



SEEK WISDOM, ELEVATE YOUR INTELLECT AND SERVE HUMANITY!



ADDIS ABABA INSTITUTE OF TECHNOLOGY

School of Electrical and Computer Engineering

Computer stream

SOFTWARE ENGINEERING SEMESTER PROJECT SOFTWARE REQUIREMENT SPECIFICATION (SRS) DOCUMENTATION

PROJECT TITLE:

APARTMENT RENTAL MANAGEMENT SYSTEM (ARMS)

**Submitted to - Dr. Fitsum Assamnew
Submission date - 03/04/2023**

User side

Name	ID number
1, Matewos Tegete (CEO)	UGR/6649/12
2, Emnet Mamo (Manager)	UGR/0545/12
3, Biruk Zewdu	UGR/0161/12
4, Muluken Walle	UGR/9931/12
5, Gezhagn Teramedew	UGR/1855/12
6, Yinges Damtie	UGR/7186/12

Manager side

Name	ID number
1, Hussen Mohammed (Manager)	UGR/6859/12
2, Mikiyas Mohammed	UGR/3095/12
3, Meksud Reshid	UGR/0511/12
4, Biniyam Abera	UGR/9138/12
5, Yosef Alemu	UGR/6607/12

Security and Owner side

Name	ID number
1, Salas Delil (Manager)	ETR/9857/12
2, Abel Ayalew	UGR/6225/12
3, Seifegebreal Mosisa	UGR/6747/12
4, Misgan Moges	UGR/5779/12
5, Mintesnote Bankisra	UGR/4174/12
6, Alelgne Eshetie	UGR/6977/12

Table Of Contents

1.INTRODUCTION.....	1
1.1 PURPOSE.....	1
1.2 SCOPE.....	1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS.....	2
1.4 REFERENCES.....	2
1.5 OVERVIEW.....	3
2.OVERALL DESCRIPTION.....	3
2.1 PRODUCT PERSPECTIVE.....	3
2.2 PRODUCT FUNCTION.....	4
2.2.1 Tenant management:.....	4
2.2.2 Apartment listing and details management:.....	5
2.2.3 Security management:.....	5
2.2.4 Maintenance management:.....	5
2.2.5 Communication:.....	5
2.2.6 Reporting and analytics:.....	6
2.2.7 Payment processing:.....	6
2.2.8 User management:.....	6
2.3 User CHARACTERISTICS:.....	6
2.4 CONSTRAINTS:.....	7
2.5 ASSUMPTIONS AND DEPENDENCIES:.....	7
2.6 APPORTIONING OF REQUIREMENTS.....	8
3 SPECIFIC REQUIREMENTS.....	9
3.1 EXTERNAL INTERFACES:.....	9
3.2 FUNCTIONS:.....	11
3.2.1 Use cases for Tenant side.....	12
3.2.2 Use cases for Manager side.....	22
3.2.3 Use cases for Owner side.....	33
3.2.4 Use cases for Security side.....	38
3.3 PERFORMANCE REQUIREMENTS:.....	44
3.4 LOGICAL DATABASE REQUIREMENTS.....	45
3.4.1 Types of information used by various functions:.....	45
3.4.2 Accessing capabilities:.....	46
3.4.3 Frequency of use:.....	46
3.4.4 Data entities and their relationships:.....	47
3.4.5 Integrity constraints:.....	48
3.4.6 Data retention requirements:.....	48
3.4.7 Design constraints.....	49
3.4.8 Software system attributes.....	50
4 APPENDIX.....	51

4.1 INPUT AND OUTPUT SAMPLE FORMATS.....	51
4.2 Use case diagram.....	52
4.3 BACKGROUND AND IMPACTS.....	53

1.INTRODUCTION

1.1 PURPOSE

The purpose of the apartment rental management system is to provide a centralized platform for property managers to manage and streamline the rental process for the apartments they manage. The system provides a range of features, including tenant management, apartment listing and details, lease management, payment tracking, and security management.

The system aims to simplify the rental process by automating routine tasks and providing a comprehensive overview of apartment occupancy, maintenance, and financial data. This information helps Property Managers make informed decisions about pricing, occupancy rates, and maintenance schedules, among other things.

The system also aims to enhance the security and safety of apartments by providing a way to manage access to the building and track the activities of security guards. Additionally, the system helps Property Managers maintain a good relationship with tenants by providing a streamlined way to communicate and manage tenant concerns or complaints.

Overall, the purpose of the apartment rental management system is to help Property Managers efficiently and effectively manage their rental properties, while providing tenants with a safe and comfortable living environment.

1.2 SCOPE

The scope of the Apartment (Rental) Management System is to develop a web-based application that will assist apartment managers in managing and maintaining their rental properties. The system will provide features that will enable the managers to perform tasks related to apartment rental management, such as managing tenant information, tracking rent payments, managing maintenance requests, and generating reports.

The system will be designed to support multiple users, including managers, tenants, and security guards. The managers will have access to all the features of the system, while tenants will have access to limited features, such as viewing their rental information and making maintenance requests. Security guards will have access to a subset of the system features, such as managing visitor access.

The system will also provide interfaces for integration with external systems, such as accounting and payment processing systems, to facilitate payment processing and financial management.

The scope of the project includes the development of the system software, database design, and web-based user interfaces. The system will be developed using industry-standard web technologies and will be hosted on a web server.

The system will not include any physical hardware, such as servers or networking equipment. The system will also not include any physical security measures, such as surveillance cameras or alarm systems. The system will only provide interfaces for managing visitor access through the security guard interface.

The system will be designed to be adaptable to various site requirements and configurations. However, the system will not include any customization services to adapt to specific tenant requirements. The system will be designed to be scalable to support an increasing number of users and rental properties as required.

In summary, the scope of the Apartment (Rental) Management System is to develop a web-based application that will assist apartment managers in managing and maintaining their rental properties, providing interfaces for integration with external systems, and support for multiple users. The system will not include any physical hardware or security measures, and will be designed to be adaptable and scalable to various site requirements and configurations.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Definitions:

- Property Manager: The person responsible for managing and maintaining a rental property on behalf of the property owner.
- Tenant: The person who rents an apartment within a rental property.
- Security Guard: A person responsible for maintaining the security and safety of a rental property.

Acronyms and Abbreviations:

- TRM: Tenant Relationship Management
- API: Application Programming Interface
- UI: User Interface
- UX: User Experience
- SQL: Structured Query Language
- SSO: Single Sign-On
- HTTPS: Hypertext Transfer Protocol Secure
- DNS: Domain Name System
- SMTP: Simple Mail Transfer Protocol
- SSL: Secure Sockets Layer

Note: Additional acronyms and abbreviations may be used throughout the development process, and will be defined as needed.

1.4 REFERENCES

- 1) Teacher's slide: slide 3(Requirements): by Dr. Fitsum Assfaw
- 2) IEEE Recommended Practice for Software Requirements Specifications: IEEE Computer Society.
- 3) ChatGPT

1.5 OVERVIEW

The apartment rental management system is a web-based software solution designed to simplify the process of managing rental properties. The system enables property managers to manage tenant information, lease agreements, rental payments, apartment listings and details, security management, maintenance management, and communication with tenants, all in one place.

The system includes a user-friendly interface that makes it easy for property managers to access the information they need and perform the necessary tasks. The system also provides a way for tenants to access their lease information, report security issues, and pay rent.

The apartment rental management system is designed to be scalable, allowing property managers to manage multiple rental properties of varying sizes and complexities. The system is secure, with features such as single sign-on (SSO), HTTPS, DNS, SMTP, and SSL encryption to ensure the privacy and security of user data.

Overall, the apartment rental management system streamlines the rental process, automates routine tasks, and provides a comprehensive overview of property-related data, allowing property managers to make informed decisions about pricing, occupancy rates, and maintenance schedules, among other things.

2.OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

(a) System interfaces: The apartment management system will interface with various external systems such as payment gateways, email servers, and SMS gateways for sending notifications and alerts to tenants, managers, and owners. It will also interface with databases to store and retrieve information about apartments, tenants, owners, and managers.

(b) User interfaces: The apartment management system will provide a user-friendly web-based interface for tenants, owners, and managers to interact with the system. The interface will be

easy to navigate and will provide quick access to all the features and functions of the system. The interface will be designed using modern web technologies to ensure compatibility with all major web browsers and devices.

(c) Hardware interfaces: The apartment management system will be hosted on a web server that will be accessible over the internet using standard web protocols. The server will require hardware components such as CPU, memory, and storage to support the system's performance and scalability requirements.

(d) Software interfaces: The apartment management system will be developed using modern web technologies such as HTML, CSS, JavaScript, React and NestJs. It will interface with various software components such as databases, web servers, and email servers.

(e) Communications interfaces: The apartment management system will use various communication channels such as email, SMS, and in-app notifications to communicate with tenants, owners, and managers. It will also use APIs to integrate with external systems such as payment gateways and SMS gateways.

(f) Memory: The apartment management system will require a certain amount of memory to store data about apartments, tenants, owners, and managers. The system will also need memory to support its processing and caching requirements.

(g) Operations: The apartment management system will be designed to support various operations such as apartment management, tenant management, owner management, and manager management. The system will also provide various features such as rent collection, maintenance requests, and security management.

(h) Site adaptation requirements: The apartment management system will be designed to support easy adaptation to different site configurations and requirements. The system will be flexible enough to adapt to different apartment layouts, security arrangements, and payment schedules. It will also be scalable to support future growth and expansion.

2.2 PRODUCT FUNCTION

2.2.1 Tenant management:

- Tenant information management: The system allows property managers to store and manage tenant information such as name, contact details, move-in date, lease expiration date, and rental history.
- Lease agreement management: Property managers can upload and manage lease agreements, track lease terms, and handle lease renewals and extensions.

- Rental payment management: The system provides tools to manage rent payments, including setting up automatic payments, tracking payments, and sending reminders to tenants who are late on payments.

2.2.2 Apartment listing and details management:

- Apartment listing management: The system enables property managers to upload and manage apartment listings, including details such as apartment type, size, location, and availability status.
- Apartment details management: The system allows property managers to add and manage apartment details such as photos, floor plans, amenities, and pricing information.

2.2.3 Security management:

- Security guard management: The system allows property managers to manage security guards, including assigning guards to specific buildings or shifts, monitoring their activity, and tracking their hours worked.
- Security issue management: Property managers can track and manage security issues reported by tenants, such as break-ins, theft, or other incidents.

2.2.4 Maintenance management:

- Maintenance request management: The system enables tenants to submit maintenance requests online, which are then assigned to maintenance staff for resolution.
- Maintenance scheduling: Property managers can schedule repairs and maintenance work, set deadlines, and monitor progress.
- Maintenance expense tracking: The system allows property managers to track maintenance expenses, including costs for repairs, replacements, and upgrades.

2.2.5 Communication:

- Tenant communication: The system provides a way for property managers to communicate with tenants, either through email or the system's messaging feature, to inform them of important information such as rent increases, policy changes, or maintenance schedules.
- Security guard communication: Property managers can communicate with security guards, assign tasks or shifts, and provide updates on security issues.

- Maintenance staff communication: The system enables property managers to communicate with maintenance staff, assign work, and monitor progress.

2.2.6 Reporting and analytics:

- Occupancy rate reporting: The system generates reports on occupancy rates, including vacancy rates, rent-to-income ratios, and unit turnover rates.
- Rental income reporting: Property managers can track and report on rental income, including rent payments, security deposit refunds, and other fees.
- Maintenance expense reporting: The system allows property managers to track maintenance expenses, including costs for repairs, replacements, and upgrades.

2.2.7 Payment processing:

- Online rent payment: The system provides a way for tenants to pay rent online, either through a credit card or bank transfer.
- Payment tracking: Property managers can track and monitor rent payments, including overdue payments, and send reminders to tenants who are late on payments.

2.2.8 User management:

- Login and authentication: The system requires users to log in and authenticate their identity before accessing any information or functionality.
- Password reset: Users who forget their password can reset it through the system's password reset feature.
- User roles and permissions: The system includes role-based access control, enabling property managers to grant or restrict access to certain functions based on a user's role or job responsibilities.

The apartment rental management system is designed to be used by Property Managers who manage multiple rental properties, as well as security personnel and tenants who use the system to manage access to the building, report security issues, and pay rent or view lease details.

The system is intended to simplify the rental process, automate routine tasks, and provide a comprehensive overview of property-related data, allowing Property Managers to make informed decisions about pricing, occupancy rates, and maintenance schedules, among other things. Additionally, the system helps ensure the safety and security of the apartment building and its residents by allowing Property Managers to manage access to the building and track the activities of security personnel.

2.3 User CHARACTERISTICS:

The intended users of the apartment rental management system are diverse and include apartment owners, property managers, security personnel, and tenants. As such, users are

expected to have varying levels of computer literacy and technical expertise. The system has been designed to be user-friendly and easy to navigate. Therefore, users are expected to be able to perform basic tasks, such as logging in, viewing property details, generating reports, and making payments. Property managers and security personnel are expected to have some level of experience in property management and security operations. Tenants are expected to have a basic understanding of the rental application process.

2.4 CONSTRAINTS:

The apartment rental management system is subject to regulatory policies related to data privacy and security. Therefore, the system must comply with industry standards and best practices for secure data management. Hardware limitations include signal timing requirements such as response time for security alerts and the need for compatible devices to access the system. The system must be able to interface with other applications, such as accounting software and email clients. The system must support parallel operations to handle multiple requests simultaneously. The system must have audit and control functions to ensure the accuracy and integrity of data. The system must support higher-order language requirements to enable customization and localization for different regions and languages. Signal handshake protocols must be supported to ensure reliable data transmission. The system must be available 24/7 and must be able to handle large volumes of data traffic. The criticality of the application requires that the system is designed to handle emergencies and respond to security incidents in a timely and effective manner. Safety and security considerations require that the system is designed to prevent unauthorized access, data breaches, and other security risks.

2.5 ASSUMPTIONS AND DEPENDENCIES:

The apartment rental management system assumes that users have access to compatible devices, reliable internet connectivity, and a basic level of technical expertise. The system also assumes that the operating system and hardware designated for the software product will be available and compatible with the system. Dependencies include third-party applications such as email clients and accounting software, which must be able to interface with the system. The system is also dependent on the availability and reliability of third-party services such as cloud storage and data analytics tools. Any changes to these assumptions and dependencies may affect the requirements specified in the SRS.

The system has been designed to cater to users with varying levels of technical expertise. The system's user interface is intuitive and user-friendly, making it easy for users to navigate the system. The system is also subject to various constraints related to regulatory policies, data privacy, security, and reliability. These constraints ensure that the system is secure and reliable, and data is managed in accordance with industry standards and best practices. The system's assumptions and dependencies relate to hardware and software compatibility and the availability and reliability of third-party services. Any changes to these assumptions and dependencies may affect the system's requirements specified in the SRS.

2.6 APPORTIONING OF REQUIREMENTS

The requirements for the apartment rental management system can be apportioned into different categories based on their priority and functional importance. The following categories can be used to apportion the requirements:

1. **Critical Requirements:** These are requirements that are essential for the basic functioning of the system. They must be implemented before any other requirements. Critical requirements for the system include:
 - **User authentication and authorization:** The system must ensure that only authorized users have access to the system and its features.
 - **Property management:** The system must enable property managers to manage and update property details, including rent rates, availability, and maintenance requests.
 - **Tenant management:** The system must enable tenants to view and pay rent, submit maintenance requests, and communicate with property managers.
2. **High-Priority Requirements:** These are requirements that are important for the efficient functioning of the system. They should be implemented soon after critical requirements. High-priority requirements for the system include:
 - **Security management:** The system must enable security personnel to monitor security alerts, track incidents, and communicate with property managers and tenants.
 - **Payment management:** The system must enable property managers to view payment history, generate invoices, and process payments from tenants.
 - **Reporting and analytics:** The system must enable property managers to generate reports and analyze data to make informed decisions.
3. **Medium-Priority Requirements:** These are requirements that are less critical but still important for the system's overall functionality. Medium-priority requirements for the system include:
 - **Messaging and notification:** The system must enable property managers and tenants to communicate through messaging and receive notifications for important events.
 - **Document management:** The system must enable property managers to store and manage property-related documents such as lease agreements and tenant applications.
 - **Integration with third-party services:** The system must enable integration with third-party services such as accounting software and email clients.
4. **Low-Priority Requirements:** These are requirements that are not essential but can enhance the system's functionality. Low-priority requirements for the system include:
 - **Social media integration:** The system can integrate with social media platforms to enable property managers to reach a wider audience.
 - **Mobile application:** The system can have a mobile application to enable tenants and property managers to access the system on-the-go.

- Gamification features: The system can have gamification features to encourage tenants to engage with the system and incentivize timely rent payments.

By apportioning the requirements into these categories, the development team can prioritize the implementation of the requirements and ensure that critical and high-priority requirements are implemented before medium and low-priority requirements. This can help to ensure that the system meets the most important needs of the users and is delivered on time and within budget.

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACES:

The apartment rental management system will have several inputs and outputs, both from internal and external sources. These external interfaces include:

1. Payment gateways: This interface will allow tenants to make payments for their rent and other charges online. It will be integrated with different payment methods such as credit cards, bank transfers, and mobile money payments.
2. Email clients: The system will interface with email clients to allow for the sending and receiving of emails related to the management of the apartment complex. This will include notifications of rent payments, maintenance requests, and other important information.
3. Accounting software: The system will interface with accounting software to manage financial transactions related to the rental business. This will include tracking of expenses, generating invoices, and managing payments.
4. Property listing websites: The system will interface with property listing websites to update information about available units, vacancies, and rental rates.
5. Security systems: The system will interface with security systems such as CCTV cameras, motion detectors, and access control systems to monitor and control access to the apartment complex.
6. Communication systems: The system will interface with communication systems such as intercoms and telephones to enable communication between tenants, property managers, and security personnel.
7. Mobile applications: The system will have a mobile application interface that will allow tenants to access the system and perform various tasks such as paying rent, submitting maintenance requests, and viewing apartment details.
8. Social media platforms: The system will interface with social media platforms to enable marketing and communication with tenants and prospective tenants.

External interfaces refer to all inputs and outputs of the software system that are not part of the internal system. This includes interactions with other software systems, hardware devices, and

human users. External interfaces are crucial for the success of the system as they enable communication with external entities and facilitate the exchange of information.

The external interfaces of the apartment rental management system include both inputs and outputs. Inputs refer to information that is received by the system, while outputs refer to information that is produced by the system.

Inputs:

1. Property details - The system receives information about the properties managed by the users, such as the number of units, location, amenities, and lease terms. This information is used to manage the properties effectively and efficiently.
2. Tenant information - The system receives information about the tenants who rent the properties, such as their personal details, lease agreements, and payment history. This information is used to manage the tenants and ensure that rent is paid on time.
3. Security alerts - The system receives alerts from security devices such as CCTV cameras, motion sensors, and fire alarms. These alerts are used to notify property managers and security personnel of potential security breaches or emergencies.
4. Accounting software - The system receives data from accounting software such as QuickBooks or Xero. This data includes financial information such as invoices, receipts, and bank statements. The system uses this information to manage the financial aspects of the properties, such as rent collection and expense tracking.
5. Email clients - The system receives emails from tenants, property owners, and other stakeholders. These emails are used to communicate with the users of the system and keep them informed of important updates and changes.

Outputs:

1. Property listings - The system generates property listings that include details such as location, price, and amenities. These listings are used to market the properties to potential tenants.
2. Tenant invoices - The system generates invoices for tenants that include details such as rent amount, due date, and payment method. These invoices are used to ensure that tenants pay rent on time.
3. Reports - The system generates reports that provide insights into the performance of the properties and the tenants. These reports include information such as occupancy rates, revenue, and maintenance costs. They are used by property managers and owners to make informed decisions about the properties.
4. Security notifications - The system sends notifications to property managers and security personnel when security alerts are triggered. These notifications include information

such as the type of alert, location, and time of the alert. They are used to respond to security breaches and emergencies.

5. Email notifications - The system sends email notifications to tenants, property owners, and other stakeholders. These notifications include updates on lease agreements, maintenance requests, and other important information.

The valid range, accuracy, and tolerance of the inputs and outputs depend on the specific requirements of the system. For example, the property details input may require the property location to be within a specific city or state, while the tenant invoice output may require the rent amount to be accurate to two decimal places. The units of measure depend on the type of input or output. For example, the property details input may include the property area in square feet, while the tenant invoice output may include the rent amount in dollars.

The timing of the inputs and outputs is also an important aspect of the external interfaces. Inputs such as security alerts and email messages may need to be processed in real-time to ensure a timely response, while outputs such as reports may be generated on a daily, weekly, or monthly basis. The relationships between the inputs and outputs are also important to consider, as they can impact the performance of the system. For example, if the accounting software input is not compatible with the system, it may result in delayed processing times or errors in the output.

The screen and window formats/organization of the external interfaces are also important to consider. The layout and design of the user interface can greatly impact the usability and efficiency of the system. The system should be designed to provide a clear and intuitive interface that enables users to quickly access the information they need and perform necessary tasks. The data formats and command formats should also be specified to ensure compatibility with other applications and to prevent errors in data transfer.

Finally, end messages should be specified to provide feedback to the user on the status of their input or request. End messages should indicate whether the input was successfully processed or if there were errors that need to be corrected. The end message should be clear and concise to ensure that the user understands the status of their request.

In summary, the external interface requirements for the apartment rental management system must specify the purpose, source/destination, valid range/accuracy/tolerance, units of measure, timing, relationships to other inputs/outputs, screen and window formats/organization, data formats, command formats, and end messages. These requirements must be designed to ensure compatibility with other applications, provide clear and intuitive interfaces, and enable timely and efficient processing of inputs and outputs.

3.2 FUNCTIONS:

The functional requirements of the apartment rental management system are as follows:

- a) Validity checks on inputs: The system shall validate all inputs to ensure that they are within valid ranges and conform to required data formats.
- b) Exact sequence of operations: The system shall follow a specific sequence of operations for each function to ensure proper execution.
- c) Responses to abnormal situations: The system shall handle abnormal situations such as overflows, communication failures, and error handling and recovery.
- d) Effect of parameters: The system shall accurately calculate and display the effect of different parameters such as rental rates, maintenance fees, and security charges.
- e) Relationship of outputs to inputs: The system shall maintain accurate records of inputs and outputs, including input/output sequences and formulas for input to output conversion.

The functional requirements may be further divided into sub functions or subprocesses to ensure proper execution of the system.

3.2.1 Use cases for Tenant side

Use case 1: Tenant Sends Register Request to Manager

Primary Actor: Tenant

Goal: To send a register request to the manager to become a tenant of a property

Preconditions:

- The tenant is not currently registered as a tenant of the property
- The tenant has a valid email address and phone number.
- The manager has a system in place to receive tenant register requests

Main Success Scenario:

1. The tenant navigates to the property management system and clicks on the "Register Request" button.
2. The system prompts the tenant to enter their personal and contact information, such as their full name, phone number, email address, and desired move-in date.
3. The tenant fills in the required information and submits the form.
4. The system sends the register request to the manager for approval.

5. The manager receives the register request and reviews the tenant's information.
6. If the manager approves the request, the system sends a confirmation to the tenant with further instructions on the next steps to take.
7. If the manager denies the request, the system sends a notification to the tenant explaining the reason for the denial.

Alternate Scenarios:

- If the tenant fails to fill in all the required information in the registration form, the system prompts the tenant to fill in the missing information before submitting the form
- If the manager is not available to approve the request immediately, the system stores the request and sends a notification to the manager when they become available.
- If the tenant fails to fill in a valid email address, the registration will not be recorded and sent to the manager, instead the tenant is prompted to enter a valid email address.

Use case 2: Tenant Login

Primary Actor: tenant

Stakeholders and Interests:

- tenant: wants to access their account and manage their rental information.
- Manager: wants to ensure that only authorized tenants can access their rental information.

Preconditions:

- The tenant must have a registered account in the system.
- The tenant must know their login credentials.

Main Success Scenario:

1. The tenant navigates to the login page and enters their username and password.
2. The system verifies the login credentials and authenticates the tenant.
3. The system displays the tenant dashboard, which shows their rental information, including lease terms, rent payments, and maintenance requests.
4. The tenant can view and update their personal information, such as their contact details and emergency contacts.
5. The tenant can view and pay their rent online through the system's payment gateway.
6. The tenant can submit maintenance requests through the system, which are automatically forwarded to the maintenance team.

Postconditions:

- The tenant has successfully accessed their account and can manage their rental information.

Extensions:

- If the tenant enters invalid login credentials, the system displays an error message and prompts them to try again.
- If the tenant forgets their password, they can use the "Forgot Password" feature to reset their password.
- If the tenant encounters any issues with the system, they can contact the manager or tenant support for assistance.

Use case 3: Forgot Password

Primary Actor: tenant

Preconditions:

- The tenant is registered in the system.
- The tenant has forgotten their password.

Main Flow:

1. The tenant clicks on the "Forgot Password" link on the login page.
2. The system displays a form to enter the registered email address.
3. The tenant enters their registered email address and submits the form.
4. The system verifies the email address and generates a temporary password.
5. The system sends an email to the registered email address with the temporary password.
6. The tenant receives the email and logs in using the temporary password.
7. The system prompts the tenant to change their password to a new one.
8. The tenant enters a new password and confirms it.
9. The system updates the tenant's password and redirects them to the dashboard.

Postconditions:

- The tenant can log in using the new password.
- The temporary password is invalidated and cannot be used to log in again.

Use case 4: Update Password

Primary Actor: tenant

Preconditions:

- The tenant must be logged into their account.

Main Flow:

1. The tenant navigates to the "Account Settings" section of the website.
2. The tenant selects the "Change Password" option.
3. The system prompts the tenant to enter their current password.
4. The tenant enters their current password and submits the form.
5. If the current password is correct, the system prompts the tenant to enter a new password.
6. The tenant enters their new password and submits the form.
7. The system confirms the password change and logs the tenant out of all devices.

Postconditions:

- The tenant's password is updated and they must log in with their new password on all devices.

Use Case 5: Process Payment

Primary Actor: tenant

Preconditions:

- the tenant must be logged in.
- tenants must have a valid payment method saved in their account.
- The payment amount must be accurate and in the correct currency.

Postconditions:

- Payment is successfully processed.
- The transaction details are saved in the tenant's account.

Main Flow:

1. The tenant selects the "Make Payment" option.
2. The system presents the payment form, showing the payment amount and requesting confirmation.
3. The tenant confirms the payment.
4. The system retrieves the payment information and sends the payment request to the payment gateway.
5. The payment gateway processes the payment and returns a response to the system.
6. The system records the transaction details in the tenant's account.
7. The system presents a confirmation message to the tenant, indicating that the payment was successful.

Alternate Flows:

2a. Invalid Payment Method - If the tenant has not saved a valid payment method in their account, the system will prompt the tenant to add a payment method before proceeding.

4a. Payment Gateway Error - If the payment gateway returns an error response, the system will display an error message to the tenant, indicating that the payment could not be processed.

6a. Payment Record Error - If the system is unable to record the transaction details in the tenant's account, the system will display an error message to the tenant, indicating that the payment was successful but the transaction could not be recorded.

Use Case 6: Select Payment Method

Primary Actor: tenant

Goal in Context: The tenant wants to select a payment method to pay for their rent.

Preconditions:

- The tenant is logged into their account.
- The tenant has initiated the rent payment process.

Main Success Scenario:

1. The system presents the payment options available to the tenant, such as credit card, bank transfer, or online payment services.
2. The tenant selects the desired payment method.
3. If necessary, the tenant enters the payment details, such as credit card number, bank account information, or payment service login credentials.
4. The system verifies the payment details and informs the tenant of any errors or issues.
5. If the payment details are correct, the system processes the payment and updates the tenant's account and the apartment owner's account.
6. The system generates a receipt or confirmation of the payment for the tenant.

Alternate Flows:

4a. Payment verification error

6a. Payment detail error

Postconditions:

- The tenant's rent payment is processed and recorded in the system.
- The apartment owner receives the payment.
- The tenant receives a receipt or confirmation of the payment.

Use Case 7 : Online Payment

Primary Actor: tenant

Preconditions:

- The tenant must have a registered account with the system.
- The tenant must have a valid credit/debit card or any other online payment method supported by the system.

Postconditions:

- The payment is processed and confirmed by the system.
- The tenant receives a receipt for the payment.

Main Success Scenario:

1. The tenant selects the apartment they want to pay for from their account dashboard.
2. The system displays the payment details, including the amount due and the payment methods accepted.
3. The tenant selects the online payment option and enters the necessary details such as card number, expiration date, and CVV.
4. The system verifies the payment details and confirms the payment.
5. The system updates the payment status of the apartment in the database.
6. The system generates and sends a receipt of the payment to the tenant's registered email address.

Alternative Scenarios:

- The payment method is declined or invalid:
 1. The system displays an error message to the tenant, asking them to update their payment method or try a different one.
 2. The tenant updates their payment method and repeats the payment process.
- The tenant cancels the payment:
 1. The system cancels the payment and displays a message to the tenant indicating that the payment was not processed.
- The system encounters an error during the payment process:
 1. The system displays an error message to the tenant, asking them to try again later or contact tenant support for assistance.
 2. The tenant tries the payment process again later or contacts tenant support.

Use Case 8: Pay Rent in Cash

Primary Actor: tenant

Stakeholders:

- tenant: wants to pay rent in cash
- Manager: needs to collect rent from the tenant

Preconditions:

- The tenant is a customer in the building.
- The tenant has cash to pay rent.

Main Flow:

1. The tenant goes to the manager's office to pay rent in cash.
2. The manager asks for the tenant's apartment number to verify the amount of rent due.
3. The tenant gives the manager the cash to cover the rent.
4. The manager counts the cash and provides a receipt to the tenant.
5. The manager updates the rent payment record for the tenant in the system.

Postconditions:

- The tenant has paid rent in cash and received a receipt.
- The manager has updated the rent payment record for the tenant in the system.

Use case 9 : Update tenant Profile

Primary Actor: tenant

Goal in Context: The tenant updates their personal information.

Preconditions:

- The tenant must be registered in the system.
- The tenant must be logged in to their account.

Trigger: The tenant chooses to update their profile information.

Main Success Scenario:

1. The tenant selects the "Profile" option from the main menu.
2. The system displays the tenant's current profile information.
3. The tenant selects the "Edit" button.
4. The system displays a form where the tenant can update their personal information.
5. The tenant updates their information and selects the "Save" button.
6. The system updates the tenant's profile information and displays a success message.

Alternate Flows:

3a. If the tenant is not logged in, the system prompts them to log in.

3b. If the tenant is not registered, the system prompts them to register.

5a. If the tenant does not enter valid information, the system displays an error message and prompts the tenant to correct the errors.

Postconditions:

- The tenant's profile information has been updated in the system.
- The system has displayed a success message to the tenant.

Use case 10 : Submit Maintenance Request

Primary Actor: tenant

Goal in Context: The tenant requests for maintenance to be performed in their apartment.

Preconditions:

- The tenant must be logged into their account.
- The tenant must have an apartment in the building.
- The tenant must have a maintenance request to be performed.

Main Success Scenario:

1. The tenant navigates to the maintenance request page.
2. The tenant selects the type of maintenance request they require.
3. The tenant inputs a description of the problem they are experiencing.
4. The tenant submits the maintenance request.
5. The system confirms that the request has been received.

Postconditions:

- The maintenance request is added to the system and assigned a unique ID.
- The tenant is notified that the request has been received.

Alternative scenario:

4a. If the maintenance request is missing required information, the system displays an error message and prompts the tenant to input the missing information.

4b. If the maintenance request cannot be submitted due to a technical issue, the system displays an error message and prompts the tenant to try again later.

Use case 11 : Submit Security Issue Report

Primary Actor: tenant

Goal in Context: The tenant reports a security issue they have encountered within the apartment complex.

Preconditions:

- The tenant is logged into their account.
- The tenant has encountered a security issue within the apartment complex.
- The tenant has access to the "Report Security Issue" feature in the user interface.

Trigger: The tenant clicks on the "Report Security Issue" button in the user interface.

Main Success Scenario:

1. The system presents the tenant with a form to fill out with the details of the security issue they have encountered.
2. The tenant fills out the form with the following information:
 - Description of the security issue.
 - Location of the security issue.
 - Date and time the security issue was encountered.
 - Any additional comments or details.
3. The tenant submits the form.
4. The system sends a confirmation message to the tenant that their security issue report has been received.

Alternate Flows:

3a. The tenant cancels the report before submitting it.

- The system returns the tenant to the previous screen.

3b. The tenant submits an incomplete form.

- The system presents an error message and prompts the tenant to fill out all required fields.

3c. The tenant submits a report for a security issue that has already been reported.

- The system informs the tenant that the issue has already been reported and provides any available updates on the status of the issue.

Postconditions:

- The system records the security issue report and forwards it to the appropriate personnel for further action.

The tenant receives a confirmation message that their security issue report has been received.

-

Use Case 12 : Enter Building

Primary Actor: tenant

Preconditions:

1. The tenant is registered and has a valid access card or digital key to enter the building.
2. The tenant is standing in front of the building's entrance.

Main Scenario:

1. The tenant presents their access card or digital key to the building's access control system.
2. The access control system verifies the validity of the card or key.
3. If the card or key is valid, the access control system unlocks the door.
4. The tenant enters the building.

Postconditions:

1. The tenant is inside the building.

Alternative Scenarios:

1a. If the access card or digital key is invalid, the access control system does not unlock the door, and the tenant is denied entry. The tenant may contact the building management to resolve the issue.

1b. If the access control system fails to verify the validity of the access card or digital key, the door remains locked, and the tenant may contact the building management for assistance.

1c. If the access control system is temporarily unavailable, the door may be manually opened by the security guard on duty. The tenant must present their ID and provide their name and contact information to the security guard for record-keeping purposes. The security guard must inform the building management of the incident as soon as possible.

Use Case 13: Allow Guest to Enter Building

Primary Actor: tenant

Stakeholders and Interests:

- tenant: Wants to allow their guests to enter the building easily.
- Guest: Wants to be able to enter the building without any issues.
- Security Guard: Wants to ensure only authorized individuals enter the building.

Preconditions:

- The tenant is present at the building entrance with their guest.
- Guests do not have their own building access key or ID.

Postconditions:

- Guests are allowed to enter the building.

Main Success Scenario:

1. the tenant approaches the building entrance with their guest.
2. Security Guard asks for the tenant's building access key or ID.
3. The tenant informs the guard that they have a guest and would like to allow them entry.
4. Guard requests the name of the guest and confirms with the tenant that they are authorized to allow the guest entry.
5. The tenant provides the name of the guest and confirms authorization.
6. Guard verifies the information provided by the tenant and allows the guest to enter the building.

Extensions:

- 3a. the tenant is not authorized to allow their guest entry.
 - Guard informs the tenant that they are not authorized to allow their guest entry.
 - Use case ends.
- 4a. Guard is unable to verify the guest's information.
 - Guard requests further information from the tenant or guest.
 - Use case continues from step 4 with the additional information.
- 6a. Guard is still unable to verify the guest's information.

- Guard denies entry to the guest.
- Use case ends.

3.2.2 Use cases for Manager side

Use Case 1: Login

Primary Actor: Manager

Preconditions:

- The Manager has a valid username and password

Postconditions:

- The Manager is logged into the system and has access to the functionalities available for their role.

Main Scenario:

1. The Manager enters their valid username and password on the login page.
2. The system verifies the Manager's credentials.
3. The system redirects the Manager to their dashboard page.

Alternative Scenarios:

- Invalid Credentials:
 1. The system displays an error message indicating that the provided credentials are invalid.
 2. The Manager is prompted to enter their username and password again.
- Account Locked:
 1. If the Manager enters invalid credentials multiple times, the system locks their account.
 2. The system displays an error message indicating that the account is locked and to contact the administrator.

Use Case 2: Forgot Password

Primary Actor: Manager

Preconditions:

- The Manager has previously created an account and has forgotten their password.

Postconditions:

- The Manager is able to reset their password and regain access to the system.

Main Scenario:

1. The Manager clicks on the "Forgot Password" link on the login page.
2. The system prompts the Manager to enter their email address associated with their account.
3. The system sends a password reset link to the Manager's email address.
4. The Manager clicks on the password reset link in their email.
5. The system directs the Manager to a page where they can enter and confirm a new password.
6. The Manager enters and confirms a new password.
7. The system updates the Manager's account with the new password and redirects the Manager to the login page.
8. The Manager logs in using their new password.

Alternative Scenario:

- Invalid Email Address:
 1. If the email address provided by the Manager is not associated with an account, the system displays an error message indicating the email address is invalid.
 2. The Manager is prompted to enter a valid email address.

Use Case 3: Update Profile

Primary Actor: Manager

Preconditions:

- The Manager is logged into the system.

Postconditions:

- The Manager's profile information is updated in the system.

Main Scenario:

1. The Manager clicks on the "Profile" link in the navigation menu.
2. The system displays the Manager's current profile information.
3. The Manager makes the necessary changes to their profile information.
4. The Manager clicks on the "Save" button to update their profile information.
5. The system confirms the successful update of the Manager's profile information and displays the updated profile information.

Alternative Scenario:

- Invalid Input:
 1. If the Manager enters invalid input into any field, the system displays an error message indicating that the input is invalid.
 2. The Manager is prompted to enter valid input.

use case 4: Update Password

Primary Actor: Manager

pre condition:

The manager login to the system.

postconditions:

- The Manager's password is updated in the system.

Main Scenario:

1. The Manager navigates to the "Security" section of the system.
2. The system displays the "Change Password" option.
3. The Manager selects the "Change Password" option.
4. The system prompts the Manager to enter their current password and new password and confirm the new password.
5. The Manager enters their current password and new password and confirms the new password.
6. The Manager selects the "Save" option.
7. The system saves the updated password.
8. The system displays a success message.

Alternative Scenario:

- 4a. The Manager forgets their current password.
 1. The Manager selects the "Forgot Password" option.

2. The system prompts the Manager to enter their registered email address.
3. The system sends a password reset link to the Manager's registered email address.
4. The Manager follows the link to reset their password.

Use Case 5: Upload Room Information

Primary Actor: Manager

Preconditions:

- The Manager is logged in to the system.
- The Manager has the necessary information and media files to upload room information.

Postconditions:

- The room information is uploaded and visible in the system.

Main Scenario:

1. The Manager navigates to the "Rooms" section of the system.
2. The system displays the list of existing rooms.
3. The Manager selects the "Add Room" option.
4. The system prompts the Manager to enter the room information such as name, description, price, availability, and media files.
5. The Manager enters the required information and uploads the media files.
6. The Manager selects the "Save" option.
7. The system saves the new room information and displays a success message.
8. The new room information is now visible in the system.

Alternative Scenario:

- 4a. The Manager wants to update the information of an existing room.
 1. The Manager navigates to the "Rooms" section of the system.
 2. The system displays the list of existing rooms.
 3. The Manager selects the room to be updated.
 4. The system displays the room information.
 5. The Manager edits the desired room information.
 6. The Manager selects the "Save" option.
 7. The system saves the updated room information and displays a success message.

Use Case 6: Manage Bookings

Primary Actor: Manager

Preconditions:

- The Manager is logged in to the system.
- There are existing bookings in the system.

Postconditions:

- The selected booking is managed and updated in the system.

Main Scenario:

1. The Manager navigates to the "Bookings" section of the system.

The system displays the list of existing bookings.

The manager selects the booking.

2. The system displays the selected booking.
3. The Manager can perform various actions on the selected booking, depending on their level of permission and the current status of the booking.

The Manager saves the changes made to the booking by clicking on a "Save" or "Update" button.

4. The system updates the selected booking with the changes made by the Manager.

Alternative Scenarios:

If the Manager selects a booking that is already canceled or completed, the system may display an error message or restrict certain actions from being performed.

If the Manager encounters an issue with the system or their internet connection while managing a booking, they may choose to save their progress and return to it later, or contact technical support for assistance.

Use Case 7: Update Tenant Information

Primary Actor: Manager

Precondition: Manager is logged into the system and has the necessary permissions to edit tenant information.

Postcondition: Tenant information is updated in the system.

Main Scenario:

1. Manager selects the “Tenants” tab.
2. Manager searches for the tenant whose information they wish to update.
3. Manager clicks on the tenant’s name to view their details.
4. Manager edits the necessary fields in the tenant’s profile.
5. Manager saves the changes.
6. System updates the tenant information and displays a confirmation message to the manager.

Alternative Scenario:

5a. Manager decides not to save the changes and cancels the operation.

5b. Manager enters incorrect data or leaves required fields blank and is prompted to correct the errors.

Use case 8: Approve Maintenance Request

Primary Actor: Manager

Precondition: Manager is logged into the system and has the necessary permissions to approve maintenance requests.

Postcondition: Maintenance request is approved and assigned to the appropriate maintenance personnel.

Main Scenario:

1. Manager selects the “Maintenance Requests” tab.
2. Manager views the list of pending maintenance requests.
3. Manager selects the maintenance request they wish to approve.
4. Manager reviews the details of the request and assigns it to a maintenance personnel.
5. Manager clicks on the “Approve” button.
6. System sends a notification to the assigned maintenance personnel and updates the maintenance request status as “Approved.”

Alternative Scenario: 4a. Manager decides to decline the maintenance request and selects the “Decline” button. 4b. Manager assigns the maintenance request to the wrong personnel and needs to reassign it.

Use Case 9: Generate Monthly Rent Report

Primary Actor: Manager

Precondition: Manager is logged into the system and has the necessary permissions to generate monthly rent reports.

Postcondition: System generates a report of monthly rent payments and displays it to the manager.

Main Scenario:

1. Manager selects the "Reports" tab.
2. Manager selects the "Monthly Rent" report.
3. Manager inputs the necessary date range for the report.
4. Manager clicks the "Generate Report" button.
5. System generates the report and displays it to the manager.

Alternative Scenario: 3a. Manager inputs an invalid date range and is prompted to input valid dates. 3b. Manager selects a different type of report.

Use Case 10: Add New Room

Primary Actor: Manager

Precondition: Manager is logged into the system and has the necessary permissions to add new rooms.

Postcondition: New room is added to the system.

Main Scenario:

1. Manager selects the "Rooms" tab.
2. Manager clicks the "Add New Room" button.
3. Manager inputs the necessary information for the new room (e.g., room number, size, rent amount, etc.).
4. Manager saves the new room information.
5. System confirms the new room has been added to the system and displays a confirmation message to the manager.

Alternative Scenario: 4a. Manager decides not to save the new room information and cancels the operation. 4b. Manager enters incorrect data or leaves required fields blank and is prompted to correct the errors.

Use Case 11: Send Announcement

Primary Actor: Manager

Precondition: The manager is logged into the system and has access to the announcement feature.

Postcondition: The announcement is successfully sent to all tenants.

Main Scenario:

1. The manager selects the "Send Announcement" option from the dashboard.
2. The system prompts the manager to enter the announcement content.
3. The manager enters the announcement content and clicks the "Send" button.
4. The system sends the announcement to all tenants.
5. The system displays a confirmation message to the manager.

Alternative Scenario: 3a. If the manager decides not to send an announcement, they can choose to cancel the process. 4a. If the system encounters an error while sending the announcement, it displays an error message to the manager and allows them to try again.

Use Case 12: Assign Staff to Maintenance Request

Primary Actor: Manager

Precondition: The manager is logged into the system and has access to the maintenance request feature.

Postcondition: The manager assigns a maintenance staff member to a maintenance request.

Main Scenario:

1. The manager selects the "View Maintenance Requests" option from the dashboard.
2. The system displays a list of all maintenance requests.
3. The manager selects a maintenance request from the list to view the details.
4. The system displays the maintenance request details, including the description and apartment number.
5. The manager selects the "Assign Staff" option.
6. The system displays a list of available maintenance staff members.
7. The manager selects a staff member from the list and clicks the "Assign" button.
8. The system assigns the staff member to the maintenance request and displays a confirmation message to the manager.

Alternative Scenario:

6a. If there are no available maintenance staff members, the system displays a message to the manager stating that there are no staff members available to assign.

Use Case 13: Warning the Tenant

Primary Actor: Manager

Preconditions:

- The manager is logged into the system

- The tenant has violated a rule or policy
- The manager has already attempted to resolve the issue with the tenant verbally or through other means

Postconditions:

- The warning has been sent to the tenant
- The tenant has been notified of the violation and potential consequences if the behavior continues

Main Scenario:

1. The manager navigates to the tenant's profile or record in the system
2. The manager selects the option to send a warning to the tenant
3. The system prompts the manager to enter details about the violation and potential consequences
4. The manager enters the necessary information and submits the warning
5. The system sends a notification to the tenant's account and email, informing them of the warning and the reason for it

Alternative Scenarios:

- If the manager determines that the issue is severe enough to warrant more severe action, they may choose to escalate the situation to higher authorities or involve law enforcement. In this case, the warning would not be the final step in the process.
- If the tenant responds to the warning and takes corrective action, the manager may choose to withdraw the warning or take other steps to resolve the issue.

Use Case 14: Delete Tenant

Primary Actor: Manager

Precondition: The manager must be logged in and have appropriate authorization to delete a tenant account.

Postcondition: The tenant account is deleted from the system and all associated data is removed.

Main Scenario:

1. The manager navigates to the tenant management page.
2. The manager selects the tenant account they want to delete.
3. The manager confirms that they want to delete the tenant account.
4. The system removes the tenant account and all associated data from the system.

5. The system displays a message confirming that the tenant account has been deleted.

Alternative Scenarios:

1a. tenant account cannot be found:- The system displays an error message indicating that the tenant account cannot be found.

2a. Manager does not have appropriate authorization:- The system displays an error message indicating that the manager does not have appropriate authorization to delete a tenant account.

3a. Manager cancels deletion:

- The manager cancels the deletion of the tenant account.
- The system returns the manager to the tenant management page.

Use Case 15: Delete guard

Primary Actor: Manager

Precondition: The guard has requested to quit their job or be fired.

Postcondition: The guard's account and access to the building have been removed.

Main Scenario:

1. Manager receives a resignation request from a guard.
2. Manager verifies that the request is authentic.
3. Manager disables the guard's access card to the building.
4. Manager removes the guard's account from the system.
5. Manager informs other staff members of the guard's departure.
6. Manager archives the guard's employment information.

Alternative Scenario:

- 2a. Manager determines that the request is fraudulent.
- 2b. Manager denies the request and informs the guard of the decision.
- 2c. Manager takes appropriate disciplinary action against the guard if necessary.

Use case 16: Feedback

Primary Actor: Security guard

Preconditions:

- The security guard is logged into the security system.
- The guard has identified a security issue that needs to be reported to the manager.

Postconditions:

- The manager has received the security report from the guard.

Main Scenario:

1. The security guard observes a potential security issue.
2. The guard logs into the security system and accesses the "Report issue" feature.
3. The guard selects the type of issue and enters a description of the problem.
4. The guard attaches any relevant photos or videos to the report.
5. The guard submits the report to the manager for review.
6. The manager receives the report and reviews the details of the issue.
7. The manager takes appropriate action to address the issue.

Alternate Scenario:

1. If the manager is unavailable, the guard may escalate the issue to a higher authority for resolution.
2. If the issue is urgent and requires immediate attention, the guard may contact emergency services before submitting the report to the manager.

Use Case 17: Manager approves tenant registration request and schedules meeting for filling forms and signing in person

Primary Actor: Manager

Preconditions:

- Tenant has submitted a registration request.
- Manager has reviewed the request and determined that the tenant is eligible for renting the apartment.
- Manager has access to a calendar and can schedule appointments.

Main Flow:

1. Manager receives the tenant's registration request.
2. Manager reviews the request and determines that the tenant is eligible for renting the apartment.

3. Manager sends a message to the tenant, informing them that their request has been approved.
4. Manager suggests a few available dates and times for the tenant to come in for filling out the necessary forms and signing the lease agreement in person.
5. Tenants respond with their preferred date and time.
6. Manager schedules the appointment and adds it to their calendar.
7. Manager confirms the appointment with the tenant.

Postconditions:

- Tenant is informed that their registration request has been approved and has a scheduled appointment with the manager for filling out forms and signing the lease agreement in person.
- Manager's calendar is updated with the scheduled appointment.

Exceptions:

- If the manager determines that the tenant is not eligible for renting the apartment, they inform the tenant that their request has been denied and provide a reason.
- If the tenant does not respond to the manager's message within a reasonable timeframe, the manager follows up with a reminder message or phone call to schedule the appointment.
- If the scheduled appointment needs to be rescheduled for any reason, the manager notifies the tenant and suggests alternative dates and times.

3.2.3 Use cases for Owner side

Use case 1: Login

Primary Actor: Owner

Preconditions:

- The owner has a valid account in the system.

Main Scenario:

1. The owner navigates to the login page.
2. The system presents a form for the owner to enter their credentials.
3. The owner enters their username and password.
4. The system verifies the owner's credentials.
5. The system redirects the owner to their account dashboard.

Postconditions:

- The owner is successfully logged in to their account.

Alternative Scenarios:

4a. Invalid Credentials: If the system cannot verify the owner's credentials, an error message is displayed, and the owner is asked to try again or reset their password.

4b. Account Lockout: If the owner enters incorrect login information multiple times, their account is locked out for a set period of time to prevent unauthorized access.

Use case 2: Reset

Primary Actor: Owner

Preconditions:

- Owner has an existing account in the system.
- Owner has forgotten their password.

Main Scenario:

1. Owner navigates to the login page.
2. Owner clicks on the "forgot password" button.
3. System prompts Owner to enter their registered email address.
4. Owner enters their registered email address and clicks "submit".
5. System sends an email to the registered email address with a link to reset the password.
6. Owner clicks on the link provided in the email.
7. System prompts Owner to enter a new password.
8. Owner enters a new password and clicks "submit".
9. System displays a message that the password has been successfully reset.
10. Owner can now login to their account using the new password.

Alternative Scenario: 5a. If the email address entered by the Owner is not registered in the system, the system displays an error message and prompts the Owner to try again or contact the system provider.

Postconditions:

- Owner can login to their account using the new password.
- System logs the password change event for auditing purposes.

Use Case 3 : View Report from the Manager:

Primary Actor: Owner

Pre-Condition: The owner is logged into their account and has access to the system.

Post-Condition: The owner can view the report provided by the manager.

- Main Scenario:
 1. The owner selects the "View Report" option from their dashboard.
 2. The system displays the list of available reports.
 3. The owner selects the desired report to view.
 4. The system displays the report.
 5. The owner can review the report and take necessary action.
- Alternative scenario: If there are no reports available, the owner will be notified with a message indicating that there are no reports available at the moment.

Use Case 4 : Feedback for Complain:

Primary Actor: Owner

Pre-Condition: The owner is logged into their account and has access to the system.

Post-Condition: The owner has provided feedback to the complainant.

- Main Scenario:
 1. The owner selects the "Feedback" option from the dashboard.
 2. The system displays the list of complaints.
 3. The owner selects the desired complaint to address.
 4. The system displays the details of the complaint.
 5. The owner provides feedback to the complainant.
 6. The system records the feedback and updates the status of the complaint.

Alternative scenario: If the owner is unable to resolve the issue, they may need to escalate the complaint to a higher authority, such as a legal representative.

Use Case 5 : Delete the Manager:

1. Primary Actor: Owner
2. Pre-Condition: The owner is logged into their account and has access to the system.
3. Post-Condition: The manager is dismissed from their position.
4. Main Scenario:
 1. The owner selects the "Manage Managers" option from the dashboard.
 2. The system displays the list of available managers.
 3. The owner selects the desired manager to dismiss.
 4. The system displays the details of the manager and their contract.
 5. The owner selects the "Delete" option.
 6. The system displays a confirmation message.
 7. The owner confirms the dismissal of the manager.

8. The system removes the manager from their position and updates the contract details.

Use case 6 : Complain from Users:

Primary Actor: Tenant, Manager

Pre-Condition: The tenant or manager has access to the system and can submit a complaint.

Post-Condition: The complaint is recorded and sent to the owner for review.

- Main Scenario:
 1. The tenant or manager selects the "Submit Complaint" option from their dashboard.
 2. The system displays the complaint form.
 3. The user fills in the details of the complaint, including the type of complaint, description, and any supporting documentation.
 4. The user submits the complaint.
 5. The system records the complaint and sends it to the owner for review.
 6. The owner receives the complaint and selects the "View Complaint" option from their dashboard.
 7. The system displays the details of the complaint.
 8. The owner can provide feedback and take necessary action.

Alternative scenario: If the complaint is related to a specific tenant, the manager may need to speak with the tenant and gather more information before addressing the issue with the owner. Additionally, if the complaint is related to a specific property, the manager may need to conduct an inspection to determine if any repairs or maintenance are necessary.

Use case 7 : View Payment History

Primary Actor: Owner

Preconditions:

- The owner has logged in to their account.
- The payment history exists in the system.

Main Scenario:

1. The owner navigates to the "Payment History" section.
2. The system displays a list of all the payments made to the manager and security guards for each property.
3. The owner can view the payment amount, payment date, and payment method for each transaction.

4. If the owner wishes to view more details about a specific payment, they can click on that payment and the system will display more detailed information about that payment.

Postconditions:

- The owner has viewed the payment history.

Alternative Scenario:

- If the payment history does not exist in the system, the owner will be notified that there is no payment history to display.

Use case 8 : Make Payment:

Primary Actor: Owner

Preconditions:

- Owner must be logged into the system
- Owner must have a valid payment method registered in the system

Postconditions:

- Payment is made and recorded in the system

Main Scenario:

1. Owner selects the option to make a payment.
2. System presents a form for the owner to enter the payment details such as payment amount, payment method, and payment date.
3. Owner fills out the payment form and submits it.
4. System verifies the payment details and confirms the payment with the owner.
5. System records the payment in the system and updates the payment history.

Alternative Scenario: 4a. Payment details are not valid. 1. System displays an error message to the owner and prompts them to correct the payment details. 2. Owner corrects the payment details and resubmits the payment form.

Use Case 9 : add new manager to the system

Primary Actor: Owner

Preconditions:

- The owner is logged in to the system.

- The owner has the necessary permissions to add a new manager.

Main Scenario:

1. The owner selects the "Add Manager" option from the dashboard.
2. The system presents a form for the owner to fill in with the new manager's details such as name, email, and contact information.
3. The owner fills in the form and submits it.
4. The system verifies the information and creates a new account for the manager.
5. The system sends an email to the new manager with instructions on how to log in and set up their account.

Postconditions:

- The new manager is added to the system and can access their account.
- The owner receives a confirmation message indicating that the new manager has been added successfully.

Alternative Scenario:

- If the system is unable to verify the information provided by the owner, it will prompt the owner to correct the errors before proceeding.

3.2.4 Use cases for Security side

Use case 1: Login

Primary Actor: Security Guard

Preconditions:

- The Security Guard must have valid login credentials.
- The system must be up and running.

Postconditions:

- The Security Guard is authenticated and has access to the security features of the system.

Main Scenario:

1. The Security Guard navigates to the login page of the system.
2. The system prompts the Security Guard to enter their login credentials.
3. The Security Guard enters their username and password.
4. The system validates the credentials and redirects the Security Guard to the dashboard page.
5. The Security Guard has access to the system features.

Alternative Scenario: 3a. The Security Guard enters invalid login credentials. 1. The system displays an error message indicating that the credentials are invalid. 2. The Security Guard is prompted to re-enter their credentials. 3. The Security Guard repeats step 3 or cancels the login process.

Use Case 2: Security Guard Forgot Password

Primary Actor: Security Guard

Preconditions:

- Security Guard is registered in the system.
- Security Guard is not logged in to the system.
- The Security Guard has forgotten their password.

Postconditions:

- Security Guard is able to reset their password and log in to the system.

Main Scenario:

1. Security Guard navigates to the login page of the system.
2. Security Guard clicks on the "Forgot Password" link.
3. System prompts the Security Guard to enter their registered email address.
4. Security Guard enters their email address and clicks on the "Submit" button.
5. System sends an email to the Security Guard's registered email address with a link to reset their password.
6. The Security Guard checks their email and clicks on the link.
7. System displays a page where the Security Guard can enter a new password.
8. The Security Guard enters a new password and confirms it.
9. System validates the new password and updates it in the database.
10. System redirects Security Guard to the login page.
11. Security Guard logs in with their new password.

Alternative Scenarios:

- 3a. If the email address entered by Security Guard is not registered in the system, the system displays an error message and prompts Security Guard to enter a valid email address.
- 5a. If the email fails to send, the system displays an error message and prompts the Security Guard to try again later.
- 7a. If the link has expired or is invalid, the system displays an error message and prompts security Guard to try again with a new link.
- 9a. If the new password does not meet the system's password requirements, the system displays an error message and prompts Security Guard to enter a valid password.

Use case 3: Update profile

Primary actor: Security guard Preconditions:

- The security guard must be logged in.
- The security guard must have permission to update their profile.
- The system must provide an interface for the security guard to update their profile.

Postconditions: The security guard's profile is updated with the new information.

Main scenario:

1. The security guard selects the "Update Profile" option.
2. The system presents the security guard with a form to update their profile.
3. The security guard updates the necessary fields and submits the form.
4. The system validates the information and updates the security guard's profile.
5. The system confirms that the security guard's profile has been updated and displays the updated information.

Alternative scenarios:

- If the security guard enters invalid information, the system will prompt them to correct the errors and resubmit the form.
- If the system encounters an error while updating the security guard's profile, it will display an error message and prompt the security guard to try again later.

Use Case 4: Update Password

Primary Actor: Security Guard

Goal in Context: The security guard wants to update their password for their user account to maintain the security and integrity of the apartment management system.

Preconditions:

- The security guard must have a valid user account.
- The security guard must be logged into their account.

Trigger: The security guard clicks on the "Update Password" button in their account settings.

Main Success Scenario:

1. The system displays the "Update Password" form.
2. The security guard enters their current password, new password, and confirms the new password.
3. The system validates that the current password is correct and that the new password and confirmation match.

4. The system updates the security guard's password in the user database.
5. The system displays a confirmation message that the password has been successfully updated.

Alternate Flows:

3a. If the current password is incorrect:

1. The system displays an error message and prompts the security guard to re-enter their current password.

3b. If the new password and confirmation do not match:

1. The system displays an error message and prompts the security guard to re-enter their new password and confirmation.

Postconditions:

- The security guard's password is updated in the user database.

Business Rules:

- The new password must meet the system's password complexity requirements (e.g. minimum length, use of special characters, etc.).
- The security guard's new password cannot be the same as their previous password.

Notes:

- It is recommended that the security guard changes their password periodically to maintain the security of their account and the system overall.

Use Case 5: Monitor Visitors

Primary Actor: Security Guard

Precondition: The Security Guard is logged in and monitoring visitors at the entrance.

Postcondition: The Security Guard has monitored and recorded all visitors entering and leaving the building.

Main Scenario:

1. The Security Guard logs in to the system.

2. The Security Guard monitors visitors entering and leaving the building.
3. If a visitor is not registered, the Security Guard asks them to provide their name and purpose of visit.
4. The Security Guard checks the visitor's information against the list of approved visitors.
5. If the visitor is approved, the Security Guard records their information and allows them to enter.
6. If the visitor is not approved, the Security Guard denies them entry and notifies the relevant authorities.

Alternative Scenario: 3a. If a visitor refuses to provide their name and purpose of visit, the Security Guard denies them entry and notifies the relevant authorities.

Use Case 6: Emergency Response

Primary Actor: Security Guard

Precondition: An emergency situation has occurred in the building.

Postcondition: The Security Guard has responded to the emergency and notified the relevant authorities.

Main Scenario:

1. The Security Guard is alerted to the emergency situation.
2. The Security Guard contacts the relevant authorities (police, fire department, ambulance, etc.).
3. The Security Guard assesses the situation and provides any necessary assistance to those affected.
4. The Security Guard follows any emergency protocols established by the building management.

Alternative Scenario: 2a. If the Security Guard is unable to contact the relevant authorities, they escalate the situation to their supervisor and continue to provide assistance to those affected.

Use Case 7: Monitor CCTV

Primary Actor: Security Guard

Precondition: The Security Guard is logged in and monitoring the CCTV cameras in the building.

Postcondition: The Security Guard has monitored and recorded any suspicious activity captured on the CCTV cameras.

Main Scenario:

1. The Security Guard logs in to the system.
2. The Security Guard monitors the CCTV cameras for any suspicious activity.

3. If suspicious activity is detected, the Security Guard takes appropriate action (contacting authorities, alerting building management, etc.).
4. The Security Guard records any suspicious activity for future reference.

Alternative Scenario: 3a. If the suspicious activity is a false alarm, the Security Guard records the incident and continues monitoring the cameras.

Use Case 8: Clearance

Primary actor: Security guard

Preconditions:

- The tenant has requested to completely leave the apartment and delete it.
- The security guard has been informed of the tenant's request.
- The tenant has already removed all of their belongings from the apartment.

Postconditions:

- The security guard has inspected the apartment and all of its components.

Main scenario:

1. The tenant requests to completely leave the apartment and delete it.
2. The security guard is informed of the tenant's request.
3. The security guard goes to the apartment to inspect it.
4. The security guard checks all of the components in the apartment, such as furniture, appliances, and fixtures, to ensure they are in good condition.
5. If any components are damaged or missing, the security guard notes it down in their report.
6. If everything is in good condition, the security guard approves the tenant's request to leave the apartment and delete it.

Alternative scenario: 4a. If the security guard finds any damage or missing components, they inform the manager and take necessary steps to address the issue. The tenant may be held responsible for any damages or missing components.

Use Case 9: Feedback

Primary Actor: Security Guard

Preconditions:

- The tenant has submitted a security complaint to the management.
- The management has assigned the complaint to a security guard to investigate.
- The security guard has access to the complaint details.

Postconditions:

- The security guard has provided feedback on the complaint to the management and the tenant.

Main Scenario:

1. The security guard logs into the system using their credentials.
2. The system displays the assigned complaint details for investigation.
3. The security guard investigates the complaint and collects evidence as required.
4. The security guard updates the complaint status and enters their feedback in the system.
5. The system notifies the management and the tenant of the updated complaint status and feedback.

Alternative Scenarios:

- If the security guard finds the complaint to be invalid or false, they update the complaint status as such and provide a reason for the decision.
- If the security guard needs additional information to investigate the complaint, they request it from the management or the tenant.
- If the tenant provides additional information that affects the complaint status or feedback, the security guard updates the system accordingly.
- If the security guard is unable to resolve the complaint, they escalate it to the management for further action.

3.3 PERFORMANCE REQUIREMENTS:

The performance requirements of the apartment rental management system include both static and dynamic numerical requirements. Static numerical requirements include the number of terminals and simultaneous users to be supported, as well as the amount and type of information to be handled.

Dynamic numerical requirements include the number of transactions and tasks to be processed within certain time periods for both normal and peak workload conditions. All performance requirements must be measurable and stated in specific numerical terms. For example, the system must be able to process 95% of transactions within 1 second.

Numerical limits applied to specific functions must be specified as part of the processing description of that function. The system must be able to handle both normal and peak workload conditions and ensure that all transactions are processed accurately and in a timely manner.

3.4 LOGICAL DATABASE REQUIREMENTS

3.4.1 Types of information used by various functions:

- Tenant Information: This includes tenant name, contact information, lease start and end dates, rent payment history, and any outstanding balances.
- Property Information: This includes property name, address, type of property, number of units, and any amenities.
- Lease Information: This includes lease start and end dates, rent amount, security deposit, and any special conditions or addendums.
- Maintenance Requests: This includes information about maintenance requests made by tenants, including the type of request, date submitted, and status of the request.
- Payment Information: This includes information about rent payments made by tenants, including the amount paid, date paid, and payment method.
- Accounting Information: This includes information about expenses related to the property, such as utilities, repairs, and maintenance.
- Marketing Information: This includes information about marketing campaigns, rental rates, and any special offers or promotions.
- Employee Information: This includes information about employees who work at the property, including their names, job titles, and contact information.
- Legal Information: This includes information about any legal issues related to the property, such as evictions, lawsuits, or zoning regulations.

Each of these types of information is used by various functions within the apartment rent management system, such as tenant screening, lease management, rent collection, maintenance management, accounting, and marketing.

3.4.2 Accessing capabilities:

Accessing capabilities refer to the ways in which users can interact with the database to retrieve, modify, and manipulate the data stored within it. Here are some examples of accessing capabilities that might be required for an apartment rent management system:

1. User authentication: The database should require users to log in with appropriate credentials before they can access the data. This ensures that only authorized personnel can view or modify sensitive information.
2. Search and filtering: Users should be able to search for specific data within the database, using criteria such as tenant name, apartment number, rental amount, etc. The database should also allow users to filter data based on various parameters.
3. CRUD operations: CRUD stands for Create, Read, Update, and Delete. Users should be able to perform all these operations on the data stored within the database, as needed.

4. Reporting: The database should be able to generate various types of reports, such as rent roll reports, occupancy reports, financial statements, etc.
5. Tenant access: Tenants should be able to view their account information, including payment history, lease terms, and maintenance requests.
6. Landlord access: Landlords should be able to view tenant information, including contact details, lease terms, and payment history. They should also be able to manage maintenance requests and send out rent reminders.
7. Property manager access: Property managers should have access to all tenant and landlord information, as well as financial records, maintenance logs, and other important data. They should be able to manage leases, set rental rates, and create reports.
8. Maintenance staff access: Maintenance staff should be able to view and update maintenance requests, including status updates and notes on the work that has been done.
9. Accounting staff access: Accounting staff should be able to view financial records, including rent payments, expenses, and invoices. They should be able to create reports and perform other financial tasks as needed.
10. Administrative access: Administrative staff should have full access to the database and be able to manage user accounts, security settings, and other aspects of the system. They should also be able to perform backups and other maintenance tasks to ensure the system runs smoothly.

Overall, the accessing capabilities of the apartment rent management system should be designed to allow different types of users to access and manage the information they need to perform their jobs effectively, while also ensuring that sensitive data is protected and secure.

3.4.3 Frequency of use:

1. Tenant information - frequently accessed by property managers and leasing agents, occasionally accessed by maintenance staff.
2. Lease agreements - frequently accessed by property managers and leasing agents, occasionally accessed by tenants.
3. Rent payment records - frequently accessed by property managers and accounting staff, occasionally accessed by tenants.
4. Maintenance requests - frequently accessed by maintenance staff, occasionally accessed by property managers and tenants.
5. Security incident reports - occasionally accessed by property managers and security staff.
6. Move-in/move-out inspection reports - frequently accessed by property managers and maintenance staff.
7. Vendor contracts - occasionally accessed by property managers and accounting staff.
8. Building/property information - occasionally accessed by property managers and leasing agents.

3.4.4 Data entities and their relationships:

- Apartment:
 - Apartment Number: Integer
 - Address: String
 - Rent Amount: Decimal
 - Number of Bedrooms: Integer
 - Number of Bathrooms: Integer
 - Square Footage: Decimal
 - Availability: Boolean
 - Lease Start Date: Date
 - Lease End Date: Date
- Tenant:
 - Tenant ID: Integer
 - Name: String
 - Phone Number: String
 - Email: String
 - Lease Start Date: Date
 - Lease End Date: Date
 - Monthly Rent Amount: Decimal
 - Security Deposit: Decimal
- Maintenance Request:
 - Request ID: Integer
 - Tenant ID: Integer
 - Apartment Number: Integer
 - Description: String
 - Request Date: Date
 - Status: Enum (Pending, In Progress, Completed)
- Payment:
 - Payment ID: Integer
 - Tenant ID: Integer
 - Apartment Number: Integer
 - Payment Date: Date
 - Amount Paid: Decimal
- Manager:
 - Manager ID: Integer
 - Name: String
 - Phone Number: String
 - Email: String
 - Password (string, hashed and salted)
- Security:
 - Security ID: Integer

- Name: String
- Phone Number: String
- Email: String
- Password (string, hashed and salted)

Relationships:

- Each tenant belongs to one apartment.
- Each apartment belongs to one building.
- Each building has one manager.
- Each apartment can have multiple maintenance requests.
- Each apartment can have multiple complaints.
- Each complaint can be assigned to one security personnel.

3.4.5 Integrity constraints:

- Tenant information should be unique, and each tenant should have a unique identifier.
- Apartment information should be unique, and each apartment should have a unique identifier.
- Lease agreements should be unique, and each lease should have a unique identifier.
- Payment information should be associated with a valid tenant and a valid lease agreement.
- Maintenance requests should be associated with a valid tenant and a valid apartment.

3.4.6 Data retention requirements:

- Tenant information and lease agreements should be retained for a certain period after the lease term ends for record-keeping purposes.
- Payment information and maintenance requests should be retained for a certain period for accounting and legal compliance purposes.

3.4.7 Design constraints

1. Standards compliance:
 - Compliance with data privacy regulations such as GDPR, CCPA, etc.
 - Compliance with accounting standards and procedures.
 - Compliance with property management laws and regulations.
2. Hardware limitations:
 - The system should be designed to work with the hardware and software specifications provided by the client.
 - The system should be scalable to accommodate future hardware upgrades.
3. User interface constraints:

- The user interface should be intuitive and easy to use for both tenants and managers.
 - The system should be designed with accessibility standards in mind to ensure it can be used by people with disabilities.
4. Security constraints:
 - The system should use secure encryption algorithms to protect sensitive data.
 - The system should be designed with a multi-level security model to ensure only authorized users can access certain data.
 - The system should have an audit trail feature to track user activity and detect any security breaches.
 5. Integration constraints:
 - The system should be designed to integrate with other property management software and tools, such as accounting software and maintenance management tools.
 - The system should be designed to integrate with third-party services, such as payment gateways and background check providers.
 6. Performance constraints:
 - The system should be designed to handle a large volume of data and transactions.
 - The system should be optimized to provide fast response times and minimize downtime.
 7. Data constraints:
 - The system should be designed to handle different types of data, including text, numbers, images, and files.
 - The system should be designed to handle data validation and data consistency checks to ensure data accuracy.
 - The system should be designed to handle data backup and recovery to prevent data loss.

3.4.8 Software system attributes

3.4.8.1 Reliability

- The system must be reliable in handling large amounts of data, and be able to perform critical functions without failure, such as processing rent payments or maintenance requests.
- The system must have a backup and recovery plan in place to ensure that data is not lost in the event of a system failure.

3.4.8.2 Availability

- The system must be available to tenants and property managers 24/7, with minimal downtime for maintenance or upgrades.
- The system should have a high level of uptime, with appropriate measures in place to ensure rapid recovery from any downtime.

3.4.8.3 Security

- The system must protect sensitive tenant information, such as contact information and rent payment history, from unauthorized access.
- The system must enforce strong password policies to prevent unauthorized access to tenant and property manager accounts.
- The system must encrypt sensitive data in transit and at rest.
- The system must have auditing capabilities to monitor user activity and identify potential security breaches.

3.4.8.4 Maintainability

- The system must be modular and well-organized to facilitate easy maintenance and updates.
- The system should be designed with clear separation of concerns and well-defined interfaces to minimize the impact of changes on other parts of the system.
- The system should have thorough documentation and code comments to aid in maintenance and debugging.

3.4.8.5 Portability

- The system should be designed with a modern, widely-supported technology stack to ensure portability across different operating systems and hosting environments.
- The system should minimize the use of platform-specific APIs or libraries to ensure maximum portability.
- The system should be designed with scalability in mind, to ensure that it can easily be expanded to support a growing user base.

4 APPENDIX

4.1 INPUT AND OUTPUT SAMPLE FORMATS

Register


First Name *
John

Last Name *
Daniel


Email Address *
sampl@gmail.com


Password *
.....

Confirm Password *
.....

Meeting Time *
04 / 03 / 2023 , -- :-- --


Phone Number *
123456789

Number of Family Members Attending *
6

very good apartment

☒ I agree to the terms and conditions

REGISTER


Sign in

Email Address *

Password *

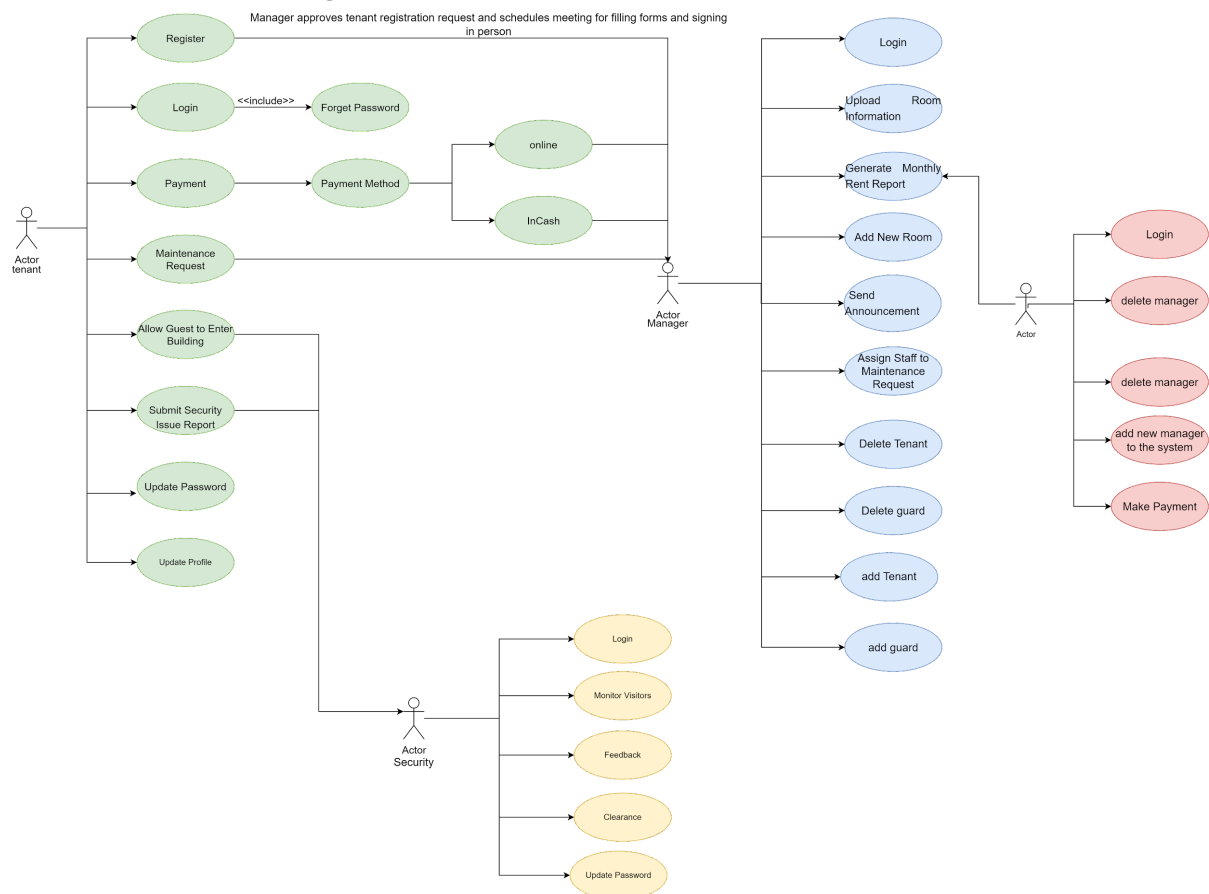
☐ Remember me

SIGN IN

[Forgot password?](#) [Don't have an account? Sign Up](#)

Powered by Material UI

4.2 Use case diagram



4.3 BACKGROUND AND IMPACTS

Background:

Apartment rental management systems are software applications that provide property managers and landlords with a comprehensive platform for managing their rental properties. These systems typically include features such as tenant screening, lease management, rent collection, maintenance tracking, and financial reporting. The goal of these systems is to streamline the rental management process, reduce administrative overhead, and improve the overall efficiency of property management operations.

Impact:

The impact of an apartment rental management system can be significant for property managers and landlords. By automating many of the routine tasks associated with rental management, such as rent collection and maintenance tracking, these systems can save managers and landlords significant amounts of time and effort. Additionally, by providing a centralized platform for managing all aspects of rental operations, these systems can help improve communication between tenants, managers, and landlords, resulting in a better overall tenant experience.

From a financial perspective, an apartment rental management system can help landlords and property managers reduce costs and increase revenue. By automating rent collection, for example, these systems can help reduce late payments and ensure that rent is collected on time. Similarly, by providing tools for tracking maintenance and repairs, these systems can help prevent costly repairs and ensure that properties are well-maintained, reducing the risk of vacancies and turnover.

Overall, an apartment rental management system can have a significant impact on the efficiency and profitability of rental property operations, making it an essential tool for property managers and landlords.