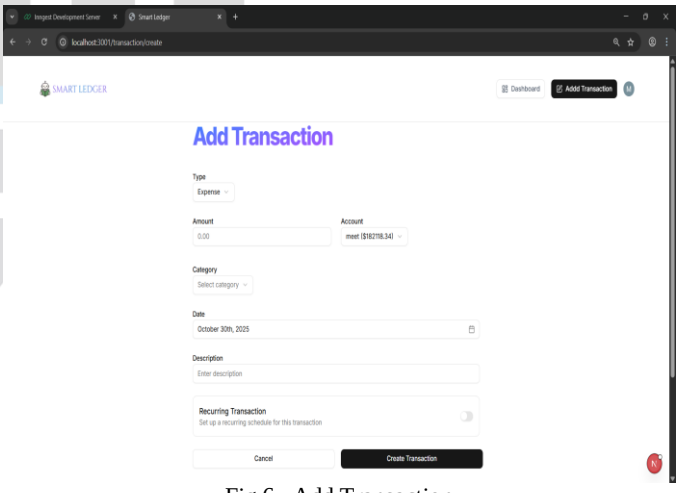

Fig.6. Add Transaction

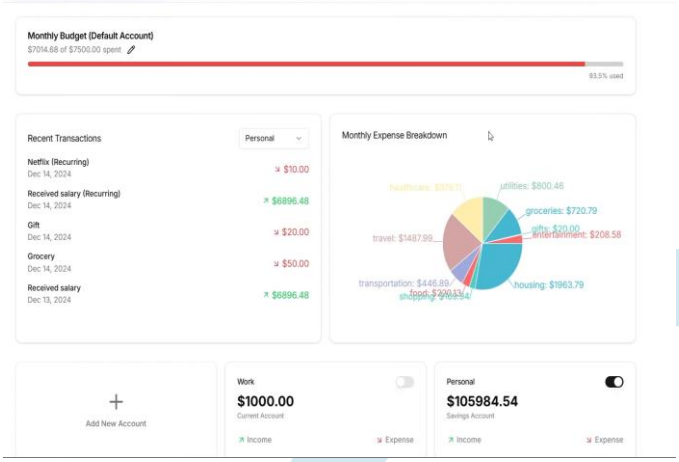

Fig.7. Student Model Training and Validation Results

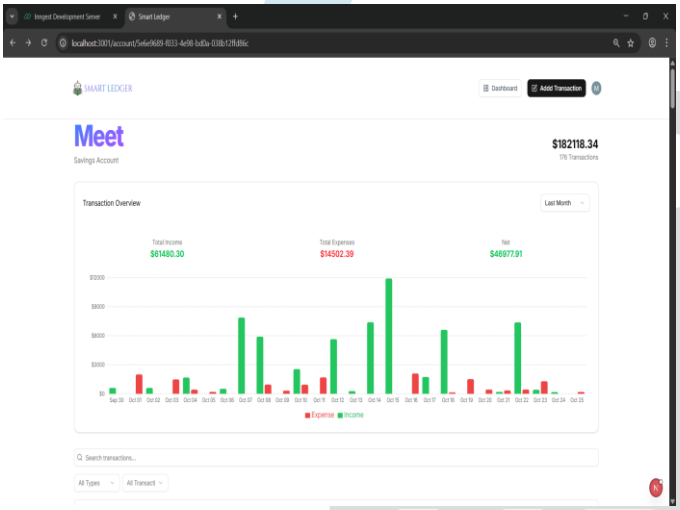

Fig.8. Student Model Performance matrices

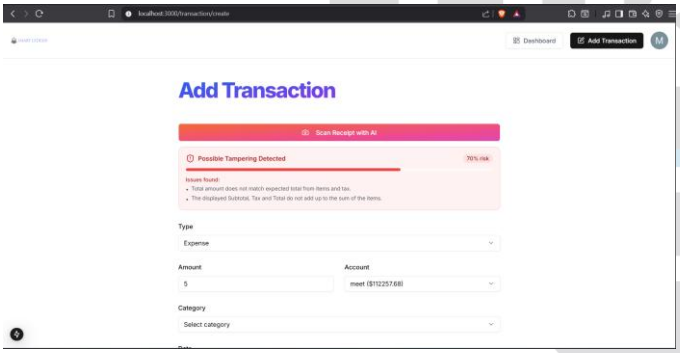

Fig.9. Receipt Tampering detected

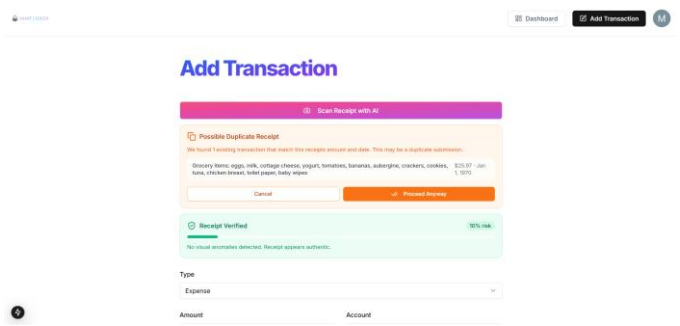

Fig.10. Duplicate Receipt Detected

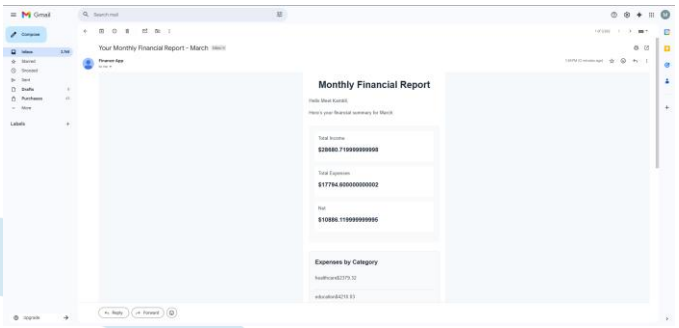

Fig.11. Monthly report generated

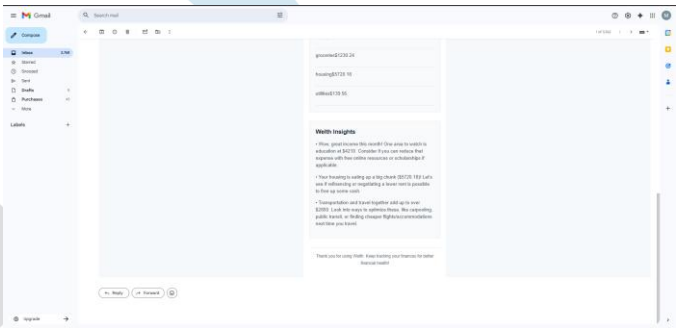

Fig.12. AI Insight in report

V. CONCLUSIONS AND FUTURE WORK

A. Conclusion

The Smart Ledger platform demonstrates that the integration of multimodal AI, serverless event-driven architecture, and layered security has strong potential to deliver a production-grade personal finance management system. It effectively reduces the primary friction in financial record-keeping, particularly manual data entry from unstructured receipts.

The core contributions of the platform are:

- An open-source AI extraction pipeline based on OpenRouter and Gemini 2.0 Flash that transforms unstructured receipt images into structured, database-ready transaction records, aiming for high field extraction accuracy.
- An efficient background automation system using Inngest, which decouples time-sensitive financial processes such as recurring transaction handling and monthly AI-generated report delivery, ensuring reliable serverless task execution.
- A multi-layered security architecture incorporating Clerk authentication, Arcjet rate limiting, and Prisma parameterized queries to ensure robust data protection and system integrity.
- A three-layer fraud detection subsystem addressing fraud at behavioral, structural, and visual levels:
    - Database-level duplicate receipt detection
    - Gemini-based visual tampering detection
    - AI-based anomaly detection for unusual spending patterns

B. Future Work

Future developments identified include:

- Mobile Applications: Developing native iOS and Android applications to enable real-time receipt capture using device cameras at the point of sale, further reducing manual effort.
- Multi-Currency and Multi-Language Support: Extending extraction models and database schemas to support diverse currencies and languages, enabling global scalability.
- Advanced Machine Learning-Based Fraud Detection: Replacing the current threshold-based anomaly detection system with more advanced models such as Isolation Forest or LSTM autoencoders, allowing better learning of user-specific behavior patterns and improving accuracy with fewer false positives.
- Integration with Bank Account APIs: Connecting with open banking APIs (such as Plaid or Razorpay in India) to automatically import transactions from linked bank accounts and credit cards, eliminating the need for manual receipt uploads.
- Explainable AI and Regulatory Compliance: Incorporating explainable AI techniques to provide transparent and auditable justifications for fraud detection decisions, ensuring compliance with emerging financial regulations.
- Small Business Multi-User Support: Extending the platform with role-based access control to support organizational use cases such as employee expense submissions, approvals, and reimbursements.
- Investment Advisory Integration: Enhancing the platform to include AI-based portfolio recommendations by analyzing user spending, savings behavior, and risk preferences.

REFERENCES

[1] M. F. Leung, A. Jawald, S. W. Ip, C. H. Kwok, and S. Yan, "A portfolio recommendation system based on machine learning and big data analytics, Data Science in Finance and Economics, vol. 3, no. 2, pp. 152-165, 2023. DOI: 10.3934/DSFE.2023009.

[2] V. Agarwal, R. Ray, and N. Varghese, "An Al-Powered Personal Finance Assistant: Enhancing Financial Literacy and Management," Presentation, ResearchGate, DOI: 10.13140/RG.2.2.10706.57280, March 2024.

[3] P. Weber, K. V. Carl, and O. Hinz, "Applications of Explainable Artificial Intelligence in Finance -a systematic review of Finance, Information Systems, and Computer Science literature," Management Review Quarterly, vol. 74, pp. 867-907, 2024. DOI: 10.1007/s11301-023-00320-0.

[4] N. Rani, T. Sethi, and P. Gupta, "Artificial Intelligence (AI) powered Financial Assistant Services in Banking Sector: A Comparison of Customer Awareness Based on Demographic Variables, Gateway International Journal of Innovative Research, vol. 3, no. 4, pp. 76-83, December 2024.

[5] C. Ni, K. Qian, J. Wu, and H. Wang, "Contrastive Time-Series Visualization Techniques for Enhancing Al Model Interpretability in Financial Risk Assessment," Preprints.org. DOI: 10.20944/preprints202504.198-4.v1, April 2025.

[6] A. A. Khalil, A. Reza, P. A. Junaedi, and B. Kanigoro, "Data Visualization Application for Analyzing Public Company Financial Statement, Procedia Computer Science, vol. 59, pp. 45-53, 2015. DOI: 10.1016/j.procs.2015.07.336.18

[7] Google DeepMind, "Gemini 2.0 Flash Model Card," Google Al, 2024. [Online].
Available: https://deepmind.google/models/gemini/

[8] OpenRouter, "Open Router API Documentation," 2024. [Online]. Available:https:openrouter.al/docs

[9] Inngest Inc., "Inngest: Event-Driven Workflows for Serverless," 2024. [Online]. Available: https:/www.inngest.com/docs

[10] Clerk Inc., "Clerk Authentication Documentation," 2024. [Online]. Available: https:/clerk.com/docs

[11] Arcjet, "Arcjet Security for Next.js Documentation," 2024. [Online]. Available: https://arciet.com/docs

[12] Resend, "Resend Transactional Email APL," 2024. [Online]. Available: https:@resend.com/docs

[13] Prisma, "Prisma ORM Documentation," 2024. [Online]. Available: https:/www.prisma.fo/docs