**Software Requirements Specification**

**for**

**<Apartment Management System>**

**Version 1.0 approved**

**< Group 1 >**

| **Member** | **Actor** |
| --- | --- |
| Nguyễn Huyền Diệu | Leader |
| Nguyễn Văn Phan | Member |
| Phạm Hoàng Huy | Member |
| Đặng Hoàng Sơn | Member |
| Vũ Hoàng Tùng | Member |

**< Spring 25 >**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1.** **Introduction 2**

1.1 Purpose [2](#_k0skr4hgfg3p)

1.2 Document Conventions [2](#_2od91qi8rwfk)

1.3 Project Scope [2](#_qptqzh1l6zyp)

1.4 References [2](#_hukneumv9bl6)

**2.** **Overall Description 2**

2.1 Product Perspective [2](#_eusnba183tav)

2.2 User Classes and Characteristics [2](#_f1y8bamwbnz2)

2.3 Operating Environment [2](#_f6si6ewdinkf)

2.4 Design and Implementation Constraints [2](#_ghz2ph6usyp4)

2.5 Assumptions and Dependencies [2](#_ib7ttk3gn0dr)

**3.** **System Features 2**

3.1 System Feature 1 [2](#_27y8rqol3q0)

3.2 System Feature 2 (and so on) [2](#_xiog87oog4vl)

**4.** **Data Requirements 2**

4.1 Logical Data Model [2](#_u4hxaa1p9hwe)

4.2 Data Dictionary [2](#_jrqmktqwecbs)

4.3 Reports [2](#_v03kfyu6h2pi)

4.4 Data Acquisition, Integrity, Retention, and Disposal [2](#_c3obmk3d9g4q)

**5.** **External Interface Requirements 2**

5.1 User Interfaces [2](#_xj093x9kdtw)

5.2 Software Interfaces [2](#_v21ds8wo4h84)

5.3 Hardware Interfaces [2](#_ig6mkrgzi5eu)

5.4 Communications Interfaces [2](#_yxqt315c8x50)

**6.** **Quality Attributes 2**

6.1 Usability [2](#_73cfkd2aehbl)

6.2 Performance [2](#_p85iodd685dp)

6.3 Security [2](#_nqhin4dxwl53)

6.4 Safety [2](#_tpiopqh97i1)

6.5 [Others as relevant] [2](#_thwvelvuoe8o)

**7.** **Internationalization and Localization Requirements 2**

**8.** **Other Requirements 2**

**Appendix A: Glossary 2**

**Appendix B: Analysis Models 2**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Nguyễn Văn Phan | 21/05/2025 | 1. Introduction | 1.0.0 |
| Phạm Hoàng Huy | 21/05/2025 | 1.1 Purpose | 1.0.0 |
| Nguyễn Huyền Diệu | 21/05/2025 | 1.2 Document Conventions | 1.0.0 |
| Đặng Hoàng Sơn | 21/05/2025 | 1.3 Project Scope | 1.0.0 |
| Vũ Hoàng Tùng | 21/05/2025 | 1.4 References | 1.0.0 |
| Vũ Hoàng Tùng | 26/05/2025 | 2.1 Product Perspective | 1.0.0 |
| Đặng Hoàng Sơn | 26/05/2025 | 2.2 User Classes and Characteristics | 1.0.0 |
| Nguyễn Huyền Diệu | 26/05/2025 | 2.3 Operating Environment | 1.0.0 |
| Phạm Hoàng Huy | 26/05/2025 | 2.4 Design and Implementation Constraints | 1.0.0 |
| Nguyễn Văn Phan | 26/05/2025 | 2.5 Assumptions and Dependencies | 1.0.0 |
| Nguyễn Huyền Diệu | 30/05/2025 | 3.1 Resident | 1.0.0 |
| Nguyễn Huyền Diệu | 30/05/2025 | 3.2 Admin | 1.0.0 |
| Phạm Hoàng Huy | 30/05/2025 | 3.3 Sanitation Management | 1.0.0 |
| Nguyễn Văn Phan | 30/05/2025 | 3.4 Accountant | 1.0.0 |
| Đặng Hoàng Sơn | 30/05/2025 | 3.5 Security Staff | 1.0.0 |
| Vũ Hoàng Tùng | 30/05/2025 | 3.6 Technical Staff | 1.0.0 |
| Vũ Hoàng Tùng | 30/05/2025 | 3.7 Guest | 1.0.0 |
| Vũ Hoàng Tùng |  | 4. Data Requirements | 1.0.0 |
| Nguyễn Văn Phan |  | 5. External Interface Requirements | 1.0.0 |
| Phạm Hoàng Huy |  | 6. Quality Attributes | 1.0.0 |
|  |  | 7. Internationalization and Localization Requirements | 1.0.0 |
|  |  | 8. Other Requirements | 1.0.0 |
| Đặng Hoàng Sơn |  | Context Diagram |  |
| Nguyễn Huyền Diệu |  | System Feature Tree |  |
| Nguyễn Văn Phan |  | Swim lane |  |
| Phạm Hoàng Huy |  | Activity Diagram |  |
| Vũ Hoàng Tùng |  | State Diagram |  |

# **Introduction**

This Software Requirements Specification (SRS) for the Apartment Management System developed by Group 1. The purpose of this system is to support apartment management by handling tasks such as managing residents and guests, service payments, maintenance requests, announcements, security logs, and feedback. This document serves as a shared reference for developers, managers, and users, guiding the system’s development and future improvements.

## **Purpose**

Apartment Management System is built to support the management, operation and interaction between the parties in an apartment area. This system helps to digitize the operation process, from information management, receiving requests, sending notices to online payment, in order to improve efficiency and minimize manual work. The main purpose of the system:

**Manage residential information:**

* Store and update the information of the apartment tenant scientifically.
* Provides the function of viewing the history of service, sending feedback, and receiving notifications.
* Allow residents to send remote repair/maintenance requests.

**Guest management:**

* Allow temporary access registration to ensure security control.
* Display the available services for guests to find out information in advance.

**Maintenance and technical management:**

* Receive and assign repair requirements for technicians.
* Allow technical staff to update the problem of troubleshooting in real time.
* Help the Management Board monitor the maintenance status of the whole apartment area.

**Sanitation management:**

* Assign the task of cleaning each specific area.
* Cleaning staff update the work progress.
* Help ensure the clean environment and quality of life of residents.

**Security management:**

* Ensuring safety, security and quality of life of residents through strict management of activities in the apartment area.
* Support information access quickly when needed.

**Payment and accounting:**

* Residents can pay online service fees through integration of payment gateway (for example: MOMO).
* Accounting for invoice management, cash flow tracking and financial statements update.
* Reduce errors in the fee collection process and increase transparency.

**System administration:**

* System configuration, account creation, decentralization and all data management in the system.
* Send regular notifications to residents via email.
* Monitor system status, reports, monitoring and integrating payments.
* Improve the efficiency of troubleshooting, maintenance and service payment quickly and conveniently.

## **Document Conventions**

* Text styles: Times New Roman.
* All heading: written in **boldly**, size 18.
* All subheading: written in **bold**, size 15.
* Regular text: size 13.
* Important terms are written **boldly**.
* The agents of the system are written *in italic*.
* Priority Level:

- High: Essential for the system to function.

- Medium: Important but can be postponed.

- Low: Not urgent, can be added later.

## **Project Scope**

* The apartment management system is a web application (and can be extended to a mobile application) to manage and maintain all activities in the apartment building. The main functions include:
* Residents: Register and manage personal information, use, pay for services, send feedback and evaluate service quality.
* Maintenance, technical support: Residents send maintenance requests. The management board assigns tasks to technical staff.
* Notifications: The management board can send notifications and updates to residents via the system.
* Service management: Users can view available services and residents can pay online via the payment gateway.
* Data management and reporting of technical staff: Managers can set up system configuration, manage resident information, and monitor system status.
* Sanitation management: Can receive tasks and assign cleaning work to cleaning staff, update on-call schedules and view reports from service users.
* Security management: Security staff can record vehicle information and track access records.
* Finance, accounting: Accounting can handle the creation of invoices, update financial records.
* Management can compile statistics, approve requests, manage personnel and apartments.
* The system aims to digitize manual management processes, improve operational efficiency, improve resident experience and increase transparency in financial and operational activities.

## **References**

* **Use Case Diagram Guidelines** Guide to develop Use case diagrams for agents (Resident, Guest, Admin, Technical Staff, Accountant, Security Staff).

Link: <https://app.diagrams.net>

* **Payment Gateway API Documentation** ZaloPay API Documentation

MoMo API Documentation

# **Overall Description**

## **Product Perspective**

* The Apartment Management System is a newly developed web-based solution, with the potential to expand to a mobile platform. It is not a replacement for any existing system, but rather a digital transformation effort aimed at replacing manual and fragmented management processes in apartment buildings. The system will act as a centralized platform integrating key management functions such as resident services, maintenance, communication, and finance.
* If integrated into a larger smart city management platform in the future, this system would serve as a modular component focused on residential building operations. Interfaces to payment gateways, authentication services, and possibly smart home or security systems are anticipated and will be developed according to standard integration practices.

## **User Classes and Characteristics**

* **Admin**: Manages resident info, system configuration, status viewing.
* **Resident**: Submits maintenance requests, pays fees, views service, gives feedback.
* **Guest**: Registers temporary access, views services.
* **Accountant**: Manages invoices, processes payments, updates records.
* **Security Staff**: Verifies visitors, logs vehicles.
* **Management Board**: Views reports, manages data, sends notifications.
* **Technical Staff**: Receives tasks, updates maintenance progress.
* **Payment Gateway**: Processes payments, sends statuses.
* **Sanitation Managemen**t: Task reception and cleaning execution, work progress update.

## **Operating Environment**

The **Apartment Management System (AMS)** is designed to function in a modern, cloud-based or on-premises computing environment with support for multiple device types and platforms. It ensures compatibility and ease of use for all stakeholders, including residents, building staff, and administrators.

* Hardware Platform: desktop PCs, laptops, tablets, and smartphones
* Operating Systems: Windows, macOS, Android, iOS.
* Organization of storage: Managed by the management board or service management such as AWS, Azure, GCP.
* Related software: API of payment, notifications, maps, camera/protection systems.

## **2.4 Design and Implementation Constraints**

The following constraints must be considered:

* Technology Stack: The system will be developed using standard web technologies (e.g., JavaScript, Node.js, React, or Angular) and supported by a relational database (e.g., MySQL, PostgreSQL).
* Security Compliance: Must follow local data protection regulations.
* Platform Compatibility: Must support both desktop and mobile browsers.
* Language: Multilingual support is expected, with Vietnamese as the default language.
* Integration Requirements: Must integrate with third-party payment and notification APIs.

## **2.5 Assumptions and Dependencies**

Key assumptions and dependencies include:

* The availability and reliability of third-party services (payment gateways, authentication providers).
* Users will have access to a stable internet connection.
* The hosting infrastructure (servers, storage) will be provisioned and maintained by a dedicated IT team.
* All stakeholders will participate in user training and accept proposed workflows.
* Government regulations or technical standards may affect implementation and must be monitored throughout the project.

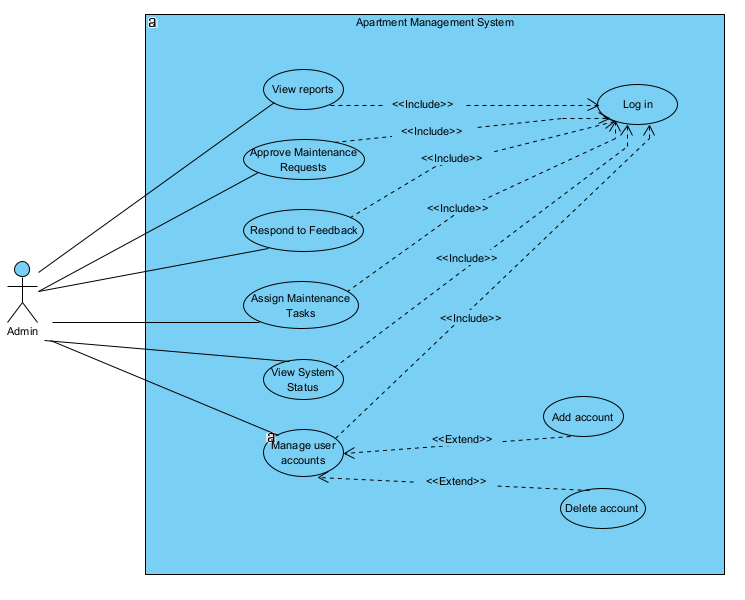
# **3. System Features**

**Context diagram:**

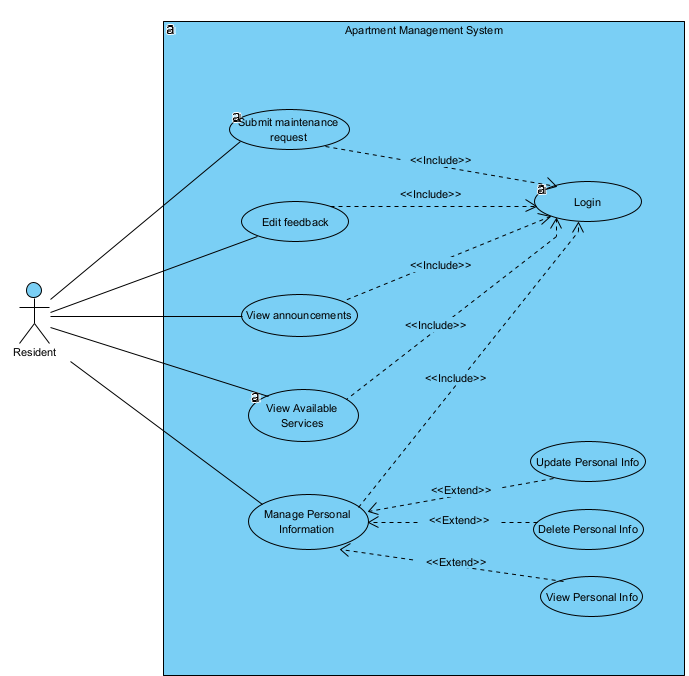
****

*Figure 3-1: Context diagram*

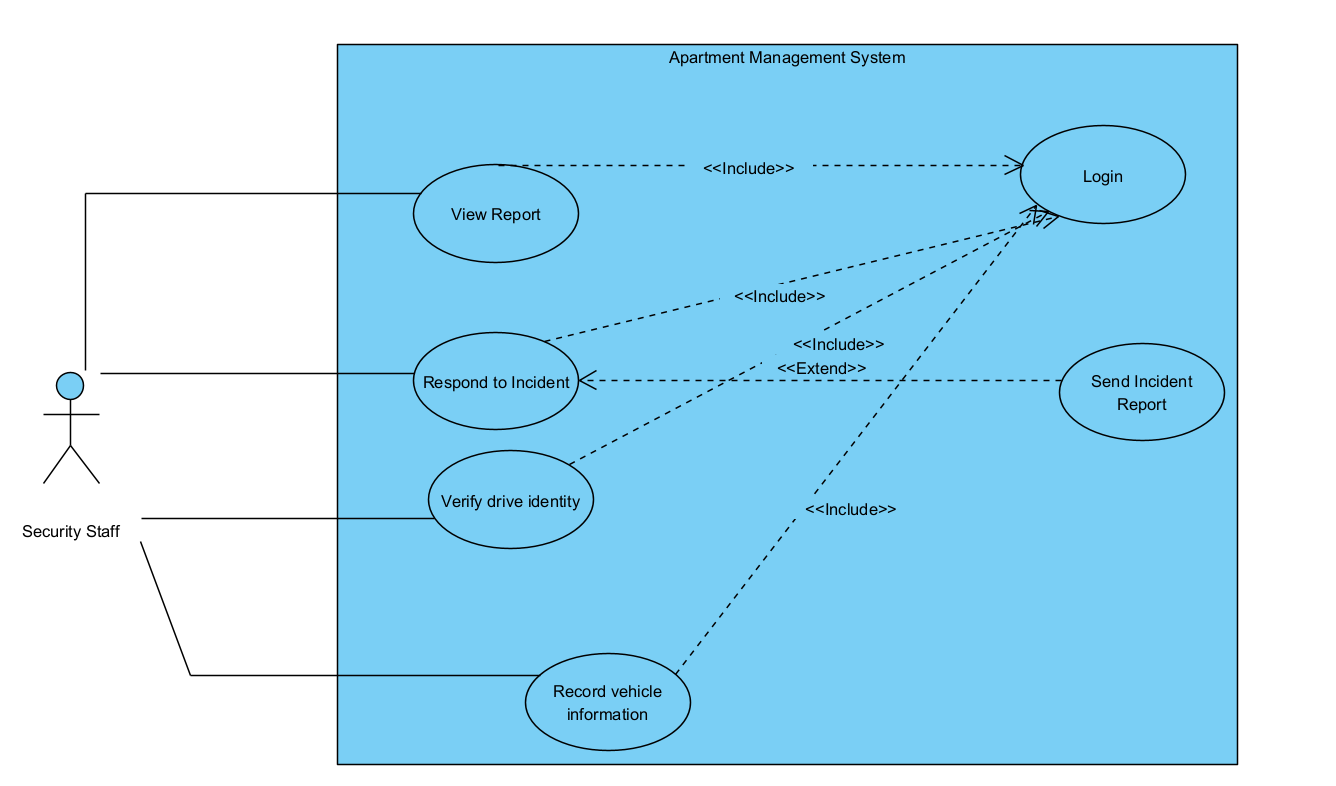
**Use case diagram:**

****

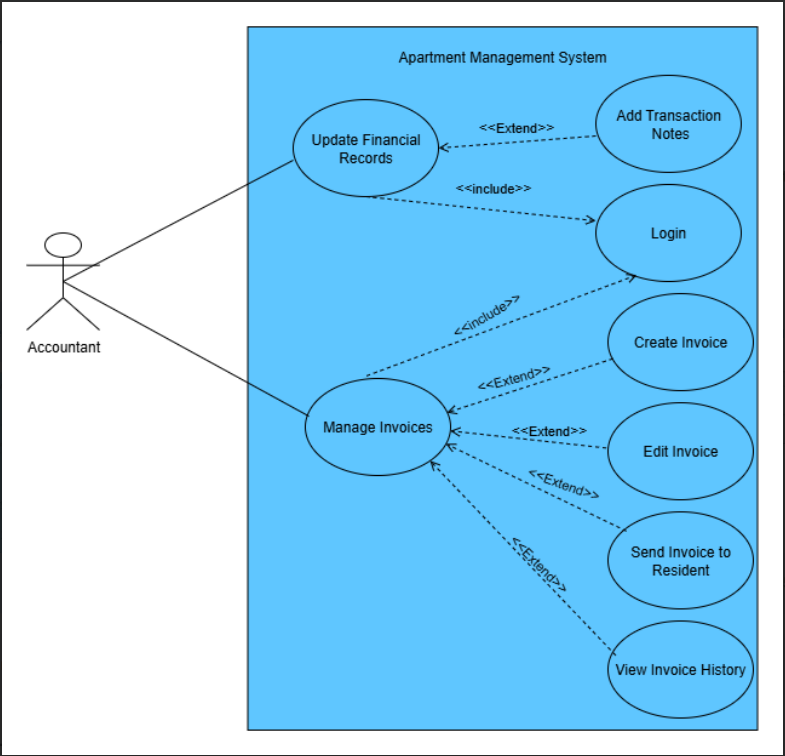
*Figure 3-2: Admin use case diagram*

****

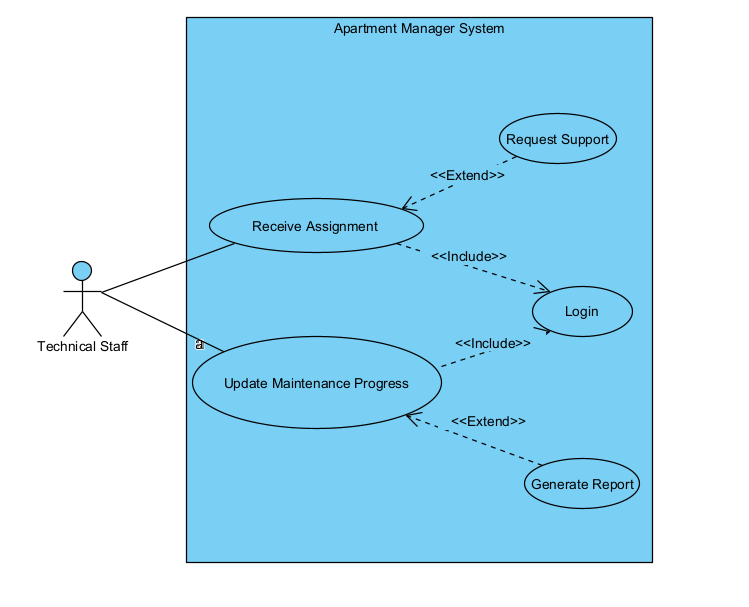
*Figure 3-3: Resident use case diagram*

****

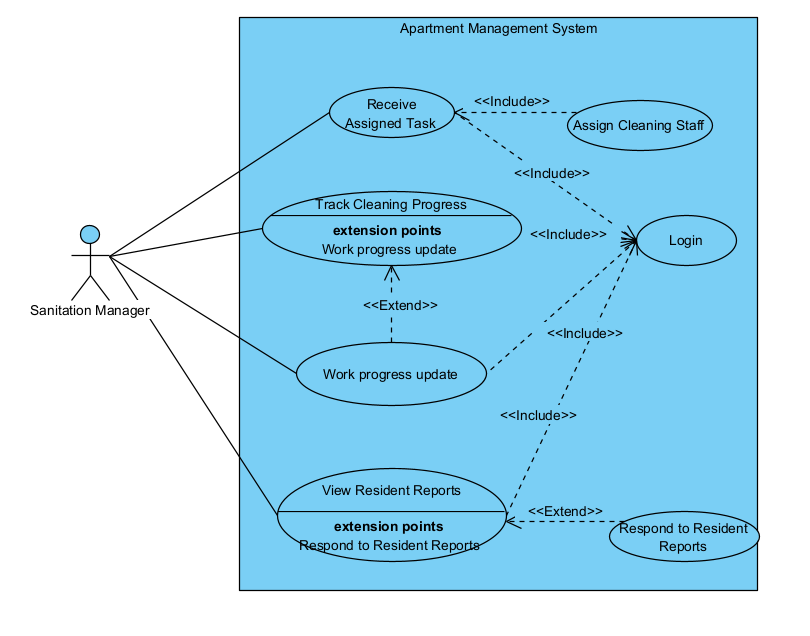
*Figure 3-4: Security Staff use case diagram*

****

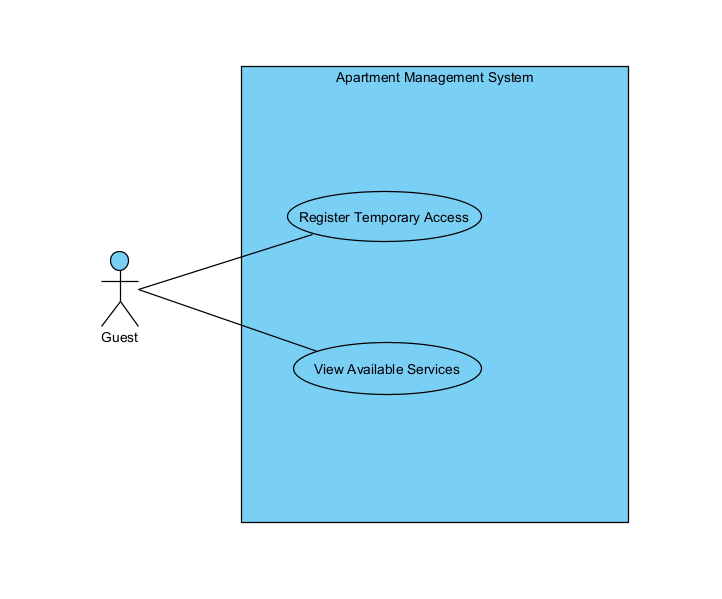
*Figure 3-5: Accountant use case diagram*

****

*Figure 3-6: Technical Staff use case diagram*

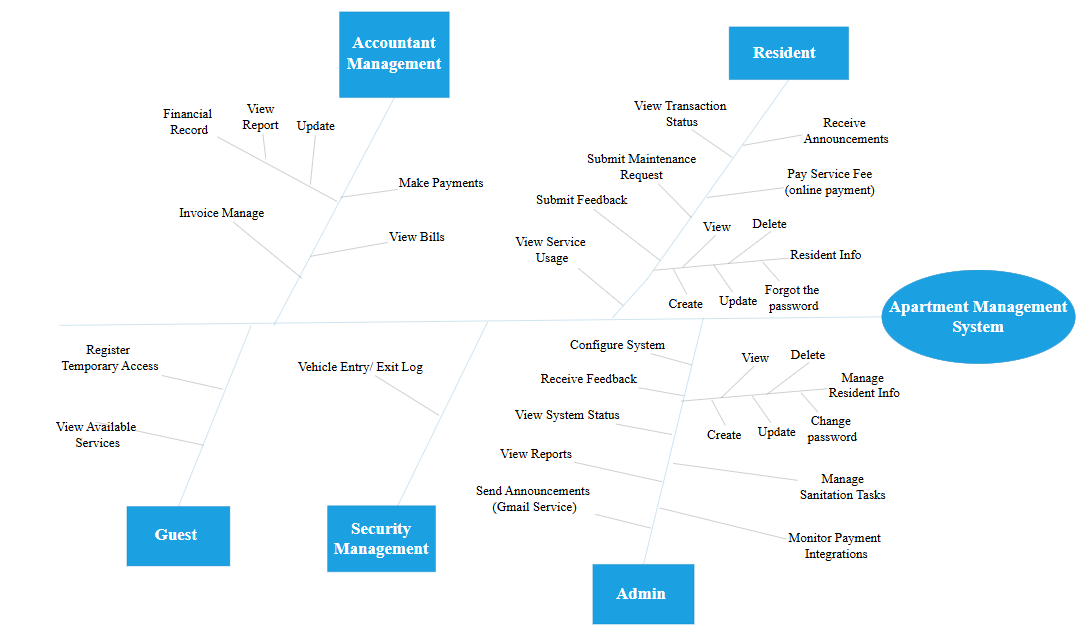
****

*Figure 3-:7 Sanitation Management use case diagram*

****

*Figure 3-10: Guest use case diagram*

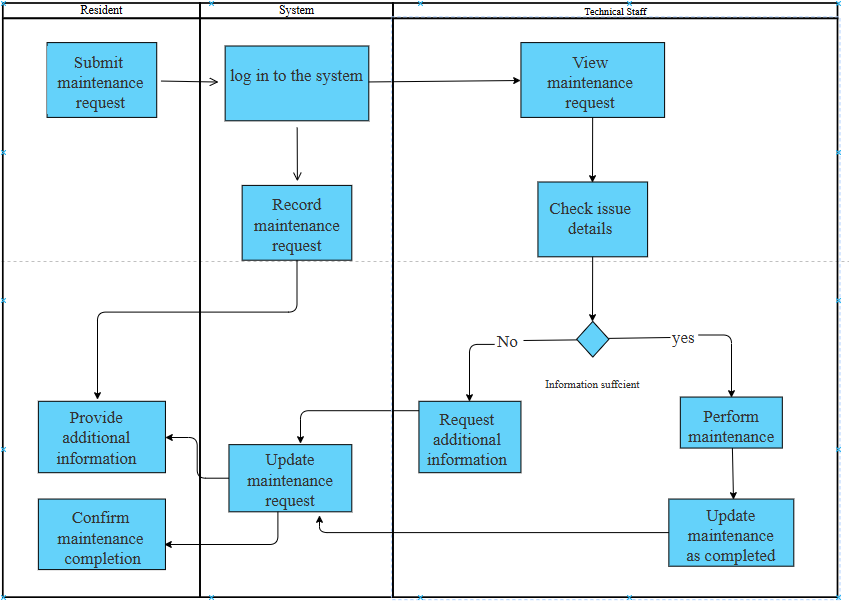
**System feature tree:**

*****Figure 3-11: System feature tree*

| **Feature Group** | **Feature** | **Description** |
| --- | --- | --- |
| **Resident** | Submit Maintenance Request | Submit maintenance requests for equipment or apartment issues. |
| Submit Feedback | Provide feedback about services or the building condition. |
| View Transaction Status | View the status of previously processed transactions (e.g., payments). |
| View Service Usage | Check service usage history. |
| Receive Announcements | Receive announcements from the management board. |
| Pay Service Fee (online payment) | Pay service fees through an integrated online payment gateway. |
| Resident Info: Create / Update / View / Delete / Forgot Password | Manage personal information and recover forgotten passwords. |
| **Admin** | Configure System | Set system settings such as email services, operating parameters, etc. |
| Receive Feedback | Receive and review feedback submitted by residents. |
| View System Status | Monitor system performance and status in real-time. |
| View Reports | Access reports related to maintenance, finance, residents, etc. |
| Send Announcements (Gmail Service) | Send announcements via integrated Gmail service to residents. |
| Manage Resident Info | Create, update, or delete resident records. |
| Change Password | Allow admin to change their login password. |
| Manage Sanitation Tasks | Assign and schedule cleaning/maintenance tasks. |
| Monitor Payment Integrations | Track and verify integrations with payment gateways (e.g., MoMo). |
| **Accountant Management** | Financial Record | Maintain and update financial data and balance sheets. |
| View Report | Access financial reports and payment summaries. |
| Update | Edit or revise financial records. |
| Invoice Manage | Manage billing information and generate service invoices. |
| Make Payments | Record or process payment transactions from residents. |
| View Bills | View detailed billing and charges for each resident or apartment unit. |
| **Security Management** | Vehicle Entry/Exit Log | Record vehicle entry and exit times for both residents and guests. |
| **Guest** | Register Temporary Access | Register and authorize guest access to the apartment during specific periods. |
| View Available Services | View the list of services provided by the apartment (read-only access). |

**Swim lane:**

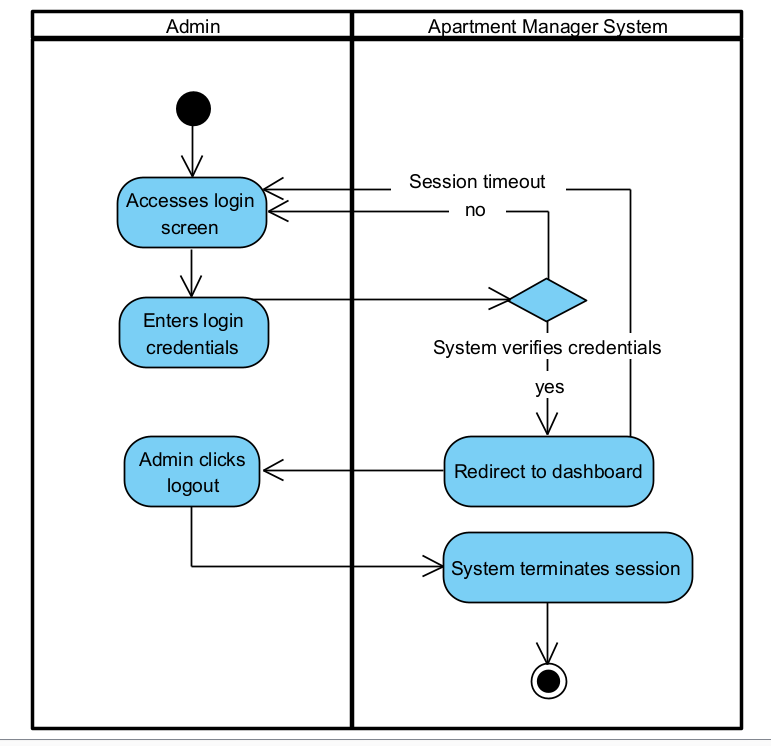
1. **Maintenance Request Workflow for Resident**

****

*Figure 3.12 – Resident Maintenance Request Process Swimlane Diagram*

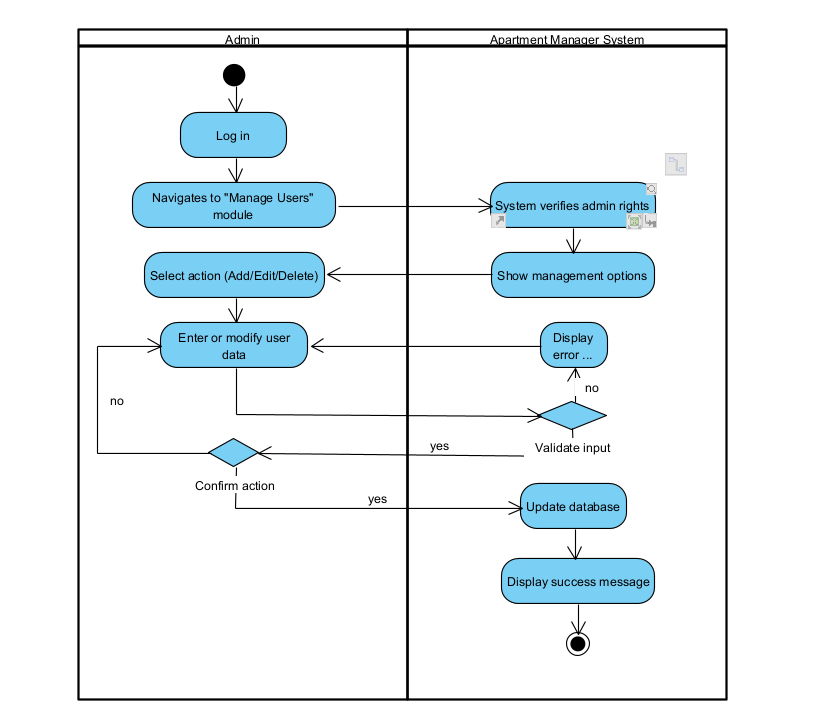
**Activity Diagram:**

1. **Admin Login and Session Management Process**

****

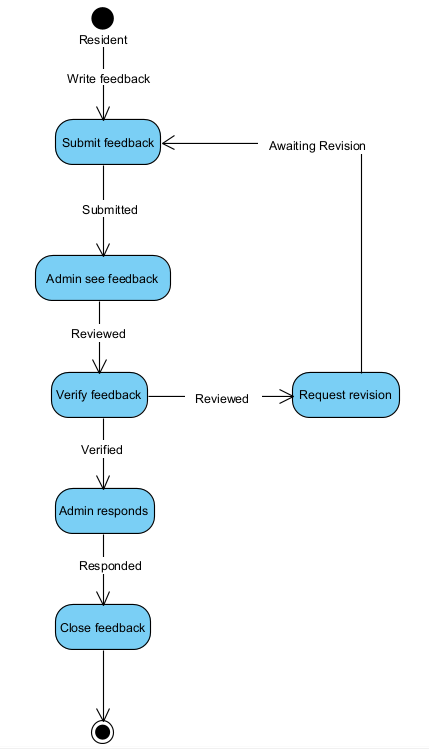
*Figure 3-12: Admin Activity Diagram*

1. **Admin User Management Activity Process**

****

*Figure 3-13: Admin Activity Diagram*

**State diagram:**

****

*Figure 3-14: State Transitions of Feedback*

## **3.1 Resident**

### **3.1.1 Description**

Residents are apartment users and the main user of the system. They interact with the system to monitor personal information, manage invoices, send maintenance requirements, and receive notifications from the Management Board. The goal is to improve the transparency and convenience in living in the apartment.

### **3.1.2 Stimulus/Response Sequences**

| **Event (Stimulus)** | **System Response** |
| --- | --- |
| Resident logs in | System verifies credentials and displays the dashboard |
| Resident logs out | System ends the session and returns to login screen |
| Resident submits a maintenance request | System saves the request and notifies the maintenance team |
| Resident views maintenance status | System retrieves and displays current status of the request |
| Resident views announcements | System shows a list of current and past announcements |
| Resident gives feedback | System stores feedback and optionally notifies admin |
| Resident views payment history | System displays history of payments and related invoices |
| Resident pays service fee | System processes payment and updates payment record |
| Resident updates personal information | System validates and stores updated information |
| Resident registers a vehicle | System records vehicle information and generates a tag (if applicable |
| Resident logs vehicle entry/exit | System records the timestamp and location for the vehicle |
| Resident changes password | System verifies old password and updates to new password |

### **3.1.3 Functional Log in**

| UC ID and Name: | UC01 - Log in | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Resident | Secondary Actors: | N/A |
| Trigger: | The resident opens the system and selects "Log in". | | |
| Description: | Allows residents to authenticate and access the system. | | |
| Preconditions: | The resident has valid login credentials. | | |
| Postconditions: | * The resident is successfully authenticated. * Granted access to other functions. | | |
| Normal Flow: | 1. Residents enter username and password. 2. System validates credentials. 3. The resident is logged in. | | |
| Alternative Flows: | * 2a. Incorrect credentials → show error message. * 2b. Account is locked → show lock notice. | | |
| Exceptions: | * Login service is temporarily unavailable. | | |
| Priority: | High | | |
| Frequency of Use: | Every session | | |
| Business Rules: |  | | |
| Other Information: |  | | |
| Assumptions: | Residents already registered in the system. | | |

### **3.1.4** **Function Submit Maintenance Request**

| UC ID and Name: | UC02 - Submit Maintenance Request | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Resident | Secondary Actors: |  |
| Trigger: | The resident notices an issue and wants to report it. | | |
| Description: | Allows the resident to submit a maintenance request to the management team. | | |
| Preconditions: | * The resident is logged in. | | |
| Postconditions: | * A maintenance request is submitted and logged in the system. | | |
| Normal Flow: | 1. Resident selects "Submit Maintenance Request". 2. Fills in details and submits. 3. The system confirms submission. | | |
| Alternative Flows: |  | | |
| Exceptions: | * System error during submission. | | |
| Priority: | High | | |
| Frequency of Use: | As needed | | |
| Business Rules: | BR-01, BR-02, BR-03 | | |
| Other Information: |  | | |
| Assumptions: | * The request contains complete details. | | |

### **3.1.5** **Function Manage Feedback**

| UC ID and Name: | UC03 - Manage Feedback | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Resident | Secondary Actors: |  |
| Trigger: | Residents want to give or update feedback about the apartment or services. | | |
| Description: | Allows Residents to add new feedback or edit previously submitted feedback. | | |
| Preconditions: | * Residents have successfully logged in. * There are feedback forms available on the system. | | |
| Postconditions: | * Feedback is stored in the system. * Feedback is added or updated successfully in the system | | |
| Normal Flow: | 1. Resident opens "Manage Feedback". 2. Chooses to either:   a. Add Feedback:  • Residents write new feedback and submits.  • The system stores the new feedback.  b. Edit Feedback:  • Residents select existing feedback.  • Makes changes and resubmits.  • The system updates the feedback. | | |
| Alternative Flows: | * 2a. Resident cancels action → Return to main screen. | | |
| Exceptions: | * Feedback not found for editing. * Validation error (e.g., empty content). * System error on submission. | | |
| Priority: | Medium | | |
| Frequency of Use: | Regular | | |
| Business Rules: | BR-04, BR-15, BR-16 | | |
| Other Information: |  | | |
| Assumptions: | * Residents are allowed to provide multiple feedback entries over time. | | |

### **3.1.6** **Function View announcement**

| UC ID and Name: | UC04 - View announcement | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Resident | Secondary Actors: |  |
| Trigger: | Residents log in the system and select the "view Announcement" function. | | |
| Description: | Allow residents to view important notices from the apartment management board such as repair notice, toll calendar, events, or security warnings. The notice is displayed in the order of time and clearly classified. | | |
| Preconditions: | * Residents have logged into the system. * The system has an active notice (posted by Admin). | | |
| Postconditions: | * Residents have seen the notice. * The state of "read" can be recorded if the system supports. | | |
| Normal Flow: | 1. Residents logged in to the system. 2. Access the screen "View Announcement". 3. The system displays the latest notice list. 4. Residents choose each notice to see details. 5. (Optional) Mark the notice read. | | |
| Alternative Flows: | * Residents filter notice by type: maintenance, fees, events, etc. * Search for notifications by keyword or date of posting. | | |
| Exceptions: | * There is no announcement in operation. * Connection error or data download error from the database. | | |
| Priority: | High | | |
| Frequency of Use: | Frequent | | |
| Business Rules: | BR-05, BR-06, BR-13, BR-14 | | |
| Other Information: | * There is an integrated notification system via email or push notification. * Support displaying notifications in language. | | |
| Assumptions: | * Residents use phones or computers with internet connection. * Admin regularly updates notifications on the system. | | |

## 

## **3.2** **Admin**

### **3.2.1 Description**

Admins are system administrators in the Apartment Management System. They are responsible for managing user accounts, assigning maintenance tasks, monitoring system status, viewing system reports, and logging in/out. The objective of the Admin is to ensure system stability, accurate data management, and efficient maintenance task delegation.

### **3.2.2 Stimulus/Response Sequences**

| **Event (Stimulus)** | **System Response** |
| --- | --- |
| Admin accesses the system | Login screen is displayed |
| Admin logs in successfully | Redirected to the main dashboard |
| Admin selects "Manage user accounts" | System shows user list and account creation functions |
| Admin selects "Assign maintenance" | Displays maintenance requests and available technical staff |
| Admin selects "View reports" | Displays report selection and relevant statistics |
| Admin selects "View system status" | Shows real-time system performance/status |

### **3.2.3 Functional Login system**

| UC ID and Name: | UC05 - Login | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | None |
| Trigger: | Admin opens the application and chooses to log in. | | |
| Description: | Allows admin to authenticate and access the system. | | |
| Preconditions: | * Users have a valid account in the system. * Applications or browsers are compatible and supported. | | |
| Postconditions: | Admin is successfully authenticated and redirected to the dashboard. | | |
| Normal Flow: | 1. Admin enters username and password.  2. System verifies credentials.  3. Admin is logged in. | | |
| Alternative Flow: | 2a. Invalid credentials → Show error message and retry. | | |
|
| Exceptions: | - Server error during login.  - Account locked. | | |
| Priority: | High | | |
| Frequency of Use: | Several times daily | | |
| Business Rules: | BR-11 | | |
| Other Information: | Use secure HTTPS, implement timeout and logging for security | | |
| Assumptions: | Admin account has been pre-registered in the system | | |

### 

### **3.2.4 Functional View reports**

| UC ID and Name: | UC06 - View reports | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | None |
| Trigger: | Admin selects “View reports”. | | |
| Description: | Allows admin to view various reports (e.g., maintenance history, user activity). | | |
| Preconditions: | Admin must be logged in. | | |
| Postconditions: | Reports are displayed. | | |
| Normal Flow: | 1. Admin login.  2. Admin selects type of report.  3. The system retrieves and shows the report. | | |
| Alternative Flows: | 2a. No data available → Show “No records found.” | | |
| Exceptions: | * Database connection failure * Unauthorized access attempt * Report generation failure. | | |
| Priority: | Medium | | |
| Frequency of Use: | Occasionally | | |
| Business Rules: | BR-17, BR-18 | | |
| Other Information: |  | | |
| Assumptions: | Report data exists in the system. | | |

### **3.2.5 Functional Approve Maintenance Requests**

| UC ID and Name: | UC07 - Approve Maintenance Requests | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | None |
| Trigger: | Admin views pending maintenance requests. | | |
| Description: | Admin can review and approve/reject maintenance requests from residents. | | |
| Preconditions: | * Admin is logged in. * Maintenance requests exist. | | |
| Postconditions: | The request status is updated. | | |
| Normal Flow: | 1. Admin login  2. Admin views list of pending requests.  3. Selects a request.  4. Approves or rejects it.  5. System updates status. | | |
| Alternative Flows: | 3a. Admin postpones decision. | | |
| Exceptions: | * Request not found. * Database update error. | | |
| Priority: | High | | |
| Frequency of Use: | Frequently | | |
| Business Rules: | BR-11, BR-12 | | |
| Other Information: |  | | |
| Assumptions: | Valid request exists. | | |

### **3.2.6 Functional Respond to Feedback**

| UC ID and Name: | UC08 - Respond to Feedback | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | Technical Staff |
| Trigger: | Admin reviews user-submitted feedback. | | |
| Description: | Admin can view and respond to feedback submitted by residents. | | |
| Preconditions: | Admin is logged in and feedback entries exist. | | |
| Postconditions: | Response is saved and sent to the resident. | | |
| Normal Flow: | 1. Admin selects a feedback item.  2. Read it and write a response.  3. Send the response. | | |
| Alternative Flows: | 2a. Admin marks feedback as read without responding. | | |
| Exceptions: | * Feedback not found. | | |
| Priority: | Medium | | |
| Frequency of Use: | Regular | | |
| Business Rules: | BR-04, BR-15, BR-16 | | |
| Other Information: |  | | |
| Assumptions: | Residents are allowed to submit feedback. | | |

### **3.2.7 Functional Assign Maintenance Tasks**

| UC ID and Name: | UC09 - Assign Maintenance Tasks | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | None |
| Trigger: | Admin decides to assign a task to a maintenance staff. | | |
| Description: | Admin assigns tasks to maintenance staff after request approval. | | |
| Preconditions: | * Request is approved. * Staff list is available. | | |
| Postconditions: | The task is assigned to a staff member. | | |
| Normal Flow: | 1. Admin login  2. Admin selects a request.  3. Chooses a staff.  4. Assign the task.  5. System updates status. | | |
| Alternative Flows: | 2a. No staff available → System shows notification. | | |
| Exceptions: | * Assignment fails due to server error. | | |
| Priority: | High | | |
| Frequency of Use: | Frequently | | |
| Business Rules: | BR-09, BR-10, BR-12 | | |
| Other Information: |  | | |
| Assumptions: | Staff have user accounts. | | |

### **3.2.8 Functional View System Status**

| UC ID and Name: | UC10 - View System Status | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | Technical Staff |
| Trigger: | Admin accesses the system health/status section. | | |
| Description: | Displays current system health, usage, and logs. | | |
| Preconditions: | Admin is logged in. | | |
| Postconditions: | Status info is shown. | | |
| Normal Flow: | 1. Admin opens System Status.  2. The system shows current metrics. | | |
| Alternative Flows: |  | | |
| Exceptions: | * Data not available. | | |
| Priority: | Medium | | |
| Frequency of Use: | Occasionally | | |
| Business Rules: | BR-08 | | |
| Other Information: |  | | |
| Assumptions: | System metrics are being tracked. | | |

### **3.2.8 Functional Manage user accounts**

| UC ID and Name: | UC11 - Manage user accounts | | |
| --- | --- | --- | --- |
| Created By: | DieuNH | Date Created: | 26/05/2025 |
| Primary Actor: | Admin | Secondary Actors: | Technical Staff |
| Trigger: | Admin selects the "Manage User Accounts" option from the main dashboard. | | |
| Description: | Admin can view the list of user accounts, add new users, update accounts or delete existing ones. | | |
| Preconditions: | * Admin is logged in. * Admin has sufficient privileges. | | |
| Postconditions: | The user account list is updated according to the admin’s actions (account added or deleted). | | |
| Normal Flow: | 1. Admin selects "Manage User Accounts" from the dashboard.  2. The system displays the list of current users.  3. Admin chooses to either:  a. Add Account:  • Admin clicks “Add User”.  • Enters user details (e.g., name, email, role, password).  • Click “Save”.  • The system validates and creates the account.  b. Delete Account:  • Admin selects a user from the list.  • Click “Delete”.  • The system asks for confirmation.  • Admin confirms deletion.  • The system removes the user. | | |
| Alternative Flows: | * 3a. Admin cancels the add or delete operation → System returns to the user list. * 3b. Admin performs search/filter instead of editing. | | |
| Exceptions: | * Email already exists when adding a user. * User not found when attempting to delete. * Database connection issue. | | |
| Priority: | High | | |
| Frequency of Use: | Frequently used when there are changes in personnel or account management is needed. | | |
| Business Rules: | BR-07 | | |
| Other Information: |  | | |
| Assumptions: | * The system provides real-time validation for user input. * Admin has full access to account management functionality. | | |

### 

## **3.3 Sanitation Management**

### **3.3.1 Description**

The **Sanitation Management** module is responsible for ensuring the cleanliness and hygiene of common areas within the apartment complex, such as hallways, staircases, elevators, lobbies, parking areas, and shared restrooms. This module enables the Management Board to schedule cleaning tasks, assign sanitation staff, track cleaning progress, and respond to resident-reported sanitation issues.

It provides a centralized system where residents can report sanitation problems (e.g., trash overflow, foul odor, or dirty areas), and the Management Board can approve and assign staff accordingly. Sanitation staff members can view their assigned tasks, update the status of cleaning jobs, and report any obstacles they encounter during execution.

### **3.3.2 Stimulus/Response Sequences**

### Below is a table listing the events (stimuli) and the corresponding responses of the Apartment Management System for the Sanitation Manager:

### 

| **Event (Stimulus)** | **System Response** |
| --- | --- |
| Sanitation Manager logs into the system | The system authenticates and grants access to the Sanitation Manager dashboard. |
| Sanitation Manager receives a task | The system displays task details and allows assignment to cleaning staff. |
| Sanitation Manager tracks cleaning progress | The system displays current progress and work points for selected tasks. |
| Sanitation Manager updates work progress | The system saves the update and notifies relevant staff if needed. |
| Sanitation Manager views resident reports | The system displays report details and allows response options. |
| Sanitation Manager responds to resident report | The system records the response and updates the report status. |

### **3.3.3 Schedule Cleaning Task**

| UC ID and Name: | UC12 - Schedule Cleaning Task | | |
| --- | --- | --- | --- |
| Created By: | HuyPH | Date Created: | 26/05/2025 |
| Primary Actor: | Sanitation Manager | Secondary Actors: | Resident, Technical Staff |
| Trigger: | The Sanitation Manager wants to create or update the cleaning schedule. | | |
| Description: | This use case describes how the Sanitation Manager sets or updates the cleaning schedule for various areas in the apartment complex (e.g., stairwells, lobbies, parking lots). | | |
| Preconditions: | - Sanitation Manager is authenticated.  - Areas to be scheduled exist in the system. | | |
| Postconditions: | - A cleaning schedule is created or updated.  - Notifications (if any) are sent to sanitation staff. | | |
| Normal Flow: | 1. Sanitation Manager logs in.  2. Navigates to "Sanitation Schedule" module.  3. Selects area and time slot.  4. Creates or modifies cleaning schedule.  5. Confirms and saves schedule.  6. System updates and notifies sanitation staff. | | |
| Alternative Flows: | - If schedule conflicts occur, the system suggests alternative times or staff. | | |
| Exceptions: | E1: System failure while saving schedule – error displayed, logs recorded. | | |
| Priority: | Medium | | |
| Frequency of Use: | Daily or weekly, depending on cleaning cycles. | | |
| Business Rules: |  | | |
| Other Information: | - Recurring schedules can be created (e.g., every Monday, Friday). | | |
| Assumptions: | - Management has access to cleaning zones and staff availability. | | |

### **3.3.4 Report Sanitation Issue**

| UC ID and Name: | UC13 - Report Sanitation Issue | | |
| --- | --- | --- | --- |
| Created By: | HuyPH | Date Created: | 26/05/2025 |
| Primary Actor: | Resident | Secondary Actors: | Sanitation Manager |
| Trigger: | A resident observes a sanitation-related issue and reports it. | | |
| Description: | This use case allows residents to report hygiene-related problems (e.g., overflowing garbage, spills, dirty elevators) to the management. | | |
| Preconditions: | - Resident is authenticated and logged in. | | |
| Postconditions: | - Sanitation issue is recorded.  - Management is notified. | | |
| Normal Flow: | 1. Resident logs into the system.  2. Navigates to "Report Issue" section.  3. Enters details, selects location, optionally attaches photo.  4. Submits report.  5. System stores report and notifies management. | | |
| Alternative Flows: | A1: If image attachment fails, system allows submission without it. | | |
| Exceptions: | E1: Invalid input – system prompts user to correct fields. | | |
| Priority: | High | | |
| Frequency of Use: | Occasionally – based on need | | |
| Business Rules: |  | | |
| Other Information: | - Residents can track the status of their report. | | |
| Assumptions: | - Residents are aware of how to use the reporting module. | | |

### **3.3.5 Update Cleaning Progress**

| UC ID and Name: | UC14 - Update Cleaning Progress | | |
| --- | --- | --- | --- |
| Created By: | HuyPH | Date Created: | 26/05/2025 |
| Primary Actor: | Sanitation Staff | Secondary Actors: | Sanitation Manager |
| Trigger: | Sanitation staff completes or updates the status of a cleaning task. | | |
| Description: | This use case allows sanitation staff to confirm, update, or report issues related to their assigned cleaning tasks. | | |
| Preconditions: | - A sanitation task has been assigned.  - Staff is authenticated. | | |
| Postconditions: | - Task status is updated (e.g., Completed, In Progress, Blocked). | | |
| Normal Flow: | 1. Staff logs in and views assigned tasks.  2. Selects a task.  3. Updates progress status.  4. Optionally adds notes (e.g., obstacles encountered).  5. System records the update and notifies Management. | | |
| Alternative Flows: | A1: If area is inaccessible, staff can report the reason with optional photo. | | |
| Exceptions: | E1: Task already marked as completed – system prevents re-editing. | | |
| Priority: | High | | |
| Frequency of Use: | Multiple times per day | | |
| Business Rules: | BR-31 | | |
| Other Information: | - Management uses this data to track sanitation performance. | | |
| Assumptions: | - Staff have access to mobile app or web portal for status update. | | |

## **3.4 Accountant**

### **3.4.1 Description**

Accountants are responsible for managing the financial operations of the apartment complex. They interact with the Apartment Management System to log in, update financial records, and manage invoices. Their tasks include adding transaction notes, creating and editing invoices, sending invoices to residents, and viewing invoice history. The goal is to ensure accuracy and transparency in all financial records and billing activities.

### **3.4.2 Stimulus/Response Sequences**

| Event (Stimulus) | System Response |
| --- | --- |
| Log in | Authenticate and display the accountant dashboard |
| Select "Update Financial Records" | Display the financial records interface |
| Add transaction notes | Open transaction note form for input |
| Submit financial updates | Validate and save to the financial database |
| Select "Manage Invoices" | Open the invoice management interface |
| Create/edit invoice | Save or update invoice and notify resident |
| View invoice history | Display the list of past invoices |
| Send invoice to resident | Deliver invoice via notification or email |

**3.4.3 Functional Requirement: Update Financial Records**

| UC ID and Name: | UC15 - Financial record update | | |
| --- | --- | --- | --- |
| Created By: | PhanNV | Date Created: | 26/05/2025 |
| Primary Actor: | Accountant | Secondary Actors: | None |
| Trigger: | Accountant logs in to update financial information | | |
| Description: | This use case describes the process by which an accountant updates financial records in the system, including payments, revenue logs, expenses, and related notes. | | |
| Preconditions: | - Accountant is authenticated.  - The system is online and connected to the financial database. | | |
| Postconditions: | - Financial records are updated and saved.  - Changes are logged with user info and timestamp. | | |
| Normal Flow: | 1. The accountant logs in.  2. Navigates to the "Update Financial Records" module.  3. Adds or edits a financial entry.  4. Optionally adds transaction notes.  5. Save the changes.  6. The system confirms the update. | | |
| Alternative Flows: | - A1: Operation is canceled → no changes made.  - A2: Existing data is being updated → system prompts for overwrite confirmation | | |
| Exceptions: | - E1: System/database error → display error message and log failure.  - E2: Invalid data format → highlight field and request correction. | | |
| Priority: | High | | |
| Frequency of Use: | Daily | | |
| Business Rules: | BR-19, BR-20 | | |
| Other Information: | - System can generate reports automatically from financial records | | |
| Assumptions: | - Accountants have sufficient system training.  - The system includes validation and security mechanisms. | | |

**3.4.4 Functional Requirement: Invoicing Management**

| UC ID and Name: | UC16 - Invoicing Management | | |
| --- | --- | --- | --- |
| Created By: | PhanNV | Date Created: | 26/05/2025 |
| Primary Actor: | Accountant | Secondary Actors: | Resident |
| Trigger: | Accountant initiates invoice creation or management. | | |
| Description: | This use case defines how the accountant creates, edits, sends, and views invoices within the system. This includes charges for services, utilities, and other fees | | |
| Preconditions: | - Accountant is authenticated. - Resident and billing data must be available. | | |
| Postconditions: | - Invoices are created or updated.  - Residents are notified via the system. | | |
| Normal Flow: | 1. Accountant accesses "Manage Invoices" module.  2. Selects residents or units for billing.  3. Enter invoice details.  4. Creates or edits the invoice.  5. Send an invoice to the resident.  6. Views invoice history.  7. The system confirms and stores the invoice. | | |
| Alternative Flows: | - A1: Resident has unpaid invoices → system alerts accountant.  - A2: Accountant downloads or prints invoice.. | | |
| Exceptions: | - E1: Resident data missing → system prevents invoice creation.  - E2: System/database error → retry and show error. | | |
| Priority: | High | | |
| Frequency of Use: | Monthly or as needed. | | |
| Business Rules: | BR-21, BR-22 | | |
| Other Information: | - Invoices can be exported as PDFs.  - Residents can access invoices via the portal. | | |
| Assumptions: | - Accurate billing data is available.  - Residents are notified upon invoice generation. | | |

## **3.5** **Security Staff**

### **3.5.1 Description**

Security Staff are responsible for monitoring safety and controlling access within the apartment complex. They interact with the system to perform tasks such as surveillance viewing, incident response, guest verification, and vehicle tracking. The system supports quick and accurate decision-making to ensure security for all residents.

### **3.5.2 Stimulus/Response Sequences**

| **Event (Stimulus)** | **System Response** |
| --- | --- |
| Security staff logs in | System verifies credentials and displays the security dashboard |
| Security staff logs out | System ends the session and returns to login screen |
| Security staff logs vehicle entry | System records vehicle entry with timestamp and location |
| Security staff logs vehicle exit | System records vehicle exit with timestamp and location |
| Security staff views vehicle log history | System retrieves and displays list of vehicle entries and exits |
| Security staff updates access information | System validates and saves access data to resident or visitor profile |
| Security staff reports incident | System stores incident report and notifies the management/admin |

**3.6.3** **Function View Surveillance**

| UC ID and Name: | UC17 – View Surveillance | | |
| --- | --- | --- | --- |
| Created By: | SonDH | Date Created: | 01/06/2025 |
| Primary Actor: | Security Staff | Secondary Actors: | None |
| Trigger: | Staff selects “Surveillance” from dashboard | | |
| Description: | Displays live and recorded security camera feeds | | |
| Preconditions: | Staff is authenticated and cameras are connected | | |
| Postconditions: | Surveillance data is displayed and may be recorded if needed | | |
| Normal Flow: | 1. Log in  2. Select “Surveillance” from dashboard  3. Choose desired camera  4. View live or recorded footage | | |
| Alternative Flows: | If connection to camera fails, system alerts user | | |
| Exceptions: | Video playback error or camera offline | | |
| Priority: | High | | |
| Frequency of Use: | As needed for real-time monitoring | | |
| Business Rules: | BR-23 | | |
| Other Information: | Footage stored securely for up to 30 days | | |
| Assumptions: | Cameras are positioned and connected properly | | |

**3.5.4** **Function Respond to Incidents**

| UC ID and Name: | UC18 – Respond to Incidents | | |
| --- | --- | --- | --- |
| Created By: | SonDH | Date Created: | 01/06/2025 |
| Primary Actor: | Security Staff | Secondary Actors: | Residents |
| Trigger: | An incident is reported through the system | | |
| Description: | Allows Security to view, acknowledge, and respond to incidents | | |
| Preconditions: | Incident has been logged by resident or system | | |
| Postconditions: | Incident status is updated (e.g., responded, resolved) | | |
| Normal Flow: | 1.Notification received through the system  2. Security staff views incident details  3. Takes appropriate action (e.g., contact resident, inspect location) | | |
| Alternative Flows: | Incident escalated to Management Board if unresolved or critical | | |
| Exceptions: | Incomplete incident data | | |
| Priority: | Very High | | |
| Frequency of Use: | Upon any reported or detected safety/security incident | | |
| Business Rules: | BR-24 | | |
| Other Information: | All actions and responses must be logged and timestamped | | |
| Assumptions: | Security staff is available and on-site when incident occurs | | |

**3.5.5** **Function Verify Identity**

| UC ID and Name: | UC19 – Verify Guest Identity | | |
| --- | --- | --- | --- |
| Created By: | SonDH | Date Created: | 16/07/2025 |
| Primary Actor: | Security Staff | Secondary Actors: | Guests |
| Trigger: | A guest requests temporary access | | |
| Description: | Confirms guest identity using system-registered information | | |
| Preconditions: | Guest information already submitted by resident | | |
| Postconditions: | Guest is approved or denied access | | |
| Normal Flow: | 1. Security opens guest verification  2. Enters guest details  3. Confirms with system data | | |
| Alternative Flows: | Manual identity check if system is down | | |
| Exceptions: | Invalid or unmatched identity information | | |
| Priority: | Medium | | |
| Frequency of Use: | For each guest access request | | |
| Business Rules: | BR-28 | | |
| Other Information: | Photo ID may be requested for verification | | |
| Assumptions: | Residents pre-register their guests | | |

**3.5.6 Functional Log Vehicle Entry/Exit**

| UC ID and Name: | UC20 – Log Vehicle Entry/Exit | | |
| --- | --- | --- | --- |
| Created By: | SonDH | Date Created: | 01/06/2025 |
| Primary Actor: | Security Staff | Secondary Actors: | None |
| Trigger: | Vehicles of residents or guests enter or exit the apartment building. | | |
| Description: | Allows security staff to record vehicle details (e.g., license plate, type, time in/out, and owner info) for monitoring and protection. | | |
| Preconditions: | * Security system is active and staff is logged in * Residents have registered their vehicles (for resident entries) | | |
| Postconditions: | * Entry/exit logs are stored successfully in the system * Data can be retrieved for review when needed | | |
| Normal Flow: | 1. Security staff logs in 2. At the gate, a vehicle approaches 3. Staff performs inspection and enters or scans code (QR/barCode) if available 4. Record vehicle info: license plate, type, time in/out, and owner (resident/guest) 5. System saves the record to the database 6. Print temporary entry card for guest vehicles | | |
| Alternative Flows: | Manual entry if QR/barcode is unreadable or not available. | | |
| Exceptions: | System offline or failure to retrieve vehicle. | | |
| Priority: | High | | |
| Frequency of Use: | Every time a vehicle enters or exits the apartment. | | |
| Business Rules: | BR- 26 | | |
| Other Information: | Entry/exit data retained securely and maybe reviewed by management. | | |
| Assumptions: | Vehicles are registered correctly and security staff are trained ọn system usage. | | |

**3.5.7 Functional record vehicle information**

| UC ID and Name: | UC21 – Record Vehicle Information | | |
| --- | --- | --- | --- |
| Created By: | SonDH | Date Created: | 01/06/2025 |
| Primary Actor: | Security Staff | Secondary Actors: | Residents |
| Trigger: | A resident registers a new vehicle or updates vehicle information through the system. | | |
| Description: | Allows security staff to view, verify, and record resident vehicle information for access control and parking management | | |
| Preconditions: | -Resident must be registered in the system  -Vehicle registration request or update must be submitted | | |
| Postconditions: | -Vehicle information is stored in the system and linked to the resident  -Vehicle access permissions (e.g., entry, parking area) are updated | | |
| Normal Flow: | 1.Security staff receives notification of vehicle registration request 2.Security staff views and verifies submitted vehicle details 3.Confirms and records vehicle information in the system 4.Grants appropriate access/parking permissions | | |
| Alternative Flows: | -Vehicle information rejected due to invalid or incomplete details → resident notified  -Request forwarded to Management Board for manual approval if required (e.g., oversized vehicle) | | |
| Exceptions: | -Missing or invalid documents (e.g., license plate, proof of ownership)  -Resident not found in system | | |
| Priority: | High | | |
| Frequency of Use: | Whenever a resident registers or updates vehicle information | | |
| Business Rules: | BR-27 | | |
| Other Information: | All updates must be logged and timestamped for audit purposes | | |
| Assumptions: | -Security staff has system access and training  -Vehicle registration data is provided accurately by the resident | | |

## **3.6 Technical taff**

### **3.6.1 Description**

Technical Staff manage and maintain the apartment management system. They handle technical issues, perform updates, and support residents and management to ensure smooth system operation and security.

### **3.6.2 Stimulus/Response Sequences**

| **Stimulus (Event)** | **Response (Action)** |
| --- | --- |
| System error or malfunction | Diagnose and fix the issue promptly |
| Request for software update | Plan and carry out the update |
| Resident or management reports a problem | Receive, assist, and resolve the issue |
| Scheduled system maintenance | Perform checks and report results |
| Security threat detected | Apply fixes and alert users |
| Request for technical help | Provide support and guidance |

### **3.6.3** **Function Receive assignment**

| UC ID and Name: | UC22 - Receive assignment | | |
| --- | --- | --- | --- |
| Created By: | Tungvh | Date Created: | 26/05/2025 |
| Primary Actor: | Technical Staff | Secondary Actors: | Resident, Management Board |
| Trigger: | Technical Staff receives a new task or assignment related to maintenance or technical support from the management system. | | |
| Description: | This use case describes how Technical Staff receive and acknowledge assignments related to maintenance, repair, or system support in the condominium management system. | | |
| Preconditions: | Technical Staff must be logged into the system and have an active account. | | |
| Postconditions: | Assignment is logged, Technical Staff acknowledges receipt, and task details are available for follow-up. | | |
| Normal Flow: | 1. The system notifies Technical Staff of a new assignment.  2. Technical Staff reviews the task details.  3. Technical Staff acknowledges the assignment.  4. The system updates the task status as "Received" or "In Progress". | | |
| Alternative Flows: | - If the system notification fails, Technical Staff can manually check the assignment list.  - If assignment details are incomplete, Technical Staff requests clarification from management. | | |
| Exceptions: | System failure preventing notification delivery. | | |
| Priority: | High | | |
| Frequency of Use: | Frequent, whenever new tasks are assigned. | | |
| Business Rules: | BR-10, BR-12 | | |
| Other Information: | - Technical Staff assignments cover maintenance, repairs, system updates, and technical support. | | |
| Assumptions: | - System is operational and connected.  - Technical Staff have valid permissions. | | |

### **3.7.4** **Function Update maintenance processing request**

| UC ID and Name: | UC23 - Update maintenance processing request | | |
| --- | --- | --- | --- |
| Created By: | Tungvh | Date Created: | 26/05/2025 |
| Primary Actor: | Technical Staff | Secondary Actors: | Resident, Management Board |
| Trigger: | Technical Staff updates progress or status of a maintenance request. | | |
| Description: | Technical Staff updates status, progress, and details of maintenance requests in the system. | | |
| Preconditions: | Technical Staff is logged in and assigned the request. | | |
| Postconditions: | Maintenance request status and updates are saved and visible to stakeholders. | | |
| Normal Flow: | 1. Access assigned request.  2. Update status (e.g., in progress, completed).  3. Add work details and comments.  4. Attach documents/photos if needed.  5. Save updates and notify relevant parties. | | |
| Alternative Flows: | - If required information is missing, the system prompts Technical Staff to provide it.  - If the system fails to save the update, Technical Staff retries or reports the issue to IT support. | | |
| Exceptions: | System failure during the update process. | | |
| Priority: | High | | |
| Frequency of Use: | Frequent, whenever maintenance progresses. | | |
| Business Rules: | BR-29, BR-30, BR-31 | | |
| Other Information: | -Updates enhance transparency and accountability in maintenance processes. | | |
| Assumptions: | - System is operational and accessible.  - Technical Staff has necessary permissions. | | |

## **3.7 Guest**

### **3.7.1 Description**

A non-resident who visits the apartment temporarily. Guests register to request access and receive security notices. Their access is controlled by the Management Board to ensure safety.

### **3.7.2 Stimulus/Response Sequences**

| **Event** | **System Response** |
| --- | --- |
| Guest submits registration | System saves info and notifies management. |
| Guest requests access | The system verifies and grants or denies access. |
| Management approves access | System updates status and notifies security. |
| Guest receives notifications | The system sends security rules and updates. |
| Guest enters the complex | The system checks access and logs entry. |
| Guest leaves the complex | System logs exit time. |
| Unauthorized entry attempt | The system denies access and alerts security. |

### **3.7.3** **Function Register Temporary Access**

| UC ID and Name: | UC24 - Register Temporary Access | | |
| --- | --- | --- | --- |
| Created By: | Tungvh | Date Created: | 26/05/2025 |
| Primary Actor: | Guest | Secondary Actors: | Admin, Resident |
| Trigger: | Guest requests access or Admin/Resident adds Guest | | |
| Description: | Temporary user with limited access for visitor registration or short-term use. Cannot change resident data. | | |
| Preconditions: | Verified by the Management Board or hosting resident. | | |
| Postconditions: | Guest info stored with access rights and expiry. | | |
| Normal Flow: | 1. Request access or add a Guest account.  2. Verify info.  3. Create an account with limited rights.  4. Use access during allowed time. | | |
| Alternative Flows: | Deny account if verification fails. | | |
| Exceptions: | Invalid or incomplete Guest info. | | |
| Priority: | Low | | |
| Frequency of Use: | Occasional | | |
| Business Rules: | BR-25 | | |
| Other Information: | Access is temporary and time-limited. | | |
| Assumptions: | Managed by Admin or authorized residents only. | | |

### **3.7.4** **Function View available services**

| UC ID and Name: | UC25 - View available services | | |
| --- | --- | --- | --- |
| Created By: | Tungvh | Date Created: | 26/05/2025 |
| Primary Actor: | Guest | Secondary Actors: | Admin, Resident |
| Trigger: | Guest requests to view available services. | | |
| Description: | Guests can view a list of services offered without making changes. | | |
| Preconditions: | Guests have valid temporary access. | | |
| Postconditions: | Services displayed to guests successfully. | | |
| Normal Flow: | 1. Guest opens services page.  2. The system shows available services.  3. Guest views details. | | |
| Alternative Flows: | Show error if services fail to load. | | |
| Exceptions: | Access denied if Guest access expired. | | |
| Priority: | Medium | | |
| Frequency of Use: | Frequent | | |
| Business Rules: | BR-32 | | |
| Other Information: | Service info is updated and verified. | | |
| Assumptions: | System operational and Guest online. | | |

### **Table Business rule:**

| **ID** | **Rule Definition** | **Type of rule** | **Static or dynamic** | **Source** |
| --- | --- | --- | --- | --- |
| BR-01 | Each requirement must be associated with residents ID and time sent. | Fact | Static |  |
| BR-02 | Requirements cannot be edited after sending, and can only be canceled if not processed. | Constraint | Static |  |
| BR-03 | The system must classify the priority level of the requirements based on the type of problem. | Action Enabler | Dynamic |  |
| BR-04 | Each response must contain at least text content. | Constraint | Static |  |
| BR-05 | The notice must have titles, content, and date of posting. | Constraint | Static |  |
| BR-06 | Some notifications can attach documents (PDF, Photo). | Fact | Dynamic |  |
| BR-07 | Each user must have a unique email - Only Admin can manage users | Constraint | Static |  |
| BR-08 | Only Admin can access the system status panel - Status must be updated in real-time or near real-time. | Constraint | Dynamic |  |
| BR-09 | Only unassigned requests can be assigned. | Constraint | Static |  |
| BR-10 | Each request can only be assigned to one staff at a time. | Constraint | Static |  |
| BR-11 | Only the Management Board can approve or reject maintenance requests. | Constraint | Static |  |
| BR-12 | Approved requests must be assigned to a qualified technical staff member. | Constraint | Dynamic |  |
| BR-13 | Announcements must include a title, content, and posting date. | Constraint | Static |  |
| BR-14 | Critical announcements (e.g., security alerts) must be marked as high priority. | Action Enabler | Static |  |
| BR-15 | Feedback must be responded to within 1 business day. | Constraint | Dynamic |  |
| BR-16 | Responses must be professional and include actionable steps if applicable. | Constraint | Static |  |
| BR-17 | Reports must include timestamps and detailed data for accuracy. | Constraint | Static |  |
| BR-18 | Exported reports must be secure and not editable. | Constraint | Static |  |
| BR-19 | Only authorized users (Accountants) may update financial records. | Constraint | Static |  |
| BR-20 | All changes must be logged for audit purposes. | Action Enabler | Dynamic |  |
| BR-21 | Only accountants can manage invoices. | Constraint | Static |  |
| BR-22 | Invoices must include service breakdown and payment deadline. | Constraint | Static |  |
| BR-23 | Only staff with camera access privileges may view footage. | Constraint | Static |  |
| BR-24 | Must be acknowledged within 5 minutes. | Constraint | Dynamic |  |
| BR-25 | Access is only granted during registered time slots. | Constraint | Static |  |
| BR-26 | Protection only has the right to record the car diary. | Constraint | Static |  |
| BR-27 | The data must be recorded with the exact date and time. | Constraint | Static |  |
| BR-28 | The system must distinguish residents and guests. | Fact | Static |  |
| BR-29 | Only assigned Technical Staff can update the request. | Constraint | Static |  |
| BR-30 | Status updates must follow company-approved categories. | Constraint | Static |  |
| BR-31 | Updates must be timely and accurate. | Constraint | Dynamic |  |
| BR-32 | Guests have read-only access to services. | Constraint | Static |  |

# **4. Data Requirements**

This section explains the types of data the system will handle, how it will store and process this data, and what the system will output.

## **4.1 Logical Data Model**

**4.2 Data Dictionary**

The Data Dictionary describes the structure of the data, like the type, length, and allowed values for each data element. It helps to clearly define what each piece of data means and how it should be used.

* Example:
* Resident ID: Data type: Integer, Length: 6 digits, Description: A unique ID for each resident.
* Payment Status: Data type: String, Allowed values: "Paid", "Unpaid", "Pending".

The data dictionary should be kept separate for easier reference and reuse in other projects.

## **4.3 Reports**

If the system generates reports (like maintenance status, payment summaries), this section should describe them and their characteristics.

* Example Report:
* Maintenance Status Report: A list of open maintenance requests, sorted by priority, showing their status and assigned technician.

You don’t need to define the exact layout of the reports now, but you should describe what the reports will include and any specific requirements for sorting or organizing the data.

## **4.4 Data Acquisition, Integrity, Retention, and Disposal**

This section describes how data will be collected, maintained, and safely deleted when no longer needed.

* Acquisition: How the system will collect and enter data (e.g., through forms, uploads).
* Integrity: Ensures the accuracy and correctness of data (e.g., checking for missing or incorrect information).
* Retention: How long the data will be stored (e.g., transaction records for 7 years).
* Disposal: How data will be securely deleted when it's no longer needed (e.g., deleting old resident data).

This part also covers techniques like backups or data accuracy checks to protect data.

## **5. External Interface Requirements**

### **5.1 User Interfaces**

The Apartment Management System will provide a web-based Graphical User Interface (GUI) for all user roles, including Admin, Resident, Technical Staff, Security Staff, Guest, Accountant, and Sanitation Management.

* **User Roles & Interfaces**:
  + **Residents**: Login, view service usage, submit maintenance requests, make payments, and send feedback.
  + **Admin**: Manage user info, configure system, monitor status.
  + **Guests**: Register temporary access and view available services.
  + **Staff**: Access assignments, update maintenance or sanitation progress.
  + **Accountant**: Manage invoices and financial updates.
  + **Security Staff**: Log vehicle entry/exit.
* **Common UI Features**:
  + Consistent layout with a top navigation bar and left sidebar menu.
  + Standard buttons: **Submit**, **Cancel**, **Back**, **Help**.
  + Error messages displayed in red under the relevant fields.
  + Keyboard shortcuts: Tab (navigate), Enter (submit), Esc (cancel).
  + Toast notifications for success/failure messages.
  + Responsive design for mobile/tablet compatibility.
* **UI Standards**:
  + Follow Material Design Guidelines.
  + Font: Roboto, Size: min 14px.
  + Color Theme: Blue-Gray with accent colors for status indicators.
* **Sample Screens** (to be included in the UI Specification Document):
  + Login Screen
  + Resident Dashboard
  + Maintenance Request Form
  + Admin Configuration Panel
  + Payment Confirmation Page

### **5.2 Software Interfaces**

The system will integrate with the following software components:

* **Database**:
  + **MySQL 8.0** used for storing user data, requests, transaction records, and feedback.
  + Communication via JDBC with SQL queries and stored procedures.
* **Payment Gateway (MOMO)**:
  + Purpose: Handle online payment transactions.
  + Format: JSON-based REST AP
  + Sent data: amount, userId, invoiceId, timestamp, signature
  + Response: status, transactionId, message
  + Secure communication using HTTPS with OAuth2 token authentication
* **Mail Sender Service**:
  + Used to send announcements and feedback confirmation.
  + Format: SMTP Protocol, MIME encoding for email content.
  + Required fields: to, subject, body, from
* **Authentication**:
  + Role-based access control integrated with Spring Security Framework.
  + JWT (JSON Web Token) used for authentication and session management.
* **Front-end**:
  + Vue.js or React-based single-page application (SPA) consuming RESTful APIs.

### **5.3 Hardware Interfaces**

This is a web-based application with minimal direct hardware interaction, except:

* **Security Staff Devices**:
  + Tablet or mobile devices used at entry/exit gates.
  + Used to log vehicle information.
  + Interface through responsive web forms.
* **Recommended Hardware for Hosting**:
  + CPU: Quad-core (min 2.5 GHz)
  + RAM: 16GB
  + Storage: SSD 256GB+
  + OS: Ubuntu Server 20.04+ or Windows Server 2019+
* **Interaction with Barcode/QR Scanner (Optional)**:
  + Communication via USB or Bluetooth input stream.
  + Input sent to designated field in vehicle logging module.

### **5.4 Communications Interfaces**

The system supports the following communication methods:

* **HTTP/HTTPS**:
  + All API calls use HTTPS for secure transmission.
  + RESTful API standard with JSON data format.
* **Email**:
  + SMTP used to send notification emails (maintenance updates, invoice confirmations).
  + Emails will not contain attachments due to security constraints.
  + SMTP Server: smtp.gmail.com (or institutional mail server)
  + Port: 587 with TLS encryption
* **Web Access**:
  + Web interface accessible via modern browsers (Chrome, Firefox, Edge).
  + Supported browser versions: last 2 major releases.
* **Security Protocols**:
  + All sensitive communication encrypted using TLS 1.2+.
* **Data Transfer Constraints**:
  + Maximum file upload (if allowed): 5MB (e.g., image proof for feedback)
  + Timeout for request/response: 30 seconds.

# **6. Quality Attributes**

## **6.1 Usability**

* Simple, user-friendly, and intuitive interface for users of all ages.
* New users can become familiar and perform basic functions within 10 minutes.
* Supports multiple languages, with Vietnamese as the default.
* Compatible with both desktop and mobile devices, following responsive design standards.

## **6.2 Performance**

* Homepage loads within 3 seconds with a 1 Mbps connection.
* The system supports at least 500 concurrent users without interruption.
* Payment transaction processing time does not exceed 5 seconds.

## **6.3 Security**

* AImplements strong password authentication for login.
* Encrypts sensitive data (AES-256 for storage, HTTPS for transmission).
* Clear role-based access control for Admin, Accountant, Technical Staff, and Residents.
* Complies with personal data protection regulations.

## **6.4 Safety**

* Two-step confirmation required for critical actions (data deletion, large payments).
* Automatic data backup every 24 hours to prevent information loss.
* Alerts for system errors or unauthorized access attempts.

## **6.5 Others as relevant**

* **Reliability:** Ensures 99.9% uptime per month.
* **Scalability:** Supports 50% user growth without architecture changes.
* **Portability:** Compatible with major browsers (Chrome, Firefox, Edge) and operating systems (Windows, Android, iOS).

**7**. **Internationalization and Localization Requirements**

The **Apartment Management System (AMS)** is designed to support internationalization and localization to ensure usability across multiple languages, regions, and cultural contexts. These requirements are essential to facilitate deployment beyond the initial domestic (Vietnamese) audience.

# **8. Other Requirements**

# The system must provide responsive design to support various screen sizes, including mobile devices, tablets, and desktops.

# The system must allow role-based access control, ensuring that each user (Resident, Admin, Accountant, etc.) can only access authorized features.

# The system should support backup and restore functionality to prevent data loss.

# All user actions must be logged for audit and troubleshooting purposes.

# System downtime for maintenance must be scheduled and notified in advance to all users.

# All communication between client and server must be encrypted using TLS 1.2+ for data security.