Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Design and Implementation of an Online Lost and Found Portal for the Babcock University Campus Community

Adelowo, Opeyemi Joshua
Department of information Technology, Babcock University

Ndukuba Daniel Elochukwu, Adebiyi Abdulmuiz Adedayo, Falaye Olaoluwa Michael
Department of Information Technology, School of Computing, Babcock University, Ilishan-Remo, Ogun State, Nigeria

Abstract **- This paper presents the design, development, and evaluation of an Online Lost-and-Found Portal for the Babcock University campus community. The existing manual process for managing lost items — relying on word-of-mouth, handwritten notices, and physical office visits — is inefficient, opaque, and unscalable for a growing university population. The proposed system is a web-based platform developed using React 18 with TypeScript on the frontend and Supabase as a Backend-as-a-Service (BaaS) platform providing a managed PostgreSQL database, JWT-based authentication, Row-Level Security (RLS), file storage, and real-time subscriptions. The Waterfall Software Development Life Cycle model was adopted. Key features include user registration, lost and found item reporting with image uploads, full-text keyword search, a claim and notification system, personalized user dashboards, and an administrator dashboard for claim verification. System evaluation encompassed functional testing (28/28 test cases passed), usability testing with 15 participants yielding a System Usability Scale (SUS) score of 83.4 (Excellent), and Google Lighthouse performance audits returning scores above 87 across all metrics with an average page load time of 1.3 seconds. Results confirm that professional-grade institutional web systems can be built using open-source and free-tier cloud tools at near-zero infrastructure cost, providing a replicable model for Nigerian universities.**

Keywords - Lost and Found; Web Portal; Babcock University; Supabase; React; PostgreSQL; Row-Level Security; Usability Testing; Campus Information System

I. INTRODUCTION

Universities are complex communities where thousands of students, academic staff, and administrative personnel interact daily across lecture halls, hostels, cafeterias, and administrative buildings. The constant movement of people and possessions makes the misplacement of personal belongings inevitable. Commonly lost items include student identity cards, mobile phones, wallets, flash drives, keys, laptops, and academic materials.

At Babcock University, the management of lost and found items has traditionally relied on manual processes. Found items are submitted to security offices or departmental secretaries, while owners must rely on word-of-mouth inquiries or physical notice boards to trace their property. These conventional methods are inefficient, time-consuming, lack proper documentation, and create administrative bottlenecks [1].

The growing influence of Information and Communication Technology (ICT) in modern education presents an opportunity to address these inefficiencies. According to Adebayo and Nwachukwu [4], institutions adopting ICT-based systems experience measurable improvements in reliability and service quality. A centralized digital portal would not only facilitate the return of misplaced items but also promote accountability and foster trust among campus users.

This study therefore, focuses on the design and implementation of an Online Lost-and-Found Portal tailored to the Babcock University campus community. The remainder of this paper is structured as follows: Section II reviews related literature; Section III describes the system design and methodology; Section IV presents the implementation and results; Section V discusses findings; and Section VI concludes with recommendations.

II. RELATED WORK

Several studies have explored digital lost and found systems in university environments, providing benchmarks that informed the design of this portal.

Tan and Chong [7] developed a hybrid web and mobile lost and found system for Universiti Kebangsaan Malaysia (UKM), replacing scattered paper-based logs across faculties. The platform introduced account-based login, item categorization, picture uploads, automatic deletion of records older than 30 days, and fraud-prevention security questions. User feedback during pilot testing confirmed reduced administrative workload and high interface satisfaction.

Ahmad, Ismail, and Hassan [8] implemented an RFID-enabled lost and found platform at Universiti Teknologi MARA (UiTM) Arau. Each recovered item was tagged with an RFID card linked to a database entry, and an automated email notification was sent to the owner. Response time between item submission and owner notification was reduced to under 10 minutes, demonstrating the viability of sensor-assisted automation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Jiang, Mao, and Kang [9] proposed a campus lost and found platform integrating Geographic Information System (GIS) digital map data and push notifications. Location-tagged reports improved recovery accuracy by more than 20% compared to conventional keyword-based search.

Mukisa, Nkurunziza, and Ssekibuule [11] developed an online platform at Makerere University using React.js, Node.js, and MongoDB, with role-based access control and an administrative analytics dashboard. User testing across three faculties confirmed high acceptance among students already accustomed to online portals.

Eze [13] demonstrated the feasibility of a web-based solution in a Nigerian tertiary context with an Automated Campus Lost and Found Management System at the University of Uyo, confirming that even low-budget institutions benefit from digitalization when user training is adequately provided.

While these studies collectively demonstrate a global progression toward digital lost and found systems, a gap remains in the Nigerian higher education context where modern Backend-as-a-Service (BaaS) platforms have not been applied to this problem domain. This study addresses that gap.

III. SYSTEM DESIGN AND METHODOLOGY

A. Research Design

The Waterfall Software Development Life Cycle (SDLC) model was adopted as the primary development methodology. This model was selected because the requirements were well-defined and stable from the outset, the sequential nature aligns with academic evaluation timelines, and the project was developed within a fixed academic semester [14]. The phases implemented were: requirements analysis, system and UI/UX design, implementation, testing, and deployment.

B. System Architecture

The system employs a three-tier architecture comprising a presentation layer, application logic layer, and database layer. An API-first approach was adopted for flexibility and potential mobile app integration in future iterations.

The frontend was developed using React 18 with TypeScript, providing a component-based architecture that promotes reusability and maintainability. Vite was adopted as the build tool, offering fast development server startup and hot module replacement. The UI stack includes Tailwind CSS for responsive layouts, React Hook Form with Zod for schema-based validation, React Router v6 for client-side routing, and shadcn/ui for accessible pre-built components.

Rather than a custom backend, the project adopted Supabase as a Backend-as-a-Service platform. Supabase provides a fully managed PostgreSQL database, built-in JWT-based authentication, real-time WebSocket subscriptions, file storage, and auto-generated RESTful APIs via PostgREST. Row-Level Security (RLS) policies were enforced directly at the database layer to ensure data ownership independent of the application layer.

C. Database Design

The logical data model consists of four core tables: profiles (extending Supabase's auth.users table through a foreign key), items (the central entity storing all lost and found reports), notifications (tracking in-app alerts), and admin_roles (controlling administrative privileges). The items table includes a generated TSVECTOR column for PostgreSQL full-text search, and GIN and B-tree indexes were created on the search_vector, status, and category columns to optimise query performance.

D. Security Design

Security was implemented in depth across all layers. At the database level, Row-Level Security policies govern all four tables: authenticated users may read all item records; users may only insert, update, or delete records they own; and administrators are granted broader update privileges through a security definer function that bypasses RLS to prevent infinite recursion. At the transport layer, HTTPS/TLS encryption is enforced with HSTS headers. At the application layer, passwords are hashed using bcrypt, inputs are validated using Zod schemas, and parameterized queries prevent SQL injection.

E. Requirements

Key functional requirements include: user registration with email verification; lost and found item reporting with image uploads; keyword and category-based item search; claim submission and real-time notification; a personal user dashboard; and an administrator dashboard for claim verification and resolution. Non-functional requirements specify page load times under 1.5 seconds on a campus network, 99.5% uptime during academic terms, GDPR-aware data handling, and modular maintainable code with documented APIs.

IV. IMPLEMENTATION AND RESULTS

A. Development Environment

The development environment was configured on a Windows 11 machine using Visual Studio Code. Node.js version 22 was installed to support the Vite development server. The project was initialised using the Vite React-TypeScript template. The Supabase JavaScript client library (@supabase/supabase-js) was configured to read environment variables at build time. Version control was established through Git with the repository hosted on GitHub, enabling continuous deployment through Netlify.

B. Core Modules

The Report Item module presents a multi-field form built with React Hook Form and Zod validation. On submission, item images are compressed client-side before upload to a Supabase Storage bucket, and the returned URL is stored alongside the report metadata. The Browse module fetches item records using PostgreSQL full-text search with the @@ operator against a plainto_tsquery expression, with additional category and status filter dropdowns. The Claim module triggers a status update and dispatches real-time notifications to the original reporter via Supabase Realtime WebSocket channels. The Administrator Dashboard, protected by both

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

RLS policies and a frontend route guard, presents a pending claims queue with resolve controls and aggregate analytics.

C. User Interface

The interface was designed around a dark navy and gold design system. The primary background is deep navy-black (#0A0B10) with gold accents (rgb(212, 170, 90)) for interactive elements and calls-to-action. Playfair Display is used for headings and Cabinet Grotesk for body text. The layout is fully responsive, adapting from single-column mobile to multi-column grid layouts using Tailwind CSS responsive breakpoints. Glass morphism effects and smooth CSS transitions were applied to elevate the visual quality.

D. Deployment

The portal was deployed on Netlify with continuous deployment configured via GitHub integration. Every push to the main branch automatically triggers a production build using the Vite pipeline, served from Netlify's global CDN. A _redirects file was configured to route all paths to index.html, enabling React Router's client-side navigation. Environment variables are stored in Netlify's secure configuration and are never committed to the repository.

E. Functional Testing

Functional testing was performed manually by executing 28 defined test cases against the deployed application covering the authentication, item reporting, browse and search, claim and notification, and administrator modules. All 28 test cases passed on the final build. Table I presents a sample of the test cases executed.

TABLE I. Sample Functional Test Cases

TC#	Test Case	Expected Result	Status
TC01	Register with valid email and password	Account created, verification email sent	PASS
TC02	Login with incorrect password	Error message displayed, access denied	PASS
TC03	Report lost item with all required fields	Item saved to DB, appears on Dashboard	PASS
TC04	Submit report with missing required field	Validation error shown, form not submitted	PASS
TC05	Upload image with item report	Image stored in Supabase Storage, URL saved	PASS
TC06	Search items by keyword	Matching items returned in real time	PASS
TC07	Non-admin user accesses admin dashboard URL	Redirected to Dashboard, access denied	PASS
TC08	Admin marks claimed item as resolved	Status updated, both users notified	PASS

F. Usability Testing

Usability testing was conducted with 15 participants drawn from Babcock University students and staff, representing the primary user personas. Participants completed five core tasks without assistance: registering an account, reporting a lost item with an image, searching for an item by keyword, submitting a claim, and logging out. Table II summarises the results.

TABLE II. Usability Testing Results Summary

Task	Completion Rate	Avg. Time (s)	Satisfaction (1-5)
Register an account	100%	48	4.7
Report a lost item with image	93%	112	4.4
Search for item by keyword	100%	22	4.8
Submit a claim on found item	87%	35	4.3
Log out of the system	100%	8	4.9

The overall System Usability Scale (SUS) score was 83.4 out of 100, which falls within the 'Excellent' range according to Bangor, Kortum, and Miller's SUS grading scale [15]. The two participants who did not complete the item claim task reported confusion between the 'Report Found Item' and 'Claim' functions, identifying a labelling improvement for future iterations.

G. Performance Testing

Performance testing was conducted using Google Lighthouse against the deployed Netlify URL [17]. Results are presented in Table III.

TABLE III. Google Lighthouse Performance Audit Results

Metric	Score	Interpretation
Performance	87/100	Fast initial load; Vite code-splitting effective
Accessibility	91/100	ARIA labels and contrast ratios largely met
Best Practices	95/100	HTTPS enforced, no deprecated APIs used
SEO	89/100	Meta tags and semantic HTML correctly applied

Page load time on a standard campus network averaged 1.3 seconds for the initial load, within the 1.5-second non-functional requirement. Subsequent navigations were near-instantaneous due to React Router's client-side routing.

V. DISCUSSION

The results of this study demonstrate four significant findings with implications for both the specific institutional context of Babcock University and the broader domain of campus information systems in developing economies.

First, the successful development and deployment of a fully functional web portal confirms that Nigerian universities can implement professional-grade digital administrative systems

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

without large infrastructure investments. By leveraging Supabase and Netlify, the total infrastructure cost is near zero for moderate traffic volumes, challenging the assumption that digital transformation requires significant capital expenditure.

Second, the adoption of Supabase as a BaaS platform proved highly effective for an academic project context. Managed PostgreSQL, built-in authentication, RLS, and file storage eliminated the need for a custom server-side application while delivering enterprise-grade security. This finding has practical implications for future student developers and institutional system developments at resource-constrained universities across sub-Saharan Africa.

Third, the SUS score of 83.4 confirms that the user-centred design process produced an interface aligned with real user expectations and mental models. The two tasks with completion rates below 100% — specifically the item claim task at 87% — point to labelling ambiguities that represent clear targets for iterative improvement in subsequent versions.

Fourth, the implementation of Row-Level Security policies at the PostgreSQL layer, rather than relying solely on frontend access controls, proved essential for data integrity. Testing confirmed that direct API requests attempting to access or modify another user's data were correctly rejected by the RLS policies, even when frontend route guards were bypassed, reinforcing the principle of defence-in-depth security.

A challenge encountered during implementation involved RLS policy infinite recursion: the admin check queried a table also governed by RLS, causing a circular dependency. This was resolved by creating a SECURITY DEFINER function to perform the admin lookup outside the RLS context. A secondary challenge involved Netlify returning 404 errors for non-root routes, resolved by adding a _redirects file routing all paths to index.html.

VI. CONCLUSION

This study successfully achieved its stated aim of designing, developing, and evaluating an Online Lost-and-Found Portal for Babcock University. All three specific objectives were met in full: a conceptual and structural framework was designed; a functional and interactive digital system was developed and deployed; and the system's performance, usability, and effectiveness were systematically evaluated.

The portal addresses the core deficiencies of the existing manual process by providing a centralized platform for reporting, searching, and claiming lost and found items with real-time notifications and transparent administrative oversight. The excellent usability score, 100% functional test pass rate, and strong Lighthouse performance metrics collectively demonstrate that the portal is ready for pilot deployment within the Babcock University community.

Future enhancements recommended include: a React Native mobile application sharing the same Supabase backend; an AI-powered image matching feature using machine learning similarity models; SMS push notifications via Africa's Talking or Twilio; QR code tagging for high-value items; and integration with Babcock University's existing student identity management system for automatic account provisioning. A longitudinal study measuring actual item recovery rates before and after portal deployment would provide empirical evidence of the system's real-world impact.

ACKNOWLEDGMENT

The authors thank Mr. Joshua Adelowo O. (Project Supervisor), the Department of Information Technology, School of Computing, Babcock University, and all students and staff who participated in usability testing for their invaluable contributions to this research.

REFERENCES

[1] A. Adeyemi, "Digital inefficiencies in Nigerian tertiary institutions," J. Educ. Technol. Dev., vol. 5, no. 2, pp. 33-42, 2020.

[2] K. Yusuf and L. Olatunji, "Manual administrative systems and their challenges in higher institutions," Nig. J. Manage. Stud., vol. 8, no. 3, pp. 55-67, 2021.

[3] T. Adebayo, "ICT adoption and transformation in Nigerian universities," Afr. J. Comput. Inf. Syst., vol. 9, no. 4, pp. 88-102, 2020.

[4] T. Adebayo and C. Nwachukwu, "Integrating technology in higher education: A case study of Nigerian universities," Afr. J. Comput. Inf. Syst., vol. 10, no. 3, pp. 120-132, 2022.

[5] S. Bello, "Administrative inefficiencies and record management in tertiary institutions," Nig. J. Educ. Admin., vol. 12, no. 1, pp. 14-26, 2020.

[6] N. Okafor, "Digital innovation in university administration: Challenges and opportunities," Int. Rev. Educ. Technol., vol. 11, no. 4, pp. 59-72, 2022.

[7] M. C. Tan and S. Y. Chong, "Effective lost and found system for university campus," International Journal of Computer Applications, vol. 180, no. 12, pp. 1-6, 2018.

[8] N. Ahmad, A. Ismail, and R. Hassan, "RFID-based lost and found system for campus management," J. Comput. Sci. Inf. Technol., vol. 6, no. 2, pp. 45-58, 2019.

[9] Y. Jiang, W. Mao, and L. Kang, "Campus lost and found platform based on digital map data," IEEE Access, vol. 7, pp. 112345-112353, 2019.

[10] P. Rathnayaka, "Mobile-based campus lost and found application with location services," Int. J. Mob. Comput. Multimed. Commun., vol. 11, no. 3, pp. 33-47, 2020.

[11] R. Mukisa, J. Nkurunziza, and B. Ssekibuule, "Online lost and found system for university campuses," African J. Comput. Inf. Syst., vol. 5, no. 1, pp. 12-24, 2021.

[12] Z. Liu, H. Wang, and X. Chen, "Lost-Net: Deep learning for lost item image matching on campus platforms," Pattern Recognit. Lett., vol. 145, pp. 89-97, 2021.

[13] A. Eze, "Automated campus lost and found management system: A case study of University of Uyo," Nigerian Journal of Technology, vol. 39, no. 2, pp. 412-419, 2020.

[14] I. Sommerville, Software Engineering, 11th ed. London: Pearson Education, 2020.

[15] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," Journal of Usability Studies, vol. 4, no. 3, pp. 114-123, 2009.

[16] Supabase Inc., "Supabase Documentation: Row Level Security," 2024. [Online]. Available: https://supabase.com/docs/guides/auth/row-level-security. [Accessed: Mar. 2026].

[17] Netlify Inc., "Netlify Documentation: Redirects and rewrites," 2024. [Online]. Available: https://docs.netlify.com/routing/redirects. [Accessed: Mar. 2026].