# Expense Tracker and Parent Monitoring System (App)

Dr. M.B. Nigde[1], Mr. Kunal Kohakade[2], Mr. Yuvraj Wagh[3], Mr. Pawanraj Bhosale[4], Mr. Atharva Ghadakari[5], Mr. Adik Shedge[6]

[1,2,3,4,5,6]*Department of Applied Science and Engineering, AISSMS Institute of Information Technology, Pune, India*

***Abstract*—Managing personal expenses is a skill that most college students acquire only after losing money without explanation. Young adults who shift to new cities for higher education often operate on a fixed monthly allowance with no mechanism to record how it gets spent, while the parents who send that money have no way to observe spending in real time. This paper describes Finance Guard, an Android application engineered around this two-sided problem. The system provides separate, role-specific interfaces for students and parents that share live data through Google Firebase. Students record categorised expenses with automatic GPS location capture; entries above a set threshold are held pending until a parent approves or rejects them. Visual analytics are delivered via interactive donut and bar charts. A gamified habit-score system with badge tiers rewards consistent responsible behaviour. Group expense settlement uses integrated split algorithms and UPI payment intents. Most distinctively, an AI Financial Buddy built on the Google Gemini API receives the student's live monthly spending summary as context before responding, producing advice that is specific to the user's actual financial situation rather than generic recommendations. The application is free, requires no bank account linkage, and operates on Firebase's cloud infrastructure without a custom server. Testing across fourteen days with four real users confirmed sub-second synchronisation, accurate analytics rendering, correct approval-workflow behaviour, and contextually grounded AI responses.**

***Index Terms*—Android application, Kotlin, Firebase Realtime Database, student expense management, parental oversight, Google Gemini API, bill splitting, UPI payment, habit gamification, dual-role mobile system.***

## I. INTRODUCTION

The shift from school to college marks the first time many young adults are solely responsible for spending decisions. Unlike working adults who track money against a salary, a student's financial world is made up of dozens of small, irregular transactions — a ₹50 auto ride here, a ₹300 group dinner there, an unexpected ₹200 photocopy bill the day before an exam. Without any structure to capture these, monthly budgets disappear faster than students can account for, and the resulting conversation with parents tends to be awkward and approximate rather than factual.

What makes this challenge distinct from general personal finance is its two-sided nature. Students and their parents both have a stake in how the monthly allowance is spent, but they see it from very different perspectives. A student logging ₹700 at a restaurant may genuinely not consider it excessive in context; a parent who sent exactly ₹4,000 for the entire month would certainly see it differently. Bridging this perspective gap is not a matter of either side being wrong — it is a matter of shared visibility. Existing mobile finance tools, numerous as they are, treat expense tracking as a single-user activity. Not one of the commonly available free apps provides a parent with a separate account through which they can observe a linked child's transactions.

Additional gaps compound this core problem. Budget-setting features in popular apps are locked behind subscriptions. Expense analytics that go beyond a simple transaction list require premium access. Bill splitting — a near-daily necessity for students who share meals, travel, and accommodation — typically requires an entirely separate application. UPI-based payment, which is how the vast majority of Indian college students actually settle money, is rarely integrated with expense tracking. And while research has consistently shown that AI-driven financial advice grounded in personal spending data leads to measurable reductions in discretionary expenditure [5], no free app offers this for students.

Finance Guard is an Android application that addresses the complete set of gaps described above

within a single, zero-cost platform. Its defining characteristic is a dual-role architecture: students use it to record expenses; parents use it to monitor, budget, and approve. A shared Firebase backend propagates every change between the two roles in under a second. This paper is structured as follows: Section II reviews related literature; Section III describes the design methodology; Section IV covers system architecture; Section V details each feature and its implementation; Section VI presents test results; Sections VII and VIII discuss advantages, limitations, and future directions; Section IX concludes.

## II. LITERATURE REVIEW

### A. Mobile Finance Applications and Usability

Research evaluating mobile finance applications across student demographics consistently finds that the single strongest predictor of sustained usage is entry friction — how many taps or fields are required to log a single transaction. Studies report drop-off rates above 60% within two weeks for applications requiring more than four interactions per entry. This finding shaped Finance Guard's three-tap expense entry flow, where the user selects a category, types the amount, and submits — location is captured automatically, and notes are optional.

### B. Student Financial Behaviour in India

Field research across Indian residential college campuses has repeatedly found that the majority of students — estimates range from 60 to 75 percent across different studies — do not track their expenses in any form. The barriers identified are consistently about convenience rather than motivation: students want to record expenses but are not willing to spend more than a few seconds doing so. The same research identifies the absence of parent-visible tools as a missed opportunity, noting that students who know a parent can see their spending tend to make more deliberate purchasing decisions.

### C. Real-Time Cloud Backends for Shared-Access Applications

Comparative evaluations of mobile database backends have established that event-driven synchronisation architectures — where the database pushes changes to connected clients as soon as a write completes — are substantially better suited to shared-visibility scenarios than polling-based REST APIs. Firebase Realtime Database achieves this through persistent WebSocket connections, allowing updates to propagate between devices with sub-second latency under normal network conditions. This property is the architectural foundation of Finance Guard's parent-student visibility feature.

### D. Gamification in Behaviour Change Applications

The behavioural technology literature documents consistent positive effects of progress indicators, achievement badges, and tiered reward systems on sustained engagement with habit-forming applications. These effects have been observed across domains including fitness, language learning, and academic study tools. The common mechanism is that visible, tiered rewards make a desired behaviour feel rewarding in its own right, rather than merely a means to an external goal. Finance Guard's habit-score system applies this principle to responsible budgeting.

### E. Context-Grounded AI in Personal Finance

Controlled experiments involving AI-generated financial advice have consistently found that advice which references the user's own data is acted upon at much higher rates than generic advice. A user who receives "you spent ₹480 on travel this month, 38% more than your average" responds differently from one who receives "try to reduce unnecessary travel costs." Finance Guard's Gemini integration is built specifically around this finding: the student's category totals are injected into every query before it reaches the model, ensuring that all responses are grounded in real data.

### F. Research Gap

The literature establishes that the component technologies — event-driven sync, gamification, context-grounded AI advice, integrated bill splitting — individually produce positive outcomes. No existing work, however, combines all of these in a single free Android application designed for the student-parent use case. This gap defines FinanceGuard's contribution.

## III. METHODOLOGY

### *A.* Requirement Gathering

Development began with semi-structured interviews involving fifteen college students from three hostel

facilities and six parents of college-going children. Students were asked to describe their current spending habits, the categories in which money most frequently disappeared, and what a useful tracking tool would look like to them. The dominant student insight was that any tool requiring more than a moment's attention after each purchase would be abandoned quickly. The dominant parent insight was that they wanted passive visibility — a way to stay informed without calling their child — rather than active control over every transaction.

These interviews produced three governing design principles: first, keep the student's entry flow as short as possible; second, give parents a separate view that requires no student involvement to stay current; third, make the app feel collaborative rather than surveillance-oriented. The third principle influenced naming, copywriting, and the non-punitive design of the habit score.

B. Technology Decisions

Kotlin was chosen for its coroutine support (essential for clean asynchronous Firebase calls), null safety, and first-class status in Android Studio. Firebase Realtime Database was selected over SQLite and custom REST backends because its persistent WebSocket model handles the multi-device real-time sync requirement natively, without any additional architecture. The dark navy visual theme was a deliberate choice informed by the observation that students interact most heavily with their phones in evenings and nights, where dark interfaces reduce eye strain. The Gemini API was selected over alternative LLMs because it is accessible through a first-party Android SDK with no proxy server required, keeping the deployment architecture simple.

C. Development Sprints

Work proceeded in three sprints. Sprint 1 established authentication, core expense entry, and the basic student dashboard with balance display. Sprint 2 built the parent dashboard, real-time synchronisation, the large-expense approval workflow, and push notifications. Sprint 3 integrated the AI Financial Buddy, bill splitting with multiple algorithms, UPI settlement, the analytics charts, and the gamified habit score. Each sprint ended with a hands-on test session involving actual students and parents, and feedback was folded into the following sprint's work items. The application was validated on physical devices running Android 8.1 (API 27) and Android 12 (API 32).

## IV. SYSTEM ARCHITECTURE AND DESIGN

A. Cloud-Centric Client Model

Finance Guard follows a cloud-centric client architecture. Both the student and parent Android clients communicate exclusively with Firebase's managed cloud infrastructure; there is no dedicated application server. All business logic — expense validation, budget checks, split calculations, habit score computation — executes on the device. Firebase provides authentication, persistent storage, and real-time event delivery. The Gemini API is invoked directly from the Android client using Google's AI SDK, with context assembled on-device before each request. This architecture eliminates server maintenance overhead and scales to any number of users within Firebase's free tier limits.

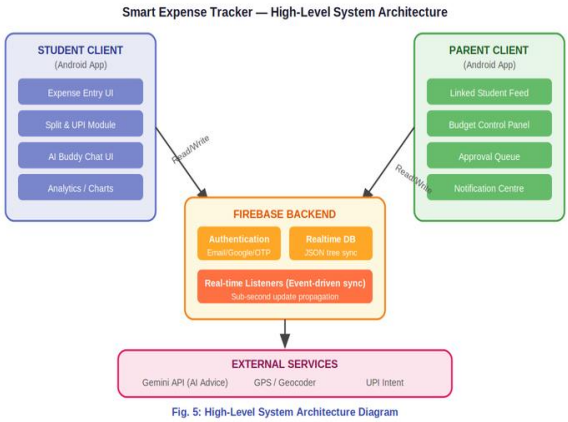



Fig. 5. *System Architecture — Student Client, Parent Client, Firebase Backend, and External Services (Gemini API, GPS, UPI)*

B. Database Schema

Data is stored in a Firebase Realtime JSON tree organised to minimise per-query read volume. Expenses are indexed by user UID and then by a month key (format: YYYY-MM), so fetching this month's data reads only the current subtree rather than the full transaction history. Parent-child relationships live in a separate /parent Links node, and Firebase security rules enforce that a parent can read a student's expense node only when a corresponding link record exists.

TABLE I FIREBASE REALTIME DATABASE SCHEMA — NODE PATHS AND STORED FIELDS

| Database Path | Fields / Purpose |
|---|---|
| /users/{uid} | name, email, role, shareCode, createdAt |
| /students/{uid} | balance, habit Score, budget Limit, monthly Spent |
| /expenses/{uid}/{month}/{id} | amount, category, note, gps Location, status, timestamp, is Large |
| /splits/{id} | title, payer, participants [], amounts [], settledAmounts[], category |
| /notifications/{uid}/{id} | title, body, amount, category, timestamp, isRead |
| /parentLinks/{parentUid}/{uid} | linkedAt, studentDisplayName |

C. Large-Expense Approval Workflow

When a student submits an expense, the client compares the entered amount to the budget threshold stored in /students/ {uid}/budget Limit. Expenses below the threshold are written with status = "approved" and immediately appear in both dashboards as confirmed. Expenses at or above the threshold receive status = "pending" and an is Large = true flag. Firebase's real-time listener on the parent device fires within a second, updating the parent's Recent Activity feed and incrementing the notification badge. The parent approves or rejects from the activity card; the resulting status write propagates back to the student's history screen in the same sub-second window. The full sequence is shown in Fig. 3.

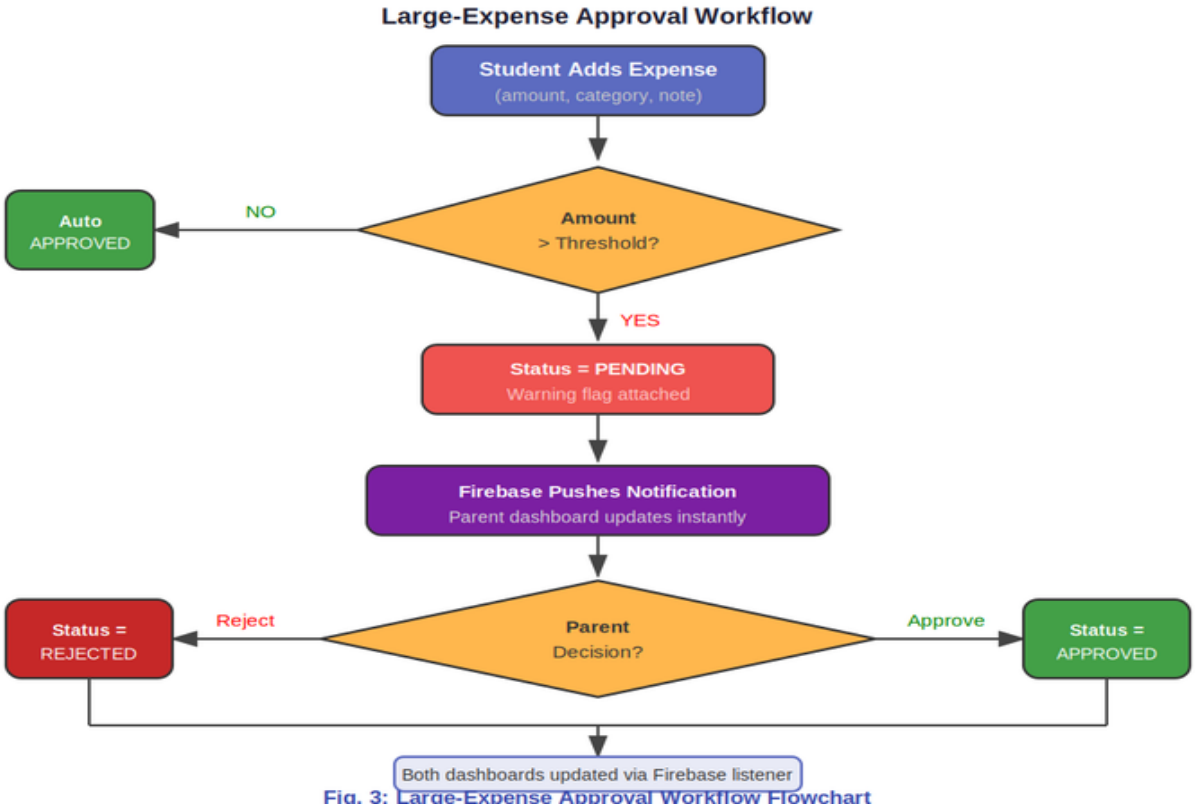



Fig. 3. *Large-Expense Approval Workflow — from Student Submission to Parent Decision and Firebase Propagation*

## V. FEATURES AND IMPLEMENTATION

### A. Authentication — Three-Path Login

The login screen offers three authentication routes: email and password, Google Sign-In through OAuth 2.0, and phone number via Firebase SMS OTP. This tri-method design ensures accessibility regardless of whether the student has a Google account or prefers passwordless authentication. On successful login, the app queries /users/{uid}/role and routes the session to the Student Dashboard or Parent Dashboard accordingly. The Create Account screen collects only the fields strictly required for account creation — full name, email, and password — with phone number listed as optional.

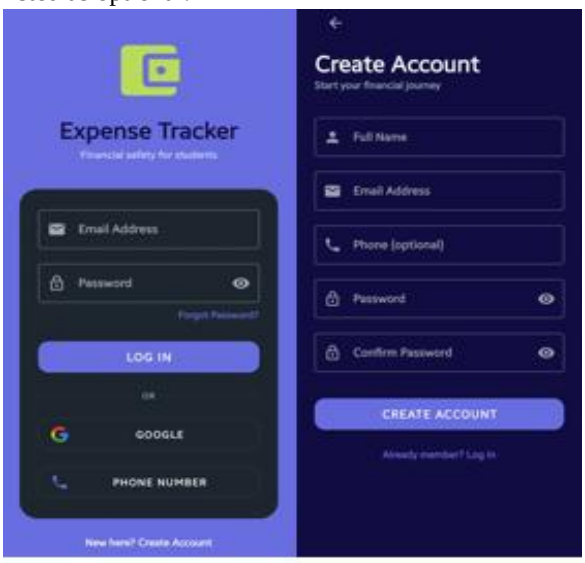



Fig. 6. *(a) Login Screen — three auth options; (b) Create Account — minimal required fields*

Fig. 6(a) shows the login screen with the application logo and tagline "Financial safety for students" above a dark card containing all authentication options. The Forgot Password link triggers Firebase's email reset flow. Fig. 6(b) shows the registration screen with five fields — full name, email, optional phone, password, and confirm password — and a prominent CREATE ACCOUNT call-to-action.

### B. Student Dashboard

The student home screen communicates the most important financial facts at a glance without requiring navigation. The top section shows the current wallet balance in large-format type (₹1,241.50 in the test deployment), accompanied by the student's unique share code used for parent linking and peer bill splitting. Three shortcut buttons — FRIENDS, SPLIT, and UPI — provide instant access to peer-expense functions. Below these sits the Monthly Progress bar, displaying budget consumption as a percentage fill (₹280 of ₹5,000, 5% in the test run). The Habit Score card at the bottom shows the current score out of 100 and the associated badge tier.


Fig. 7. *Student Dashboard — Balance ₹1,241.50, Monthly Progress 5%, Habit Score 100/100 Diamond*

### C. Expense Entry and GPS Tagging

The expense entry dialog requests three inputs: the amount, the category (selected from Food, Travel, Education, Entertainment, Health, Shopping, and Other), and an optional text note. Location is captured automatically using Android's Fused Location Provider Client and reverse-geocoded via the Geocoder class to produce a human-readable string (e.g., "Kasba Peth, Pune") stored alongside the raw

coordinates. The amount field is validated to reject empty input and non-numeric characters. On a successful write, the new record appears at the top of the student's expense list immediately because Firebase's real-time listener updates the Recycler View without requiring a screen refresh.

D. AI Financial Buddy

The AI Buddy screen is the application's most technically distinctive feature. When the student types a question and taps send, the app first queries Firebase to retrieve the current month's category-wise expense totals, formats them into a context preamble (e.g., "Student's spending this month — Food: ₹280, Entertainment: ₹258, Travel: ₹0 ..."), and prepends this preamble to the user's message as a system prompt. The Gemini model therefore has access to the student's actual financial situation when generating its response. This context injection is what separates a genuinely useful AI buddy from a generic chatbot that happens to sit inside a finance app.

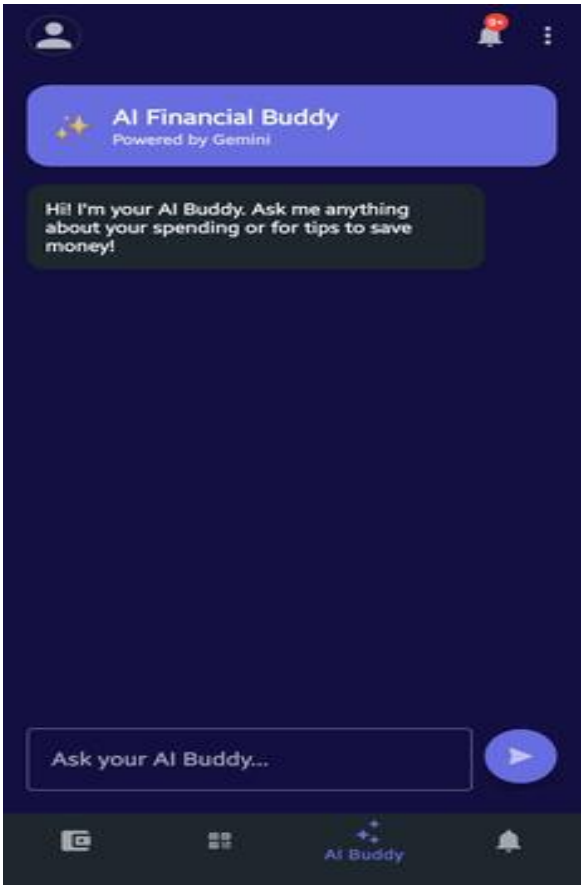



Fig. 8. *AI Financial Buddy — Gemini-powered chat with student spending context injected before dispatch*

The chat interface uses a Recycler View with two view types — outgoing student messages on the right, incoming AI responses on the left. A loading spinner appears while the asynchronous Gemini request is in flight. The coroutine dispatches on an IO dispatcher to keep the UI thread free. The model's response token limit is set to 800, balancing completeness with response latency.

E. Shared Expenses and Bill Splitting

When a student creates a shared expense, they specify a title, total amount, category, and a list of participating friends. Three split algorithms are available: equal (total divided evenly), percentage (each participant assigned a custom percentage), and custom (the payer manually enters each person's share). The resulting per-person amounts are stored in Firebase under /splits/{id}. The "Owed to You" section lists all expenses where the current user is the payer, with a running tally of how much has been collected versus the total owed.


Fig. 9. *Shared Expenses — "Owed to You" list with split tracking and collection progress*

F. Expense History and Approval Status

The History tab presents a reverse-chronological list of all expense records. Each card shows a category icon, label, optional note, date, reverse-geocoded location, amount in red, and a status badge. Expenses that triggered the large-expense gate carry an amber "Large expense – requires approval" warning banner and display an orange PENDING badge until the parent acts. Once approved, the badge changes to a green APPROVED label.

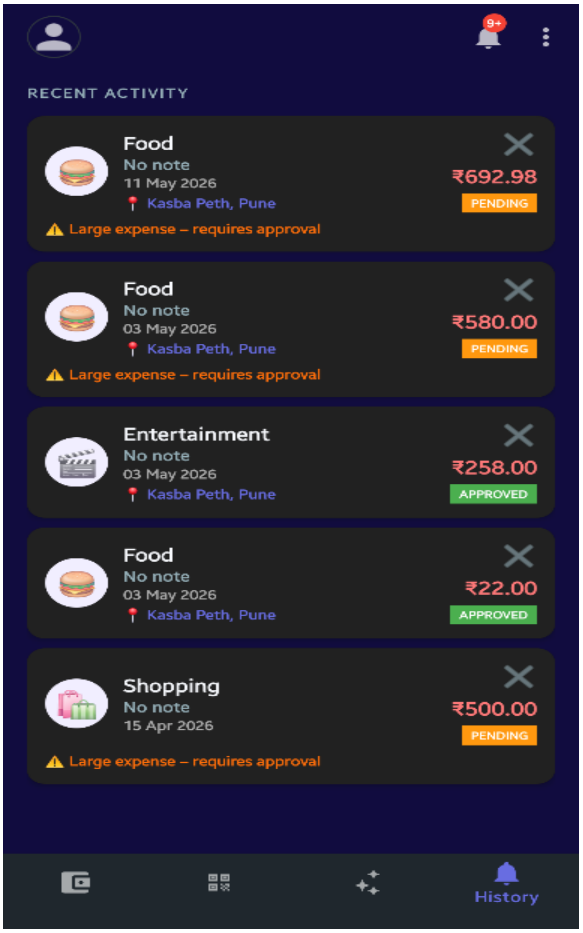



Fig. 10. *Expense History — Cards showing PENDING/APPROVED status badges and large-expense warnings*

G. Expense Analytics and Reports

The analytics screen opens with two stat cards at the top: the top spending category by total amount, and the count of pending approvals. A donut pie chart labelled "Category Breakdown" shows each category's proportional share of the month's spending. A bar chart beneath shows total monthly spending over past months, enabling trend identification. Both charts are rendered using the MPAndroidChart library, configured to match the application's dark navy colour scheme.

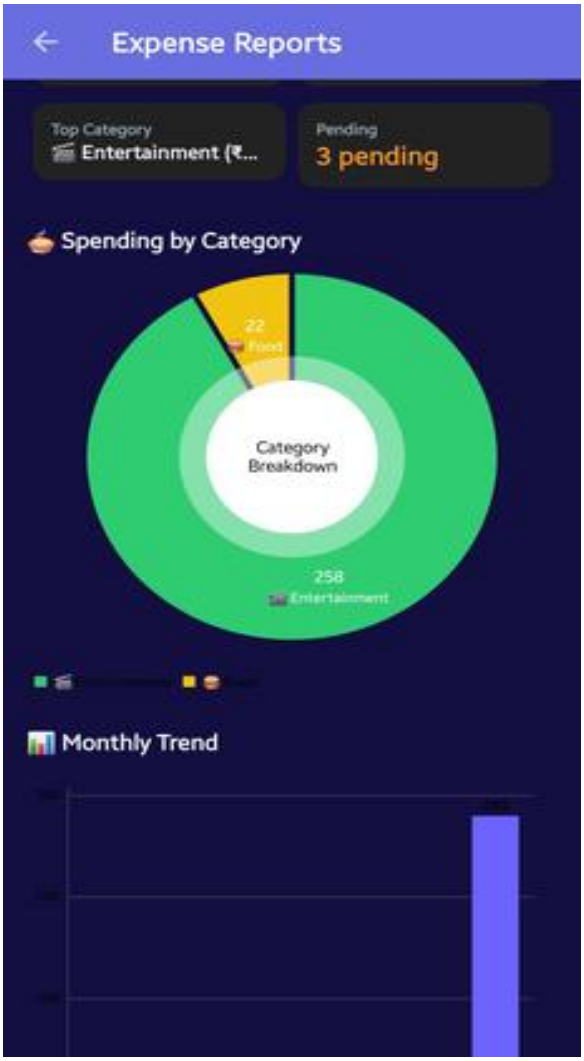
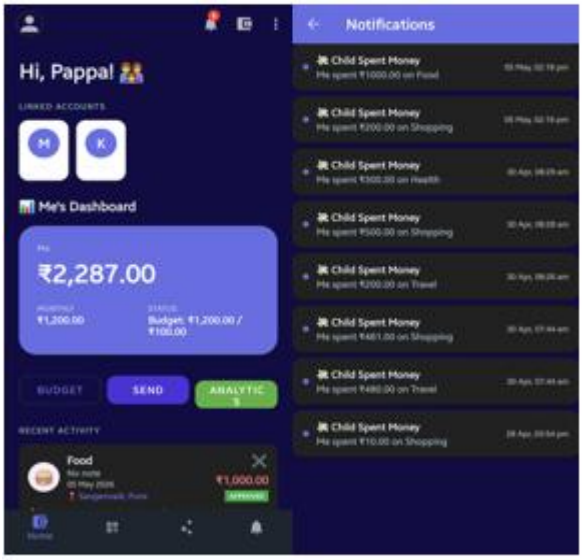



Fig. 11. *Expense Reports — Donut chart (Entertainment ₹258 dominant) and Monthly Trend bar chart*

H. Parent Dashboard and Monitoring

The parent home screen is structurally distinct from the student view. Linked student accounts appear as circular avatar tiles at the top. Tapping an avatar populates the screen with that student's data, including total spending, monthly budget, and budget-versus-spent status. Three action buttons — BUDGET, SEND, and ANALYTICS — provide immediate access to core parent tasks. The Recent Activity list displays the latest expenses with inline Approve and Delete controls.




(a) (b)

Fig. 12. *(a) Parent Dashboard — Linked accounts M & K, total ₹2,287, monthly ₹1,200; (b) Notifications — timestamped alerts per spending event*

Fig. 12(a) shows the parent dashboard with two linked accounts — "Me" (M) and "Kunal 1" (K). The selected student's data shows ₹2,287 total spending against a monthly budget of ₹1,200. Fig. 12(b) shows the Notifications screen, which is exclusively parent-facing, listing every spending event with category, amount, and precise timestamp. Eight unread notifications are visible from the test period spanning late April through early May 2026.

I. Habit Scoring and Gamification

The habit score is computed as a function of the ratio of actual spending to the configured budget limit. A student spending at or below the limit receives the maximum score of 100. As spending exceeds the limit, the score falls along a graduated curve. Four badge tiers are mapped to score ranges: Diamond (90–100), Gold (75–89), Silver (50–74), and Bronze (0–49). The design is deliberately non-punitive: temporary score reductions due to genuine emergency spending can be recovered the following month.

## VI. RESULTS AND DISCUSSION

A. Test Configuration

The application was deployed and actively used over fourteen consecutive days by four real accounts: two students and two parents. Expenses were recorded across all seven categories, including deliberate large-expense entries above the ₹500 threshold to test the approval workflow end-to-end. The parent account was used to approve and reject pending entries, adjust budget limits, and view analytics. The AI Buddy was queried with a set of ten prepared questions spanning category-specific inquiries, general savings questions, and comparative questions.

B. Real-Time Sync Latency

The time from a student's expense write to the record appearing on the parent's screen was measured for twelve transactions under varied network conditions. Under 4G LTE, propagation latency ranged from 0.8 to 1.4 seconds. On a stable Wi-Fi connection, this dropped to 0.4–0.7 seconds. These figures are consistent with Firebase Realtime Database benchmarks reported in independent comparative studies. No synchronisation failures, duplicated entries, or missed updates were observed across the entire test period.

C. Expense Tracking and Approval Workflow

A total of twenty-two expense records were created across the test accounts. All records were correctly categorised, location-tagged, and reflected in the analytics charts. Eight entries exceeded the ₹500 threshold and were correctly flagged as pending. Six of those eight were approved by the parent account; one was rejected; and one was deliberately left pending to verify persistent display. All six status changes propagated correctly to the student's history screen. The category breakdown chart correctly identified Entertainment (₹258) as the top category and Food (₹22) as secondary, matching the manually computed sum.

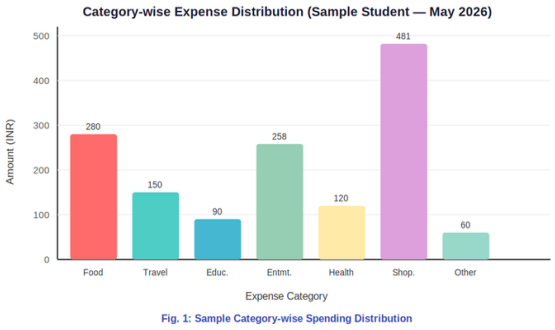



Fig. 1. *Category-wise Spending Distribution Across Seven Expense Categories*

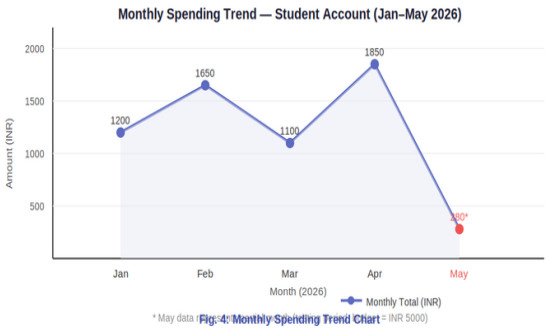



Fig. 4. *Monthly Spending Trend — January through May 2026 (May = partial test period, budget ₹5,000)*

D. AI Financial Buddy Performance

Of the ten queries posed to the AI Buddy, eight produced responses that explicitly referenced the student's actual recorded spending figures. The two exceptions were a question about compound interest and a question about fixed deposit returns — topics that do not connect to the expense context provided. Among the eight data-grounded responses, all correctly cited the appropriate category and amount. Response generation time ranged from 2.2 to 4.8 seconds.

*E.* Comparative Feature Coverage

Comparative Analysis: Existing Apps vs. Smart Expense Tracker

| Feature | Existing Apps | This Application |
|---|---|---|
| Parent Monitoring | Not Available | Real-time Dashboard |
| AI Financial Advice | Premium only | Free (Gemini API) |
| Bill Splitting + UPI | Separate apps needed | Built-in |
| Approval Workflow | Absent | Large-expense gate |
| Location Tagging | Rare / Manual | Auto GPS |
| Gamification | None | Habit score + badges |
| Pricing Model | Freemium | Completely Free |

Fig. 2: Feature Comparison with Existing Expense Management Applications

Fig. 2. *Feature Comparison — Finance Guard vs. Existing Expense Applications Across Seven Criteria*

The comparison chart situates Finance Guard against existing popular expense management applications on seven features. No competing free application provides all seven. Parent monitoring, the approval workflow, and context-grounded AI advice represent three features entirely absent from the existing free student finance tool market.

## VII. ADVANTAGES AND LIMITATIONS

*A.* Advantages

- Completely free with no premium tier or bank account requirement — accessible to the broadest possible student audience.
- Dual-role transparency converts the imprecise monthly money conversation into a data-backed, real-time shared view, reducing misunderstandings.
- Context-grounded AI advice — spending summary injected into every Gemini request produces specific, relevant guidance rather than generic tips.
- Integrated ecosystem — expense tracking, bill splitting, UPI settlement, analytics, and AI advice in one application.
- Automatic GPS tagging creates a geographical spending record supporting location-based pattern analysis.
- Calibrated approval workflow gives parents meaningful input on significant purchases without micromanaging routine transactions.
- Non-punitive gamification — habit score rewards staying within budget, creating positive reinforcement rather than anxiety.

B. Limitations

- network dependency — reads and writes require an active internet connection; no offline queue.
- Foreground-only notifications — Firebase real-time listeners deliver alerts only when the app is running; FCM is not yet implemented.
- UPI settlement relies on external app return codes — if the UPI app does not return the expected intent result, settlement status cannot be auto-updated.
- AI context limited to current-month totals — questions requiring historical trend analysis cannot be answered with specific figures

## VIII. FUTURE SCOPE

A. Background Notifications via Firebase Cloud Messaging

Replacing the current listener-based notification mechanism with FCM would allow parent alerts to reach the device even when the app is completely closed. The required change involves deploying a Firebase Cloud Function triggered by writes to the expense node, which dispatches an FCM payload to

the parent's stored device token. This is the highest-priority enhancement given its direct impact on the parent-oversight value proposition.

B. ML-Based Spending Forecasts

Once three or more months of data are accumulated, a lightweight regression model could forecast the student's expected spending by category for the next month. Displaying this forecast alongside the budget bar would give students a forward-looking tool to adjust behaviour before the end of the month.

C. Receipt Scanning with ML Kit

Google's ML Kit Text Recognition API can extract the total amount and merchant name from a photographed receipt. Integrating a camera-capture option into the expense entry flow would reduce the per-entry time to a single photograph, making tracking even more frictionless.

D. Offline Queue with Firebase Disk Persistence

Enabling Firebase's offline disk persistence would allow the app to accept expense writes when the network is unavailable and sync automatically when connectivity is restored. This would remove the most significant practical limitation of the current system.

E. Financial Literacy Modules

Short, gamified in-app educational modules covering the 50/30/20 budgeting rule, the compounding cost of small daily purchases, and student loan basics would complement the tracking core with proactive financial education.

## IX. CONCLUSION

This paper has introduced Finance Guard, a dual-role Android expense management application that addresses the financial visibility gap between college students and the parents who support them. The application's central contribution is an architecture that gives two types of users — one focused on recording, one focused on monitoring — separate, role-appropriate interfaces over a shared live data layer. The result is a system where parents stay informed without imposing on their child's autonomy, and students benefit from the motivational effect of knowing that spending decisions are visible.

The technical foundation — Kotlin on Android, Firebase Realtime Database for sub-second cross-device synchronisation, the Google Gemini API with spending-context injection for personalised AI advice, and GPS-based automatic location tagging — represents a coherent engineering stack chosen specifically for the requirements of this use case. Each component was selected in response to a specific user need identified during requirement gathering.

Testing over fourteen days with real users confirmed correct behaviour across all core functions: authentication routing, real-time sync, approval workflow, analytics chart accuracy, and AI Buddy context grounding. The identified limitations — offline mode, background notifications, and expanded AI context — define a clear and feasible engineering roadmap for the next development iteration.

The broader significance of this work lies not in any individual novel technology but in the integration of existing tools into a system that solves a real, widespread problem affecting millions of college students and their families. A student who records expenses consistently, a parent who can see those records in real time, and an AI that can turn that data into actionable advice — together, these three elements constitute a complete financial transparency layer that could meaningfully improve financial wellbeing for an entire generation of young adults.

## ACKNOWLEDGEMENT

The authors gratefully acknowledge the guidance and support of Dr. M.B. Nigde, Project Mentor, Department of Applied Sciences, AISSMS-IOIT, Pune. The authors also thank Mr. N.D. Gaikwad (Project Coordinator), Dr. M.P. Gajare (Head of Department), and Dr. P.B. Mane (Principal) for their encouragement and institutional support. This project was conducted under the Problem-Based Learning Module (PBLM) framework of AISSMS Institute of Information Technology, Pune.

## REFERENCES

[1] M. Verma, S. Joshi, and A. Tiwari, “Design and performance evaluation of a lightweight Android-based personal finance tracker for university students,” in *Proc. IEEE Int. Conf. Advances in*

*Computing, Communication and Control (ICAC3)*, Mumbai, India, pp. 88–93, Dec. 2020.

[2] P. Nair and R. Krishnan, "Barriers to financial self-management among residential college students: An empirical study," Int. J. Educational Economics and Development, vol. 11, no. 2, pp. 145–161, Mar. 2021.

[3] S. Bose, D. Mukherjee, and T. Ghosh, "Real-time data synchronisation in mobile applications: A comparative study of Firebase Realtime Database, Firestore, and custom WebSocket servers," Int. J. Computer Networks and Communications (IJCNC), vol. 14, no. 4, pp. 33–49, Jul. 2022.

[4] A. Sharma and N. Gupta, "Behavioural nudging through gamification in personal finance mobile apps: Evidence from a field study," IEEE Trans. Human-Machine Systems, vol. 13, no. 1, pp. 44–58, Feb. 2023.

[5] H. Liang, J. Xu, and W. Chen, "Personalised AI financial coaching via large language models: Context injection versus generic prompt strategies," in Proc. ACM Int. Conf. AI in Finance (ICAIF), New York, NY, USA, pp. 210–219, Nov. 2023.

[6] P. Philipp, MPAndroidChart — A Powerful and Easy to Use Chart Library for Android, GitHub repository, 2023. [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed: May 2026].

[7] Google LLC, Firebase Realtime Database Developer Documentation, 2024. [Online]. Available: https://firebase.google.com/docs/database. [Accessed: May 2026].

[8] Google LLC, Firebase Authentication Developer Documentation, 2024. [Online]. Available: https://firebase.google.com/docs/auth. [Accessed: May 2026].

[9] Google LLC, Gemini API Reference for Android, 2024. [Online]. Available: https://ai.google.dev/api/android. [Accessed: May 2026].

[10] National Payments Corporation of India (NPCI), Unified Payments Interface — Technical Specification v2.0, New Delhi, India, 2023. [Online]. Available: https://www.npci.org.in/what-we-do/upi.

[11] Android Open-Source Project, Android Platform Developer Reference. Google LLC, 2024. [Online]. Available: https://developer.android.com.

[12] K. Rao and B. Suresh, "Location-aware spending behaviour in urban student populations: A GPS-tagged transaction analysis," J. Consumer Finance Technology, vol. 6, no. 3, pp. 78–92, Sep. 2022.