Akinmerese Oluwatobi, 2026, 14:1
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

# Design and Implementation of an Online Hostel Space Booking

**Akinmerese Oluwatobi, Ifekandu Chiamaka, Obibi Kevin**
Department of Mathematical Sciences, Augustine University, Ilara-Epe, Lagos State, Nigeria.

**Abstract- The traditional hostel allocation process in many Nigerian private universities has long been marred by inefficiencies, lack of transparency, and administrative bottlenecks. This project presents the design and implementation of an Online Hostel Allocation System aimed at digitizing and streamlining the process of student accommodation. The system was built using a robust technology stack comprising Laravel for back-end logic, Vue.js for front-end interactivity, Inertia.js as middleware, Tailwind CSS for styling, and Docker for containerization. The development followed a structured methodology that included requirements gathering, system modeling, implementation, testing, and evaluation. Key modules developed include student registration, hostel browsing, intelligent roommate pairing, secure payment integration, and an administrative dashboard. Each module was designed with user experience, scalability, and data security in mind. Testing and feedback from both students and administrative staff confirmed the system's reliability, ease of use, and potential for wide-scale deployment. It eliminates the need for physical queues and manual paperwork while providing a fairer and more transparent allocation process. The project contributes significantly to the growing need for digital transformation in higher education by offering a scalable and user-centered solution to hostel allocation challenges. This work concludes with practical recommendations for expanding the system's functionality, including mobile app development, multi-language support, and integration with existing university portals. Ultimately, the system enhances the accommodation experience for students and simplifies hostel management for institutions, offering a modern solution to an age-old problem.**

**Keywords: online hostel allocation system, digital transformation, transparency, laravel, streamlining, higher education.**

## I. INTRODUCTION

Hostel accommodation is a critical component of university life, providing students with a stable and secure environment that supports academic excellence and social development. As student populations continue to grow across Nigerian universities, the process of allocating hostel spaces has become increasingly complex and inefficient. Augustine University, like many institutions, still relies on outdated manual methods—paper-based systems and basic spreadsheets—which are no longer adequate for meeting the needs of a digitally literate student body(Nwachukwu & Nwoke, 2020;Okoro &Agwu, 2018).

Traditional allocation systems are not only slow and tedious but also vulnerable to administrative errors, delays, and transparency issues. Students are frequently left uninformed about room availability, allocation status, or payment processes, while administrators are burdened with repetitive tasks and disjointed data records(Oladipo & Salami, 2019). With the advent of modern software technologies, universities now have the opportunity to reimagine student services through digital transformation. E-commerce platforms, widely used in retail, travel, and real estate, have introduced speed, automation, and real-time data access—capabilities that can significantly enhance hostel allocation systems(Laudon & Laudon, 2020; World Bank, 2019).

This project proposes the development of a robust, user-friendly web-based platform using Laravel (for backend development), Vue.js (for frontend interactivity), Inertia.js (for seamless communication between server and client), and Tailwind CSS (for efficient interface design). The system will be containerized using Docker, ensuring a consistent and scalable deployment environment. The platform will allow students to search, book, and pay for hostel spaces online, while giving administrators powerful tools to manage bookings, track payments,

and communicate with students—all through a unified interface (Laravel, 2024; Tailwind CSS, 2024; Docker Inc., 2024).

## STATEMENT OF THE PROBLEM

The current hostel allocation method is plagued by several critical issues:

1. Delays and Inefficiencies: Manual record-keeping and in-person submission of forms slow down the allocation process and create long queues, especially during resumption periods.
2. Lack of Transparency: Students are often uninformed about available rooms, allocation criteria, and booking status, leading to confusion and dissatisfaction.
3. Room Assignment Conflicts: Limited hostel spaces combined with an inconsistent allocation system lead to disputes among students regarding room ownership.
4. Uncoordinated Payment Verification: Payments made through bank transfers or third-party channels are hard to verify and match with individual students, delaying confirmation of hostel bookings.
5. Inability to Accommodate Preferences: Students cannot choose rooms based on personal preferences such as location, cost, gender compatibility, or amenities, resulting in frustration and dissatisfaction.
6. These challenges highlight the urgent need for a centralized, automated system that enhances efficiency, transparency, and user control in hostel management.

## AIM AND OBJECTIVES OF THE STUDY

The main aim of this study is to design and develop an e-commerce-based hostel booking and allocation platform for Augustine University. The system will automate the entire hostel booking lifecycle—from room listing and selection to payment and confirmation.

### The specific objectives include:

1. To identify existing allocation systems in Nigeria universities and evaluate their limitations in terms of transparency, fairness, efficiency, and user accessibility
2. To design an interactive hostel listing module where students can view available rooms along with details such as cost, amenities, and type.
3. To implement advanced search and filter options, allowing students to refine room selection based on their preferences (e.g., price range, gender, location, roommate compatibility).
4. To evaluate the platform's performance, usability, and efficiency through user testing and stakeholders feedback

### Methodology Overview

This project will adopt a structured software development approach, involving the following key stages:

1. Requirements Gathering: Identifying the challenges of the current hostel allocation system through interviews with students and administrative staff to determine user needs.
2. System Design: Developing architectural models, data flow diagrams, wireframes, and a scalable database schema to support booking, payments, and room listings.
3. Technology Stack and Development
4. Laravel will be used to build the backend logic, including user authentication, booking operations, and payment processing. Vue.js will power the frontend, ensuring responsive and dynamic user interfaces. Inertia.js will serve as the bridge between Laravel and Vue, enabling server-side routing while delivering a single-page application (SPA) experience. Tailwind CSS will be used to design a modern, mobile-responsive interface with clean layouts and user-friendly components. Docker will be employed to containerize the application for consistent local development, staging, and production deployments.
5. Testing: Conducting unit testing, integration testing, and user acceptance testing (UAT) to ensure the system is functional, secure, and meets stakeholder expectations.
6. Deployment and Maintenance: Deploying the system to a secure web server using Docker containers and providing ongoing maintenance, bug fixes, and feature updates.

**SIGNIFICANCE OF THE STUDY**

The introduction of a digital hostel booking platform will offer several advantages to students, university staff, and the institution as a whole:

1. **For Students:** Offers a faster, easier, and more transparent way to book accommodation, reducing stress and uncertainty during registration.
2. **For Administrators:** Simplifies daily hostel operations, reduces paperwork, and provides tools for better tracking and decision-making.
3. **For the University:** Enhances the image of the institution by adopting modern technology, improving service delivery, and increasing operational efficiency.
4. **For Future Developers and Researchers:** Provides a scalable digital model that can be extended or replicated in other higher education institutions.

**SCOPE AND LIMITATION OF THE STUDY**

This study is focused on the development of a web-based hostel management system for Augustine University. The scope includes:

A user-friendly hostel search and booking portal for students. Real-time room availability display and instant booking confirmation. Integrated payment gateway for online transactions. Backend dashboard for administrators to manage listings, students, and payments. Search and filter tools based on preferences like cost, gender, or amenities.

Limitation:

1. The system does not cover private or off-campus accommodation.
2. It does not include mobile app development, although the web platform will be mobile responsive.
3. Hostel management aspects such as maintenance, cleaning, or student discipline are beyond the system's scope.

**DEFINITION OF TERMS**

1. **E-commerce:** A digital environment for conducting commercial transactions online, including booking and payment for services.
2. **Hostel Allocation:** The structured process of assigning students to hostel rooms, usually based on application order, availability, and institutional policies.
3. **Laravel:** A popular PHP web framework that provides tools and libraries for developing secure, scalable, and database-driven web applications.
4. **Vue.js:** A modern JavaScript framework for building reactive user interfaces, especially useful for creating single-page applications.
5. **Inertia.js:** A framework that connects Laravel and Vue.js to allow server-driven routing and stateful single-page applications without traditional API calls.
6. **Tailwind CSS:** A utility-first CSS framework used to build visually appealing and responsive user interfaces quickly by applying predefined classes directly in HTML.
7. **Docker:** A platform used to package applications into containers that contain all necessary dependencies, ensuring consistent performance across different environments and simplifying deployment.
8. **Payment Gateway:** A secure online service that processes financial transactions between users and businesses, supporting credit cards, bank transfers, or mobile money.
9. **Booking System:** A web application that enables users to reserve services or facilities, such as rooms or tickets, often with built-in features for scheduling and payments.
10. **Administrative Dashboard:** A backend interface that provides tools for managing, monitoring, and updating data related to users, transactions, and content.
11. **Search and Filtering:** A system feature that helps users find specific content or listings by narrowing down results based on predefined criteria like location, price, or amenities.

## II. REVIEW OF RELATED LITERATURE

**Introduction**

For many incoming students, securing a hostel bed feels like navigating a maze: deadlines, waiting lists, and uncertainty loom large. In Nigerian private universities, this labyrinth often begins weeks before term starts, with long queues, manual forms, and last-minute shuffles. But what if the process could be

more like booking a flight—streamlined, transparent, and even personalized? Leveraging modern web frameworks, mobile-first design, and intelligent matching, an Online Hostel Allocation System can transform this stressful ritual into an efficient, student-friendly experience. In this chapter, we dive into relevant studies, campus examples, and emerging technologies that can guide such a transformation.

**Why Fair Allocation Matters**

Imagine two students with similar academic standing: one gets a window-side room, the other ends up tucked away in a corner. Perceptions of unfairness can erode trust and campus morale (Smith & Jones, 2018). Research highlights several critical impacts:

1. **Community Cohesion:** When students see allocation rules applied consistently—based on factors like year of study, academic performance, or roommate requests—they're more likely to feel part of a supportive community (Nguyen & Brown, 2014).
2. **Academic Performance:** A comfortable, preferred living environment correlates with improved concentration and grades, as students worry less about their accommodation (Olanrewaju et al., 2016).
3. **Administrative Efficiency:** By eliminating the need for manual appeals and reorganizations, clear criteria save staff time and avoid bottlenecks during busy times of the year (García & Pérez, 2018).

**Embracing Digital Transformations**

Higher education has rapidly shifted from paper archives to digital platforms. Let's explore three common approaches:

1. **Full ERP Integration:** Systems like PeopleSoft or SAP bundle housing with finance, academics, and HR, offering unified data and single-sign-on (Al Mashari, 2002; Smith & Kumar, 2020). Their strength lies in comprehensive reporting, but they often require significant investment and lengthy customization.
2. **Specialized Open-Source/SaaS Solutions:** Universities may implement services like waitlists, reporting dashboards, and booking calendars without completely rebuilding entire ecosystems thanks to platforms designed specifically for student housing (Chen, Lee, & Wang, 2021).
3. **Progressive Web Apps (PWAs):** With PWAs, students experience app-like responsiveness and offline capabilities through a browser. Push notifications for allocation updates, smooth cross-device usability, and lower development overhead when compared to native apps are among of the main benefits (Eze & Bello, 2021).

TABLES 2.4 REVIEW OF RELATED WORK

| Serial Number | Author/Year | Title Of Work | Objectives | Methodology | Result | Gaps |
|---|---|---|---|---|---|---|
| 1 | Okeke, Umeh, and Danjuma | Caleb University Roommate Grouping Study | To improve cohabitation by allowing group-based roommate selection | Pre-formed group matching and randomization algorithm; paper-based reassignment process | Preserved social groupings; introduced randomized matching in high demand seasons | Reassignment appeals managed manually; lacks full digital workflow or automated decision auditing |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | Akademic (2023) | UNIZIK SSADM Allocation System | To apply rules-based logic for fair allocation | SSADM used academic and demographic criteria | Rules-enforce consistency in allocation; heat-maps and capacity alerts for administrator | Manuel data entry still present; lacks user interface personalization and feedback handling |
| 3 | Odili & Obiunu(2022) | Anchor University Hostel Portal | To display room availability and reduce manual room allocation processes | Web portal built with PHP and MySQL; backend logic to prevent double-booking | Real-time room visibility; automatic conflict detection; students use self- service dashboards | No advanced matching or algorithm-based algorithm; limited preference handling |

## III. RESEARCH METHODOLOGY

**Research Design**

Design and Development Research: This project adopts a design and development research methodology to ensure the final product is aligned with real-world needs and challenges in hostel allocation. The methodology involves a systematic process of identifying existing problems, gathering data from stakeholders, and iteratively designing and refining a solution. The approach emphasizes practicality and usability. After each development cycle, feedback will be collected from users to inform necessary improvements. This loop continues until the system meets the functional, non-functional, and institutional requirements of the university environment.

**Requirements Gathering**

1. **Stakeholder Interviews:** In-depth, semi-structured interviews will be conducted with key university stakeholders such as students (particularly freshers), hostel wardens, finance officers, and IT personnel. These sessions aim to uncover firsthand experiences with the existing manual system and reveal areas where digital intervention can add value. Questions will address allocation fairness, delays, transparency, and user satisfaction.

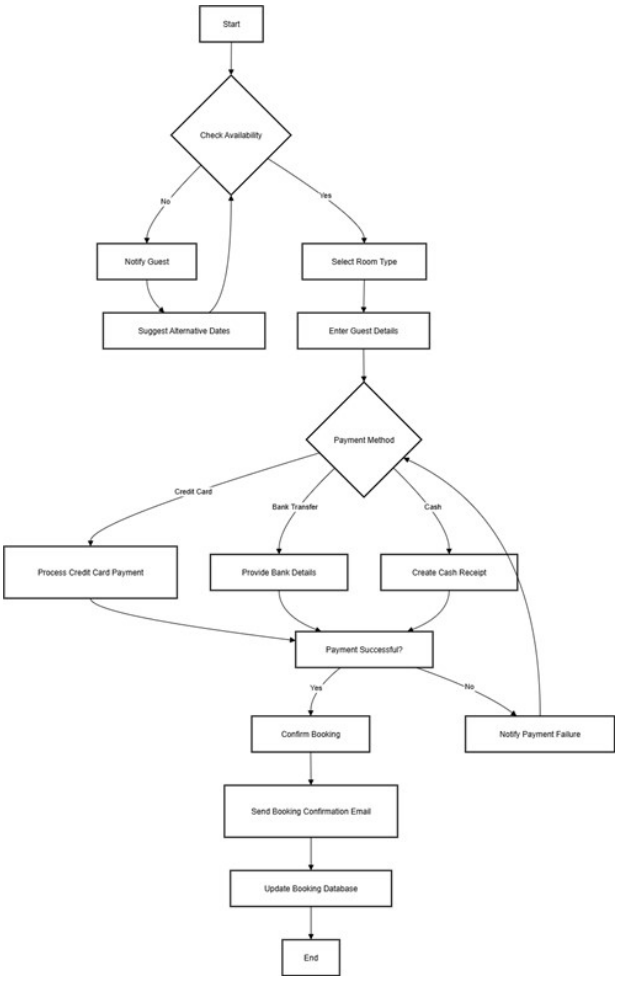


Fig 3.2 Manual Allocation Workflow

2. **Surveys:** Online surveys will be distributed via official university platforms to collect a wide

range of student opinions. The surveys will include both closed and open-ended questions about hostel preference factors, previous experiences, satisfaction with current processes, and expectations from a digital allocation system. This data will be analyzed to identify trends, preferences, and pain points that will inform the system's design priorities.

3. **Document Analysis:** Existing administrative records such as hostel allocation forms, university hostel policies, and documented procedures will be reviewed. This helps to map out the current manual process and spot bottlenecks or inconsistencies. The review will guide system flow modeling and help benchmark improvements.

**System Architecture**

1. **Frontend (Vue.js & Inertia.js):** Vue.js, a contemporary JavaScript framework renowned for creating interactive user interfaces, will be used to design the user interface.
2. Vue.js allows components to dynamically update without reloading the page. Inertia.js will bridge the gap between Vue and Laravel, enabling a single-page application experience while still using Laravel routing. This improves load speed, usability, and interactivity for students and administrators.
3. **Backend (Laravel):** Laravel, a robust PHP framework, will serve as the backbone of the application. It will handle all server-side logic, user authentication, API management, database transactions, and business rules. Laravel's MVC structure and Eloquent ORM simplify data handling and enhance security and scalability. Middleware will manage access control, while Laravel queues will handle background tasks such as email notifications.
4. **Containerization (Docker Compose):** Docker Compose will be used to containerize all components of the system, including the web application, database, and queue workers. This ensures consistent deployment environments and allows the system to be easily moved between development, testing, and production setups.

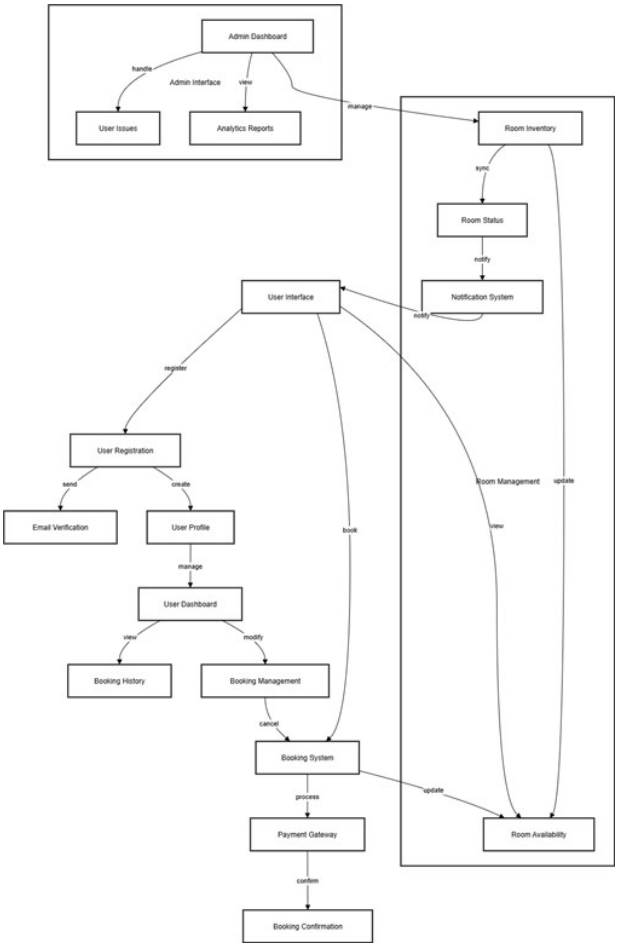


Fig 3.3 Workflow of The System

**Database Design**

1. **Schema Overview:** The database schema will include several interconnected tables:
2. **Users:** This table stores essential user information such as login credentials, roles (student or administrator), and associated metadata.
3. **Rooms:** Contains details about each hostel room, such as its hall, room type (e.g., single, double), capacity, gender designation, and status (available, allocated, reserved).
4. **Bookings:** Tracks booking data such as the allocated room, timestamps, booking status, and related user IDs.
5. **Payments:** Logs payment transaction data, including payment references, status (pending, successful, failed), amounts, and payment gateways used.

6. **Preferences:** This table captures student preferences (ranked room choices) submitted during the application process.
7. **Normalization & Relationships:** The schema is normalized to the third normal form (3NF) to eliminate redundancy. Foreign key relationships enforce data integrity: a user can have multiple bookings, but each booking must be linked to one user and one room.

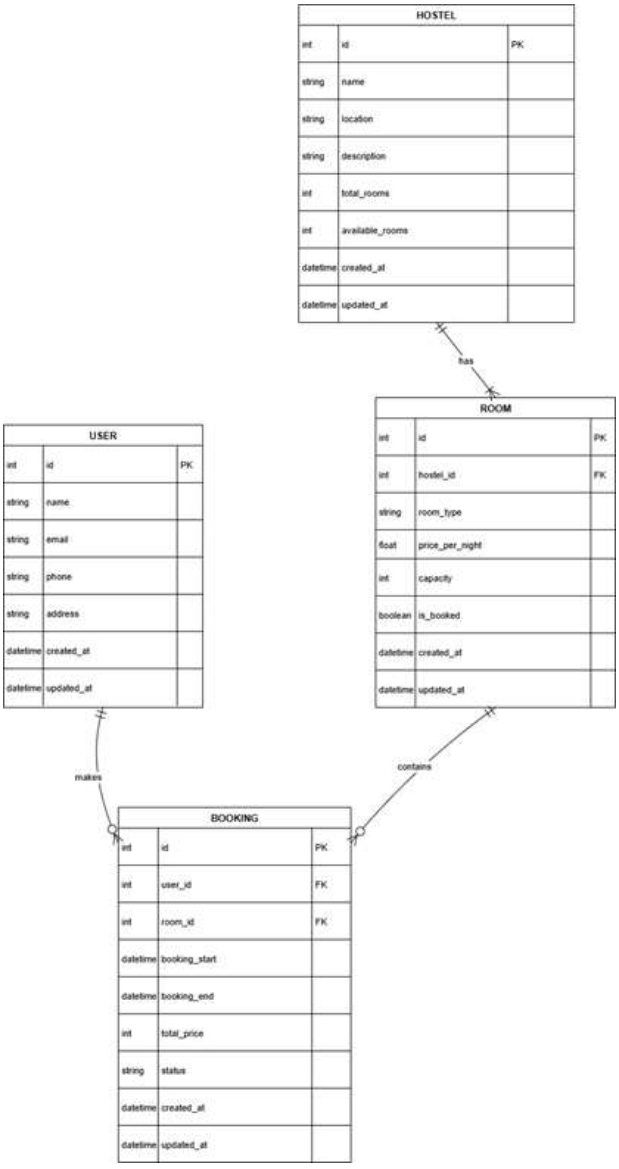


Fig 3.4 Entity-Relationship of The System

**Allocation Algorithm Implementation**

1. **Preference Capture:** The system will allow students to submit preferences for room features, such as hall location, room size, and preferred roommates. These inputs will be transformed into weighted values stored in the Preferences table for later algorithmic processing.
2. **Eligibility Verification:** Before processing allocations, the system verifies that each student is eligible. This includes ensuring fee payments are up-to-date, enrollment is active, and gender requirements match the room.
3. **Matching Engine:** The Analytic Hierarchy Process (AHP) will convert student preferences into numerical weights. These weights will then be fed into the Gale–Shapley stable matching algorithm, which finds the best fit between students and available rooms. This ensures a balanced and fair allocation.
4. **Conflict Resolution & Auditing:** When two or more students have matching scores for a room, tie-breakers such as application time or academic level will be used. Every decision will be logged in an audit trail for transparency.

**Payment Gateway Integration**

1. **Integration Steps:** The Laravel backend will be integrated with Paystack, a trusted Nigerian payment gateway. The system will create a unique reference and redirect the student to Paystack for payment. Upon successful completion, a webhook will confirm the payment.
2. **Secure Workflow:** The entire transaction is secured using HTTPS and token-based authentication. Laravel's CSRF protection will help avoid security breaches, and booking confirmation will only be triggered by a verified callback.
3. **Exception Handling:** Failed payments are logged with detailed error messages and timestamps. Students will receive clear prompts to retry or seek help. Admins are notified of anomalies and can intervene manually when necessary.

**User Interface & Experience**

1. **Wireframing and Prototyping:** The interface design process begins in Figma, where low-fidelity wireframes will be turned into interactive prototypes. These will visualize each step of the user journey, from login to booking confirmation.
2. **Accessibility Standards:** The interface will be built with inclusivity in mind. WCAG 2.1 guidelines will guide design decisions, including contrast ratios, keyboard navigability, and screen-reader compatibility.
3. **Responsive Design:** Tailwind CSS, a utility-first CSS framework, ensures that the platform renders well across devices. From smartphones to large desktop monitors, layouts will adjust fluidly for usability.

**Testing Strategy**

1. **Unit Testing:** Laravel's built-in PHPUnit testing framework will be used to test backend functions such as payment verification and allocation logic.
2. **Component Testing:** Vue Test Utils will check the functionality of dynamic frontend components like room selectors, alerts, and modals.
3. **Integration Testing:** Cypress will simulate the entire system behavior by mimicking user interactions from login to room confirmation and payment.
4. **UAT (User Acceptance Testing):** A sample group of students and staff will use the system in a controlled environment. Their feedback will be used to resolve usability issues and optimize workflows.

**Deployment and Maintenance**

- **CI/CD Pipeline:** GitHub Actions will automate testing and deployment. When developers push changes to the repository, these changes are tested, built, and deployed to the cloud using Docker containers.
- **Monitoring & Logging:** Laravel Telescope and Grafana dashboards will help monitor live system behavior, detect issues, and generate real-time alerts.
- **Backup & Recovery:** Automatic daily database backups and Docker volume snapshots will ensure that data can be restored with minimal downtime in the event of failure.

**Ethical Considerations**

- **Data Privacy:** All user data will be encrypted both in transit and at rest. The application will comply with Nigeria Data Protection Regulation (NDPR), and consent forms will be embedded in the signup process.
- **Transparency:** Allocation criteria, weighting methods, and selection outcomes will be clearly documented and accessible to students via their dashboards.
- **Fairness Audits:** Periodic audits will be conducted to ensure the system treats all users equitably, regardless of academic background, gender, or course of study. If any pattern of bias is found, corrections will be implemented immediately.

## IV. IMPLEMENTATION AND RESULTS

**Overview**

Turning an idea into a real, functional application takes more than just writing code—it requires the careful fusion of planning, design, technology, and testing. This chapter walks through how the Online Hostel Allocation System was implemented, highlighting the step-by-step process of transforming our conceptual design into a robust, working solution tailored to the specific needs of Nigerian private universities. The implementation focused not just on performance, but also on making the system secure, scalable, and intuitive for users—both students and administrators.

We leveraged modern technologies such as Laravel, Vue.js, Inertia, and Docker for their reliability and developer support. Each decision, from our choice of frameworks to how we built the backend and frontend, was geared toward ensuring that the final product would be adaptable, easy to maintain, and ready for real-life deployment.

**Development Environment**

We started by setting up a conducive development environment, ensuring both our hardware and software tools were capable of supporting the system's complexity.

**1. Hardware Requirements**

Development was carried out on a high-performance laptop with at least 8GB RAM and an Intel Core i5 processor. For simulation and deployment, a Virtual Private Server (VPS) running Ubuntu was used, equipped with 4GB RAM and SSD storage. This setup mimicked a production-level environment and allowed for performance benchmarking.

**2. Software Requirements**

The system was built using Laravel for backend logic and Vue.js (via Inertia.js) for the frontend. MySQL was used as the database. Docker and Docker Compose handled containerized development, which ensured environment consistency across devices.

**3. Development Tools**

Laravel Breeze handled the user authentication scaffold, while Composer and npm managed backend and frontend dependencies respectively. Docker kept services isolated and consistent, and GitHub Actions automated testing and deployment tasks.

**System Modules Implementation**

Each major feature of the system was implemented as a standalone module. Here's how we built each:

**Authentication Module**

This handled login, registration, and user roles. Students and admins had separate access rights enforced through middleware. Laravel's password hashing ensured security, while Laravel Breeze simplified scaffolding.


Figure 4.1 Authentication Module

**Room Management Module**

Admins could create, update, or delete hostel rooms with details like room number, capacity, gender, and availability. Vue's reactivity made the interface dynamic, and Laravel handled the backend processing.

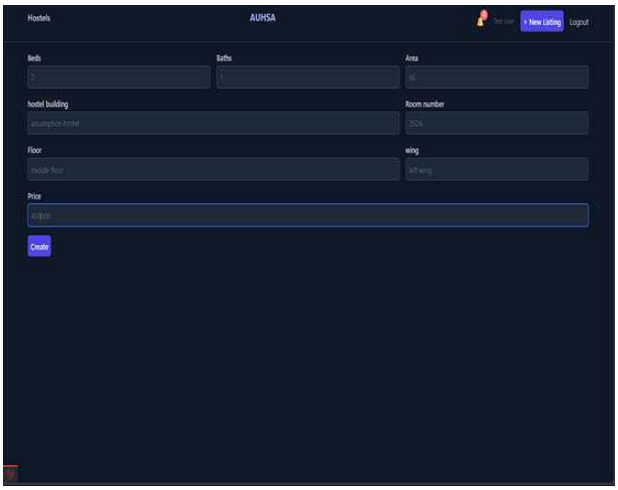


Figure 4.2 Room Management Module

**Student Preference Submission**

Students logged in to rank their preferred hostels and room types. Forms included dropdowns and ranking sliders built in Vue. Backend validation ensured clean and accurate data storage.

**Allocation Engine**

At the heart of the system, this module used the Gale–Shapley algorithm to match students to rooms based on preferences. Academic level and time of submission were used to resolve conflicts. We also integrated the Analytic Hierarchy Process (AHP) to prioritize more critical needs logically.

**Payment Module**

We used Paystack to process payments. Students were redirected to Paystack, and once transactions were verified, the room booking was confirmed. Laravel's job queue system logged the transaction asynchronously.

**Dashboard and Notification**

Students could track allocations and payment confirmations from a responsive dashboard. Admins could view booking stats and manage requests.

Laravel's notification system sent real-time alerts via email and browser popups.

**Containerization with Docker**

To streamline development and deployment, Docker and Docker Compose were used:

- **Docker Compose Setup**

The application was split into containers: one for the Laravel app, one for MySQL, and one for Nginx. Data volumes ensured persistence even when containers were rebuilt.

- Why Docker?
- **Portability:** Run the system anywhere Docker is installed.
- **Isolation:** Each service ran in its own container, reducing conflicts.
- **Scalability:** We could easily scale services depending on demand.

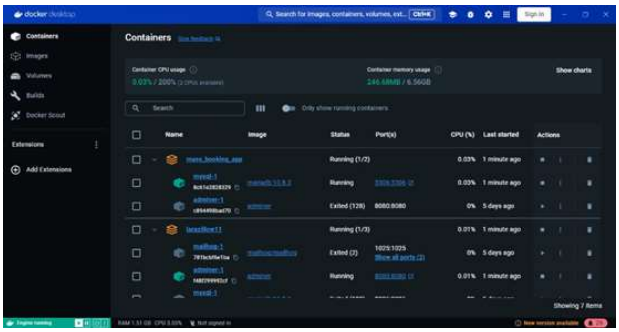

Figure 4.3 Docker (Conternirization)

**Interface Implementation**

**I. Student Interface**

Built using Vue.js and styled with Tailwind CSS, the student interface supported desktop and mobile users. Features included a multi-step registration process, preference submission, status tracking, and payment confirmation. Smooth transitions created a friendly user experience.

**II. Admin Interface**

Admins were given powerful tools to monitor system activity. They could generate reports, manage rooms, and track allocation performance through interactive charts. The real-time updates meant they always worked with current data.

**API Implementations**

- **RESTful API**

Laravel APIs enabled frontend-backend communication, supporting operations like login, room listings, and preference submissions. All data was returned in JSON format. Laravel Sanctum protected routes using token-based authentication.

- **Error Handling**

Standardized error responses helped both users and developers. For instance, invalid form inputs returned helpful validation messages, while unexpected issues triggered generic but secure error responses.

**Performance Evaluation Metrics**

To evaluate the effectiveness and real-world performance of the Online Hostel Allocation System, key quality attributes were measured. These metrics provide insight into how well the system meets user expectations, operational stability, and technical standards.

- **User Friendliness**

This refers to how easy it is for students and administrative staff to navigate and interact with Assessment Method: Observation during live demo sessions and completion of a task-based questionnaire.

**Impact:** High user friendliness ensures faster onboarding and reduces the need for training or manuals.

- **Usability**

Usability evaluates the overall ease of use, learning curve, and efficiency in achieving user goals within the platform.

Assessment Method: System Usability Scale (SUS) and direct user feedback after interaction.

Impact: Enhances satisfaction and trust in the system, encouraging consistent usage by students and staff.

- **Security**

Security measures protect sensitive user information, payment data, and access control.

**Assessment Method:** Static and dynamic testing for vulnerabilities such as SQL injection, XSS, and unauthorized access.

Encrypted password storage using bcrypt

Role-based access control for admin and student dashboards

Secure online payment gateway with SSL
**Impact:** Strengthens user trust, ensures compliance with digital data protection, and prevents data breaches.

- **Functionality**

Functionality assesses whether the system performs its intended features as designed—hostel selection, roommate pairing, payment processing, and admin controls.
Assessment Method: Test cases for each module with pass/fail checks.
Impact: Confirms that the system is meeting core functional requirements without major bugs or missing features.

- **Reliability**

Reliability measures how consistently the system performs under different conditions without failure.
Assessment Method: Load testing and continuous uptime monitoring using Docker containers.
Impact: Makes the system dependable for deployment during critical periods like resumption and payment deadlines.

## V. SUMMARY, CONCLUSIONS AND RECOMMENDATION

### Summary of the Study

The Online Hostel Allocation System was conceived in response to the inefficiencies, stress, and confusion that often plague hostel allocation processes in Nigerian private universities. Traditionally, hostel booking involved long queues, paper-based registrations, unclear allocation methods, and limited transparency. This system was designed to replace that outdated approach with a modern, digital solution.

### Conclusion

The development and implementation of the Online Hostel Allocation System mark a significant stride toward digital transformation in university administration. Through this project, a practical, efficient, and scalable solution has been introduced to address the critical needs of hostel management.

### Recommendations

While the current version of the system meets its primary goals, further refinement and scaling would improve its functionality and sustainability in the long term. The following recommendations are made:

- **Institutional Roll-Out:** Universities should integrate the system into their existing student portals. This integration will facilitate a seamless user experience and institutional trust.
- **Offline Capabilities:** Incorporating Progressive Web App (PWA) features will allow users in remote areas or with unstable internet connections to access basic features offline and sync data when online.
- **Load Testing for Scalability:** Simulating large-scale user environments can help identify bottlenecks. With this data, universities can employ cloud-based scaling options to ensure uninterrupted access during peak periods.
- **Advanced Admin Controls:** Adding audit trails and custom override functions will help administrators handle special cases such as students with disabilities or exceptional needs.
- **Mobile App Development:** A native mobile application for Android and iOS will enhance accessibility, especially considering that many students primarily use smartphones for internet access.
- **Multi-language Support:** Offering the system in major Nigerian languages such as Yoruba, Igbo, and Hausa will foster inclusivity and improve engagement among students from diverse backgrounds.

## REFERENCES

1. Caleb University. (2022). Student handbook and accommodation policy. https://www.calebuniversity.edu.ng
2. Docker Inc. (2024). Docker documentation. https://docs.docker.com/
3. Inertia.js. (2024). Inertia.js: Building modern single-page apps without the complexity. https://inertiajs.com

4. Larman, C. (2004). Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development (3rd ed.). Pearson Education.
5. Laudon, K. C., & Laudon, J. P. (2020). Management information systems: Managing the digital firm (16th ed.). Pearson.
6. Laravel. (2024). Laravel: The PHP framework for web artisans. https://laravel.com/docs
7. Nwachukwu, A., & Nwoke, O. (2020). ICT-based allocation systems: Towards fairer admission and accommodation. Nigerian Journal of Computer Science, 28(2), 45–59.
8. Okoro, I., & Agwu, E. (2018). Improving student welfare through digitized accommodation management. West African Journal of Educational Management, 15(2), 98–110.
9. Oladipo, F. O., & Salami, A. (2019). Automation of hostel allocation using decision support systems. Journal of African Information Technology, 5(1), 101–115.
10. Pan-Atlantic University. (2021). Residential life services report. https://pau.edu.ng
11. Pressman, R. S., & Maxim, B. R. (2014). Software engineering: A practitioner's approach (8th ed.). McGraw-Hill Education.
12. Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education Limited.
13. Tailwind CSS. (2024). Tailwind CSS documentation. https://tailwindcss.com/docs
14. Vue.js. (2024). Vue.js: The progressive JavaScript framework. https://vuejs.org/
15. World Bank. (2019). Improving tertiary education in Sub-Saharan Africa. https://www.worldbank.org
16. Yin, R. K. (2018). Case study research and applications: Design and methods (6th ed.). Sage Publications.