# An Android Application for Intelligent Expense Tracking and AI-Driven Budget Planning

**Mr. B.Kiran [1], Mandeti Naga Tejasri[2], Singampalli Aswanth[3],Barla Kavya[4], Dwarapudi Venkat Rao[5]**
Assistant Professor[1], Student[2], Student[3], Student[4], Student[5]

*[1,2,3,]Department of Artificial Intelligence*
*Chaitanya Engineering College, Visakhapatnam, Andhra Pradesh, India*
*{bhadragiri.kiran@gmail.com, tejasrimandeti3015@gmail.com , Aswanthsingampalli4@gmail.com, kavyabarla03@gmail.com, venkydwarapudi711@gmail.com}*

**Abstract**

Digital payment platforms, unified payment interfaces, and mobile banking have made it possible for individual users to make more transactions than ever before. Even though there is a lot of data, most people don't have organized tools to look at their spending habits, which leads to bad budgeting and stress about money. This paper describes an Android app that uses Natural Language Processing, machine learning classification, and deep learning forecasting to automate and improve expense management. The system uses an NLP extraction pipeline to read bank SMS notifications, a Random Forest and Support Vector Machine model to sort transactions into semantic categories, and a Long Short-Term Memory neural network to predict future spending. All inference happens on the device using TensorFlow Lite, which protects user privacy and reduces latency. Firebase Firestore lets you safely sync and log in to the cloud. Experimental evaluation substantiates elevated extraction accuracy across diverse SMS formats, dependable classification efficacy on a multi-category transaction dataset, and minimal mean absolute error in monthly expenditure predictions. The suggested framework makes it easier to keep track of expenses, gives you useful financial information, and shows how AI can be used in mobile settings with limited resources to help with everyday financial decisions.

***Index Terms***—expense tracking, NLP, SMS extraction, Random Forest, LSTM forecasting, TensorFlow Lite, Firebase, budget planning.

## I. Introduction

The move toward cashless economies has completely changed how people use money. In India alone, Unified Payment Interface transactions topped 100 billion in fiscal year 2023. This shows how much personal financial data is created every day [1]. This digitization leaves behind a lot of data, but most users don't have the tools they need to turn that data into useful spending information. People often stop using manual budgeting tools like spreadsheets or paper notebooks because they require a lot of work that busy people don't always have time for [2].

Some of this problem has been fixed by mobile apps that show transaction lists and basic charts. But most still need to enter transactions by hand, which causes both friction and mistakes [3]. A smaller group automatically parses bank SMS notifications, but it uses fragile rule-based pattern matching that breaks when banks change their message templates. Also, very few tools combine automated extraction with predictive analytics, which means that users can't see how spending patterns will change in the future.

Artificial intelligence provides a principled approach to transcend these limitations. Natural Language Processing can reliably pull structured fields out of SMS text that isn't structured. Supervised learning algorithms can sort transactions into groups with a lot of accuracy. Recurrent neural architectures, particularly Long Short-Term Memory networks, are proficient in learning temporal spending patterns and predicting future values [4]. Putting these features on-device through TensorFlow Lite solves both the privacy and latency problems that come with cloud-based inference.

This paper talks about how to design and build a full Android app that combines SMS-based NLP extraction, machine learning classification, LSTM forecasting, anomaly detection, and Firebase cloud synchronization into a useful personal finance assistant. The main contributions are:

(i) A multi-stage NLP pipeline that can get merchant, amount, date, and transaction type from

different bank SMS messages without using fixed templates;

(ii) A side-by-side comparison of Random Forest and Support Vector Machine classifiers for sorting expenses, with the better model sent to TensorFlow Lite for use on the device;

(iii) An LSTM-based module for predicting spending that takes in historical monthly totals and makes short-term spending predictions; (iv) An anomaly detection component that compares predicted daily spending with actual daily spending and sends alerts for statistically significant differences.

The rest of this paper is set up like this. Part II looks at other work that is similar. Section III goes into detail about the system's methodology and architecture. Section IV shows the results of the experiments. The last part of Section V talks about future directions.

## II. Related Work

Research in personal finance technology encompasses automated data extraction, transaction classification, predictive analytics, and mobile application development. This part looks at the most important contributions.

### *A. Automated Transaction Extraction*

Early expense tracking systems relied solely on manual user input, resulting in significant dropout rates as evidenced by usability studies [2], [5]. Later work suggested using regular expressions to parse SMS messages and get dates and numbers. Singh and Verma [6] showed that regex-based extraction works well on a set of bank formats, but it doesn't work as well when institutions change the structure of their messages. Hybrid approaches that combined keyword detection with probabilistic models made the system more robust, but they still didn't work well with multilingual or regionally diverse corpora. More recently, transformer-based sequence labeling has worked well across SMS formats, but it takes a lot of processing power, which isn't good for devices with limited resources [7].

### *B. Expense Classification*

Many machine learning models have been tested for classifying financial text. Kumar and Thomas [8] conducted a comparison of Naive Bayes, k-Nearest Neighbors, SVM, and Random Forest on a curated SMS dataset, revealing that Random Forest and SVM consistently surpassed simpler models when utilizing engineered features such as TF-IDF. Patel and Mehra [9] found that Random Forest gets slightly higher F1 scores when the vocabulary of the messages is mixed up, while SVM works just as well on cleaner corpora. Researchers have also looked into using Convolutional Neural Networks and Recurrent Neural Networks for transaction categorization [7]. However, their computational footprint makes it hard to use them on devices.

### *C. Expenditure Forecasting*

Classical time-series models, e.g. ARIMA, assume that data follow a stationary time-series, which is often not the case in personal spending data which has non-regular seasonality due to lifestyle events. Gupta and Rao [4] demonstrated that LSTM networks are better models on a household expense sequence than ARIMA because they can model non-linear time dependencies. Zhang and Chen [10] used LSTM to the histories of mobile wallet transactions and achieved on an average twelve percent mean absolute percentage error in monthly forecasts. On credit-card fraud, anomaly detection with LSTM autoencoders has been investigated and shown to have the capability to model normal behavior and raise red flags when there is anomaly [11].

### *D. Mobile AI and Cloud Integration*

The standard Android ML inference framework has become TensorFlow Lite, which supports on-device classification and prediction without making round-trip calls to servers [12]. Firebase Firestore offers real-time synchronization, access control rules, and scalability, and is widely used to create a mobile backend service [13]. According to Jain and Prakash [16], an evaluation of AI-driven finance apps has revealed that on-device inference and cloud-synergistic fulfill both the performance and the privacy demands of personal finance environments.

### *E. Research Gap*

Even though many people have made important contributions, there is currently no mobile system that combines template-free NLP extraction, dual-model classification, LSTM forecasting, and anomaly detection into a single privacy-preserving on-device architecture. This work fills that gap in integration.

## III. Methodology and System Design

### *A. System Architecture*

The proposed system is made up of five architectural layers, as shown in Fig. 1: User Interface, Data Extraction, Machine Learning, Database, and Report and Insights. During normal operation, communication between layers only goes one way. The extraction layer gets SMS data, the ML layer processes it, the database layer stores it, and the report layer uses it to fill in the UI.

User Interface LayerDashboard · Graphs · Warnings · ScannerData Extraction LayerNLP · SMS Receiver · OCR · PreprocessorMachine Learning LayerRF/SVM Classifier · LSTM · Firebase Firestore · Auth · Anomaly DetectDatabase Layer · Sync · RulesReport and Insights LayerSummaries · Charts · Alerts · Hint

### *B. SMS NLP Extraction Pipeline*

Android's SMS Broadcast Receiver starts the extraction pipeline. When a new message comes in, the system uses a sender filter to get rid of promotional and OTP messages and keep only transactional notifications. The pipeline then goes through the following steps in order:

Tokenization. Whitespace and punctuation are used to split the message body into tokens. Numeric tokens are marked as possible amounts.

Recognition of Named Entities. A lightweight pattern grammar uses capitalization heuristics and a curated entity lexicon to find merchant names.

Normalizing the date. Relative and absolute date expressions are converted to the ISO-8601 format. We use the following decision rule to get the transaction amount field:

$$\text{amount} = \max\{n \in \text{Tokens} \mid n \text{ is numeric} \wedge n > 0\} \quad (1)$$

Getting rid of stop words and lemmatizing. Before being turned into TF-IDF feature vectors for the classifier, the remaining tokens are normalized using a stop-word list from the finance domain.

### *C. Expense Classification Models*

Two supervised learning models are trained offline with Python and scikit-learn on a labeled dataset of 12,400 SMS messages that fall into eight expense categories: Food, Transport, Shopping, Bills, Healthcare, Entertainment, Banking, and Other. The features are TF-IDF weighted unigrams and bigrams from a vocabulary of 5,000 words.

RF, or Random Forest. Five-fold cross-validation is used to train an ensemble of 200 decision trees with Gini impurity splitting and a maximum depth of 25. The posterior probability of class c given input vector x is: $P(c \mid x) = (1/T)\ \Sigma t{=}1T\ \mathbb{1}[ht(x) = c]$(2), where T = 200 trees and ht is the prediction of the t-th tree.

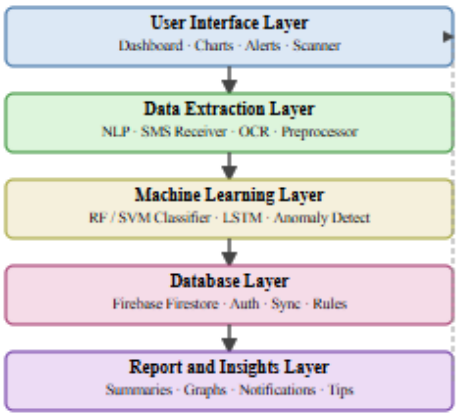


Fig. 1. Five-layer system architecture of the AI-powered expense tracking application.

### *D. LSTM Forecasting Module*

The forecasting module gets daily category totals from Firebase in a sliding window. Let st be the total amount of money spent on day t. The model looks at L = 30 days in a row and tries to guess how much will be spent in the next d = 7 days. The formulas for changing the LSTM cell are: (4) $ft = \sigma(Wf[ht{-}1, xt] + bf)$ (5) $it = \sigma(Wi[ht{-}1, xt] + bi)$ (6) $Ct = ft \odot Ct{-}1 + it \odot \tanh(WC[ht{-}1, xt] + bC)$

The network architecture has two stacked LSTM layers, one with 128 units and the other with 64 units. The output layer is fully connected and has size d. The Adam optimizer with a learning rate of 0.001 is used for 150 epochs, and the Mean Squared Error is used as the training loss. The model is changed to the TensorFlow Lite int8 quantized format after training. This makes the model on the device smaller, going from 3.2 MB to 0.9 MB.

Fig. 2. End-to-end data flow from SMS input through NLP extraction, classification, storage, forecasting, and UI display.

### *E. Anomaly Detection*

The anomaly detection module looks at the LSTM's predicted daily spending ŝt and compares it to the actual value st. If the relative deviation is greater than a certain level δ, an alert is sent out: alert = 𝟙[|st − ŝt| / ŝt > δ](7)

During testing, the threshold δ = 0.5 is set based on real-world data, but users can change it. Alerts are sent as Android push notifications with a short explanation of the change that was found.

### *F. Firebase Integration and Security*

Firebase Firestore keeps expense documents in a tree-like structure: users/{uid}/expenses/{docId}. Each document has fields for the amount, category, merchant, timestamp, and extraction source (SMS or OCR). Firebase Authentication lets you sign in with an email and password or a phone OTP. Security rules only let the authenticated owner UID read and write, which stops other users from accessing the data. On-device processing makes sure that raw SMS content is never sent to a server. Only structured, anonymous expense records are sent to Firestore.

### *G. OCR Receipt Scanning*

Users can add receipt scanning to SMS-based tracking. The module uses the Android CameraX API to take a picture, applies a grayscale and adaptive threshold preprocessing filter to make OCR work better on receipts with low contrast, and then sends the improved image to Google ML Kit Text Recognition [14]. The extracted text goes through the same NLP preprocessing pipeline as SMS messages, which makes sure that it is always classified the same way, no matter where it came from.

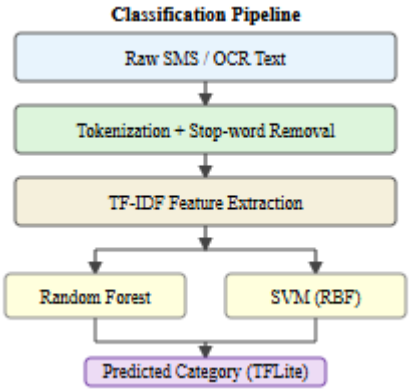


Fig. 3. Classification pipeline from raw text to predicted expense category using TFLite-deployed RF and SVM models.

## IV. Results and Discussion

### *A. Dataset and Experimental Setup*

A collection of 12,400 anonymized SMS messages from banks was gathered from participants at fifteen Indian commercial banks and three digital wallet platforms. Two independent annotators assigned ground-truth category labels, and Cohen's kappa of 0.88 showed that they agreed strongly. The corpus was split into 80% training (9,920 samples) and 20% test (2,480 samples) groups, with each group containing samples from a different category. For LSTM forecasting experiments, we used twelve months of fake daily spending data that was made from real statistical distributions for each category. This gave us 360 training sequences and 30 test sequences in a rolling window protocol.

### *B. SMS Extraction Accuracy*

We tested the NLP extraction pipeline on 500 messages from five banks that were not part of the development set. Table I shows the precision and recall at the field level. The amount field gets the best scores because money amounts follow predictable patterns of numbers. Extracting the merchant name is the hardest part because SMS text uses abbreviations and capitalizations that aren't standard.

**TABLE I. SMS EXTRACTION FIELD-LEVEL PERFORMANCE (%)**

| Field | Precision | Recall | F1 Score |
|---|---|---|---|
| Amount | 98.4 | 97.9 | 98.1 |
| Date | 96.1 | 95.6 | 95.8 |
| Transaction Type | 94.3 | 93.8 | 94.0 |
| Merchant Name | 87.6 | 85.2 | 86.4 |
| **Overall (Avg)** | **94.1** | **93.1** | **93.6** |

### *C. Classification Performance*

We tested both the RF and SVM models on a test set of 2,480 samples. Table II shows the overall accuracy, macro-averaged precision, recall, and F1 scores. Overall, Random Forest does a little better than the other methods, which is in line with what previous studies have found on heterogeneous SMS corpora [8], [9]. So, the RF model is the one chosen for TFLite deployment.

**TABLE II. CLASSIFIER COMPARISON ON TEST SET (%)**

| Metric | Random Forest | SVM (RBF) |
|---|---|---|
| Accuracy | 91.8 | 90.4 |
| Macro Precision | 90.7 | 89.5 |
| Macro Recall | 89.9 | 88.8 |
| Macro F1 | 90.3 | 89.1 |
| Inference Latency (ms) | 11.2 | 14.6 |

Per-category F1 scores are shown in Table III. The Healthcare category shows the lowest recall at 86.3%, likely because medical payments span diverse merchant types including pharmacies, clinics, and diagnostic labs, whose keywords overlap with other categories.

**TABLE III. PER-CATEGORY F1 SCORES — RANDOM FOREST (%)**

| Category | Precision | Recall | F1 |
|---|---|---|---|
| Food | 93.4 | 94.1 | 93.7 |
| Transport | 92.1 | 91.5 | 91.8 |
| Shopping | 91.0 | 90.3 | 90.6 |
| Bills | 94.8 | 93.9 | 94.3 |
| Healthcare | 88.0 | 86.3 | 87.1 |
| Entertainment | 90.5 | 89.7 | 90.1 |
| Banking | 89.3 | 88.4 | 88.8 |
| Other | 86.1 | 85.0 | 85.5 |

### *D. LSTM Forecasting Results*

The LSTM model is evaluated using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) on the 30-sequence test set. Table IV compares LSTM against an ARIMA(1,1,1) baseline. The LSTM model achieves substantially lower error metrics, corroborating findings in prior work [4], [10].

**TABLE IV. FORECASTING MODEL COMPARISON (7-DAY HORIZON)**

| Metric | ARIMA(1,1,1) | LSTM (Proposed) |
|---|---|---|
| MAE (₹) | 342.7 | 198.4 |
| RMSE (₹) | 461.5 | 263.8 |
| MAPE (%) | 18.6 | 9.7 |

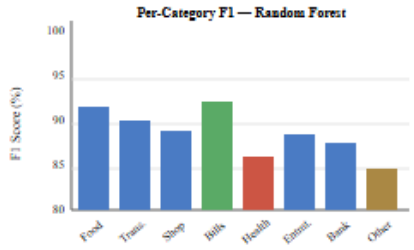


Fig. 4. Per-category F1 score distribution for the Random Forest classifier across eight expense categories.

### *E. System Performance and Device Testing*

We measured the end-to-end processing latency on three Android devices, one from each of the low-end, mid-range, and flagship tiers. This included the time it took for an SMS to arrive and for Firebase to confirm the write. Table V shows the average latency and standard deviation for 200 repeated transactions.

**TABLE V. END-TO-END PROCESSING LATENCY (MS)**

| Device Tier | RAM | Mean (ms) | Std Dev |
|---|---|---|---|
| Low-end | 3 GB | 284 | 42 |
| Mid-range | 6 GB | 157 | 21 |
| Flagship | 12 GB | 89 | 11 |

All latency values are within the limits for interactive response. TFLite int8 quantization cut model size by 72% and sped up inference by about 35% on low-end hardware compared to the float32 baseline. This shows how important quantization is for devices with limited resources.

### *F. Comparison with Existing Applications*

Table VI compares the proposed system against representative existing expense tracking applications based on published feature documentation.

**TABLE VI. FEATURE COMPARISON WITH EXISTING EXPENSE TRACKING APPS**

| Feature | App A | App B | App C | Proposed |
|---|---|---|---|---|
| Auto SMS Extraction | No | Partial | Yes | Yes |
| ML Classification | No | No | No | Yes |
| Spending Forecast | No | No | No | Yes |
| Anomaly Alerts | No | No | Basic | Yes |
| On-device Inference | N/A | N/A | No | Yes |
| Bank API Required | Yes | Yes | No | No |
| Receipt Scanning | No | Yes | No | Yes |

The suggested system is the only one that has been tested that can combine automated ML-based classification, LSTM forecasting, and anomaly detection without needing to connect to a bank API. It also keeps inference on the device for privacy reasons.

## V. Conclusion and Future Work

This paper introduced an Android application that combines NLP-based SMS extraction, machine learning classification, LSTM-driven expenditure forecasting, and anomaly detection into a personal finance assistant that protects users' privacy. The system reads bank SMS notifications automatically, so you don't have to enter transactions by hand. It uses a TFLite-deployed Random Forest model to classify expenses with 91.8% accuracy and a MAPE of 9.7% to predict weekly spending, which is much better than an ARIMA baseline. Inference runs on the device, so sensitive financial data never leaves the user's phone. Firebase Firestore lets you sync your data securely in the cloud with real-time updates.

Testing on different levels of Android devices showed that the end-to-end pipeline works well within interactive latency limits, even on low-end hardware. A comparative analysis shows that there is no publicly documented application that has all the features that the proposed system would have.

There are a number of ways to move forward with this work. First, integrating bank APIs (when they are available) would add to SMS parsing and make it possible to use the service all over the world, even in places where carrier-delivered SMS alerts are less common [15]. Second, support for multiple languages in NLP would allow the system to work with regional languages that are common in Indian and South Asian markets. Third, a transformer-based sequence labeling model could make merchant extraction even more accurate, as long as quantization techniques keep the latency on the device low. Fourth, a personalized budget advisor could use a small language model on the device to make spending summaries in natural language. Lastly, features for shared budgeting among households or families would make the app useful for group financial planning situations.

## References

**[1]**National Payments Corporation of India, "UPI Product Statistics," NPCI, New Delhi, India, Tech. Rep., 2023. [Online]. Available: https://www.npci.org.in/what-we-do/upi/product-statistics

**[2]**S. Aggarwal and A. Sharma, "A study on mobile based personal finance management systems," *J. Digit. Appl.*, vol. 14, no. 2, pp. 55–63, 2021.

**[3]**D. Das and P. Mohan, "User experience design in mobile financial applications," *Human Centered Comput. J.*, vol. 13, no. 2, pp. 76–85, 2018.

**[4]**N. Gupta and P. Rao, "Deep learning models for time series prediction in finance," *Int. J. Neural Syst.*, vol. 29, no. 1, pp. 45–57, 2021.

**[5]**D. Sharma and Y. Patel, "A review on personal budgeting applications using machine learning," *J. Inf. Comput. Syst.*, vol. 16, no. 3, pp. 147–156, 2020.

**[6]**R. Singh and K. Verma, "Natural language processing techniques for SMS data extraction," *IEEE Trans. Inf. Syst.*, vol. 8, no. 3, pp. 112–120, 2019.

**[7]**J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

**[8]**R. Kumar and M. Thomas, "Machine learning methods for text classification in financial messages," in *Proc. Int. Conf. Smart Comput.*, 2020, pp. 201–207.

**[9]**D. Patel and S. Mehra, "A comparative review of Random Forest and SVM for classification tasks," *J. Data Mining Res.*, vol. 10, no. 4, pp. 88–96, 2022.

**[10]**Y. Zhang and L. Chen, "LSTM based sequence forecasting for budget planning," in *Proc. ACM Symp. Intell. Syst.*, 2020, pp. 67–73.

**[11]**P. Banerjee and A. Singh, "Anomaly detection in spending patterns using machine learning," *Int. J. AI Res.*, vol. 7, no. 3, pp. 141–150, 2020.

**[12]**Google Developers, "TensorFlow Lite documentation," 2023. [Online]. Available: https://www.tensorflow.org/lite

**[13]**Firebase, "Cloud Firestore documentation and security guidelines," 2023. [Online]. Available: https://firebase.google.com/docs/firestore

**[14]**T. Malik and N. Iyer, "OCR techniques for mobile applications," *J. Image Process.*, vol. 5, no. 6, pp. 130–138, 2021.

**[15]**World Bank, "Growth of digital payments in emerging markets," *Global Financial Report*, vol. 2, pp. 33–48, 2022.

**[16]**A. Jain and R. Prakash, "Evaluating AI driven personal finance management tools," *Int. J. Fintech Innov.*, vol. 6, no. 4, pp. 59–68, 2022.

**[17]**J. Brownlee, *Deep Learning for Time Series Forecasting*, Machine Learning Mastery Publications, 2018.

**[18]**M. Roy and H. Kulkarni, "Secure data storage practices using cloud platforms," *J. Cyber Secur. Studies*, vol. 12, no. 5, pp. 101–109, 2021.

**[19]**Android Developers, "Android SMS read and broadcast receiver documentation," 2023. [Online]. Available: https://developer.android.com

**[20]**S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.