# BoardingHub Management System

# Lorna Jee M. Aure

# Surigao Del Norte State University, Philippines

# [laure@ssct.edu.ph](mailto:laure@ssct.edu.ph)

**Abstract**

*The BoardingHub Management System is a web-based platform developed to assist house owners in managing boarding houses more efficiently. This system minimizes manual administrative tasks by automating room assignments, tenant records, and payment tracking. By offering a user-friendly interface, the system streamlines house management and enhances service quality for tenants. Developed using Django, Bootstrap, and MySQL, BoardingHub simplifies operations through a secure, responsive, and accessible digital solution.*

**I. Introduction**

Managing boarding houses can be time-consuming and prone to human error, especially when done manually. Issues such as misplaced tenant records, delayed payment tracking, and inefficient room assignments often arise. The need for a digital solution has become essential, especially with the increasing number of tenants and rooms to manage. The **BoardingHub Management System** addresses these concerns by offering a centralized platform tailored for house owners, enabling them to oversee their operations seamlessly and provide better service to their tenants.

**II. Objectives**

* To develop a web-based boarding house management system for house owners.
* To reduce manual workload and administrative burden through automation.
* To enable easy monitoring of room availability, tenant data, and payments.
* To provide a secure, responsive, and easy-to-use platform for daily operations.
* To support data accuracy and improve service efficiency.

**III. Methodology**

The system was developed using the following tools and technologies:

* **Backend:** Django Framework (Python)
* **Frontend:** HTML, CSS (Bootstrap 5), JavaScript
* **Database:** MySQL
* **Authentication:** Django’s built-in user authentication system
* **Additional Features:** Crispy Forms for better UI, custom dashboards, dark mode, room and tenant management modules

**IV. Development Phases:**

1. **Planning & Requirement Gathering:** Identification of user roles (admin, house owner) and system features.
2. **System Design:** Wireframes, database schema, and interface layouts were created.
3. **Implementation:** Coding the backend and frontend features, integrating the database, and building functionalities such as CRUD (Create, Read, Update, Delete) operations for rooms and tenants.
4. **Testing:** Functionality and usability testing across various devices and user roles.
5. **Deployment:** Hosted via local server and GitHub repository for version control.

**V. Results**

The BoardingHub Management System successfully met the identified goals:

* House owners can register, log in, and manage their boarding houses.
* They can create rooms with defined capacities, add and manage tenant details, and track monthly payments.
* Tenants do not need to log in; instead, house owners input and manage their information.
* Features such as due dates, payment status, and tenant information views were implemented.
* A dark mode UI and custom branding colors were added for better usability.

**VI. System Features and Descriptions**

**1. Admin Panel**

The Admin Panel is the backend interface reserved for the system administrator. It allows for full control over all data models, including house owners, rooms, and tenants. Admins can oversee system activity, manage user accounts, and maintain data integrity across the platform. Access is restricted and secured with Django’s built-in admin authentication.

A screenshot of a computer

Description automatically generated

**2. Homepage**

The homepage serves as the public landing page of the BoardingHub system. It displays a clean, user-friendly layout with buttons for Login, Register. There is also About BoardingHub. The About section gives a brief introduction to the system's purpose and features. The design uses a white, pink, and black color scheme for visual clarity and professionalism.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**3. Login and Registration (House Owner)**

This feature allows new house owners to register by providing required information including house name and a secure passkey. Existing users can log in with their credentials. Upon successful login, users are redirected to their personalized dashboard. All data is validated and securely stored.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**4. House Owner Dashboard**

The dashboard is the central control panel for each house owner. It shows a quick overview of current room availability, number of tenants, and pending payments. Navigation options led to features like Room List, Tenant List, and Profile settings. The UI is responsive and includes dark mode support.

A screenshot of a computer screen

Description automatically generated

**5. Room List**

This section allows house owners to view all rooms registered under their boarding house. Each room includes details such as room name and maximum capacity. Owners can add new rooms, edit existing ones, or delete them as needed.

A screenshot of a computer

Description automatically generated

**6. Tenant List**

The Tenant List shows all tenants assigned to rooms under the house owner's management. Each entry includes the tenant's name, room, and payment status. Buttons allow for viewing detailed information, editing, or deleting a tenant.

A screenshot of a computer

Description automatically generated

**7. House Owner Profile**

This page displays the house owner's personal details, including their boarding house name, change password, and update profile picture. Users can update their information here. It ensures that each owner can maintain accurate personal data and manage their digital identity within the system.

A screenshot of a computer

Description automatically generated

**8. Tenant Information View**

When a house owner clicks on a tenant’s name, they are taken to a detailed information page. This includes the tenant’s full name, contact details, address, guardian information, room assignment, due date, payment status, monthly rent, and any deposits or advance payments. There is also the edit, delete, add payment, view payment history, download pdf and back to dashboard function.

A screenshot of a phone

Description automatically generated

A screenshot of a computer

Description automatically generated

**9. Edit Tenant**

This feature allows house owners to update a tenant’s details. They can change contact numbers, guardian info, payment due dates, and even reassign tenants to different rooms. Data validation ensures all edits are correctly applied and stored.

A screenshot of a computer

Description automatically generated

10. Add Payment

House owners can add payment records for each tenant. They can specify the amount paid, the payment date, and the method used. This helps maintain accurate financial records and ensures clear tracking of each tenant’s payments.

A screenshot of a computer

Description automatically generated

**11. Payment History**

This section lists all payments made by a specific tenant. It includes dates, amounts, and payment statuses (e.g., paid/unpaid). This historical view helps house owners manage finances and address payment discrepancies easily.

A screenshot of a computer

Description automatically generated

**12. Download PDF**

A built-in PDF export feature allows house owners to generate downloadable reports. This includes tenant details, payment summaries, and room data. Reports can be used for record-keeping or shared with guardians and co-owners for transparency.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a calendar

Description automatically generated**

**VII. TECHNICAL REQUIREMENTS**

The BoardingHub website system is designed to streamline boarding house management through a web-based platform. The project uses robust and modern technologies to ensure scalability, performance, and security.

Technologies Used:

* Front-End:
  + HTML5, CSS3 (Bootstrap 5)
  + JavaScript (for dynamic interaction and user feedback)
* Back-End:
  + Python 3.x
  + Django Web Framework
  + Django REST Framework (DRF) for API functionality
* Database:
  + MySQL (via mysqlclient) for data storage and management
* File Generation and Export:
  + fxhtml2pdf or generating downloadable reports and PDFs

Architecture

The BoardingHub follows a three-tier architecture aligned with modern web development best practices:

1. Presentation Layer
   * Built using HTML, CSS (Bootstrap), and JavaScript.
   * Provides an intuitive UI for house owners and admins.
   * All interactions with backend services are handled via HTTP requests and AJAX/API calls.
2. Application Layer
   * Powered by Django, this layer handles business logic, authentication, tenant management, room management, payment data and history, and file uploads.
   * APIs are exposed via Django REST Framework (DRF).
   * Houses user roles: admin and house owner.
3. Data Layer
   * Data is stored in a structured relational format using MySQL.
   * Models include HouseOwner, Tenants, Rooms, and Payments.
   * fxhtml2pdf integration enables dynamic data to be exported as downloadable PDF reports.

User Interface Design

The system’s UI adopts a clean, responsive, and user-centric design, following Bootstrap 5 guidelines:

* Color Scheme: White, Pink, and Black for clarity and branding.
* Navigation: Simple sidebar and top navigation for ease of access.
* Mobile Friendly: Optimized with responsive elements and adaptive layout.
* Dark Mode: Optional theme toggle for better user comfort.
* Accessibility: Designed with appropriate color contrast and interactive labels.

Data Model Design

The data model is implemented using Django’s ORM and structured around real-world boarding house operations:

* User: Handles user authentication and role assignment (Admin, Owner).
* HouseOwner: Stores house-specific information linked to owners.
* Room: Represents rooms with capacity, availability, and features.
* Tenant: Stores tenant information (contact, guardian info, rent, due date).
* Payment: Tracks rent fees, deposits, and proof of payment.

API Design

The RESTful API built with Django REST Framework offers endpoints for:

* Register/Login functionality
* CRUD operations for Houses and Rooms
* Tenant management (add/view/edit/delete)
* PDF report generation using fxhtml2pdf
* Payment status tracking and history
* Admin dashboard analytics

Authentication is secured with session-based login and role validation middleware.

Security Measures

* HTTPS enforced during deployment
* Role-based access control across system modules
* Input validation and CSRF protection
* Sanitization of uploaded images and file types
* Secure file handling and temporary storage for PDFs

**VIII. TEST PLANS**

Testing Approach:

1. Unit Testing
   * Models, views, serializers, and utility functions
2. Integration Testing
   * Login to dashboard, room and tenant creation, house owner set up authentication.
3. System Testing
   * End-to-end tests for complete workflows (add tenant → assign room → update payment)
4. User Acceptance Testing (UAT)
   * Feedback from users on usability and functionality

Testing Scope:

* Functional Testing
  + Adding tenant, room, payment, announcement posting
* Usability Testing
  + UI responsiveness and navigability
* Compatibility Testing
  + Works across Chrome, Firefox, Safari; desktop and mobile
* Performance Testing
  + Handles 100+ concurrent users, optimized queries
* Security Testing
  + Login/auth flows, form input validation, restricted access

Key Test Cases

Functional Tests

1. Create a room, add capacity → success message.
2. Add tenant and associate with a room → tenant listed correctly.
3. Update payment and see history → file stored securely and accessible.
4. Delete tenant → tenant removed from database.

Usability Tests

1. All navigation buttons are visible and functional.
2. Form validation messages are clear and consistent.

Compatibility Tests

1. Dark mode works across all screen sizes.
2. Page layout consistent in desktop and mobile.

Security Tests

1. Prevent direct access to owner dashboard by unauthorized users.
2. Password reset flow is secure and functional.

**IX. KNOWN BUGS AND MAINTENANCE GUIDE**

Common Bugs Encountered:

1. Shadowing Variables
2. Incorrect model relationships (e.g., ForeignKey misuse)
3. Unhandled exceptions in views
4. PDF export layout inconsistency
5. API error formatting

Maintenance Strategies:

* Conduct regular code reviews for Django views and serializers.
* Implement try-except blocks around critical database and file I/O operations.
* Use Django signals to track and log unexpected model changes.
* Validate file types and sizes during upload (especially for PDFs and proof of payment).
* Maintain test coverage with Django’s test client.
* Keep third-party libraries (Django, DRF, fxhtml2pdf) up to date.

**X. CONCLUSION**

The BoardingHub system delivers a modern, reliable, and user-friendly platform for boarding house management. Built with Django and enhanced by REST APIs and PDF exports, it supports essential features like tenant tracking, room management, and payment history in a secure and well-organized interface.

The modular structure allows for easy maintenance and future expansion. Key highlights include:

* Dynamic dashboards and reporting
* Room and Tenant management
* Payment Tracking and History
* Dark mode and responsive layout
* PDF generation for reports and records

**Future Enhancements and Recommendations**

* GCash/PayPal Payment Gateway Integration  
  Provide secure and seamless rent payment options.
* Tenant Login Portal  
  Allow tenants to view personal rent status and announcements.
* SMS Notifications  
  Send reminders for due dates or announcements.
* Calendar View for Room Availability  
  Visual scheduling tool for owners to manage tenant inflow.
* AI-based Analytics Dashboard  
  Predict overdue tenants, suggest room price adjustments, etc.
* Real-time Chat  
  Enable communication between house owners and tenants.

# GITHUB LINK

[auljee/Boarding-Hub-Project](https://github.com/auljee/Boarding-Hub-Project)