International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

AI-Driven Intelligent Expense Management Using Data Analytics for Personalized Financial Insights

Keerthana V, 3rd year BCA

3rd Year BCA, BMS College of Commerce and Management

Email: keerthana.bca.20082006@bmsccm.ac.in

Abstract - Managing shared expenses can be challenging when transactions are spread across multiple people, groups, and categories. Conventional expense-splitting tools primarily record transactions and calculate balances, while users may still need help interpreting spending patterns and planning budgets. This paper presents SplitSync, a cross-platform expense-management application developed with Flutter and Firebase, and outlines an approach for extending expense records into useful financial insights through data analytics and intelligent decision support. The application overview describes functions for user authentication, group management, expense creation and editing, automatic split calculations, real-time synchronization, offline support, notifications, localization, QR-based group joining, and PDF export. Structured records such as transaction amount, category, date, group, and participant shares can provide a foundation for descriptive analytics, budget monitoring, and future predictive features. The paper proposes a staged analytics workflow involving data validation, category aggregation, trend analysis, and user-facing explanations. Receipt extraction and automated categorization are considered supporting capabilities where implemented and tested. Because predictive recommendations require sufficient data and validation, they are treated here as proposed extensions rather than as proven outcomes. The paper discusses the system architecture, data flow, security considerations, and an evaluation plan for assessing correctness, usability, synchronization, and the usefulness of insights. SplitSync illustrates how an everyday financial application can be designed to connect expense tracking with data-driven decision support while maintaining transparency about the current implementation and future work.

1. Introduction

Digital payment methods and group activities have made it common for people to share expenses for travel, accommodation, meals, subscriptions, and household needs. Although an expense can be divided mathematically, coordinating records among several participants can become difficult. People may forget transactions, misunderstand who paid, or struggle to see how spending changes over time. An expense-management application can centralize these records, calculate each participant's share, and make group balances easier to understand.

SplitSync is a cross-platform mobile application built with Flutter and Dart, with Firebase services used for authentication and cloud-backed data operations. Its purpose is to support shared-expense tracking, group coordination, synchronization across devices, and clear presentation of balances. Beyond recording transactions, structured expense data can also support analytics such as category summaries, monthly comparisons, budget variance, and recurring-spending patterns. These capabilities connect the application to the conference theme of AI, Data & Intelligent Systems.

This paper documents the application concept and technical design and proposes a responsible path toward intelligent financial insights. It distinguishes features described in the project overview from analytical

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

capabilities that require further implementation and evaluation. No claim is made that predictive accuracy or user benefits have already been demonstrated without measured test results.

2. Problem Statement

Users who share expenses need both accurate settlement calculations and a clear view of their spending. Basic record-keeping may not adequately explain spending trends, identify budget deviations, or help users interpret historical transactions. At the same time, an intelligent expense system must avoid presenting uncertain predictions as facts and must protect personal and group financial information. The problem addressed by SplitSync is how to organize shared-expense workflows in one application and provide a practical foundation for trustworthy, data-driven financial insights.

3. Objectives

- Provide a unified interface for recording, editing, categorizing, and reviewing group expenses.
- Support group creation and membership workflows so participants can coordinate shared transactions.
- Calculate expense shares and present balances in a clear, understandable form.
- Synchronize authorized changes across devices and support offline work where the configured Firebase client permits it.
- Use structured expense records to support summaries, category trends, budget monitoring, and future intelligent insights.
- Apply authentication, access rules, and privacy-aware data handling to protect user and group information.
- Define a testable evaluation plan for functional correctness, synchronization, usability, and the quality of analytical outputs.

4. Scope and Contribution

The primary contribution is an application-oriented design that connects shared-expense management with a pathway toward data analytics. The paper describes the proposed system, its layered architecture, core workflows, and the safeguards needed before introducing predictive or AI-generated recommendations. It is not a clinical or financial-advice system, and it does not claim to replace professional financial planning. The current project overview lists several implemented application features; each should be verified against the actual repository and tested build before the paper is submitted as a final implementation report.

5. Related Concepts and Background

5.1 Expense Management and Shared Balances

Expense-management systems typically record who paid, who participated, and how the total should be allocated. A reliable system needs consistent transaction records and transparent rules for calculating each participant’s share. For group expenses, the application should preserve the relationship between a transaction, its group, its payer, and the participants responsible for the cost.

5.2 Data Analytics for Personal Finance

Descriptive analytics summarizes what has happened, for example total spending by category or month. Diagnostic analysis explores possible reasons for a change, such as an increase in dining expenses. Predictive analytics estimates future outcomes from historical patterns, while prescriptive approaches suggest possible

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

actions. These are distinct levels of analysis: a dashboard showing category totals is not, by itself, a predictive AI system. SplitSync can begin with transparent descriptive analytics and add more advanced techniques only when adequate data and evaluation are available.

5.3 Intelligent Features

Potential intelligent features include transaction categorization, anomaly flagging, and spending forecasts. Receipt text extraction can reduce manual entry, but OCR output may contain errors and should be reviewed by the user. An anomaly flag should be presented as a prompt to check a transaction, not as proof of fraud. Forecasts should communicate uncertainty and should not be generated when the user's history is too sparse or irregular.

6. Proposed System and Methodology

6.1 Development Approach

SplitSync follows a modular application-development approach. Flutter provides the cross-platform user interface, while Firebase services provide authentication and cloud data infrastructure. Provider and ChangeNotifier are identified in the project overview as the state-management approach. Services separate data access and notification handling from the widgets that display the application.

6.2 User Workflow

1. The user signs in and accesses their profile.
2. The user creates a group or joins an existing group through the supported joining workflow.
3. A member records an expense with its amount, category, date, payer, participants, and split method.
4. The application validates the entry and calculates each participant's share.
5. Authorized group members view updated expenses and balances as data synchronizes.
6. Users review expense history and export a summary where the export feature is available.
7. Structured records can be aggregated into analytics for category trends and budget monitoring.

6.3 Data Fields and Processing

A practical minimum expense record may include an expense identifier, group identifier, creator or payer identifier, description, amount, currency, category, timestamp, split method, participant shares, and optional receipt reference. Exact field names and storage paths must match the implementation. Before analysis, the system should validate amounts, timestamps, category values, group membership, and participant shares. Personal identifiers should be excluded from analytics outputs whenever they are not necessary.

6.4 Analytics Workflow

8. Data collection: read only the expense records the signed-in user is authorized to access.
9. Validation: check required fields, currency consistency, duplicate submissions, and split totals.
10. Transformation: group transactions by time period, category, or group and calculate relevant aggregates.
11. Insight generation: provide descriptive summaries first, such as month-over-month changes and budget-versus-actual comparisons.
12. Optional predictive stage: evaluate a suitable forecasting method against a simple baseline before displaying forecasts to users.
13. Presentation: explain the time period and data behind each insight and allow users to inspect the underlying transactions.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

7. System Architecture

The application can be described through the following logical layers:

- Presentation layer: Flutter screens and reusable widgets for authentication, groups, expenses, balances, insights, settings, and exports.
- State-management layer: Provider/ChangeNotifier objects expose application state and notify the interface when relevant data changes.
- Domain layer: model classes represent users, groups, expenses, splits, and settlements.
- Data/service layer: FirestoreService encapsulates database operations; NotificationService manages messaging-related tasks; export services generate reports.
- Firebase infrastructure layer: Firebase Authentication, Cloud Firestore, Firebase Cloud Messaging, and Firebase Storage, according to the services configured in the project.
- Analytics layer: aggregation and insight-generation logic, with predictive methods treated as an extension until implemented and validated.

Firestore listeners can deliver updates to subscribed clients, allowing the interface to react to remote changes. Offline behaviour depends on the platform, Firebase configuration, and the operations performed; it should be verified through tests rather than described as guaranteed for every workflow. Likewise, push notifications require appropriate permission handling, device-token management, and a trusted mechanism for sending messages.

8. Technology Stack

Layer	Technology	Purpose
UI / Framework	Flutter and Dart	Cross-platform interface and application logic
State management	Provider and ChangeNotifier	Reactive state distribution
Authentication	Firebase Authentication	User sign-in and identity
Database	Cloud Firestore	Cloud-backed structured records and listeners
Messaging	Firebase Cloud Messaging	Push-notification delivery
File storage	Firebase Storage	Receipt images and other supported files
Localization	flutter_localizations and intl	Localized interface text and formatting
PDF export	pdf and printing packages	Generate and share expense summaries
Analytics extension	Aggregation logic; ML methods if validated	Summaries and potential predictive insights

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

9. Core Application Modules

9.1 Authentication and Profiles

The authentication module provides a controlled entry point to the application. The project overview lists Google and email/password sign-in and an email-verification flow. Authentication success should be followed by authorization checks for every group and expense operation; signing in alone must not grant access to all records.

9.2 Groups and Expenses

Group management supports the organization of expenses around a shared context. Expense operations include creating, editing, and deleting records, along with calculating splits. Validation should ensure that the total of participant shares matches the expense amount within an explicitly defined rounding tolerance.

9.3 Synchronization and Offline Use

Real-time listeners can update group views when Firestore documents change. Offline persistence can improve resilience during temporary connectivity loss, but conflict behaviour and pending writes should be tested. The interface should communicate errors and avoid implying that an expense has reached the server before confirmation when that distinction matters.

9.4 Notifications, Localization, and Export

Notifications can alert users to relevant group events, subject to permission and delivery configuration. Localization is described for English, Hindi, and Kannada. PDF export can make summaries easier to share, but exported documents should be handled carefully because they may contain sensitive financial information.

10. Security, Privacy, and Responsible Analytics

Financial records can reveal habits, relationships, locations, and routines. The application should collect only necessary information, restrict access by authenticated identity and group membership, and avoid exposing receipt images or exported reports to unauthorized users. Firestore security rules must be tested with both authorized and unauthorized cases. Rules should check authentication before reading request.auth.uid, validate membership against trusted data, and enforce access at the document level. Rules that rely on a field from a document being created or deleted require special care because the resource may not exist in the same way for every operation.

The rule snippets in the supplied overview are illustrative and should not be copied into production without a security review. For example, an expense rule that reads resource.data.groupId may behave differently for create operations, where the new data is available through request.resource.data. Membership changes should not be authorized solely by a client-provided list that users can freely edit. Receipt storage rules and notification-token access also need explicit protection.

Analytics should be transparent and proportionate. Users should understand why an insight appears and which period of data it uses. Any prediction should be labelled as an estimate, avoid overconfident claims, and allow users to correct misclassified expenses. Group-level analytics should not disclose one member’s private transactions to other members unless that disclosure is necessary for the feature and clearly understood by participants.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

11. Testing and Evaluation Plan

The supplied project overview mentions a test directory, provider/service tests, static analysis, and possible continuous integration. The existence and coverage of these tests should be confirmed before claiming that the application has passed them. A reproducible evaluation can be organized around the following test groups:

- Functional tests: verify sign-in flows, group creation/joining, expense CRUD operations, split calculations, and PDF export.
- Calculation tests: test equal, exact-amount, percentage, and custom splits if those methods are supported by the current build; include rounding and invalid-input cases.
- Authorization tests: verify that non-members cannot read or change group expenses and that users cannot modify another user's private profile.
- Synchronization tests: make changes from two clients and check that the expected updates appear without duplicating transactions.
- Offline tests: create and edit records while disconnected, reconnect, and inspect whether data synchronizes correctly and conflicts are handled safely.
- Usability tests: ask representative users to complete common tasks and record completion rate, errors, and feedback.
- Analytics tests: compare computed totals with manually verified data; for forecasts, compare predictions against a baseline on held-out historical periods.

No quantitative performance or user-study results are included in this draft because no test measurements were provided. Once tests are run, report the test environment, sample size, metrics, and observed results. Do not insert estimated values as if they were measured findings.

12. Limitations

The usefulness of financial insights depends on complete and accurate expense records. Users may enter incomplete categories, duplicate transactions, or incorrect amounts. Predictive analytics may be unreliable when the available history is short or when spending is irregular. Cloud synchronization and notifications also depend on network conditions, platform permissions, Firebase configuration, and service availability. Finally, the current overview does not provide benchmark results, a user study, or evidence that machine-learning-based forecasting has already been implemented. These are important limitations for interpreting the proposed intelligent features.

13. Future Enhancements

- Add a transparent analytics dashboard with category totals, spending trends, and budget variance.
- Evaluate automatic category suggestions using receipt text and user-confirmed labels.
- Explore forecasting only after collecting adequate, consented historical data and comparing methods against simple baselines.
- Add explainable anomaly flags that direct users to review transactions rather than asserting fraud.
- Consider an opt-in data export for advanced analysis while minimizing personal identifiers.
- Complete security-rule tests, accessibility checks, and automated build/test workflows.
- Evaluate additional settlement/payment integrations only after reviewing provider availability, compliance, and user consent.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

14. Conclusion

SplitSync provides a practical application context for exploring how expense-management workflows can be connected with data analytics and intelligent decision support. Built around Flutter and Firebase, the system overview describes modules for authentication, groups, expenses, synchronization, notifications, localization, and report export. These structured workflows create a foundation for descriptive analytics and, with further implementation and evaluation, more advanced personalized insights. The central message is that an intelligent financial application should do more than calculate balances: it should present understandable information while preserving privacy, validating its outputs, and communicating uncertainty. Future work should verify implementation status, complete systematic testing, and evaluate the accuracy and usefulness of any predictive feature before claiming measurable benefits.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)