Especificación de requisitos de software

Proyecto: CondoManage: Comprehensive Condominium Management Platform

Revisión 1

Noviembre

Document sheet

Fecha	Revisión	Autor	Verificado dep. calidad.
10/11/2024	11/11/2024	noOvertime	

Documento validado por las partes en fecha: 10/11/2024

Por el cliente	Por la empresa suministradora
Fdo. D./ Dña	Fdo. D./Dña noOvertime

Content

1 Introduction

The Homeowner's Association Management System (HAMS) is designed to streamline administrative tasks for homeowner associations, enabling administrators to efficiently manage resident fee payments, schedule facility maintenance, and facilitate communication with property owners and tenants. Key features include automated invoice generation, online payment support, maintenance scheduling with notifications, and a messaging platform for easy interaction between admins and residents. The system also offers reporting and analytics on payments, maintenance activities, and communication logs, while ensuring high performance, robust security measures, and a mobile-responsive user interface to enhance usability and accessibility.

1.1 Purpose

The purpose of the Homeowner's Association Management System (HAMS) project is to create an efficient and comprehensive solution for managing the administrative tasks of homeowner associations. This system aims to automate processes such as tracking resident fee payments, scheduling and monitoring facility maintenance, and improving communication between the association's administrator and residents. By providing a centralized platform, the project seeks to enhance operational efficiency, reduce manual workloads, improve transparency, and facilitate better engagement between administrators and the community members they serve.

1.2 Scope

The Homeowner's Association Management System (HAMS) is a centralized platform designed to streamline administrative tasks for homeowner associations. It includes features for tracking resident fee payments, scheduling facility maintenance, and facilitating communication between administrators and residents. The system aligns with industry standards for property management by ensuring data security, efficient communication, and robust financial tracking. It aims to enhance operational efficiency, reduce manual workloads, and improve engagement within the community.

1.3 Personnel involved

Name	Jami Klever Leiton José Manosalvas Gabriel Maza Jardel
Role	Developer
Professional category	Developer
Responsibilities	Design, collaborate and develop
Contact information	ESPE
Approval	

1.4 Definitions, acronyms and abbreviations

This section provides definitions of terms, abbreviations, and acronyms used throughout this document to ensure clarity and proper interpretation.

- **HAMS**: Homeowner's Association Management System - The software platform designed to manage the administrative tasks of a homeowner's association.

- **Admin**: Administrator - The person responsible for managing the system, including fee payments, maintenance schedules, and resident communication.
- **Resident**: A property owner or tenant within the homeowner's association community who interacts with the system for payments, maintenance requests, and communication.
- **UI**: User Interface - The visual part of the system that users interact with, including both desktop and mobile interfaces.
- **API**: Application Programming Interface - A set of protocols and tools that allow different software components to communicate with each other.
- **SSL**: Secure Sockets Layer - A standard security protocol used for encrypting data transmitted over the internet to ensure privacy and data integrity.
- **Invoice**: A bill issued by the homeowner's association to residents for fee payments.
- **Maintenance Request**: A report or inquiry submitted by residents regarding repairs or upkeep of community facilities.
- **Backup and Recovery**: Processes involved in creating copies of data to protect against data loss and to enable data restoration in case of system failure.

1.5 References

Reference	Title	Route	Date	Author
1	Aprender a programar con Java : un enfoque práctico partiendo de cero	Paraninfo	2014	Jiménez Marín, Alfonso
2	Introducción a la programación orientada a objetos	Pearson	2010	Deitel, Paul J

1.6 Abstract

This document outlines the Software Requirements Specification (SRS) for the Homeowner's Association Management System (HAMS), detailing the system's purpose, scope, features, and specific requirements. It provides an overview of the system's functionality, including fee management, maintenance scheduling, and communication tools for administrators and residents. The document is organized into sections that cover the introduction, overall system description, specific functional and non-functional requirements, key system features, and additional non-functional considerations such as legal and security requirements. Each section is structured to offer clear and comprehensive guidance for the development and implementation of the system.

2 General Description

2.1 Product perspective

CondoManagement is a standalone system designed to efficiently manage a condominium's operations. Its primary function is to comprehensively handle tasks related to residents, vehicles, utility bills, payments, common area reservations, and access control. The system uses JSON and CSV formats to ensure data persistence and is developed following clean code principles in Java.

2.2 Product functionality

CondoManagement provides the condominium administrator with a comprehensive tool to optimize administrative operations and enhance the residents' experience. The key functionalities include:

1. **Resident Management**
   Allows registering, updating, and deleting resident information, including personal and contact details.
2. **Vehicle Management**
   Provides functionality to register vehicles, assign them to residents, and manage their parking spaces.
3. **Access Control**
   Records and manages access for residents and their vehicles to the condominium, using role-based permissions (administrator and resident).
4. **Utility Bill Management**
   Generates and manages utility bills associated with condominium services, storing details such as amounts, due dates, and payment status (pending or paid).
5. **Payment Processing**
   Allows the registration of payments made by residents and automatically updates the status of associated bills.
6. **Transaction History**
   Stores a detailed history of access logs, utility bills, payments, and reservations, enabling quick reference for future management.
7. **Common Area Reservations**
   Facilitates the creation, modification, and cancellation of reservations for common areas, including details like date, time, and associated resident.
8. **General Reports**
   Generates periodic reports on reservation statuses, completed payments, pending bills, and resident access logs, providing a comprehensive overview of the condominium's administrative status.
9. **Security and Confidentiality**
   Implements user authentication and data encryption to protect residents' confidential information and transactions.

2.3 User characteristics

User type	The main users of CondoManagement are condominium managers, in charge of managing leases, payments and occupancy of dwellings.
Training	Users typically have a background in property management or accounting, with secondary or tertiary education. Advanced technical software training is not required.
Skills	Basic skills in administrative management, computer use and communication with tenants are required.
Activities	Users will perform activities such as recording data, managing payments, generating reports and viewing lease history.

2.4 Restrictions

The development of CondoManagement must take into account the following constraints and requirements:

1. **Programming languages**
   The system must be developed using languages suitable for the creation of management applications, such as Java or Python, which are suitable for object-oriented programming and offer database support.
2. **Development Platform**
   The system must be compatible with common platforms, such as Windows or Linux, and must be guaranteed to be accessible through web browsers, without the need for complex installations.
3. **Database**
   A scalable database management system (DBMS), such as MySQL or PostgreSQL, shall be used to store lease and payment information securely and efficiently.
4. **User Interface**
   The interface should be simple and easy to use, without requiring advanced technical skills, with a design adaptable to different devices (PCs and tablets).
5. **Security**
   Security measures, such as user authentication and encryption of sensitive data, should be implemented to protect tenant information and transactions.
6. **Hardware limitations**
   The system should not require specialised hardware, but should function properly on standard equipment with minimum specifications (basic processors, 4 GB RAM)..
7. **Regulatory Compliance**
   The system should adhere to local data protection and privacy regulations, especially with regard to the management of tenants' personal information.

2.5 Assumptions and dependencies

1. **Operating System and Platform**
   It is assumed that the system will run on Windows or Linux compatible platforms, and that the web browser will be accessible to most users. If operating system versions change or if a different platform is required, adjustments to the design or technologies used may be necessary.
2. **Internet connection**
   It is assumed that users will have access to a reliable Internet connection to access the system through a web browser. If internet access is unavailable or unstable, the system may have to be adapted to a desktop application model or include offline functionalities.

3. **Technological Resources**
   It is assumed that the selected programming languages and tools (such as Python, JavaScript, MySQL) will be adequate and available during the development of the system. If for some reason these tools are not feasible or the project specifications change, the development may be affected.
4. **Users with basic skills**
   It is assumed that users have a basic level of knowledge in computer and web browser use, without the need for advanced technical training. If end users have a different skill level, the system may require a redesign to be more accessible or include additional tutorials.
5. **Accessibility of Tenant Data**
   It is assumed that the customer will be able to provide all the necessary data on tenants and dwellings for the correct functioning of the system. If this information is not available or is incomplete, it may be necessary to adjust the functionalities of the system.

2.6 Foreseeable evolution of the system

As the development of the HAMS system progresses, some enhancements and new functionalities that could be implemented in the future can be considered:

1. **Integration with Local Electronic Payment Platforms.**
   In the future, it could be considered to integrate the system with popular electronic payment platforms in Ecuador, such as Pago Ágil, Diners Club or local bank transfers, allowing tenants to make payments in a faster and more convenient way.
2. **Mobile Application**
   A mobile application could be developed to manage payments and rentals from smartphones, offering greater convenience to tenants and managers, which would be very useful in the local context.
3. **Housing Maintenance Management**
   In the future, the system could include a home maintenance management functionality, where managers can register, track and manage maintenance requests, something that can be very useful to improve the management of condominiums.
4. **Automatic Notifications to Users**
   The system could add the ability to send automatic notifications to tenants about payment dates, lease expirations, and important events, facilitating efficient communication between the manager and residents
5. **Scalability and Adaptability to New Needs**
   As demand grows, the system could scale to manage more properties and users, ensuring optimal performance as the customer base expands and new condominiums are added.

3 Specific requirements

This section outlines the detailed requirements that the **CondoManagement** system must meet to satisfy the needs of both administrators and residents. The requirements are numbered to ensure traceability and validation. Each requirement includes specific details about the need, its source, and its priority.

Número de requisito	**1**
Nombre de requisito	**Fee Payment Management**r
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Needs of administrators and residents
Prioridad del requisito	☐ Alta/Esencial ☐ ☐

Description:
The system must allow administrators to manage and record monthly payments from residents. This includes:

- Automatically generating invoices using the UtilityBill class.
- Storing and updating the status of invoices (paid or pending).
- Recording payments made by residents using the Payment class.
- Providing residents with access to view their payment history and receive reminders for overdue invoices.

Número de requisito	**2**
Nombre de requisito	Common Area Reservation Management
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Needs of administrators and residents
Prioridad del requisito	☐ Alta/Esencial ☐ ☐

Description:
The system must allow administrators and residents to manage reservations for common areas using the AreaReservation class. This includes:

- Registering, modifying, and canceling reservations.
- Providing reservation details such as area, date, time, and associated resident.
- Notifying residents about successful reservations or changes.

Número de requisito	**3**
Nombre de requisito	Access Control Management
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Needs of administrators
Prioridad del requisito	☐ Alta/Esencial ☐ ☐

Description:
The system must manage access to the condominium using the AccessControl class. This includes:

- Recording access for residents and vehicles.

- Controlling access permissions based on roles (administrator or resident).
- Generating an access history for auditing purposes.

Número de requisito	**4**
Nombre de requisito	Resident Portal
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Needs of residents
Prioridad del requisito	☐ ☐ Media/Deseado ☐

Description:
The system must provide residents with secure access to a portal where they can:

- Consult their personal information stored in the Resident class.
- View the status of their invoices, payments, and reservations.
- Update their contact information easily and securely.

Número de requisito	**5**
Nombre de requisito	Reporting and Analytics
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Needs of administrators.
Prioridad del requisito	☐ ☐ Media/Deseado ☐

Description:
The system must include functionalities to generate administrative reports using data from the UtilityBill, Payment, and AccessControl classes. Reports should include:

- Summary of completed and pending payments.
- Access history.
- Trends in payment delays and frequently reserved areas.

Número de requisito	**6**
Nombre de requisito	Secure Data Management
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Legal compliance and regulations
Prioridad del requisito	☐ Alta/Esencial ☐ ☐

Description:
The system must implement strong security measures to protect user data. This includes:

- Encrypting sensitive data during transmission (using secure protocols such as SSL).
- Authenticating users with secure passwords.
- Complying with data protection laws.
- Conducting regular audits to identify and resolve vulnerabilities.

3.1 Common interface requirements

The HAMS system will have a web browser-based user interface, accessible from both desktop computers and mobile devices. Interactions will be primarily split between condominium managers and tenants. Key inputs and outputs of the system are detailed below.

System Inputs:

Tenant Data:

1. **Full Name:**
    - The system stores the full name of the tenant in the PersonalData or Resident class, depending on the system design for resident data management.
    - Class: PersonalData or Resident.
2. **Identification Number:**
    - The tenant's identification number is stored in either the PersonalData or Resident class.
    - Class: PersonalData or Resident.
3. **Address of the Rented Dwelling:**
    - The address of the rented dwelling is stored in the Resident class or could be part of a related class associated with the dwelling.
    - Class: Resident.
4. **Contact Information:**
    - The system stores contact information such as phone number and email in the Resident or PersonalData class.
    - Class: Resident.
5. **Start and End Date of the Rental Contract:**
    - The system manages the start and end dates of rental contracts in the Resident class or could be stored in a dedicated Lease class, if created.
    - Class: Resident or Lease.

Details of the Dwelling:

1. **Full Address:**
    - The system stores the full address of the property in either the Resident class or in a dedicated Dwelling class.
    - Class: Resident or Dwelling.
2. **Description of the Property:**
    - The system stores details about the property, such as type and number of rooms, in the Dwelling class, which is associated with the resident.
    - Class: Dwelling.
3. **Status of the Property:**
    - The system manages the status of the property (e.g., rented, under maintenance) within the Dwelling or Resident class.
    - Class: Dwelling or Resident.

Payments and Payment Dates:

1. **Amount of Rent:**
    - The system tracks the rent amount in the UtilityBill or Lease class, depending on the organization of the rental data.
    - Class: UtilityBill or Lease.
2. **Due Date of Monthly Payment:**
    - The due date of monthly payments is stored in the UtilityBill class.
    - Class: UtilityBill.
3. **Information on Payments Made:**
    - Payment information, including amount, date, and method of payment, is

stored in the Payment class.
  - Class: Payment.

Maintenance Record (Future):

1. **Maintenance Requests:**
   - Maintenance requests, along with their descriptions and dates, are stored in a potential MaintenanceRequest class.
   - Class: MaintenanceRequest.
2. **Status of Requests:**
   - The status of each maintenance request (pending, in progress, resolved) is tracked in the MaintenanceRequest class.
   - Class: MaintenanceRequest.

System Outputs:

Payment History:

- Description: The system generates a summary of payments made by each tenant, including dates and amounts, and allows the generation of detailed reports. This data is retrieved from the Payment and UtilityBill classes.
- Class: Payment and UtilityBill.
- Method: The system includes methods to generate reports based on payment history.

Lease Status:

- Description: The system provides information on active and expired leases, including start and end dates. This data is managed in the Lease class, if created.
- Class: Lease (if created) or Resident.
- Method: Methods in the Lease or Resident class determine the lease status.

Tenant Notifications:

1. Reminders of Outstanding Payments:
   - The system generates reminders for outstanding payments, based on data from the UtilityBill and Payment classes.
   - Method: A method in the Payment class manages the creation and sending of payment reminders.
2. Notices about Contract Renewals or Expirations:
   - Notifications related to contract renewals or expirations are generated from the Lease class.
   - Method: Methods in the Lease class track the expiration dates and generate renewal reminders.

Maintenance Report (Future):

- Description: A report detailing all maintenance requests, their statuses, and completion dates. This data is managed by the MaintenanceRequest class.
- Class: MaintenanceRequest.
- Method: A method in the MaintenanceRequest class compiles and generates maintenance reports.

General Reports:

1. **Housing Occupancy Reports:**

- The system generates reports on housing occupancy, derived from the data in the Resident or Dwelling class.
    - Method: Methods in the Resident or Dwelling class generate occupancy data.
2. **Rental Income Summaries:**
    - The system generates summaries of rental income, derived from payments in the Payment or UtilityBill classes.
    - Method: A method in the Payment or UtilityBill class compiles and reports rental income.

3.1.1 User interfaces

The HAMS system will have an interface accessible from mobile devices and desktop computers. The design will be simple and intuitive, using a color palette of blue, gray, and white, with modern, easy-to-read fonts such as Arial or Roboto.

- **Home Screen:**
    - **Administrators:** Overview with key indicators (revenue, active/expired leases, upcoming payment dates). Quick access to leases, payments, and maintenance.
    - **Tenants:** Contract summary, amount and payment due date, payment history. Options to pay online and view details.
- **Lease Management Screen:**
    - **Administrators:** Lease management with options to add, edit, or delete records.
    - **Tenants:** View of their contract, payments, and status, without the option to edit data.
- **Payments Screen:**
    - **Administrators:** Record of payments, both manual and online.
    - **Tenants:** Option to make online payments and view payment history.
- **Maintenance Screen (future):**
    - **Administrators:** Management of maintenance requests (pending, in process, completed).
    - **Tenants:** Option to submit maintenance requests.
- **Notifications:** The system will send payment and contract renewal reminders to tenants, as well as notifications about payments and maintenance to administrators.

3.1.2 Hardware interfaces

1. **Access Control System Interface**

- **Logical Characteristics**: The system will interface with biometric scanners installed at entry and exit points, as well as in other restricted areas within the condominium. The interface will manage identification, verification, and logging of each access attempt. Communication will occur over a secure connection, with data transferred in encrypted format to ensure the privacy of resident data.
- **Configuration Characteristics**: The system will support configuring access permissions by resident type. The administrator can set up and adjust permission levels, update user profiles, and view access logs through a web-based management portal.

2. **Surveillance Camera System Interface**

- **Logical Characteristics**: The system will integrate with IP-based surveillance cameras to monitor common areas. The interface will support real-time streaming and video storage, with the ability to retrieve and display footage on demand. It will use the RTSP to handle video feeds.

- **Configuration Characteristics**: The interface will allow the administrator to set camera angles, adjust recording schedules, and access footage based on date and time. Permissions to access surveillance data can be configured based on role, such as security personnel or management.

3. **Intercom System Interface**

- **Logical Characteristics**: The system will connect to the intercom devices installed in each unit and main entrances, allowing residents to communicate with visitors or management directly. The interface will employ for audio communication and support call routing and call logs.
- **Configuration Characteristics**: Administrators will be able to configure call forwarding options, set permissions for specific intercom devices, and monitor call activity logs for security purposes.

4. **Smart Utility Meters Interface**

- **Logical Characteristics**: Integration with smart water and electricity meters will enable automatic reading of usage data, allowing for accurate billing. Communication with meters will be established using the Modbus or Zigbee protocol, depending on the device specifications.
- **Configuration Characteristics**: The interface will support interval configuration for reading data and allow administrators to set usage thresholds, monitor real-time consumption, and generate usage reports. Data from the meters will be accessible through the management system for billing purposes.

3.1.3 Software interfaces

1. **Description of Software Product Used:**
    - **Billing Software:** This software is used to generate and maintain billing records, allowing the management of maintenance fees, service payments, and account statements for condominium residents.
    - **Access Control Software:** System that controls and monitors access to the condominium's common areas, including entry and exit points, recreational areas, and parking facilities.
    - **Owner and Tenant Database:** Database where all information related to property owners, tenants, and condominium units is stored. It enables the management of contact information, payment history, and property details.
2. **Purpose of the Interface:**
    - **Billing Integration:** Enable the condominium management system to automatically generate invoices and apply recurring charges for each property owner. It facilitates the consultation of completed and pending payments, as well as the generation of financial reports.
    - **Access Control:** Synchronize data from the management system with the access control system, ensuring that only authorized individuals can enter designated areas of the condominium, based on their residency or ownership status.
    - **Owner Database Management:** Store, update, and retrieve personal and contact information, as well as track the payment status and records for each resident.
3. **Interface Definition: Content and Format**
    - **Billing Interface**
        - **Content:** User identification (ID), property details, type and amount of fees, due date, payment history.
        - **Format:** JSON for data transfer over RESTful APIs. XML may also be used when necessary to integrate with systems that require this

format.

- **Access Control Interface**
  - **Content:** Resident ID, full name, permitted access types (e.g., common areas, parking), access validity period.
  - **Format:** JSON or XML for compatibility with access control systems that accept multiple formats. Authentication will be managed through OAuth2 to ensure the security of personal information.
- **Owner Database Interface**
  - **Content:** Owner ID, property details, payment history, contact information.
  - **Format:** SQL for direct database queries, with RESTful endpoints that allow the software modules to perform CRUD (Create, Read, Update, Delete) operations on the database in JSON format.

3.1.4 Communication interfaces

The Homeowner's Association Management System will rely on various communication interfaces to interact with others. Additionally, secure backup and disaster recovery processes will ensure system resilience. These interfaces and protocols will provide seamless and secure communication across the system's components and external services.

3.2 Functional requirements

The Homeowner's Association Management System must perform essential functions, including input validation, operation sequencing, response to abnormal situations, and accurate output generation. It must validate and process resident registrations, payment submissions, and maintenance requests according to well-defined rules. The system must also respond appropriately to errors, provide flexible parameters for configurations, and ensure logical relationships between inputs and outputs. Data stored in the system must be accurate and consistent, including personal information, payment history, and maintenance records, ensuring the system can generate meaningful reports and notifications for administrators and residents.

3.2.1 Entry Validity Check

The system must validate all incoming data to ensure correctness and consistency.

3.2.2 Exact Sequence of Operations

The system must follow a specific, defined sequence of operations to ensure smooth execution of tasks and accurate results.

3.2.3 Parameters

The system must allow administrators to define key parameters such as payment deadlines.

3.2.4 Database Logical Requirements

The system must store and manage information logically in the database to ensure integrity, accuracy, and consistency.

3.3 Non functional requirements

3.3.1 Performance requirements

The Homeowner's Association Management System must meet specific performance criteria to ensure optimal system operation, even under high load conditions. These performance requirements are crucial for providing a seamless user experience and maintaining efficient system performance.

3.3.2 Security

HAMS will employ a variety of security measures to protect against malicious access, use, sabotage, and accidental modifications. Additionally, secure session management will provide further protection against external threats. These measures will ensure the system remains secure and resilient against security risks.

3.3.3 Reliability

These reliability specifications ensure that HAMS operates consistently and efficiently, with minimal downtime and high fault tolerance. Meeting these reliability factors is essential for maintaining trust and satisfaction among both administrators and residents, as well as for ensuring the smooth operation of critical association management tasks.

3.3.4 Availability

The Homeowner's Association Management System (HAMS) must maintain high availability to ensure smooth operation for both administrators and residents. These availability requirements are designed to provide a reliable and uninterrupted user experience.

3.3.5 Maintainability

The Homeowner's Association Management System (HAMS) requires regular and well-planned maintenance to ensure optimal performance, security, and user satisfaction. Corrective, adaptive, preventive, and perfective maintenance will be carried out by system administrators, developers, and users, with a structured maintenance schedule that includes daily, weekly, monthly, quarterly, and annual tasks. This ensures the system remains reliable, up-to-date, and secure while continuously improving to meet evolving needs.

3.3.6 Portability

To ensure that the Homeowner's Association Management System (HAMS) is portable, it must minimize dependencies on specific server infrastructures and operating systems. By using widely supported programming languages like Java. Additionally, the use of a cross-platform development platform, a portable database management system, and ensuring compatibility with major operating systems will facilitate the easy transfer of the system to new environments or platforms with minimal effort and disruption. These measures will help future-proof the system and make it adaptable to different technologies as they evolve.

3.4 Other requirements

HAMS must meet additional requirements related to cultural adaptability, legal compliance, security, scalability, and environmental impact. It should be locally adaptable to support multiple languages, comply with data protection and financial regulations, and provide strong security measures to protect user data. These considerations are crucial for ensuring that the system is suitable for diverse user bases and remains compliant with evolving legal and regulatory frameworks.

4 Appendices

Appendix A: System Architecture Diagram

Includes a diagram of the overall architecture of the application, describing the relationship between the main components, such as the database, the backend server, the frontend client, and third-party services (e.g., payment platforms, notification services, etc.).

Appendix B: Database Specifications

Describes the structure of the database, including:

- Main tables (e.g., users, reservations, incidents, payments).
- Attributes of each table and data types.
- Relationships between tables.

This appendix may include an entity-relationship (ER) diagram to facilitate understanding.

Appendix C: API Specifications

Details the APIs required for the system, both internal and external:

- User API: Registration, authentication, password recovery.
- Reservation API: Create, modify, cancel common area reservations.
- Payment API: Payment registration and verification.
- Notification API: Sending messages to residents and administrators.

Each entry should include the endpoints, HTTP methods, input parameters, and expected responses.

Appendix D: Security Requirements

This appendix covers the security standards that the application must meet, such as:

- Authentication: Use of secure passwords, two-factor authentication (if applicable).
- Authorization: Role-based access control (e.g., resident, administrator).
- Data encryption: Encryption of sensitive data in the database and in transit (e.g., HTTPS).
- Protection against common threats: Measures to prevent SQL injections, XSS attacks, CSRF, etc.

Appendix E: Installation Guide

A detailed guide for installing the application in the production environment, including:

- Server requirements (operating system, RAM, disk space).
- Necessary dependencies (e.g., PHP versions, Node.js, compatible databases).
- Step-by-step installation procedures, with commands and necessary configurations.
- Post-installation tests to ensure the system is functioning correctly.

Appendix F: User Manual

A manual to help users (administrators and residents) navigate the application:

- Instructions for residents: How to register, view their account status, make reservations, report incidents.
- Instructions for administrators: How to manage residents, review and approve reservations, receive incident reports, send notifications.
- Screenshots and detailed steps for common tasks.

Appendix G: Testing Strategy

Includes a testing plan with:

- Test cases: Specific cases to test key functionalities (e.g., register a new user, make a reservation).
- Security testing: To validate that security requirements are met.
- Performance testing: To verify that the application can handle multiple simultaneous users.
- Usability testing: Tests to ensure that the application is easy to use for

end-users.

Appendix H: Maintenance Plan

Includes the plan for maintaining the system once it is in production, such as:

- Security updates: Frequency of security review and updates.
- System monitoring: Tools and metrics for monitoring to ensure availability.
- Technical support: Procedures for reporting and resolving issues.