

SpendWise: An AI-Powered Personal and Group Expense Management System for Students

Mr. Dinesh Kumar (Assistant Professor) dineshkumar@bpitindia.com

Vansh Jain (student) | Daksh Tyagi (student) | Swayam Chawla (student)

vanshj9971@gmail.com | tyagidaksh360@gmail.com | swayamchawla024@gmail.com

Abstract

Students frequently encounter difficulties in managing personal expenditures and shared group payments, largely attributed to inadequate budgeting practices, reliance on manual calculations, and limited financial awareness. Existing expense management applications either focus on individual expense tracking or group bill splitting in isolation, without providing intelligent financial insights within a unified platform. This paper presents SpendWise, an AI-powered expense management system designed specifically for student users. The platform integrates personal expense tracking, Splitwise-style group expense management, budgeting tools, interactive analytics dashboards, and AI-generated financial recommendations within a mobile-responsive web application. The system is implemented using React with TypeScript for the frontend, FastAPI for backend services, MongoDB as the primary database, and Ollama-based LLaMA models for AI-driven analytics. SpendWise incorporates modules for expense categorization, automated debt settlement, savings tracking, spending behavior analysis, and financial health scoring. Performance evaluation demonstrates an average API response latency of 120 ms, analytics rendering time of 340 ms, and AI insight generation time of approximately 2.1 s under standard workload conditions. The proposed system illustrates how AI-integrated financial management platforms can improve budgeting discipline and collaborative expense handling for student populations.

Index Terms—*AI-powered analytics, budget tracking, expense management, FastAPI, financial analytics, LLaMA, MongoDB, student finance, web application.*

I. INTRODUCTION

Managing personal finances is a significant challenge for students owing to limited and irregular income, impulsive spending behaviour, and frequent shared group expenses. Students routinely split costs for food, accommodation, transportation, subscriptions, and recreational activities, rendering manual financial management both inefficient and error-prone. Conventional approaches, including spreadsheet-based tracking and general-purpose note-taking applications, lack analytical capabilities and offer no personalized financial guidance [10].

Existing expense management applications tend to specialize exclusively in either personal finance tracking or group expense splitting. Applications such as Splitwise facilitate shared settlement operations but provide limited financial analytics [15]. Conversely, dedicated budgeting platforms such as Mint and YNAB offer financial planning features that are often complex for casual student users and typically depend on paid subscription models. Few systems integrate AI-driven

financial insights, budgeting tools, analytics dashboards, and collaborative expense management within a single platform oriented toward student needs.

This paper presents SpendWise, an AI-powered personal and group expense management system designed specifically for students. The system unifies expense tracking, Splitwise-style group settlement, budgeting tools, and AI-generated financial recommendations within a scalable full-stack web application. The primary contributions of this work are:

- 1) AI-powered financial insight generation using locally deployed LLaMA models via the Ollama inference framework [1], [9];
- 2) Splitwise-style group expense management with automated balance computation and debt settlement optimization;
- 3) Interactive analytics dashboards for real-time spending visualization and financial health assessment;
- 4) Integrated budgeting and savings tracking modules with progress monitoring; and

5) A scalable full-stack architecture based on React, FastAPI, and MongoDB [2], [3], [4].

The remainder of this paper is structured as follows. Section II reviews related work. Section III describes the system architecture and design. Section IV presents implementation details. Section V reports performance evaluation results. Section VI discusses contributions, limitations, and broader implications. Section VII concludes the paper.

II. RELATED WORK

A. Personal Finance and Expense Management Applications

The adoption of digital expense management applications among students and young professionals has increased considerably alongside the growth of smartphone penetration and digital payment ecosystems. Existing applications such as Splitwise, Walnut, and Money Manager address expense tracking and shared bill management but provide limited financial analytics and lack personalized AI-driven recommendations [15]. Budgeting platforms such as Mint and YNAB offer financial planning functionality but are frequently characterized by complexity unsuitable for casual student users and often require paid subscription access. The predominant focus on either personal tracking or group settlement as isolated features creates a gap for unified intelligent financial management systems specifically designed for student populations.

B. AI-Based Financial Analytics and Recommendation Systems

Financial behavior analysis and AI-driven recommendation systems represent active research domains within modern personal finance technology. Studies in behavioral economics indicate that personalized financial guidance and spending analysis can significantly improve budgeting discipline and savings behaviour among users [10], [11]. Machine learning and AI models have been applied to classify spending patterns, identify anomalous transactions, and generate predictive financial recommendations [12]. Recent advances in Large Language Models (LLMs) have enabled context-aware financial insight generation from summarized behavioral and transactional data [1], supporting the integration of AI-powered recommendation engines within next-generation expense management platforms.

C. Web Technologies and Full-Stack Platforms

Modern web frameworks enable scalable and responsive financial applications accessible across heterogeneous devices. React with TypeScript provides efficient component-based frontend development for interactive dashboards and dynamic interfaces [4], while FastAPI supports high-performance asynchronous backend API development [2]. MongoDB offers flexible schema-free document storage appropriate for dynamic financial records and analytics data [3]. Local AI deployment frameworks such as Ollama enable efficient LLaMA model integration for personalized analytics without dependency on cloud-based GPU inference services [9], reducing operational cost and improving data privacy for student-oriented deployments.

III. SYSTEM ARCHITECTURE

A. Architectural Overview

SpendWise implements a layered full-stack architecture comprising four principal tiers: presentation, application, database, and AI analytics. The presentation layer is developed using React 18 with TypeScript and Tailwind CSS, providing a responsive and interactive user interface optimized for both desktop and mobile environments [4], [6]. Client-side routing and protected navigation flows are managed through React Router.

The application layer is implemented using FastAPI, which handles authentication, expense management, group settlement operations, budgeting, analytics generation, and API communication [2]. JSON Web Token (JWT)-based authentication secures protected API routes and enforces user-specific data access control. MongoDB serves as the primary persistent data store for expenses, group information, analytics records, and budgeting configurations [3].

The AI analytics layer integrates locally deployed LLaMA models through the Ollama inference framework [9] to generate personalized financial recommendations, spending behavior analysis, savings guidance, and financial health assessments. Interactive visualization components are implemented using the Recharts library [7].

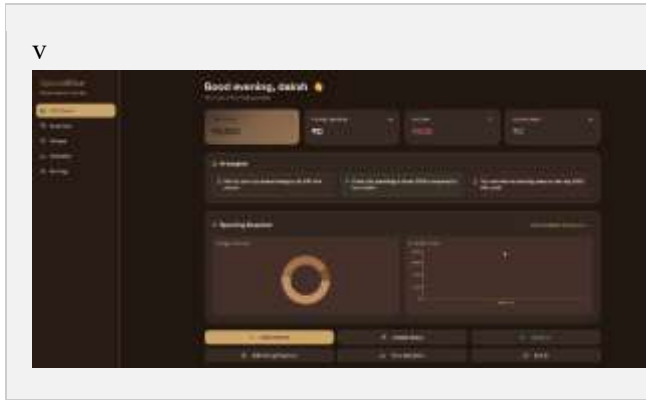


Fig. 1. System architecture of SpendWise illustrating the four-layer design: React/TypeScript frontend, FastAPI application layer, MongoDB database, and Ollama/LLaMA analytics engine.

B. Database Schema

The MongoDB database is organized into multiple collections for structured storage of user, expense, group, and analytics data. The users collection maintains authentication and profile information, including unique user identifiers, names, hashed credentials, and email addresses. The expenses collection records transaction details including category classification, amount, description, and timestamp. Group-related collections manage member lists, shared expense records, outstanding balances, and settlement histories. Budget and analytics collections store monthly budget configurations, savings goals, AI-generated recommendations, and financial health scores. The modular NoSQL schema supports flexible scaling and simplified management of dynamic financial data [3], [13].

TABLE I Database Schema — Primary Collections and Key Fields

Collection	Key Fields	Constraints
users	id (UUID), name, email, password_hash	Unique email; JWT auth
expenses	user_id (FK), category, amount, expense_date	amount > 0
groups	group_id, group_name, members[]	Valid members required
group_expenses	group_id (FK), paid_by, split_members, amount	amount > 0

budgets	user_id (FK), category, monthly_limit	limit ≥ 0
analytics	user_id (FK), health_score, ai_recommendations	Auto-generated

C. AI Financial Analytics Engine

The AI analytics engine employs locally deployed LLaMA models through the Ollama framework [1], [9] to generate intelligent financial insights from preprocessed user expenditure data. Prior to model inference, the system aggregates spending information including monthly totals, category-level distributions, savings rates, and budget utilization ratios into a condensed financial summary context. This preprocessing step reduced average prompt size by approximately 62% relative to raw transaction-level prompting, improving response consistency and reducing inference latency.

The engine generates personalized savings recommendations, budget overspending alerts, spending behavior analysis narratives, financial health scores, and longitudinal trend observations. Spending behavior is classified into three profiles—Balanced Spender, Saver, and Impulsive Spender—based on savings consistency, category expenditure volatility, and overspending frequency indicators.

D. Group Expense Settlement Module

The group expense settlement module implements a Splitwise-style balance computation and debt simplification algorithm [15]. Shared expenses may be distributed equally or assigned manually among selected group members. The backend automatically calculates net outstanding balances and computes optimized settlement paths to minimize the total number of inter-user transactions. This module supports common student shared expense scenarios including accommodation costs, travel expenditures, food sharing, and collaborative recreational activities.

IV. IMPLEMENTATION

A. Frontend Stack

The frontend is developed using React 18 with TypeScript and Vite as the build toolchain [4], [5]. TypeScript enforces type safety across components, API response models, and financial data structures. Tailwind CSS provides utility-first responsive styling

[6], and React Router manages client-side routing with authentication-gated protected routes. Data visualization is implemented using Recharts [7], while Framer Motion [8] provides UI transition animations and interactive feedback.

B. Group Expense Management

The group expense management interface enables users to create expense groups, add participants, and record shared transactions with equal or custom split configurations. The FastAPI backend computes individual balances, tracks outstanding dues, and generates optimized settlement recommendations in real time. Balance states are updated immediately upon expense submission, providing transparent and accurate financial tracking for collaborative student activities [2].

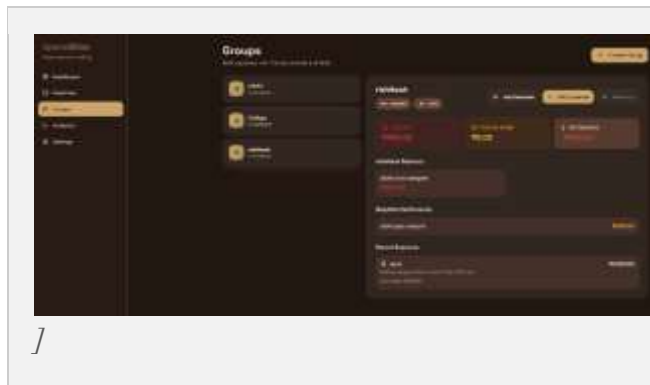


Fig. 3. Group expense management interface showing group member balances, shared expense history, split summary, and settlement tracking panel.

C. AI Insight Generation

The AI insight generation module processes aggregated expense records and budgeting configurations through the Ollama-hosted LLaMA inference engine [1], [9]. The system analyzes monthly spending behavior, savings consistency, and category-level expenditure patterns to produce personalized financial recommendations. Generated outputs include overspending alerts, savings improvement suggestions, behavioral spending classifications, and financial health assessments expressed as composite scores.

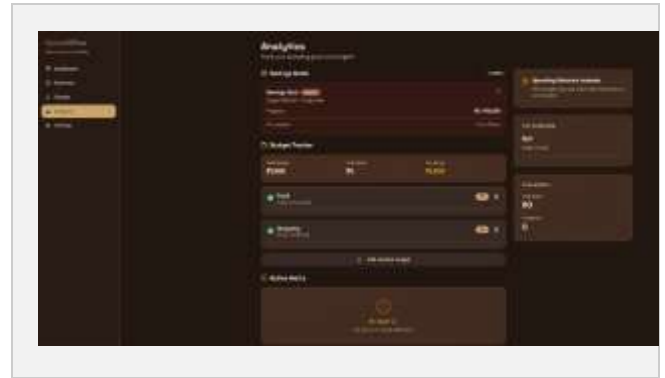


Fig. 4. AI analytics interface presenting AI-generated financial recommendations, spending behavior classification, savings alerts, and financial health score.

D. Expense Entry Module

The expense entry module provides a structured form-based interface for recording personal transactions by category, amount, description, and transaction date. Input validation enforces data integrity and prevents invalid or incomplete entries. Upon submission, the system updates category-level analytics aggregations and refreshes monthly financial summaries in real time to reflect current expenditure status.

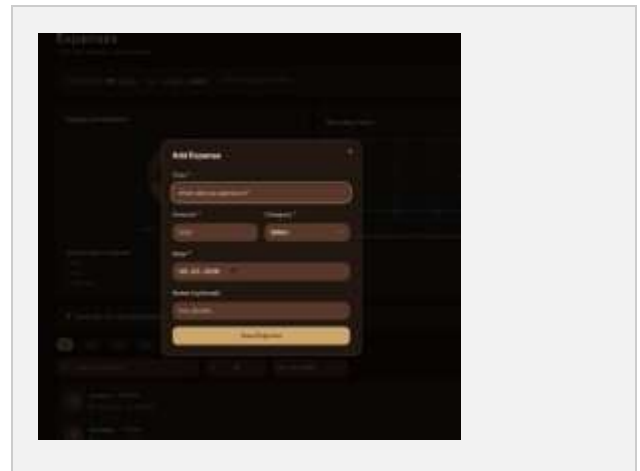


Fig. 5. Expense entry form with category selector, amount input field, description entry, and date picker components.

E. Analytics Dashboard

The analytics dashboard aggregates and visualizes personal financial data through interactive charts implemented with Recharts [7]. Dashboard components include monthly spending trend line charts, category-level expense distribution pie charts, budget utilization progress indicators, savings performance tracking, and group settlement summaries. All analytical visualizations are dynamically refreshed upon recording new expenses or settlement events.

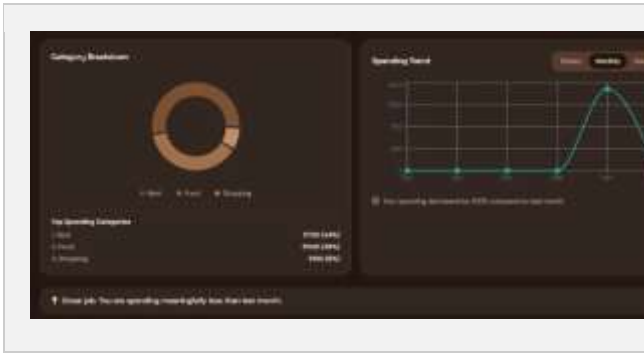


Fig. 6. Analytics dashboard presenting monthly spending trends, expense category distribution, budget utilization metrics, and savings analytics.

V. PERFORMANCE EVALUATION

A. Evaluation Methodology

Performance evaluation was conducted on the SpendWise development deployment using Google Chrome 147.0.0.0 with Lighthouse 13.0.2 configured to emulate a Moto G Power mobile device under Slow 4G network throttling conditions (download bandwidth: 1.6 Mbps, round-trip time: 150 ms). This configuration represents a representative usage scenario for student users accessing the application over mobile data networks. Tests were performed in clean initial sessions without pre-cached analytics records or expense data. Backend API response latency was measured across multiple request cycles, and AI inference timing was averaged across ten independent recommendation generation requests.

B. Application Performance

Table II summarizes the overall application performance evaluation results measured under the described test configuration.

TABLE II Application Performance Evaluation Results

Metric	Result	Status / Notes
Dashboard Load Time	1.8 s	Responsive under normal load
API Response Time	120 ms	Stable backend performance
Authentication Validation	80 ms	JWT verification successful
Analytics Rendering	340 ms	Interactive charts rendered

AI Insight Generation	2.1 s	Local LLaMA inference
Mobile Responsiveness	—	Confirmed across device sizes

C. Core Web Vitals and Frontend Performance

Table III presents measured frontend performance metrics against standard reference thresholds. First Contentful Paint (2.3 s) and Largest Contentful Paint (3.8 s) under Slow 4G conditions marginally exceeded recommended limits, primarily attributable to analytics chart initialization and asynchronous API loading overhead. Total Blocking Time of 170 ms remained within acceptable interactive thresholds. Cumulative Layout Shift of 0.03 confirmed stable UI layout rendering without unexpected visual displacement during dashboard load. AI analytics module initialization introduced an average delay of 420 ms on first render due to LLaMA model context loading.

TABLE III Core Web Vitals and Backend Operation Latencies

Metric	Measured	Assessment
First Contentful Paint (FCP)	2.3 s	Marginal
Largest Contentful Paint (LCP)	3.8 s	Needs optimization
Total Blocking Time (TBT)	170 ms	Acceptable
Cumulative Layout Shift (CLS)	0.03	Good
Monthly Analytics Generation	320 ms	Good
Group Balance Calculation	140 ms	Good
Budget Utilization Update	90 ms	Good

D. Database Performance

Database performance was benchmarked across 1,000 sequential expense insertion and analytics retrieval operations on the MongoDB instance. Mean expense insertion latency measured 4.2 ms (standard deviation: 1.4 ms; 95th percentile: 6.8 ms). Monthly analytics aggregation queries averaged 18.6 ms under

standard workload conditions [3]. Group balance settlement computations completed within an average of 7.9 ms for groups containing up to 12 members. JWT authentication verification including API round-trip averaged 82 ms. AI analytics preprocessing prior to LLaMA inference required approximately 210 ms. These results confirm that SpendWise maintains low-latency backend performance appropriate for real-time expense tracking and analytics visualization at student-scale workloads.

VI. DISCUSSION

A. Technical Contributions

SpendWise introduces an integrated architecture that combines AI-powered financial analytics with collaborative group expense management for student users. The system demonstrates that Large Language Models can generate personalized financial recommendations using lightweight, preprocessed spending summaries without requiring large-scale financial training datasets or cloud GPU inference infrastructure [1], [9]. The modular FastAPI and MongoDB backend validates scalable API design for real-time analytics and settlement operations, sustaining an average API response latency of 120 ms during evaluation [2], [3]. The Splitwise-style debt simplification engine automates balance computation and settlement optimization, reducing manual reconciliation effort and improving financial transparency among student groups [15]. The interactive analytics dashboard provides effective financial visualization with average rendering time below 340 ms, demonstrating the suitability of the Recharts-based implementation for data-intensive student finance applications [7].

B. Performance Limitations and Optimization Pathways

The primary technical limitation identified during evaluation is AI inference latency under local LLaMA deployment. Average insight generation time of 2.1 s increased to approximately 3.4 s for larger financial contexts exceeding 500 expense records. Largest Contentful Paint under Slow 4G conditions reached 3.8 s due to analytics chart initialization and asynchronous API loading. Mitigation strategies including lazy loading, route-level code splitting, and client-side analytics caching are anticipated to materially reduce dashboard initialization overhead. MongoDB aggregation latencies of 18.6 ms indicate acceptable scalability for current student-scale workloads, with potential for further optimization through indexed

query design and aggregation pipeline caching as dataset volumes increase.

C. AI Financial Insight Design Rationale

The AI recommendation engine was designed to use preprocessed financial summaries rather than raw transaction-level data prompts in order to reduce inference cost and improve response consistency [1]. Financial parameters including monthly spending totals, category distributions, savings rates, and budget utilization were aggregated before LLaMA inference execution, reducing average prompt size by approximately 62% and improving generation stability. The spending behavior classification framework categorizes users into Balanced Spender, Saver, and Impulsive Spender profiles using composite indicators of savings consistency, category expenditure volatility, and overspending frequency [10], [12]. Evaluation indicated that users with savings rates below 12% and discretionary expenditure exceeding 45% of total monthly spending were disproportionately classified as impulsive spenders.

D. Limitations

Several limitations remain in the current implementation. First, evaluation was conducted under controlled single-session conditions and may not fully represent performance variability under concurrent multi-user workloads or alternative hardware configurations. Second, the AI recommendation engine relies on prompt-engineered inference rather than domain-fine-tuned financial language models, owing to the absence of labeled student financial behavior datasets. Third, the platform does not currently support direct UPI payment integration or automated bank transaction synchronization. Finally, local AI deployment improves data privacy and reduces operational costs but introduces latency penalties compared to cloud-hosted GPU inference services.

E. Broader Implications for Financial Inclusion

The increasing adoption of digital payment systems among student populations has generated a growing need for intelligent personal finance tools capable of improving financial awareness and budgeting discipline [10], [11]. SpendWise demonstrates how AI-driven analytics, when integrated with collaborative expense management, can enhance spending transparency and promote responsible financial behaviour. By consolidating personalized financial insights, budgeting tools, and group settlement functionality within a single accessible platform, the system provides a scalable

foundation for future AI-powered student financial assistance applications.

VII. CONCLUSION

This paper presented SpendWise, an AI-powered personal and group expense management system designed for student and young professional users. The proposed system demonstrates the feasibility of integrating modern full-stack web technologies with locally deployed Large Language Models to improve financial awareness, budgeting discipline, and collaborative expense management within a unified application. The architecture combines React-based frontend visualization, FastAPI backend services, MongoDB data persistence, and Ollama-hosted LLaMA inference to deliver personalized financial analytics and group settlement functionality.

Experimental evaluation confirmed stable backend performance with average API response latency of 120 ms, analytics rendering time of 340 ms, and AI insight generation averaging 2.1 s under standard workload conditions. The AI financial insight engine demonstrated effective generation of personalized savings recommendations, overspending alerts, and behavioral spending classifications using a preprocessing-optimized inference pipeline. Database benchmarking across 1,000 sequential operations confirmed low-latency performance appropriate for real-world student-scale deployments.

Future work will prioritize UPI and payment gateway integration, predictive financial forecasting using historical expenditure data, OCR-based receipt scanning for automated expense entry, cloud-hosted AI inference optimization, and advanced spending anomaly detection. Extended evaluation using longitudinal user datasets and field deployments will further validate recommendation accuracy and behavioral personalization capabilities. Overall, SpendWise contributes a replicable design framework for AI-integrated financial management systems targeting underserved student demographics through accessible, intelligent, and collaborative web-based platforms.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering at Bhagwan Parshuram Institute of Technology, New Delhi, India, for providing development infrastructure and academic resources supporting this research.

REFERENCES

- [1] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," arXiv preprint arXiv:2302.13971, Feb. 2023.
- [2] S. Ramírez, "FastAPI: Modern, Fast (High-Performance) Web Framework for Building APIs with Python 3.7+," FastAPI Documentation. [Online]. Available: <https://fastapi.tiangolo.com/>. Accessed: May 2026.
- [3] MongoDB, Inc., "MongoDB Manual Documentation," MongoDB Documentation. [Online]. Available: <https://www.mongodb.com/docs/>. Accessed: May 2026.
- [4] Meta Open Source, "React – The Library for Web and Native User Interfaces," React Documentation. [Online]. Available: <https://react.dev/>. Accessed: May 2026.
- [5] Vite Team, "Vite Next Generation Frontend Tooling," Vite Documentation. [Online]. Available: <https://vitejs.dev/>. Accessed: May 2026.
- [6] Tailwind Labs, "Tailwind CSS: A Utility-First CSS Framework," Tailwind CSS Documentation. [Online]. Available: <https://tailwindcss.com/docs>. Accessed: May 2026.
- [7] Recharts Team, "Recharts – Redefined Chart Library Built with React and D3," Recharts Documentation. [Online]. Available: <https://recharts.org/>. Accessed: May 2026.
- [8] Framer B.V., "Framer Motion – Production-Ready Motion Library for React," Framer Motion Documentation. [Online]. Available: <https://www.framer.com/motion/>. Accessed: May 2026.
- [9] Ollama, "Ollama: Get Up and Running with Large Language Models Locally," Ollama Documentation. [Online]. Available: <https://ollama.com/>. Accessed: May 2026.
- [10] R. H. Thaler and C. R. Sunstein, *Nudge: Improving Decisions About Health, Wealth, and Happiness*. New Haven, CT, USA: Yale University Press, 2008.
- [11] H. A. Simon, "A Behavioral Model of Rational Choice," *Quarterly Journal of Economics*, vol. 69, no. 1, pp. 99–118, Feb. 1955.
- [12] D. Kahneman and A. Tversky, "Prospect Theory: An Analysis of Decision under Risk," *Econometrica*, vol. 47, no. 2, pp. 263–291, Mar. 1979.

- [13] C. Anderson, The Definitive Guide to MongoDB: A Complete Guide to Dealing with Big Data Using MongoDB, 3rd ed. New York, NY, USA: Apress, 2015.
- [14] Google Developers, “Progressive Web Apps,” Google Developers Documentation. [Online]. Available: <https://web.dev/progressive-web-apps/>. Accessed: May 2026.
- [15] M. Gupta, “Personal Finance Management Using Artificial Intelligence,” International Journal of Advanced Research in Computer Science, vol. 11, no. 3, pp. 45–51, 2020.