

DESIGN AND DEVELOPMENT OF RESORTSYNC: A SMART RESORT MANAGEMENT SYSTEM WITH AI-POWERED BUSINESS INTELLIGENCE

Mohan Rameshwar Barde
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
bardemohan08@gmail.com

Khushi Umanath Sharma
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
khushisharma9124@gmail.com

Ankit Manoj Raut
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
rautankit559@gmail.com

Krupa Ganesh Lonare
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
krupalonare07@gmail.com

Meher Ishwar Gautam
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
gautammeher27@gmail.com

Harsh Ambadas Nawale
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati.
Maharashtra, India
harshnawale8@gmail.com

Prof. Mayur S. Burange
Computer Science and Engineering
P. R. Pote Patil College of Engineering and Management Amravati. (An Autonomous Institute)
Maharashtra, India
msburange@prpoteatilengg.ac.in

ABSTRACT—Running a resort day to day means keeping track of bookings, rooms, guests, payments, housekeeping, staff, and inventory all at once, and doing it without a shared system tends to produce exactly the errors one would expect: double-booked rooms, misplaced records, and financial figures nobody can fully trust. ResortSync brings all of these functions authentication, resort and room management, customer and booking handling, check-in/check-out, payments, housekeeping, staff and inventory tracking, invoicing, dashboards, and reporting into one platform, with an AI-driven Business Brain layer added on top to turn operational history into revenue and turnover insights. Access is controlled through JWT-based authentication and Role-Based Access Control (RBAC), so each user sees only what their role needs. The system was built module by module under the Triroop Engineering System (TES), a vertical-slice methodology covering design, frontend and backend implementation, API integration, security, and testing throughout. The result is a platform aimed squarely at improving booking accuracy, cross-department coordination, day-to-day efficiency, and the financial visibility resort owners actually need to make decisions.

Keywords: *Resort Management, Cloud Computing, Business Intelligence, Booking Management, REST API, Role-Based Access Control, Project Monitoring.*

I. INTRODUCTION

A. Background

A resort's daily operation touches many moving parts at once — reservations coming in, rooms being assigned and cleaned, guests checking in and out, bills being settled, and staff coordinating across shifts. When these activities are tracked separately, on paper or across unrelated tools, small errors compound quickly: a room gets double-booked, a payment goes unrecorded, or a housekeeping request is missed. As the number of bookings grows, so does the cost of not having one shared source of truth for the property.

B. Why a Centralized Digital Platform Helps

Bringing these activities onto a single digital platform changes the picture. Front-desk staff can see room status the moment it changes, accountants can pull payment records without chasing paper trails, and management can view occupancy and revenue as they happen rather than at the end of the month. Because bookings, rooms, guests, and finances are all connected in practice, keeping them connected in software is what actually removes the friction — not simply digitizing each activity on its own.

C. What Goes Wrong Without It

In the absence of such a platform, resorts commonly run into a familiar set of problems: two

reservations landing on the same room, guest histories that live in someone's notebook instead of a shared record, housekeeping updates that reach the front desk late, and financial records that are scattered across registers and messaging apps. These are documented failure patterns in hotel and resort management studies as well [1], [6], not exotic edge cases — they are the ordinary consequence of coordinating a multi-department operation without a shared system, and they are exactly what a role-aware, real-time platform is meant to remove.

D. Proposed ResortSync

ResortSync is our response to this gap: a single platform that carries a booking from the moment a guest reserves a room through check-in, service, payment, and check-out, with an analytics layer sitting on top of the same data.

Workflow: Customer → Booking → Room Assignment → Check-In → Housekeeping/Service → Payment → Check-Out → Dashboard & Reports

Every user who touches the system — admin, reception, accountant, or housekeeping staff — signs in through JWT-based authentication and sees only what their role permits. The goal is not just to move paperwork onto a screen, but to let the resort make faster, better-informed decisions from data it already generates.

II. PROBLEM STATEMENT

A resort's booking, room, guest, payment, housekeeping, staffing, and inventory records are not independent — a change in one usually needs to be reflected in another almost immediately. When these are maintained separately, the natural result is double bookings, mismatched payment records, slow handoffs between departments, and an owner who cannot say, on any given day, exactly how the business is performing.

Most off-the-shelf hospitality software solves a narrow slice of this problem — a reservation calendar here, a billing tool there — without connecting the full chain from booking to housekeeping to finance, a limitation seen across several single-function reservation and hotel-management platforms in prior work [1], [2], [5]. What is missing is a system that treats the resort as one connected workflow, with access controlled by role rather than left open to everyone.

ResortSync is built to close that gap: one platform, one shared dataset, and an analytics layer that reads directly from live operational data instead of a separate reporting pipeline bolted on afterward.

III.

OBJECTIVES

This work sets out to build ResortSync as a single, secure environment for running resort operations end to end. Specifically, the project aims to:

1. Build a modular Resort Management System that covers the full resort operations lifecycle in one platform.
2. Bring authentication, resort, room, customer, booking, check-in/check-out, payment, housekeeping, staff, inventory, invoice, dashboard, and reporting functions under a single connected system.
3. Secure the system with JWT-based authentication and Role-Based Access Control (RBAC) so each user only reaches what their role requires.
4. Give managers real-time dashboards and reports covering bookings, occupancy, and revenue instead of end-of-period summaries.
5. Structure the backend as independent, vertically-sliced modules with concurrency control that rules out double bookings by design.
6. Add a Business Brain analytics layer that turns the system's own operational history into turnover trends, revenue projections, and other business insights.

IV.

LITERATURE REVIEW

A modular reservation and asset-scheduling platform built by Murin et al. [1] standardized administrative workflows for booking-heavy operations, though it was built purely for the web, offered no mobile access, and left out both database-level locking and any analytics capability. Acharya [2] took a step closer to hospitality specifically, building a digital reservation and resort-management tool that centralized booking data, but the work stopped short of forecasting or deeper business analysis. On the forecasting side, Lee and Kim [3] showed that ARIMA and LSTM models could predict hotel occupancy and revenue with over 90 percent accuracy — a strong result, but one that relied on a separate, costly data warehouse rather than plugging into the software resorts actually run day to day. Silva and Zhang [4] came closer to a full-stack answer, pairing Flutter with Node.js for eco-resorts and cutting development overhead by roughly 40 percent, but their choice of loose NoSQL storage led to billing inconsistencies, and the system had no type-safe way to keep relational data in sync.

Read together, these four efforts each solve one piece of the puzzle — scheduling, centralization, forecasting, or cross-platform delivery — but none combine all of them in one system. More recent hospitality-focused work follows a similar pattern: Lv [5] proposed an SOA-based hotel reservation platform that centralized channel

management but remained web-only with no mobile support, Pathak et al. [6] reviewed hotel management system design without extending into predictive or business analytics, and Zhong [7] applied big-data techniques to hotel management while keeping the analytics layer separate from day-to-day booking operations. ResortSync's contribution is to fold a mobile-capable Flutter client, a type-safe Prisma/PostgreSQL data layer with proper concurrency handling, and an analytics module that reads from the same live operational database into a single platform, rather than treating each of these as a separate concern.

V. PROPOSED METHODOLOGY

A. Proposed System Overview

At its core, ResortSync is one connected pipeline covering authentication, resort and room setup, customer and booking records, check-in/check-out, payments, housekeeping, staff and inventory tracking, invoicing, and reporting — all sitting behind a single role-based access layer so that no data has to be re-entered or reconciled between tools.

Workflow: Customer/Guest → Booking → Room Assignment → Check-In → Housekeeping/Service → Payment → Check-Out → Dashboard & Reports

B. System Architecture

The system is built to the Tiroop Engineering System (TES) and splits into four layers: a Flutter-based Presentation Layer for role-aware interaction, an API and Security Layer handling JWT authentication, refresh tokens, and Zod-based request validation, a Business Logic Layer holding the system's functional modules, and a Data Layer that talks to a Supabase PostgreSQL database through Prisma ORM. The Flutter and Node.js pairing follows the cross-platform direction explored by Silva and Zhang [4], with a type-safe Prisma/PostgreSQL data layer added specifically to avoid the billing inconsistencies their loosely-typed NoSQL approach produced.

Architecture Flow: Users → Flutter Presentation Layer → REST API (/api/v1) → Authentication Middleware → Business Modules → Prisma ORM → Supabase PostgreSQL Database

Within the backend, each module is itself split into API, logic, contract, and data sub-layers — the TES vertical-slice principle, under which a module is only considered done once it works end to end on its own. That discipline is what makes centralized data sharing, live room-availability checks, and role-specific access hold together as the system grows

C. System Modules

Fifteen modules make up the platform: authentication, resort management, room management, customer management, user management, booking management, check-in/check-out, payment management, housekeeping, staff management, inventory management, invoicing, dashboards, reports, and the Business Brain analytics module. Together they carry a guest from first booking through checkout, service delivery, and the business reporting that follows. The Business Brain module's forecasting role is informed by the strong occupancy and revenue prediction accuracy reported for time-series models in prior hospitality analytics work [3], applied here directly on ResortSync's own operational data rather than a separate warehouse.

Overall Workflow: Customer → Booking → Room Assignment → Check-In → Housekeeping/Service → Payment → Check-Out → Dashboard → Reports & Business Brain

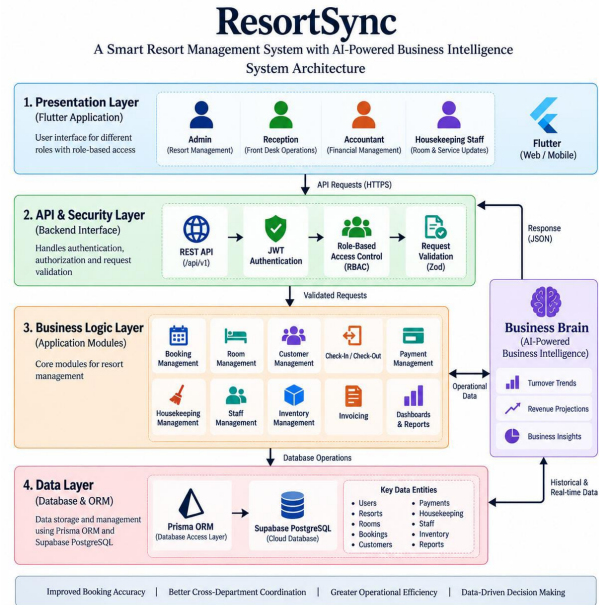


Fig. 1. Layered System Architecture of ResortSync

Fig. V.1 Layered System Architecture of ResortSync

VI. RESULTS AND DISCUSSION

A. Results

ResortSync, as built, brings authentication and role-based access, resort and room setup, guest records, conflict-free booking and room allocation, check-in/check-out handling, payment recording and invoicing, housekeeping and staff coordination, inventory tracking, and centralized dashboards and reporting into one running system. The Business Brain module reads from this same operational data to surface turnover trends and revenue-related insights, rather than depending on a separate export-and-

analyze step. Flutter, Node.js, REST APIs, Prisma ORM, and Supabase PostgreSQL together handle the frontend, backend logic, and data storage as one integrated stack.

B. Discussion

The vertical-slice, TES-based structure has held up well as a way to keep a large system maintainable — each module can be reasoned about and modified largely on its own. Tying booking, room, customer, payment, and housekeeping data together in one database is what actually removes the duplicate data entry and cross-checking that fragmented systems require, while JWT authentication and RBAC keep each role's access limited to what it needs. Because bookings pass through a single concurrency-controlled layer, two staff members can never accidentally assign the same room to two guests. The same shared database is also what lets the Business Brain module produce insights straight from live activity, instead of the disconnected, after-the-fact analytics pipelines seen in comparable forecasting-focused systems in the literature [3], [7].

VII. LIMITATIONS

A few practical constraints are worth stating plainly, since they shape where the system can be deployed today and where it will need further work.

A. Cloud and Network Dependency

Because ResortSync runs on Supabase PostgreSQL and communicates over REST APIs hosted on Vercel, its availability is tied to network connectivity and third-party cloud uptime rather than to a fully offline, on-premise setup.

B. Access Control Granularity

Role-based access is currently defined at the module level. Larger properties with more specialized staff structures may eventually need finer, action-level permissions within a module rather than module-wide access alone.

C. Scalability and Performance

The system has not yet been stress-tested against very high concurrent booking volumes or multi-resort data loads. Load testing, concurrency stress testing, and query tuning for larger deployments remain planned next steps.

D. Payment Gateway Integration

Payments are currently recorded and reconciled inside the system itself rather than processed through a live third-party payment gateway, which is left for a future integration phase.

VIII.

CONCLUSION

ResortSync was developed to bring the major activities of resort management into a single, connected platform. The system integrates booking, room management, customer records, check-in/check-out, payments, housekeeping, staff, inventory, invoicing, and reporting, reducing the need for separate tools and repeated data entry. With JWT-based authentication and Role-Based Access Control, the platform also ensures that users can access the functions relevant to their responsibilities.

An important part of ResortSync is the Business Brain analytics layer, which uses the system's operational data to provide turnover trends, revenue-related insights, and business information for management. The TES-based modular approach also helped in developing the system in manageable vertical slices while keeping the frontend, backend, security, and database components connected.

Overall, ResortSync provides a practical foundation for more organized and data-driven resort management. While the current system has limitations such as cloud dependency, scalability testing, and the absence of live payment gateway integration, these areas provide clear opportunities for future improvement. With further testing and enhancement, the system can be extended to support larger and more demanding resort operations.

IX.

REFERENCES

- [1] M. Murin, F. Jakab, and M. Michalko, "Advanced Reservation System," in 2021 International Conference on Emerging eLearning Technologies and Applications (ICETA), 2021, doi: 10.1109/ICETA54173.2021.9726673.
- [2] K. Acharya, "Resort Management and Reservation System Project," IEEE Dataport, Oct. 31, 2024, doi: 10.21227/v909-e497.
- [3] H. Lee and J. Kim, "Time-Series Predictive Modeling for Automated Revenue Management and Occupancy Forecasting in Independent Lodgings," IEEE Transactions on Hospitality and Tourism Systems, vol. 4, no. 2, pp. 115-124, Jun. 2025, doi: 10.1109/THTS.2025.3409876.
- [4] J. Silva and L. Zhang, "Evaluating Cross-Platform Flutter Architectures for Real-Time Service Coordination in Mid-Scale Hospitality Environments," in Proceedings of the International Conference on Software Engineering and Mobile Applications (ICSEMA), pp. 84-91, March 2024.
- [5] X. Lv, "Design and Implementation of Hotel Reservation Management Platform Based on SOA Framework," in 2021 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Xi'an, China, 2021, pp. 517-520, doi: 10.1109/ICITBS53129.2021.00132.

[6] A. Pathak, A. Snghal, and B. K. Rana, "Review on Hotel Management System," in 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 1834-1837, doi: 10.1109/ICAC3N53548.2021.9725658.

[7] Z. Zhong, "Implementation of Hotel Intelligent Management System Based on Big Data," in 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 2022, pp. 1074-1078, doi: 10.1109/IPEC54454.2022.9777524.