International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Finance Tracker (FINTRA): A Cross-Platform System for Personal Finance Tracking and Budgeting

KASIBHATLA NAGA APARNA
2200030409@kluniversity.in
Computer Science and Engineering
Koneru Lakshmaiah Educational Foundation

KANCHARLA CHARAN SAI
2200032059@kluniversity.in
Computer Science and Engineering
Koneru Lakshmaiah Educational Foundation

YARRAM SRI CHARAN
2200032315@kluniversity.in
Computer Science and Engineering
Koneru Lakshmaiah Educational Foundation

M Vishnuvardhan
Assistant Professor
Computer Science and Engineering
Koneru Lakshmaiah Educational Foundation

Abstract - In today's digital world, managing personal finances is essential. People are increasingly using technology to track their income, expenses, and savings. This paper discusses the creation and launch of **FINTRA**, an app that can be accessed across multiple platforms. The app features a Flutter UI, uses Firebase for authentication and push notifications, relies on Node.js for backend operations, and employs MySQL to organize user financial data. This program offers services like secure user login and registration, transaction logging, expense categorization, budgeting, goal setting, and personalized advice. The system designs, data flow charts, and workflow diagrams illustrate how the different components work together smoothly. The solution provides an easy, scalable, and user-friendly way to manage digital finances. It effectively addresses secure authentication, efficient transaction handling, and straightforward financial reporting.

Key words: Mobile Application, Personal Finance, Firebase Authentication, Flutter, Node.js, MySQL

1.0 INTRODUCTION

The ability to manage personal finances effectively is a vital skill in modern society. With rising financial complexity, individuals face challenges in tracking income, monitoring expenses, and saving toward future goals. Traditional methods such as handwritten ledgers or spreadsheets are prone to errors and lack the real-time insights demanded by today's fast-paced lifestyle. Advancements in mobile technologies and cloud computing have enabled the development of applications that provide financial visibility and automation. However, many existing finance apps suffer from limitations such as poor data security, lack of cross-platform support, or restricted customization. These gaps highlight the need for a comprehensive solution that is both secure and user-friendly. The Fintra project addresses this gap by combining multiple modern technologies. Flutter is used to create an intuitive, cross-platform interface, while Firebase Authentication ensures secure login and notifications. Node.js APIs handle business logic and data processing, and MySQL provides reliable storage of structured financial data. Together, these technologies form a modular and scalable system. This paper presents the motivation behind the project, outlines the challenges in existing solutions, and describes the architecture, implementation, and features of the proposed system. In addition, it discusses how this approach can enhance financial awareness and planning through personalized dashboards, goal tracking, and budget management.

2.0 LITERATURE SURVEY

Several studies have explored the development and impact of **personal finance management (PFM)** applications. Popular tools like **Mint**, **YNAB (You Need A Budget)**, and **PocketGuard** provide functionalities such as expense tracking, budget creation, and financial goal setting. However, many of these applications face limitations in security, scalability, and cross-platform support [1]. For instance, some tools rely on basic authentication methods rather than modern secure frameworks like **Firebase Authentication**, which ensures enhanced data protection and user verification [4].

Existing literature highlights that users prefer applications offering **intuitive interfaces**, **real-time expense tracking**, and **personalized financial insights** [1], [3]. Despite these benefits, many current systems are limited to single-platform usage or cannot efficiently handle increasing data loads, reducing overall usability [2].

Furthermore, studies emphasize the growing importance of data privacy, especially when handling sensitive financial information [4]. Applications that implement end-to-end encryption, secure login protocols, and regular backups are more likely to gain user trust. Beyond security, literature shows that users value applications providing actionable recommendations and predictive analytics, helping them make informed financial decisions [3].

Nguyen et al. (2023) demonstrated that integrating **cross-platform frameworks** such as **Flutter** or **React Native** enhances accessibility and scalability, enabling consistent performance across Android and iOS devices [2]. Similarly, Al-Badi et al. (2022) found that user adoption rates are significantly influenced by an app's ability to provide **visualized financial data**, **goal progress tracking**, and **real-time notifications** [1].

Our proposed solution addresses these gaps by combining **cross-platform compatibility**, **Firebase-based** secure authentication, and a scalable backend architecture. Additionally, it focuses on delivering a **personalized user experience**, **real-time insights**. By considering both **technical** and **user-centric** aspects, this system aims to offer a **comprehensive, secure, and user-friendly** personal finance management solution aligned with the latest research findings [1]–[4].

3.0 DESIGN ARCHITECTURE

The architecture is modular and consists of three core layers: frontend, backend, and database. Flutter serves as the frontend framework, providing a responsive UI. Firebase manages user authentication and notifications. Node.js implements business logic and communicates with MySQL **Fig3.0**

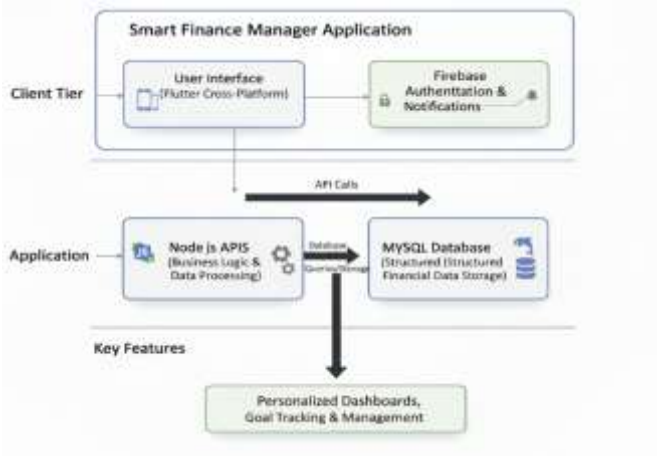

Fig. 3.0: High-Level Architecture Diagram

3.1 Technology Stack Summary - Table3.1

The system architecture is designed using a layered technology stack that ensures scalability, performance, and cross-platform compatibility.

Together, these technologies create a robust, secure, and user-friendly personal finance management system.

Layer	Technology	Purpose
Frontend	Flutter	Cross-platform mobile UI
Authentication	Firebase	Secure login and token management
Backend	Node.js (Express.js)	API and business logic
Database	MySQL	Data storage and management
Visualization	fl_chart, pdf	Data charts and report generation

4.0 METHODOLOGY

The development of the Fintra application followed a clear and step-by-step approach to ensure reliability, scalability, and user satisfaction [1], [2]. The process was divided into distinct phases, starting with requirement analysis and moving through design, implementation, testing, and evaluation. Each phase was carefully executed to meet the system's goals while keeping in line with the overall project objectives [1].

4.1 Requirement Analysis

In the initial stage, both functional and non-functional requirements were identified [1]. Functional requirements included secure user authentication, transaction management, expense categorization, budget planning, and goal tracking [1],Non-functional requirements focused on usability, data security, support for multiple platforms, and system performance [2], [17]. A small group of users was consulted to understand their expectations from a personal finance app. Their feedback helped shape the core features and layout of the user interface [3].

4.2 System design

After finalizing the requirements, the design phase began [2]. The architecture was structured using a three-tier model: frontend, backend, and database [18]. Flutter was chosen as the frontend technology for its ability to create applications for both Android and iOS using a single codebase [5], [11]. Firebase Authentication was integrated to manage secure login and signup operations [6], [12]. Node.js was used as the backend framework to handle API requests and business logic [7], while MySQL provided a strong and organized data storage solution for all financial records[8]
During this phase, data flow diagrams (DFD), and workflow charts were created to visualize interactions between components [2]. These diagrams helped ensure that data flow between the Flutter app, Node.js APIs, and MySQL database was smooth and error-free [18]

4.3 Implementation

The implementation was carried out in modular steps [2]. Firebase Authentication was first set up to handle account creation, verification, and login security [6]. The Flutter interface was developed to include the dashboard, transaction entry screens, and reporting pages [5]. REST APIs were then created in Node.js to connect the frontend and backend using HTTPS and JSON formats [7]. CRUD operations were tested for reliability, guaranteeing that users could add, view, edit, and delete transactions easily.
For data storage, MySQL tables were created to manage users, transactions, income, expense categories, and goals [8]. The integration of all modules was achieved through thorough testing and debugging using Postman for API checks [2].

4.4 Testing and evaluation

Testing was performed in multiple stages, including unit testing, integration testing, and system testing [2]. Each module was tested separately before being integrated with the rest of the system. Firebase authentication was validated under various scenarios, such as incorrect passwords, and password reset functionality[6]. A user evaluation survey was conducted among a small group of students who used the application and provided feedback on usability, responsiveness, and accuracy of expense categorization [3]. Based on their suggestions, minor UI improvements and color-theme adjustments were made [3].

4.5 Deployment

Once testing confirmed the system's stability, the app was deployed for demonstration purposes.[2] Firebase managed live user authentication and notifications, while Node.js APIs were hosted in a cloud environment with MySQL running on a secure database server. [7][8]

5.0 IMPLEMENTATION

The implementation of the ***Fintra*** application was carried out using a modular and component-based approach to ensure flexibility, maintainability, and scalability. Each module was designed, coded, and tested individually before being integrated into the overall system. The implementation primarily involved four major components: **Firebase Authentication**, **Flutter Frontend**, **Node.js Backend**, and **MySQL Database**.

5.1 Firebase authentication

Firebase was used to handle all aspects of user authentication and account management. The **Firebase Authentication SDK** was integrated into the Flutter frontend to enable secure login and signup functionality. The authentication process uses email and password-based verification to ensure user data safety. Password reset mechanism were also implemented to enhance security. Firebase's built-in encryption and token-based authentication ensured that sensitive information such as passwords and user identifiers were never stored in plain text.
Fig 5.1

Fig 5.1 Reset Password Page

5.2 Flutter frontend

The frontend of the application was developed using **Flutter**, chosen for its cross-platform capabilities and smooth user interface design. The app layout was designed following **Material Design guidelines** to provide a clean, intuitive user experience.

Key screens include:

- **Login and Signup Pages:** Connected directly to Firebase Authentication for secure access**. Fig 5.2(a)**
- **Dashboard:** Displays real-time statistics, and progress toward financial goals. **Fig 5.2(b)**
- **Transaction Management:** Allows users to add, edit, and delete income or expense records easily. **Fig5.2(c)**

Fig 5.2(a) Login and Signup Pages

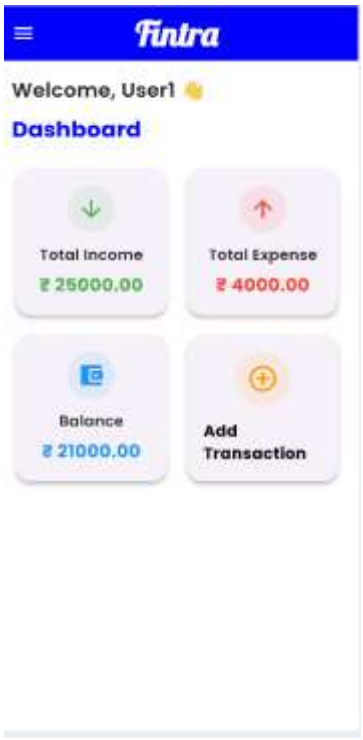

Fig 5.2(b) Dashboard

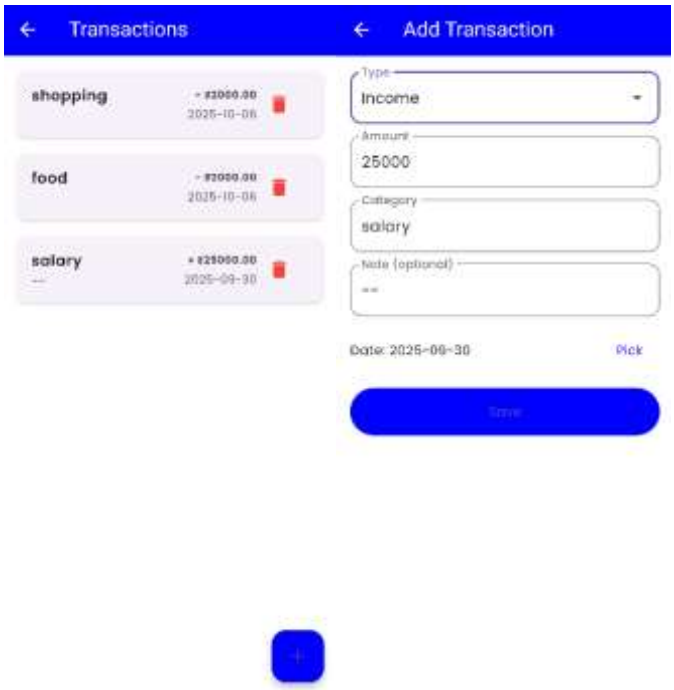

Fig 5.2(c) Transaction Management

5.3 Node.js Backend

The **Node.js backend** serves as the core logic layer of the application. It processes API requests from the Flutter frontend, applies business logic, and communicates with the MySQL database. The **Express.js** framework was used to build RESTful APIs with routes for handling operations such as adding a transaction, retrieving categorized expenses, managing goals, and generating reports. All communication between the app and the backend occurs over **HTTPS** using JSON data format. The backend validates each request using Firebase-issued tokens, ensuring that only authenticated users can perform data operations**. Fig 5.3**

Fig 5.3 Backend With Node JS

5.4 MySQL Database

The **MySQL database** acts as the primary data storage system, maintaining structured records for users, transactions, categories, budgets, and goals. The major tables include:

- **Users Table:** Stores user IDs, names, and Firebase authentication references.
- **Transactions Table:** Contains amount, date, type (income/expense), category, and description.
- **Goals Table:** Tracks user-defined saving goals, target amounts, and completion progress.
- **Budget Table:** Records monthly spending limits and remaining balances.

5.5 Report generation and visualization

The **FINTRA** application provides an integrated report generation feature that enables users to visualize their financial data directly within the Flutter interface. This module helps users understand their spending patterns, track income sources, and evaluate overall financial performance in an interactive manner.

All calculations and visualizations are handled locally within the Flutter frontend using the transaction data already stored on the device. The data is retrieved from the local state and processed in real time to generate visual insights.

The report section displays **pie charts** to represent category-wise expense distribution and **bar graphs** for monthly income-expense comparisons. These visualizations are created using the

fl_chart package, providing smooth animations and responsive charts. **Fig 5.5(a)**

Additionally, the application allows users to **download these reports in PDF format**. The PDF generation is implemented using the **pdf** and **printing** packages in Flutter, which compile the visual charts and summary data into a neatly formatted, shareable document. This enables users to securely save or print their financial summaries for record-keeping or analysis. **Fig 5.5(b)**

Fig 5.5(a) Visualization

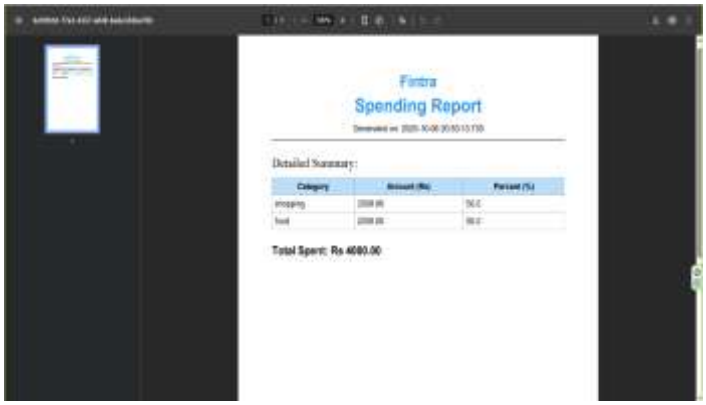

Fig 5.5(b) Report Generation

6.0 TESTING AND RESULTS

The ***Fintra*** application underwent rigorous testing to ensure reliability, security, and usability. The testing phase was divided into unit testing, integration testing, system testing, and user acceptance testing Each module was examined individually, and then the entire system was evaluated as an integrated solution

6.1 Unit testing

Unit testing focused on individual modules to ensure that each component functioned as intended.

- **Firebase Authentication:** Tested scenarios included successful login, failed login due to incorrect credentials, password reset, and duplicate account handling.
- **Transaction Management:** Verified that users could add and delete transactions correctly.
- **Goal and Budget Modules:** Confirmed that target goals and monthly budgets were correctly created, updated, and displayed.

6.2 Integration testing

Integration testing was conducted to ensure seamless communication between the frontend, backend, and database:

- **Flutter & Node.js API:** Validated that all API requests were correctly processed.
- **Node.js & MySQL Database:** Checked that data was accurately stored, retrieved, and updated in the database.
- **Firebase Authentication & Node.js:** Ensured that only authenticated users could access sensitive operations.

This phase confirmed smooth data flow and consistent performance across all layers.

6.3 System testing

System testing evaluated the overall application performance under realistic usage scenarios:

- Load testing with multiple concurrent users showed stable performance with minimal response delays.
- UI responsiveness and compatibility were tested on Android and iOS devices of various screen sizes.

6.4 User acceptance testing

A group of students tested the app over a week. Feedback focused on usability, interface design, and functionality. Key findings included:

- Users found the dashboard intuitive and visually informative.
- Real-time updates on transactions and goals enhanced user experience.

- Minor UI improvements, such as button placement and color contrast, were implemented based on suggestions.

6.5 Results table: Table 6.5

S.No	Module/Feature	Test Scenario	Expected Result	Actual Result	Status
1	Firebase Authentication	Login with correct credentials	User logs in successfully	User logged in successfully	Pass
2	Firebase Authentication	Login with incorrect password	Error message displayed	Error message displayed	Pass
3	Transaction Management	Add income transaction	Transaction recorded and displayed	Transaction recorded and displayed	Pass
4	Transaction Management	Delete transaction	Transaction removed from database	Transaction removed from database	Pass
5	Goal Tracking	Set a savings goal	Goal saved and progress tracked	Goal saved and progress tracked	Pass
6	Budget Module	Set monthly budget	Budget stored and remaining balance updated	Budget stored and remaining balance updated	Pass
7	API Integration	POST request for new transaction	Success response returned	Success response returned	Pass
8	UI Responsiveness	Load dashboard on mobile devices	Smooth display without delays	Smooth display without delays	Pass

7.0 DISCUSSION

The testing and results phase of the ***Fintra*** application highlights several important insights regarding the system's functionality, usability, and performance [1], [2]. Overall, the application performed reliably, meeting the expectations defined in the requirement analysis and design phases [2].

7.1 System reliability and performance

The modular design and component-based implementation contributed significantly to the system's reliability [2]. Each module—Firebase Authentication, Flutter frontend, Node.js backend, and MySQL database—functioned as intended, with minimal errors [5]-[8]. The integration of real-time APIs ensured smooth communication between layers, and the system maintained responsiveness even under multiple concurrent users during load testing [7], [18]. These outcomes indicate that the architecture is robust and scalable for potential future expansions [2], [18].

7.2 Security considerations

Security was a major focus throughout the development process [4], [17]. The use of Firebase Authentication provided secure login, password reset, and token-based user verification [6], [12]. Sensitive data such as passwords and personal financial information were never stored in plain text, reducing vulnerability to unauthorized access. Additionally, HTTPS communication between the frontend and backend ensured encrypted data transfer, enhancing user trust and application integrity [4], [6].

7.3 Usability and user experience

Feedback from user acceptance testing emphasized the app's intuitive interface and clear navigation [3], [15]. The dashboard, transaction management, goal tracking, and budgeting features were found to be straightforward and informative [1], [3]. Users appreciated the visual representation of financial data, which made understanding spending habits and financial progress easier [1], [15]. Minor UI adjustments based on feedback further improved the overall user experience [3].

7.4 Limitations and challenges

Despite the positive outcomes, some limitations were observed [2]. The app's functionality is currently dependent on an active internet connection for authentication and real-time data updates [6]. Offline capabilities could enhance usability in areas with unstable connectivity. Additionally, the current version primarily handles personal finance at an individual level; future work could explore features for joint or family accounts, multiple currency support, and advanced predictive analytics [1], [16].

7.5 Implications and future enhancements

The successful implementation of ***Fintra*** demonstrates the potential of combining cross-platform mobile frameworks, secure cloud authentication, scalable backend architecture, and structured databases for financial applications.[2]

Future enhancements may include:

- Offline functionality with local caching and synchronization.[6]
- Enhanced data visualization and custom reporting options[9][11]

CONCLUSION

The **Fintra** application successfully demonstrates the development of a secure, user-friendly, and scalable personal finance management system. By leveraging Flutter for cross-platform compatibility, Firebase Authentication for secure login and notifications, Node.js for backend processing, and MySQL for structured data storage, the system effectively addresses the challenges faced by traditional finance tracking methods. The application allows users to log and categorize transactions, set budgets, track savings goals, and receive personalized financial insights. Testing results confirmed that all modules function reliably, integrate seamlessly, and provide an intuitive user experience. User feedback highlighted the clarity of the dashboard, responsiveness of the interface, and usefulness of real-time updates for financial planning

References

[1] A. Al-Badi, S. Al-Harthy, and M. Al-Qayoudhi, "Personal Finance Management Applications: A Study of User Adoption and Satisfaction," *Proceedings of the 2022 International Conference on Computer and Information Sciences (ICCIS)*, IEEE, 2022.

[2] P. Nguyen, T. Tran, and D. Nguyen, "Design and Implementation of a Cross-Platform Mobile Personal Budget Application," *2023 IEEE 9th International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, 2023.

[3] M. Rahman and S. Sultana, "Enhancing Financial Literacy through Mobile Applications," *Proceedings of the 2021 International Conference on Smart Computing and Communications (ICSCC)*, IEEE, 2021.

[4] R. Sharma and A. Singh, "Security and Privacy in Mobile Finance Applications: Challenges and Solutions," *Proceedings of the 2020 IEEE International Conference on Cloud Computing (CLOUD)*, IEEE, 2020.

[5] Flutter Documentation, "Flutter: Build Apps for Any Screen," Available at: https://docs.flutter.dev

[6] Firebase Documentation, "Firebase Authentication and Cloud Messaging," Available at: https://firebase.google.com/docs

[7] Node.js Documentation, "Node.js v20.x API Reference," Available at: https://nodejs.org/en/docs

[8] MySQL Documentation, "MySQL Reference Manual," Available at: https://dev.mysql.com/doc

[9] GeeksforGeeks, "Flutter Tutorial – Learn Flutter Step by Step," Available at: https://www.geeksforgeeks.org/flutter/

[10] Stack Overflow, "Best Practices for Firebase Authentication in Flutter Apps," Available at: https://stackoverflow.com

[11] Anwar, A., & Sharma, R. (2022). *Flutter for Beginners: Build Cross-Platform Apps with Flutter and Dart 2.* Packt Publishing.

[12] Son, S. (2023). *Firebase Essentials: Building Real-Time Apps with Firebase.* BPB Publications.

[13] Casciaro, M., & Mammino, L. (2020). *Node.js Design Patterns (Third Edition).* Packt Publishing.

[14] Beighley, L. (2021). *Head First SQL: Your Brain on SQL—A Learner's Guide.* O'Reilly Media.

[15] Graham, J., & Kagan, J. (2021). *Personal Finance: Building Your Future (2nd Edition).* McGraw-Hill Education.

[16] S. Kumar and R. Patel, "A Study on the Role of Mobile Applications in Personal Finance Management," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 4, pp. 112–119, 2022.

[17] A. Gupta and M. Saini, "Security Enhancement Techniques in Mobile Banking and Finance Applications," *International Journal of Information Security and Privacy (IJISP)*, vol. 16, no. 3, pp. 45–60, 2022.

[18] R. K. Singh and A. Nair, "Cross-Platform Mobile Application Development Frameworks: A Comparative Study," *International Journal of Software Engineering and Applications (IJSEA)*, vol. 14, no. 2, pp. 67–76, 2023.