**Software Requirements Specification**

**for**

**<Apartment Management System>**

**Version 1.0 approved**

**Prepared by <>**

**<Group 1>**

**<Spring25>**

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

**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 |

# **Introduction**

This section provides an overview of the Software Requirements Specification (SRS) for the Apartment Management System. It outlines the purpose of the document, identifies its intended audience, and explains how the document is organized and should be utilized.

## **Purpose**

This document describes the software requirements for the Apartment Management System. The system aims to provide an online platform for residents, management boards, technical staff and other stakeholders to interact, manage and handle operations related to life in the apartment building such as managing residential accounts, sending feedback, monitoring invoices, technical maintenance, etc.

* Software developer: To develop functions as required.
* Project management: To control the progress, scope and product quality.
* Test staff: To write and execute test cases.
* End users (residents, management boards): To understand the functions provided.
* Technical staff and support: To deploy and maintain the system.

## **Document Conventions**

* Text styles: Times New Roman
* 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: Managers can set up system configuration, manage resident information, and monitor system status.
* User accounts and authentication: Support login function for everyone based on roles.
* 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**

* **IEEE Std 830-1998 – Software Requirements Specification (SRS)** The criteria for writing software requirements for apartment management systems.
* **Use Case Diagram Guidelines** Guide to develop Use case diagrams for agents (Resident, Guest, Admin, Technical Staff, Accountant, Security Staff) và chức năng hệ thống
* **Payment Gateway API Documentation** Document integrated online payment processing and sending status (Process online payments, Send transaction status).
* **System Security Best Practices** Instructions for accessing access, recognition and monitoring of activities (Log vehicle entry/exit).
* **Maintenance Management Standards** Management criteria and update maintenance progress (Assign maintenance tasks, Update maintenance progress).

# **Overall Description**

The **Apartment Management System (AMS)** is a comprehensive software solution developed to assist residential property managers in effectively managing apartment operations and resident-related services. This system is designed to automate and centralize key activities such as resident registration, maintenance requests, visitor management, billing, security monitoring, and administrative reporting.

## **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**

*<Identify the various user classes that you anticipate will use this product and describe their pertinent characteristics. Some requirements might pertain only to certain user classes. Identify the favored user classes. User classes represent a subset of the stakeholders described in the vision and scope document. User class descriptions are a reusable resource. If available, you can incorporate user class descriptions by simply pointing to them in a master user class catalog instead of duplicating information here.>*

## **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.

## **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.

## **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.

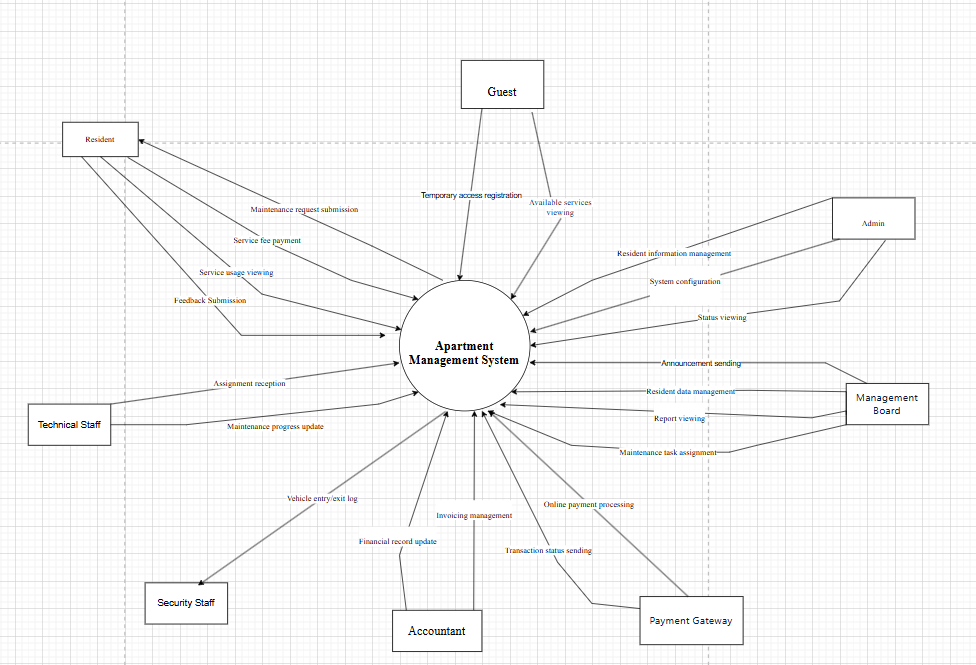
\* The hosting infrastructure (servers, storage) will be provisioned and maintained by a dedicated IT team.

\* All stakeholders will participate in user training and accept proposed workflows.

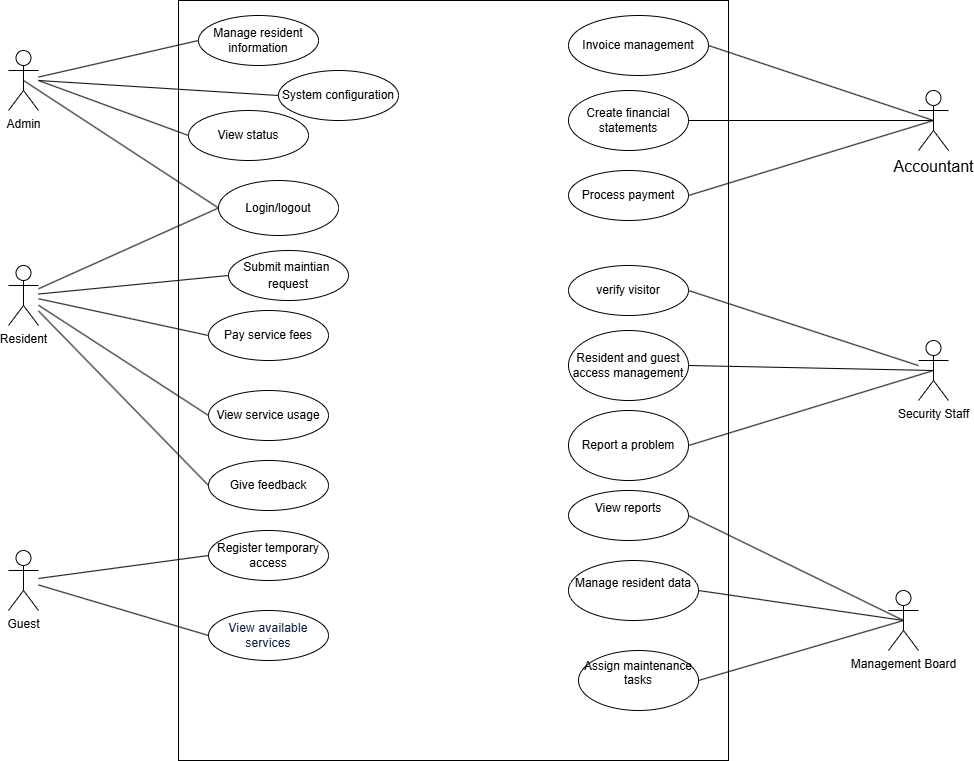
\* Government regulations or technical standards may affect implementation and must be monitored throughout the project.

# **System Features**

**Context diagram:**

****

**Use case diagram:**

****

**System feature tree:**

## **System Feature Manage Account**

### **Description**

Writing short description about manage account here

### **Stimulus/Response Sequences**

List all event and FAP system must response with the even. Should write in table formate

### **Functional Requirements Create new Account**

Draw the GUI of the screen create new Account function and insert here

| UC ID and Name: | Create new Account | | |
| --- | --- | --- | --- |
| Created By: | TrungNT | Date Created: | 12/2/2009 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: |  | | |
| Description: |  | | |
| Preconditions: | The user must have the valide email of FPT University. Example TrungNT@fpt.edu.vn | | |
| Postconditions: | After finish this function, the new account must be created and store in the database of the FAP system | | |
| Normal Flow: | 1. Fisrt, the admin user must access the screen create new account | | |
| Alternative Flows: |  | | |
| Exceptions: |  | | |
| Priority: |  | | |
| Frequency of Use: |  | | |
| Business Rules: |  | | |
| Other Information: |  | | |
| Assumptions: |  | | |

### **Function update account**

Draw the GUI of the screen update Account function and insert here

| UC ID and Name: | Update Account | | |
| --- | --- | --- | --- |
| Created By: | TrungNT | Date Created: | 12/2/2009 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: |  | | |
| Description: |  | | |
| Preconditions: | The user must already have the valid account in the FAP System. | | |
| Postconditions: | After finish this function, the account must be updated and store in the database of the FAP system | | |
| Normal Flow: | 1. Fisrt, the admin user must access the screen create new account | | |
| Alternative Flows: |  | | |
| Exceptions: |  | | |
| Priority: |  | | |
| Frequency of Use: |  | | |
| Business Rules: |  | | |
| Other Information: |  | | |
| Assumptions: |  | | |

## **System Feature Manage grade of student**

### **Function import grade**

Draw the GUI of the screen import grade function and insert here

| UC ID and Name: | Import grade | | |
| --- | --- | --- | --- |
| Created By: | TrungNT | Date Created: | 12/2/2009 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: |  | | |
| Description: |  | | |
| Preconditions: | The user must already have the valid account in the FAP System. | | |
| Postconditions: | After finish this function, the account must be updated and store in the database of the FAP system | | |
| Normal Flow: | 1. Fisrt, the admin user must access the screen create new account | | |
| Alternative Flows: |  | | |
| Exceptions: |  | | |
| Priority: |  | | |
| Frequency of Use: |  | | |
| Business Rules: |  | | |
| Other Information: |  | | |
| Assumptions: |  | | |

### **function update grade**

Draw the GUI of the screen import grade function and insert here

| UC ID and Name: | Import grade | | |
| --- | --- | --- | --- |
| Created By: | TrungNT | Date Created: | 12/2/2009 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: |  | | |
| Description: |  | | |
| Preconditions: | The user must already have the valid account in the FAP System. | | |
| Postconditions: | After finish this function, the account must be updated and store in the database of the FAP system | | |
| Normal Flow: | 1. Fisrt, the admin user must access the screen create new account | | |
| Alternative Flows: |  | | |
| Exceptions: |  | | |
| Priority: |  | | |
| Frequency of Use: |  | | |
| Business Rules: |  | | |
| Other Information: |  | | |
| Assumptions: |  | | |

### **function delete grade**

# **Data Requirements**

*<This section describes various aspects of the data that the system will consume as inputs, process in some fashion, or create as outputs.>*

## **Logical Data Model**

*<A data model is a visual representation of the data objects and collections the system will process and the relationships between them. Include a data model for the business operations being addressed by the system, or a logical representation for the data that the system itself will manipulate. Data models are most commonly created as an entity-relationship diagram.>*

## **Data Dictionary**

*<The data dictionary defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements that make up those structures. In many cases, you're better off storing the data dictionary as a separate artifact, rather than embedding it in the middle of an SRS. That also increases its reusability potential in other projects.>*

## **Reports**

*<If your application will generate any reports, identify them here and describe their characteristics. If a report must conform to a specific predefined layout you can specify that here as a constraint, perhaps with an example. Otherwise, focus on the logical descriptions of the report content, sort sequence, totaling levels, and so forth, deferring the detailed report layout to the design stage.>*

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

*<If relevant, describe how data is acquired and maintained. State any requirements regarding the need to protect the integrity of the system's data. Identify any specific techniques that are necessary, such as backups, checkpointing, mirroring, or data accuracy verification. State policies the system must enforce for either retaining or disposing of data, including temporary data, metadata, residual data (such as deleted records), cached data, local copies, archives, and interim backups.>*

# **External Interface Requirements**

*<This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements.>*

## **User Interfaces**

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

## **Software Interfaces**

*<Describe the connections between this product and other software components (identified by name and version), including other applications, databases, operating systems, tools, libraries, websites, and integrated commercial components. State the purpose, formats, and contents of the messages, data, and control values exchanged between the software components. Specify the mappings of input and output data between the systems and any translations that need to be made for the data to get from one system to the other. Describe the services needed by or from external software components and the nature of the intercomponent communications. Identify data that will be exchanged between or shared across software components. Specify nonfunctional requirements affecting the interface, such as service levels for responses times and frequencies, or security controls and restrictions.>*

## **Hardware Interfaces**

*<Describe the characteristics of each interface between the software and hardware (if any) components of the system. This description might include the supported device types, the data and control interactions between the software and the hardware, and the communication protocols to be used. List the inputs and outputs, their formats, their valid values or ranges, and any timing issues developers need to be aware of. If this information is extensive, consider creating a separate interface specification document.>*

## **Communications Interfaces**

*<State the requirements for any communication functions the product will use, including e-mail, Web browser, network protocols, and electronic forms. Define any pertinent message formatting. Specify communication security or encryption issues, data transfer rates, handshaking, and synchronization mechanisms. State any constraints around these interfaces, such as whether e-mail attachments are acceptable or not.>*

# **Quality Attributes**

## **Usability**

*<Specify any requirements regarding characteristics that will make the software appear to be “user-friendly.” Usability encompasses ease of use, ease of learning; memorability; error avoidance, handling, and recovery; efficiency of interactions; accessibility; and ergonomics. Sometimes these can conflict with each other, as with ease of use over ease of learning. Indicate any user interface design standards or guidelines to which the application must conform.>*

## **Performance**

*<State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it's appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section.>*

## **Security**

*<Specify any requirements regarding security or privacy issues that restrict access to or use of the product. These could refer to physical, data, or software security. Security requirements often originate in business rules, so identify any security or privacy policies or regulations to which the product must conform. If these are documented in a business rules repository, just refer to them.>*

## **Safety**

*<Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product. Define any safeguards or actions that must be taken, as well as potentially dangerous actions that must be prevented. Identify any safety certifications, policies, or regulations to which the product must conform.>*

## **[Others as relevant]**

*<Create a separate section in the SRS for each additional product quality attribute to describe characteristics that will be important to either customers or developers. Possibilities include availability, efficiency, installability, integrity, interoperability, modifiability, portability, reliability, reusability, robustness, scalability, and verifiability. Write these to be specific, quantitative, and verifiable. Clarify the relative priorities for various attributes, such as security over performance.>*

# **Internationalization and Localization Requirements**

*<Internationalization and localization requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created. Such requirements might address differences in: currency; formatting of dates, numbers, addresses, and telephone numbers; language, including national spelling conventions within the same language (such as American versus British English), symbols used, and character sets; given name and family name order; time zones; international regulations and laws; cultural and political issues; paper sizes used; weights and measures; electrical voltages and plug shapes; and many others.>*

# **Other Requirements**

*<Examples are: legal, regulatory or financial compliance, and standards requirements; requirements for product installation, configuration, startup, and shutdown; and logging, monitoring and audit trail requirements. Instead of just combining these all under "Other," add any new sections to the template that are pertinent to your project. Omit this section if all your requirements are accommodated in other sections. >*

**Appendix A: Glossary**

*<Define any specialized terms that a reader needs to know to understand the SRS, including acronyms and abbreviations. Spell out each acronym and provide its definition. Consider building a reusable enterprise-level glossary that spans multiple projects and incorporating by reference any terms that pertain to this project.>*

**Appendix B: Analysis Models**

*<This optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams. You might prefer to insert certain models into the relevant sections of the specification instead of collecting them at the end.>*