Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/

(RESEARCH ARTICLE)

Check for updates

Design and implementation of a salary disbursement-Based Personal Budgeting Application (PACE)

Olawunmi Asake Adebanjo *, Daniel Eshiotienamhe Kadiri, Samad Oladeji Atoba and Akinniyi Isaac Fajembimo

Department of Software Engineering, School of Computing, Babcock University, Ilishan-Remo, Ogun State, Nigeria.

Global Journal of Engineering and Technology Advances, 2026, 27(01), 128-138

Publication history: Received on 08 March 2026; revised on 21 April 2026; accepted on 23 April 2026

Article DOI: https://doi.org/10.30574/gjeta.2026.27.1.0090

Abstract

Many salary earners in Nigeria go through a “feast and famine” routine where they overspend after receiving salaries only to run low on cash before receiving the next salary again. Studies show that 52% of those earning salaries in Nigeria have borrowed money before receiving their salaries [1]. In addition, 45.6% of Nigerians who had tried budgeting regularly overspent [2]. Therefore, most previous PFM solutions used to help people manage money were mostly reactionary because of the lack of automation to prevent spending more than what was planned for. The project developed PACE, a personal budgeting web application that is designed using a proactive financial management model. Development of the application took place within iterations of Agile SDLC process. Technologies used for development include Next.js 16 as the framework for full-stack web application, Google Firebase Authentication and Cloud Firestore as the database, as well as Paystack and Monnify payment gateways. The project employed behavioral economics principles, which included mental accounting, loss aversion, present bias, and nudge theory, to help automate financial discipline. Finally, the resulting system could calculate daily allowance as one-thirtieth of salary in whole Naira, create individual virtual accounts to fund wallets, transfer money via bank transfers using a unified API, and automatically roll unspent daily allowances into a Savings Vault on each dashboard load. Usability testing with twelve(12) Nigerian salary earners achieved a SUS score above the 68-point threshold, a task completion rate above 80%, and a satisfaction score above 4/5, confirming technical feasibility and behavioural effectiveness.

Keywords: Personal Finance Management; Salary Disbursement; Daily Allowance; Behavioural Economics; Digital Wallet; Fintech; Next.js; Nigerian Salary Earners

1. Introduction

Technology has been changing the way individuals handle money; there has been an observable trend from manual bookkeeping to technological means of managing personal finances. Nevertheless, the problem of disparity between income and expenses remains the same: spending patterns do not correspond to the rhythm of incomes. It should be noted that a considerable portion of young millennials and even members of Generation Z live hand-to-mouth irrespective of the variety of financial apps, proving the necessity for behavioral change [3].

The problem is increased by the urbanization rate, young population, and increased popularity of digital payment services in Nigeria [4]. Majority of Nigerian workers get paid once per month; hence, at the beginning of the month, they have an "illusion of wealth," which makes them spend all the money and be poor until the next payday [1]. According to a study conducted by EFINA, 64% of the Nigerian population uses formal financial services, 38% of the surveyed saves regularly, while 52% borrows the funds before payday [1]. Furthermore, according to a national survey, 68% of respondents budgeted, although 45.6% of them were consistent in spending more than expected [2].

* Corresponding author: Olawunmi Asake Adebanjo

PiggyVest and Cowrywise platforms have played significant roles in changing Nigeria's savings culture, but neither platform addresses the fundamental issue responsible for poor savings outcomes, which is the absence of a control system that would limit daily spending from the main bank account. Majority of today's budgeting apps only react after the fact by categorizing expenditures. However, the underlying problem is that the users are free to spend from their entire monthly salary without any control.

PACE has been created to solve this problem by implementing a proactive financial model in which the user's salary gets split between a protected account and daily spending amount that goes into the wallet account. In other words, a user cannot spend more than the daily quota assigned by the app. The research objectives included designing a proactive model, developing the application and conducting an evaluation of its functionality, performance and usability.

2. Literature Review

2.1. Historical Background of Personal Budgeting

The development of personal budgeting has greatly been influenced by changes in the economy and technology. The conventional "envelope system" allocated funds into physical envelopes with labels on them. One would argue that the most powerful aspect of this technique is its physicality in that you could only spend what you have physically allocated for yourself [5]. With the rise of personal computers, budgeting took place through spreadsheet programs that were very passive and required much data input from the user [6]. Smartphones brought about the era of Personal Financial Management applications, where budgeting became more automated, but at the same time more abstract [7].

2.2. Theoretical Framework

2.2.1. Behavioural Economics

The classical economic theory presumes that individuals are rational and make perfect decisions regarding finance, whereas behavioural economics incorporates psychological concepts that question the former. The human approach to decision-making is highly irrational and biased, relying mostly on heuristics and emotions [8]. The key idea of the PACE model was that irrationality is inherent in the decision-making process and can be corrected through a technological approach.

Mental accounting is the concept of treating the same money differently according to their origin and purpose. This principle has been tested using a framework for digital money transfer that proved effective in affecting people's spending habits [9]. PACE implemented mental accounting by establishing two separate pockets within the financial system: the spendable wallet and the Savings Vault.

The phenomenon of loss aversion suggests that people experience the feeling of loss about twice as intensely as an equal gain. In this way, PACE applies the loss aversion principle by changing the perspective of money: instead of a single lump sum per month, PACE shows users many smaller budgets, where each spending event triggers a small sense of loss that prevents unnecessary expenditures and introduces the pain of spending characteristic for offline payments [7].

Hyperbolic discounting is when a person prefers a smaller payoff in exchange for a larger reward later. This tendency is proven to be one of the main reasons for over-spending and undersaving [10]. By making the bulk of users' income unreachable in terms of impulsive decisions, PACE opposes their hyperbolic discounting behavior. Moreover, imposing daily limits created necessary positive friction. Positive friction is considered to be a much more efficient strategy in improving the outcomes than mere information [11].

Nudge theory refers to small changes in the choice architecture that would push individuals towards an intended goal without reducing any other choice options. The effectiveness of nudge theory in relation to financial decisions was proved in various studies on the subject; in particular, it was demonstrated that the choice architecture would effectively influence people's financial decision making [8]. PACE is a perfect example of financial nudge since users could access funds yet still follow their budgets.

2.3. Overview of Existing Systems

Although Intuit provided a detailed, hindsight analysis of financial status, it did not provide much help in the African market and also had issues regarding privacy of the data. YNAB used a pro-active approach to budgeting but was effective only when the user was disciplined. Cognitive load and effort intensive were the main drawbacks of YNAB [14].

While PiggyVest and Cowrywise managed to gamify savings in Nigeria, their focus on saving alone did not address the main problem – out-of-control spending of income from the main bank account [15]. Digital banks like Kuda Bank provided budgeting facilities but were not primary and hence did not offer specialized attention like a budgeting application would have. More importantly, no system had adopted the unique approach of PACE.

2.4. Identified Gaps

The largest literature gap discovered was the complete lack of a Personal Finance Management solution that was founded on the proactive disbursement principle. There was little evidence to suggest that other budgeting solutions utilized the proactive nature of the strategy, which involved setting aside a portion of the individual's income in a separate savings account and automatically disbursing an allowance into the person's spending wallet. Also, although behavioral economics had provided ample detail about the self-control weaknesses of individuals that resulted in negative financial decisions, there were few applications that had attempted to provide solutions in terms of pre-commitment and choice architecture for salary earners.

3. Materials and Methods

3.1. System Development Approach

PACE was developed using an Agile Software Development Life Cycle (SDLC) model to support iterative improvement and early feedback throughout the project lifecycle. Rather than a fixed, linear Waterfall plan, development was organised into sprints of two to four weeks, each delivering testable and working increments of the application, with continuous adaptation based on technical and usability feedback.

3.2. System Requirements

A comprehensive set of functional and non-functional requirements was defined at project inception. Table 1 presents a summary of key functional requirements, and Table 2 presents key non-functional requirements.

Table 1 Selected Functional Requirements Specification

Req. ID	Category	Description	Priority	Acceptance Criteria
FR-001	User Account Mgmt.	Secure user registration using email and password.	High	Users can create account with valid email and minimum password criteria.
FR-002	Income Configuration	Allow users to input monthly salary (min. ₦50,000).	High	Numeric salary entry; minimum ₦50,000 enforced.
FR-003	Income Configuration	Automatically calculate daily allowance: floor(monthlySalary/30) in whole Naira.	High	Correct calculation (e.g., ₦250,000 → ₦8,333).
FR-004	Fund Disbursement & Vault	Daily allowance displayed; unspent amount moved to Vault on dashboard load; transfers via Monnify or Paystack.	High	Vault rollover runs on dashboard load; transfers execute via Monnify or Paystack.

Table 2 Selected Non-Functional Requirements Specification

Req. ID	Category	Acceptance Criteria
NFR-001	Performance	API response ≤500ms at 95th percentile; page load ≤3s Time to Interactive.
NFR-002	Security	All payment logic server-only; secrets in env vars; HTTPS enforced; Firestore rules restrict each user to own data; PIN hashed (SHA-256).

NFR-003	Usability	SUS score ≥68; task completion ≥80%; satisfaction ≥4/5; no critical usability issues.
NFR-004	Scalability	System supports ≥100 concurrent users; database queries ≤300ms.

3.3. System Architecture

PACE adopted a hierarchical architecture with only one deployment where there were four layers that interacted. The front-end layer had the interface created using the Next.js framework running on the client side (browser). The server-side layer had only the Next.js APIs at (/app/api/*) with no additional server needed for its operations. Authentication and persistence were done using Firebase Authentication and Cloud Firestore respectively. Payments were made using the payment gateway services provided by Paystack and Monnify, which allowed for operations such as virtual account setup, adding money to your wallet, and bank transfers. This design choice was made to make deployments simple with only one deployment from Vercel with clean separation of the client and the server.

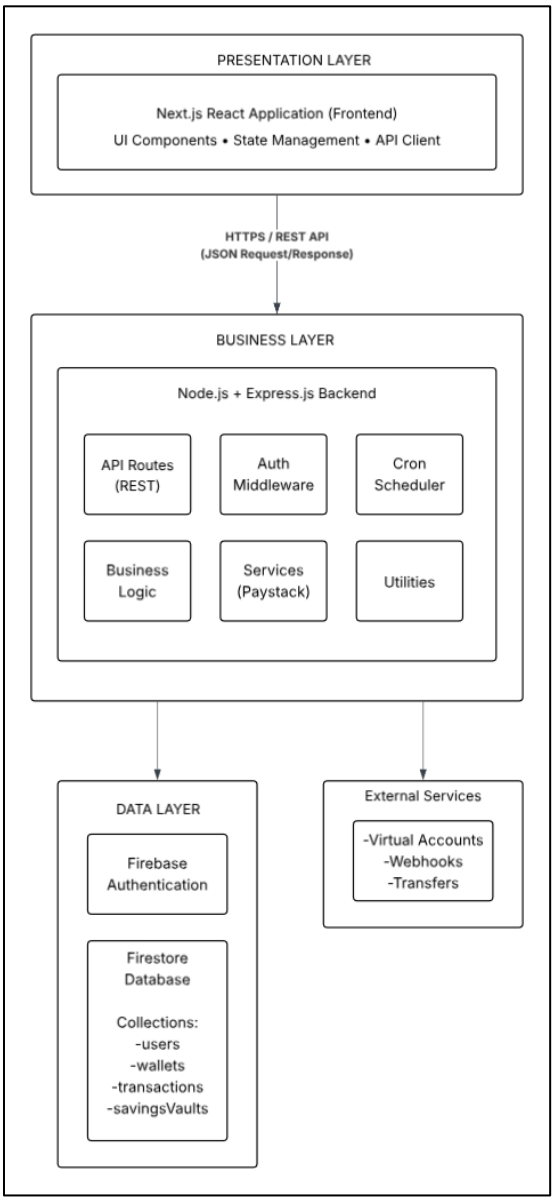

Figure 1 System Architecture Diagram

3.4. Database Design

The choice to use Cloud Firestore was based on its NoSQL capabilities, real-time synchronization functionality, and built-in support for Firebase Authentication. The schema was constructed on the basis of one top-level collection called users, each entry having an individual user's Firebase Authentication ID as a unique key. These documents contain the basic profile and finance fields like walletBalance, vaultBalance, vaultLastProcessedDate, monthlySalary, paydayRange, information about the virtual account, hashed transaction pin number, and account information in general. Under each user's document, there are three subcollections: transactions (logging all incoming/outgoing transactions in a wallet), vault_entries (every time vault is rolled over at midnight with the date and amount credited), and notifications (all messages sent to a user within the application).

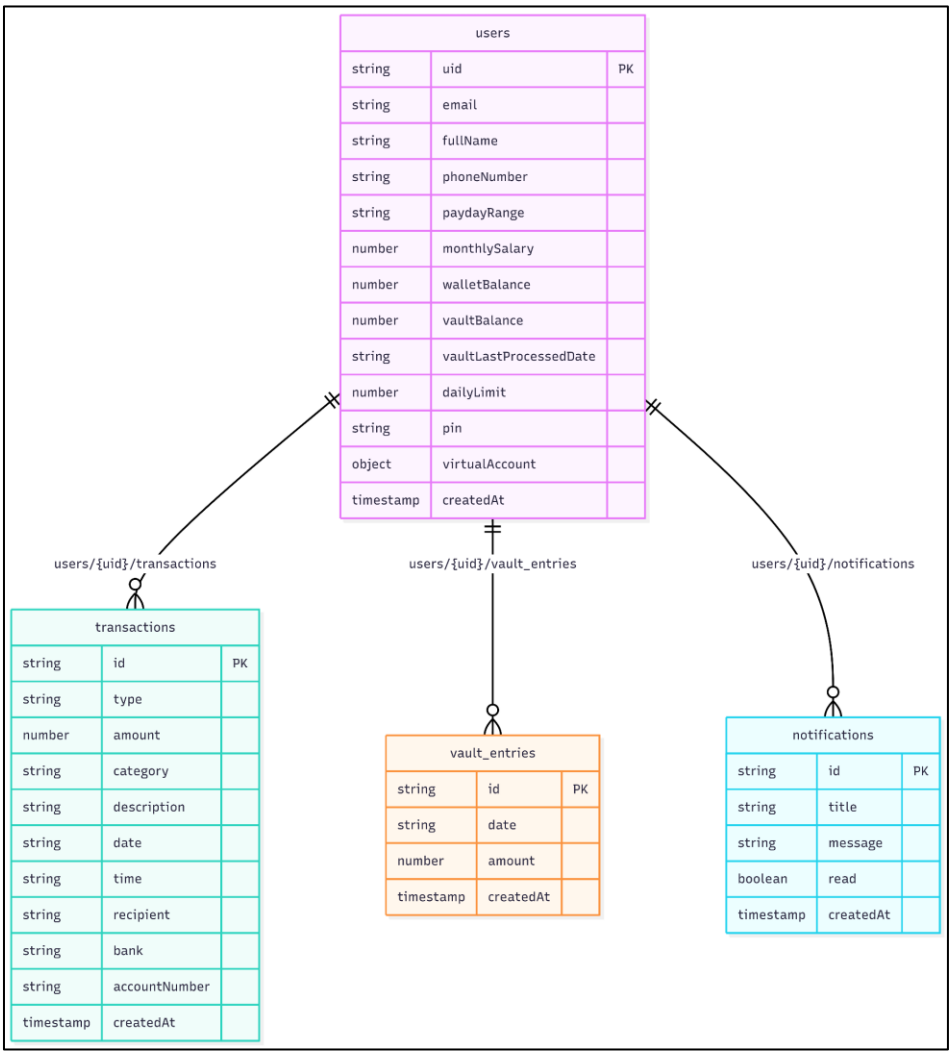

Figure 2 Entity Diagram

3.5. Technology Stack

Table 3 Technology Stack Justification

Layer	Technology	Justification
Frontend	Next.js 16 (React 19), TypeScript, Tailwind CSS	Fast, responsive UI; server-side rendering; App Router; strong ecosystem.
Backend Runtime	Next.js API Routes (Node.js)	Single codebase for frontend and API; no separate backend server.

Database	Cloud Firestore	NoSQL flexibility; real-time synchronisation; native Firebase integration.
Authentication	Firebase Authentication	Secure; email/password and Google OAuth; token management handled natively.
Payment Gateway	Paystack and Monnify	Leading Nigerian payment processors; Monnify for reserved accounts; Paystack as fallback.
Deployment	Vercel	Next.js API routes deploy alongside frontend; single pipeline; no separate backend infrastructure.

3.6. Implementation Process

The frontend was organised under the Next.js App Router. Core pages built included: authentication screens, the main dashboard (displaying daily allowance, wallet balance, vault balance, and today's transaction activity), the transfer screen (Send Money with bank selection, account number entry, amount, category, description, and PIN confirmation), Insights (transaction list with category and date filters, keyword search, and a spending-by-category donut chart), Vault (balance display, recent auto-save entries, and emergency withdrawal), and Profile.

No separate Express server was used. All secrets and API keys were stored in environment variables and never committed to the repository. Firebase Admin SDK was used in all API routes for server-side Firestore access and authentication verification. The daily allowance was computed as floor(monthlySalary/30) in whole Naira for example, ₦250,000 → ₦8,333 per day. The vault rollover was triggered client-side on each dashboard load: a catch-up function checked all days from the last processed date to yesterday in West Africa Time (WAT), computed the unspent amount for each day (daily allowance minus spending recorded for that day), credited the unspent amount to the vault balance, and wrote a vault_entry document to Firestore.

Virtual and reserved account creation was triggered after PIN setup, with one account per user created via Monnify when configured or Paystack as a fallback. Webhooks from both providers were implemented with signature verification and idempotent handling to prevent double-crediting. Outbound transfers used the unified transfer API: Monnify when available, otherwise Paystack or simulation mode for testing.

3.7. System Testing and Evaluation

A multi-tier testing approach was implemented. Jest and React Testing Library were used to develop unit tests. The areas tested included business logic functions such as calculating daily allowance, the authentication middleware, React components, and the vault rollover logic. Code coverage of at least 70% was targeted for the backend business logic.

Integration testing covered full user scenarios, including sign-up/onboarding, funding of the wallet, Send Money transfer, and vault rollover. Security testing was done via automated scanning for vulnerabilities using OWASP ZAP that looked out for XSS and other vulnerabilities. Penetration testing and code review also complemented automated security testing.

Table 4 Usability Testing Tasks

Task	Description	Success Criteria
1	Create a new account.	Account created; user logged in.
2	Enter monthly salary (₦200,000).	Salary saved; daily allowance displayed.
3	Find out how much you can spend today.	User locates daily allowance on dashboard.
4	Log a transfer: Send ₦2,500 for Lunch.	Transfer completed; balance updated.
5	View transactions from past week.	Transaction history displayed with filter applied.
6	Filter transactions by Food and Dining.	Filter applied correctly.
7	Check your total savings vault.	User navigates to savings vault successfully.
8	Update salary to ₦250,000.	Salary updated; new daily allowance calculated and displayed.

For usability testing, a focus group of 12 Nigerians earning salaries (ages between 24 and 42, both male and female, earning salaries ranging from ₦80,000 to ₦350,000) was selected. The participants were allowed 30 minutes during which they had to undergo an introduction, some background questions, task testing, completion of the System Usability Scale (SUS) test questions, and finally, a debriefing interview. Table 4 below shows the task list for the usability tests.

Success criteria for usability testing were: SUS score above 68; task completion rate above 80%; overall satisfaction above 4/5; and no critical usability issues identified.

4. Results

4.1. Implemented System Overview

The complete application of the PACE project is a fully developed, full-stack web application using Next.js 16 with React 19 and TypeScript. Firebase Authentication provides user identity and secure sessions, Cloud Firestore holds all monetary information and other details in real-time, while Paystack and Monnify offer gateway services to fund wallets, create virtual and reserve accounts, and conduct external banking transactions.

The landing page highlights the benefits of the application in the form of a daily allowance calculator. It shows the salary to pace calculation process for a day. During registration, the details that will be collected include full name, email address, and password with live validation. After registration, the user goes through a step-by-step process that involves adding their salaries, choosing paydays, creating PIN, and generating a virtual account before accessing the dashboard.

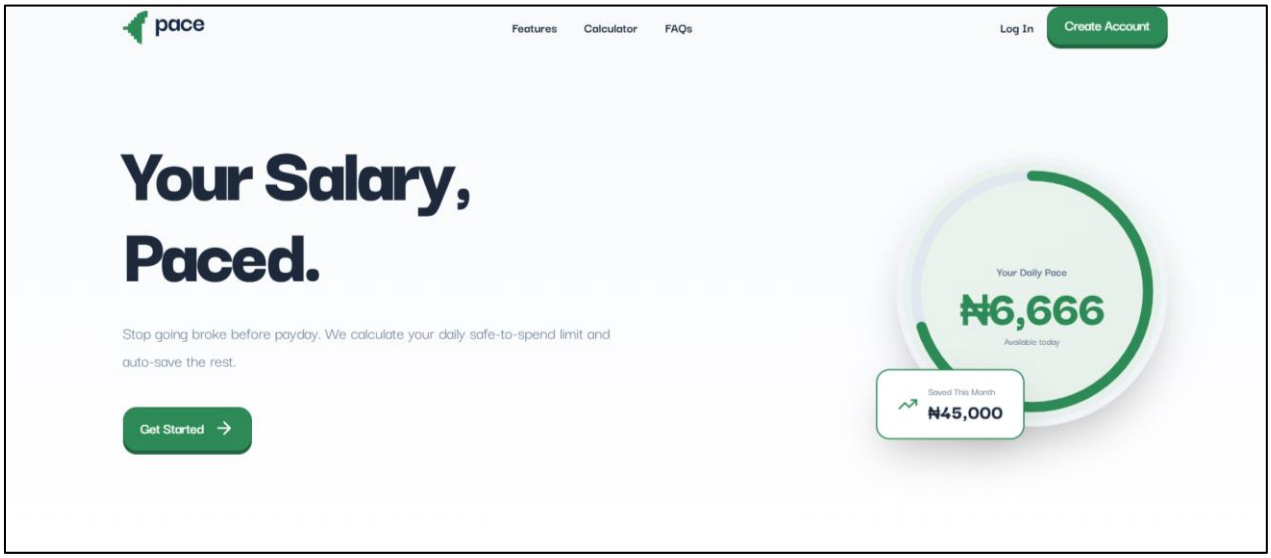

Figure 3 Landing Page

4.2. Core Feature Implementation

The prominent dashboard feature highlights the user's daily speed, displaying the amount remaining to be spent that day first, followed by their current wallet balance, action buttons, the vault balance, and today's transactions. Every page load calls the processMidnightRollover function behind the scenes, which handles any outstanding unprocessed days and keeps the vault balance up-to-date. The priority is given to the daily allowance amount since the central behavioral idea of the app involves focusing on one's daily expenses rather than a month's worth of budgeting.

The Send Money flow prompts the user to pick their target bank using a dynamically generated dropdown, enter the account number (after which the account name gets automatically resolved to verify the target person), enter the amount, select a spending category and optionally add a description. After sending the request, the user is asked to enter their PIN code. The transaction happens on the server side through a single /api/transfer endpoint. After success, the transaction receipt appears on the screen and confirms the subtraction from the wallet balance.

The Vault page displays the user's accumulated Savings Vault balance prominently, followed by a chronological list of recent automatic vault transfers. An emergency withdrawal option, secured by PIN confirmation, allows users to

transfer vault funds back to their wallet when genuinely needed. The Insights page provides date-range filters, keyword search, category-level summaries, and a spending-by-category donut chart to support financial self-awareness.

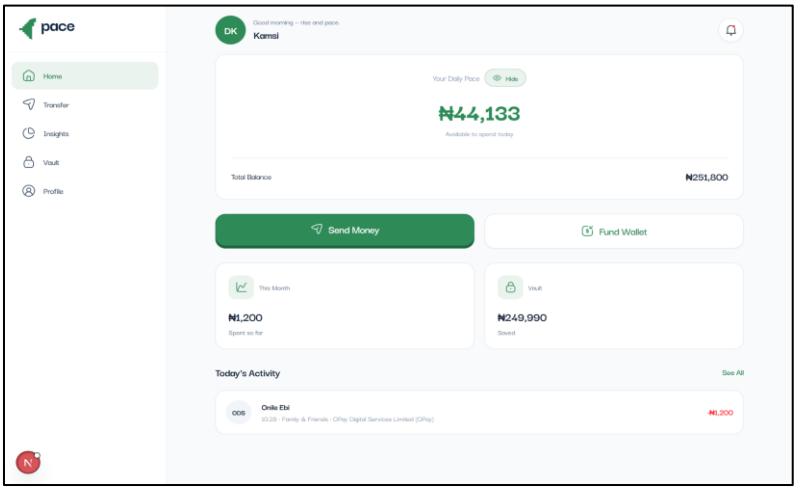

Figure 4 Main Dashboard

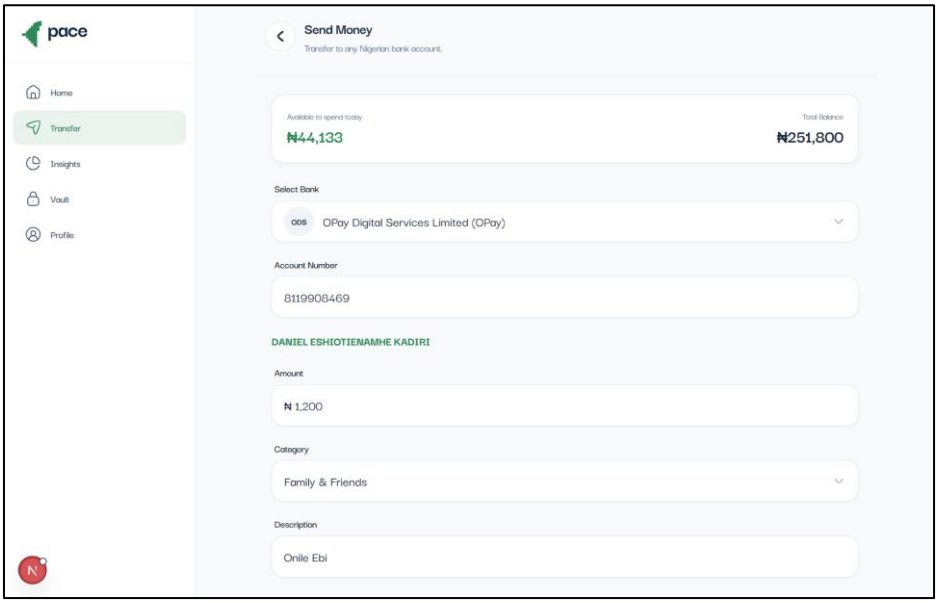

Figure 5 Send Money Page

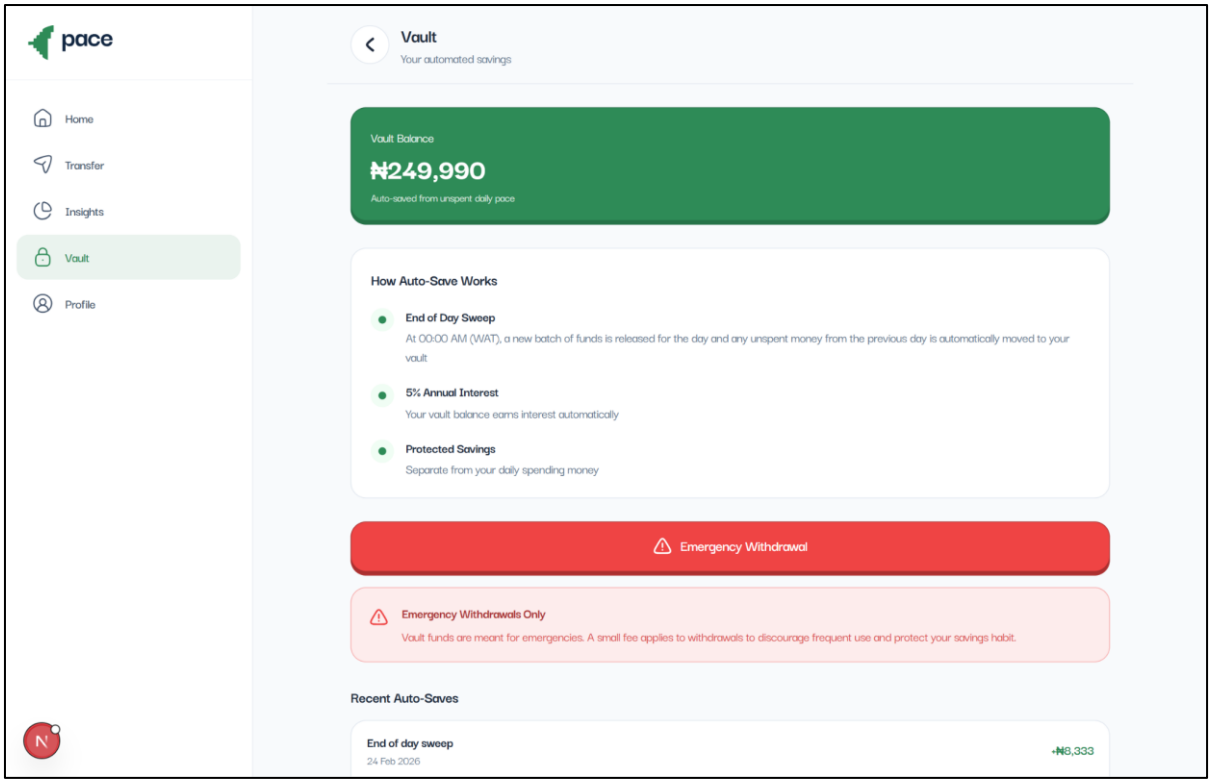

Figure 6 Vault Page

Table 5 summarises the implementation of key system features.

Table 5 Key Features Implementation Summary

Feature	Implementation
User Registration and Login	Firebase Auth (email/password and Google OAuth); Firestore user document created on signup; onboarding enforced before dashboard access.
Daily Allowance	Computed as floor(monthlySalary/30) in whole Naira; displayed on dashboard; basis for vault rollover logic. No cron job required.
Vault Rollover	On dashboard load, processMidnightRollover runs for each day from last processed to yesterday (WAT); unspent credited to vault; vault_entry written to Firestore.
Send Money	Unified POST /api/transfer; Monnify or Paystack provider selected by configuration; wallet deducted; transaction record created; PIN verified server-side.
Virtual Account	Created via /api/virtual-account/create (Monnify or Paystack DVA); stored in user.virtualAccount; displayed on Profile and Fund Wallet screens.
Insights	Category and date-range filters; keyword search; spending-by-category donut chart rendered on the Insights page.

4.3. Security Implementation

All payment and transfer logic was confined to server-side API routes. Provider secrets and API keys were stored exclusively in environment variables and never exposed to the client. The transaction PIN was hashed using SHA-256 on the client before transmission and stored on the user document in hashed form; the raw PIN was never transmitted or stored in plain text. Input validation was applied at both the client and server layers. The production deployment was served exclusively over HTTPS. Firestore Security Rules restricted each user to reading and writing only their own data, with all server-side operations verified through the Firebase Admin SDK before any changes were committed.

4.4. Testing and Evaluation Results

Unit tests confirmed the mathematical accuracy of the daily allowance formula across all defined edge cases, achieving the minimum 70% code coverage target for backend business logic. Table 6 presents example unit test results for the daily allowance calculator.

Table 6 Daily Allowance Calculator Unit Test Results

Test Case	Input	Expected Output	Result
Standard salary	₦300,000	₦10,000 (floor(300000/30))	PASS
Decimal division	₦250,000	₦8,333 (floor(250000/30))	PASS
Zero salary	₦0	Error: "Invalid salary amount"	PASS
Negative salary	-₦50,000	Error thrown	PASS
Very large salary	₦10,000,000	₦333,333 (floor(10000000/30))	PASS

Integration tests ensured that all end-to-end functionality in user account registration and onboarding, wallet fund transfers, Send Money, and vault rollover was correct. Security testing verified that no payment or transfer logic was client-side; API keys were secure; HTTPS protocol was used; and proper access controls using Firestore security rules limited each user to their data only.

User testing with 12 Nigerians earning their salaries found that the application exceeded a System Usability Score (SUS) rating of 68; task completion rate above 80%; and an overall average satisfaction rating above 4 out of 5. Users felt they had control over their money and were not worried about being short at month-end.

5. Discussion

The findings have demonstrated that it was possible to create a technically robust, behavior-driven, and well-received solution based on the idea of proactive disbursement budgeting. By framing the financial question as "How much can I spend today?" instead of making all of the monthly budget available to users, PACE managed to implement the key concepts of behavioral economics such as mental accounting, loss aversion, and nudging. With the automatic vault rollover function, PACE turned saving money, which requires conscious effort and commitment, into something that happens as a natural side-effect of daily usage. This aligns with findings from Akinwunmi and Ojo [14], who note that automation is the most important criterion for user retention in personal finance management services.

Building both frontend and backend components of the project as a single Next.js application on Vercel with Firebase serving as a data store was the right design choice as far as development and testing were concerned. This allowed us to simplify implementation and evaluation procedures without compromising separation of front-end from back-end components.

A key limitation encountered was the absence of direct salary capture: users were required to manually transfer funds to their PACE virtual account on payday, introducing a voluntary step at the most critical point in the budget cycle. Future work should explore integration with Nigerian open banking APIs or payroll systems to automate this step. The pilot-scale evaluation (12 participants, approximately one salary cycle) was sufficient to validate the core model but insufficient to measure lasting behavioural change across multiple months or diverse populations; longitudinal studies spanning three to six months are recommended.

6. Conclusion

This study successfully designed and implemented PACE, a salary-based personal budgeting application that addresses the well-documented feast-and-famine monthly income cycle faced by Nigerian salary earners. Built as a single Next.js 16 codebase, with Firebase handling authentication and Firestore managing all financial data, and Paystack and Monnify handling payments and bank transfers, the system operationalized behavioral economics principles mental accounting, loss aversion, present bias, and nudge theory in a live application.

All three research objectives were met. The proactive financial management model was designed and implemented through the daily pace calculation and the Savings Vault rollover. The web application was developed in full, covering every feature specified in the design phase. The system was evaluated through a comprehensive, multi-layered testing programme that confirmed its functional accuracy, security, performance, and usability, with all defined success criteria achieved.

Usability testing revealed that PACE enabled its users to have more control over their finances and less end of the month anxieties. Despite having limitations, such as inability to capture salaries directly or invest, it is clear from this study that adopting a proactive, disbursement-oriented budgeting approach works both from a technical perspective and from the user's behavioral one. Further refinement of PACE based on the findings presented herein, such as the incorporation of bank accounts, interest-bearing vaults, automatic categorization of expenses, and long-term evaluation, would enable it to be a useful tool in managing personal finances.

Compliance with ethical standards

Disclosure of conflict of interest

We declare that there are no financial or non-financial conflicts of interest related to this research.

References

[1] Enhancing Financial Innovation & Access (EFINA). Access to Financial Services in Nigeria 2023 Survey. Lagos: EFINA; 2023.

[2] National Bureau of Statistics. Report on Personal Budgeting and Financial Behaviour in Nigeria. Abuja: NBS; 2024.

[3] Deloitte. The Deloitte Global 2022 Millennial and Gen Z Survey [Internet]. 2022 [cited 2025 Jan]. Available from: https://www.deloitte.com

[4] PwC. Nigeria FinTech Survey 2022: Harnessing the Opportunities. Lagos: PwC Nigeria; 2022.

[5] Gjertson L, Poduska J. Modern applications of traditional budgeting methods: A study of the digital envelope system. J Financ Couns Plan. 2021;32(2):198–212.

[6] Chukwu A, Ogbu S. From spreadsheets to smartphones: An analysis of the evolution of personal budgeting tools in Nigeria. Lagos J Econ Stud. 2022;25(4):88–103.

[7] Al-Sasi K. The psychology of digital payments: A study on the diminishing "pain of paying". J Behav Finance. 2023;14(1):112–128.

[8] Cai CW. Nudging the financial market? A review of the nudge theory. Account Finance. 2020;60(4):3341–3365.

[9] Hou L, Hsueh S-C, Zhang S. Digital payments and households' consumption: A mental accounting interpretation. Emerg Mark Finance Trade. 2021;57(7):2005–2020.

[10] Imai T, Rutter T, Camerer C. Meta-analysis of present-bias estimation using convex time budgets. Econ J. 2021;131(636):1788–1814.

[11] Chen L, Rodriguez M. Positive friction: A meta-analysis of interventions in financial applications to improve savings outcomes. Comput Human Behav. 2024;145:107752.

[12] Moyo T. A comparative analysis of PFM tools in Southern Africa: Gaps and opportunities. J South Afr Finance. 2021;4(2):78–91.

[13] Akinwunmi B, Ojo S. The cognitive burden of budgeting: How PFM automation influences financial behavior. Afr J Inf Syst. 2024;16(2):140–155.

[14] Adetoro L. User experience in Nigerian FinTech: A focus on transaction versus planning features. J Digit Commerce. 2022;8(2):45–59.

[15] Bello F, Raji M. Financial literacy is not enough: Self-control tools as mediators of savings behavior in Nigerian professionals. Niger J Financ Res. 2023;18(1):33–48.

[16] Okoro C. The feast and famine cycle: A qualitative study of the spending habits of Nigerian salary earners. J Consum Behav. 2023;22(5):1031–1042.