1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) outlines the functional and non-functional requirements for the **Real Estate Management System (REMS)**, version 1.0. The REMS is being developed to automate the management of various types of real estate properties, including residential, commercial, and industrial properties. It will streamline the tasks related to property listings, tenant management, property maintenance, and financial transactions.

The purpose of this document is to provide a clear and detailed description of the system's requirements, ensuring that the project meets the needs of all stakeholders, including property managers, tenants, and system administrators. The SRS will serve as a reference for:

- **System Designers**: To design the architecture and subsystems.
- **Developers**: To implement the system.
- **Testers**: To validate the system's behavior and performance.
- **Project Managers**: To track the scope and ensure alignment with business objectives.
- **End Users**: To ensure that the final product meets their expectations and needs.

This document is meant to reduce ambiguity by providing concrete requirements, establishing clear expectations for both the client and development team. Additionally, this SRS will serve as a baseline for verifying that the system is functioning as expected throughout the development lifecycle.

The scope of the product includes the automation of the following real estate management processes:

1. Property Listings and Searches.
2. User Registration and Authentication.
3. Property Reservation and Cancellation.
4. Payment Processing for Rent and Utilities.
5. Tenant-Owner Communication.
6. User and Property Verification.
7. System Reporting and Analytics.

This document describes the system's requirements comprehensively. However, it may be updated as additional information is gathered during further project development phases.

1.2 Document Conventions

This document follows several formatting and structural conventions to maintain clarity and ease of understanding:

- **Bold text** is used to highlight important terms, concepts, or requirements.

- *Italicized text* is used for emphasis on specific parts of the document such as definitions or assumptions.
- **Requirements are uniquely numbered**, following the pattern REQ-1, REQ-2, etc., to ensure easy reference and traceability throughout the development and testing phases.
- For sections where requirements are yet to be finalized or where further clarification is required, the placeholder `<TBD>` (To Be Determined) is used. These placeholders will be replaced once the required information becomes available.
- Functional requirements are listed first, followed by non-functional requirements.
- The document uses a hierarchical structure, where major sections are numbered 1, 2, 3, etc., with subsections numbered 1.1, 1.2, and so on.

The conventions in this SRS aim to provide consistency and make it easier for the reader to follow and interpret the requirements. The format and structure are aligned with IEEE standards for software documentation, particularly IEEE 830-1998.

1.3 Intended Audience and Reading Suggestions

The primary audience of this document includes various stakeholders involved in the development, implementation, and use of the Real Estate Management System. Below is a breakdown of the different reader types and suggestions on how each group should approach this document:

- **Developers**: Developers will use this document to understand the detailed system functionality. They should focus on sections detailing the overall system design, system features, external interface requirements, and functional requirements. The sections on non-functional requirements such as performance, security, and maintainability will also be relevant for system design.
- **Project Managers**: Project managers should use this document to track project milestones and ensure the project remains within scope. Key sections for project managers include the product scope, functional overview, constraints, assumptions, and dependencies.
- **Testers**: The testing team will need to carefully review the functional requirements to ensure all use cases and system behaviors are covered during validation. Additionally, non-functional requirements such as performance, usability, and security will provide insight into the necessary testing criteria.
- **End Users**: This document serves as a reference for end users to ensure that the functionality meets their needs. Key sections for end users include the overall product description, user classes and characteristics, and functional requirements.
- **System Architects**: The system architecture team should refer to the design and implementation constraints and external interface requirements to ensure that the system design complies with these constraints. The assumptions and dependencies section is also critical for architects to address any external factors that could influence system behavior.

- **Documentation Writers**: Writers tasked with creating user manuals, help files, and guides will need to refer to the sections on user interfaces, user documentation, and functional requirements.

It is recommended that readers start with the **Overall Description** and **Product Scope** to gain a broad understanding of the system's objectives. From there, they can focus on the sections most pertinent to their role.

1.4 Product Scope

The **Real Estate Management System (REMS)** is being developed as a comprehensive solution to address the various challenges associated with property management in the real estate industry. This system is aimed at automating the core tasks of property listings, tenant management, payment processing, and communication between tenants and property managers. The primary goals of REMS are to improve operational efficiency, enhance user experience, ensure data security, and maintain regulatory compliance.

The system will be utilized by several classes of users, including property managers, tenants, maintenance staff, and financial administrators. Each user group will have a distinct role within the system, and their interactions will be facilitated through various user interfaces designed for ease of use and accessibility.

Key product benefits include:

- **Automation of Routine Tasks**: Property managers will no longer need to manually track tenant payments, maintenance requests, and lease agreements. These tasks will be automated, allowing managers to focus on strategic decision-making.
- **Improved Tenant Relations**: Tenants will be able to communicate directly with property managers through integrated messaging, pay their rent online, and submit maintenance requests in a hassle-free manner.
- **Data Security and Compliance**: The system will ensure that sensitive tenant and financial data is stored securely in compliance with industry regulations, such as GDPR and PCI DSS.
- **Scalability**: The system will be designed to scale with growing user numbers and data volumes, allowing it to handle increased demand without sacrificing performance.

The system is strategically designed to support both small and large-scale real estate operations, making it suitable for real estate firms managing a few properties as well as larger organizations managing extensive property portfolios across multiple locations.

1.5 References

The following documents, guidelines, and standards are referenced throughout this SRS. These documents provide context for the requirements and ensure that the system complies with applicable industry standards:

1. **IEEE Standard 830-1998** – IEEE Recommended Practice for Software Requirements Specifications.
2. **ISO/IEC 27001:2013** – International standard for information security management systems.
3. **NIST SP 800-53** – Guidelines for securing information systems, including privacy controls.
4. **PCI DSS** – Payment Card Industry Data Security Standard for secure processing of financial transactions.
5. **Corporate Guidelines on Property Management** – Internal guidelines provided by the client organization, detailing property management protocols, data retention policies, and service-level agreements.
6. **RESTful API Documentation** – Reference for integration with external APIs for payment gateways and property listing services.
7. **Web Accessibility Guidelines (WCAG 2.1)** – Standards for ensuring the system's accessibility for users with disabilities.

This section may be updated as additional references or sources become relevant during the course of the project. Where appropriate, the full version number and date of each document will be included to ensure accuracy and traceability.

2. Overall Description

2.1 Product Perspective

The **Real Estate Management System (REMS)** is designed to be a comprehensive, scalable, and user-friendly solution to manage the lifecycle of real estate properties. This system integrates various modules to facilitate property management, user registration, payments, and communication between tenants, property managers, and maintenance personnel.

REMS is part of a broader enterprise suite that connects seamlessly with existing software solutions such as financial management systems, customer relationship management (CRM) tools, and document storage systems. The system will replace the current manual, paper-based processes for managing properties and tenants, offering automation and integration across departments within the real estate business.

Product Context: The REMS will operate as a web-based, cloud-deployed application with a supporting mobile interface for tenants and property managers. The cloud architecture will enable property managers to access and manage properties from anywhere, and tenants will have access to self-service features, such as payment processing, maintenance requests, and lease management.

Key Components of the Overall System:

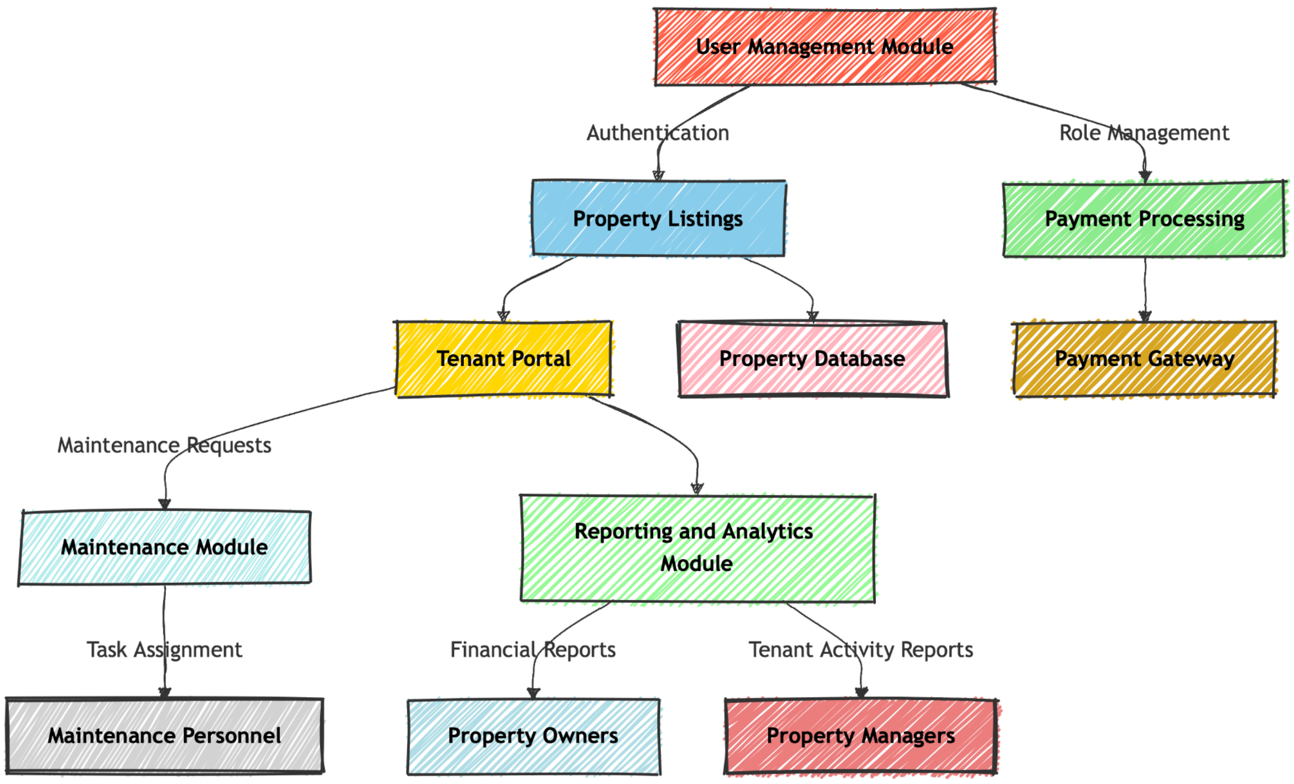
1. **Property Listing Database**: Stores all details related to available, booked, or archived properties. This component will enable searching, filtering, and booking functionalities.

2. **User Management Module**: Handles user registration, authentication, and role-based access control. This ensures that property owners, managers, tenants, and administrators can log in and access only the information and functionality relevant to their role.
3. **Payment Processing**: The system will integrate with external payment gateways (e.g., PayPal, Stripe, UPI) to handle rent payments, deposits, and fees.
4. **Maintenance and Support Module**: Allows tenants to submit maintenance requests, while property managers can assign tasks to maintenance personnel and track resolution status.
5. **Reporting and Analytics Module**: Provides real-time dashboards and reports on financial transactions, tenant occupancy rates, and property performance.

The system will interface with several external components, including:

- **Payment Gateways**: To process rent and maintenance payments.
- **Email/SMS Providers**: For communication between tenants and property managers (e.g., notifications for overdue rent, maintenance alerts).
- **Government Verification Services**: To handle tenant and property owner verification in accordance with local laws.

The following diagram illustrates the system's major components and their interconnections:



2.2 Product Functions

The system will include a wide range of functions, each designed to address specific needs for property management. These functions can be grouped into core features, each targeting different user roles within the system. Below is a summary of the key functionalities:

1. **User Management**:
    - User Registration: Allows tenants, property owners, and managers to create accounts and log into the system.
    - Authentication: Provides a secure login mechanism, supporting multi-factor authentication (MFA) and encrypted password storage.
    - Role Management: Assigns roles (e.g., Tenant, Manager, Owner) with varying levels of access to system features.
2. **Property Listings**:
    - Property Search: Enables users to search for available properties using filters such as location, size, rent price, and availability.
    - Booking System: Allows tenants to book properties directly from the listings page.
    - Property Archiving: Allows managers to archive properties that are no longer available, preserving records for future reference.
3. **Payment Processing**:
    - Rent Payments: Facilitates the online payment of rent via multiple gateways.
    - Maintenance Fee Payments: Enables tenants to pay any additional fees related to property maintenance, utilities, or amenities.
    - Payment History: Displays a history of all transactions made within the system for tenant and manager reference.
4. **Tenant-Owner Communication**:
    - Messaging System: Provides a secure, real-time messaging system between tenants and property managers for maintenance requests, queries, or complaints.
    - Notifications: Sends automatic notifications via email or SMS regarding rent due, upcoming maintenance, or lease expiration.
5. **Verification and Security**:
    - Tenant Verification: Integrates with government services for identity and background checks.
    - Secure Document Upload: Allows users to upload and manage documents securely (e.g., rental agreements, government IDs).
6. **Maintenance Requests**:
    - Request Submission: Tenants can submit maintenance requests through their user portal.
    - Task Assignment: Property managers can assign these requests to maintenance personnel and track their status.
    - Task Tracking: Maintenance personnel can update task status in real-time, allowing tenants to track the progress.
7. **Reporting and Analytics**:

- Real-Time Dashboards: Provides property managers with a high-level view of occupancy rates, tenant satisfaction, and financial performance.
- Financial Reports: Generates automated financial statements detailing rent collection, expenses, and outstanding dues.
- Tenant Activity Reports: Tracks tenant behavior, such as frequency of late payments or maintenance requests.

2.3 User Classes and Characteristics

The Real Estate Management System will cater to multiple classes of users, each with distinct roles, needs, and permissions within the system. The key user classes are as follows:

1. **Property Managers**:
    - Role: Oversee property listings, tenant relationships, lease agreements, and financial transactions.
    - Characteristics: Technologically proficient, with experience in using real estate management software. They require access to comprehensive system features, including tenant communication, payments, property updates, and reporting.
    - Primary Interaction: Regular users of the system for property and tenant management tasks, requiring access to advanced features and analytics.
2. **Tenants**:
    - Role: Rent properties and submit maintenance requests through the system.
    - Characteristics: Varying levels of technical expertise. The system should be easy to navigate, providing simple, straightforward functionality such as rent payment and maintenance requests.
    - Primary Interaction: Use the system primarily for rent payments, reviewing lease agreements, and communicating with property managers.
3. **Property Owners**:
    - Role: View property performance, occupancy rates, and financial reports.
    - Characteristics: High-level users who are interested in financial outcomes and property performance. They may require access to reporting features but will interact less frequently with the system compared to managers and tenants.
    - Primary Interaction: View-only access to analytics and financial reports.
4. **Maintenance Personnel**:
    - Role: Handle tenant maintenance requests and track task completion.
    - Characteristics: Primarily need task-related functionality and real-time updates on assignments.
    - Primary Interaction: Limited to task management and status updates.

2.4 Operating Environment

The Real Estate Management System is designed to be accessible through a variety of platforms, ensuring broad compatibility and ease of access for all user types. The software will operate in the following environments:

- **Web-Based Application**: Accessible via major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.
- **Mobile Platform**: Mobile-optimized web version available for iOS and Android devices, enabling tenants and property managers to interact with the system from smartphones or tablets.
- **Operating Systems**: Supports Windows, macOS, and Linux for desktop access. For mobile, supports iOS 13+ and Android 8.0+.
- **Cloud Hosting**: The application will be hosted on AWS or Azure for scalability and reliability, with database replication and automatic failover to ensure high availability.

2.5 Design and Implementation Constraints

Several constraints will influence the system's design and implementation. These constraints include:

1. **Compliance with Regulatory Standards**: The system must comply with regulatory requirements for data security and privacy, including GDPR, PCI DSS for financial transactions, and local property management laws.
2. **Hardware Limitations**: The application must be optimized to perform efficiently on a variety of hardware platforms, including older mobile devices.
3. **Technology Stack**: The system will use a predefined technology stack:
   - **Frontend**: React.js with Tailwind CSS for the user interface.
   - **Backend**: Node.js with Express for server-side logic.
   - **Wa Database**: MongoDB for data storage, ensuring scalability for large datasets.
4. **Integration with Existing Systems**: The system must seamlessly integrate with existing financial systems and property management tools without disruption.
5. **Performance Requirements**: The system must load web pages within 2 seconds under normal operating conditions, and payment transactions must be processed within 3 seconds.

2.6 User Documentation

The system will be accompanied by comprehensive user documentation, available in both online and downloadable formats. The documentation will include:

1. **User Manuals**: Detailed guides for property managers, tenants, and owners, covering all system features, such as property listings, payments, and maintenance requests.
2. **Online Help**: Contextual help accessible from within the application, offering tips and instructions for common tasks.
3. **Video Tutorials**: Short instructional videos explaining how to navigate the system and use key features, targeting less technically proficient users.
4. **FAQ Section**: A repository of frequently asked questions, providing quick solutions to common problems.
5. **Support Portal**: A dedicated support portal allowing users to submit queries or request technical assistance.

2.7 Assumptions and Dependencies

The successful implementation of the Real Estate Management System relies on several assumptions and external dependencies:

1. **Payment Gateway Integration**: The system assumes that third-party payment gateway APIs will be available and functional for secure processing of transactions.
2. **Government Verification API**: It is assumed that external APIs for tenant and property verification will be available for integration.
3. **Browser Compatibility**: The system assumes that end users will access the platform through modern, updated web browsers.
4. **Internet Connectivity**: The system depends on reliable internet connectivity for both tenants and property managers, as the application is cloud-based.
5. **External System Integration**: The system assumes that any third-party financial, CRM, or verification systems will remain stable and available throughout the implementation.

3. External Interface Requirements

This section provides a comprehensive breakdown of the interfaces that allow the **Real Estate Management System (REMS)** to communicate and interact with users, hardware, software, and communication protocols. These interfaces ensure seamless integration and functionality across different environments and platforms, and they govern how data is exchanged, processed, and rendered both internally and externally.

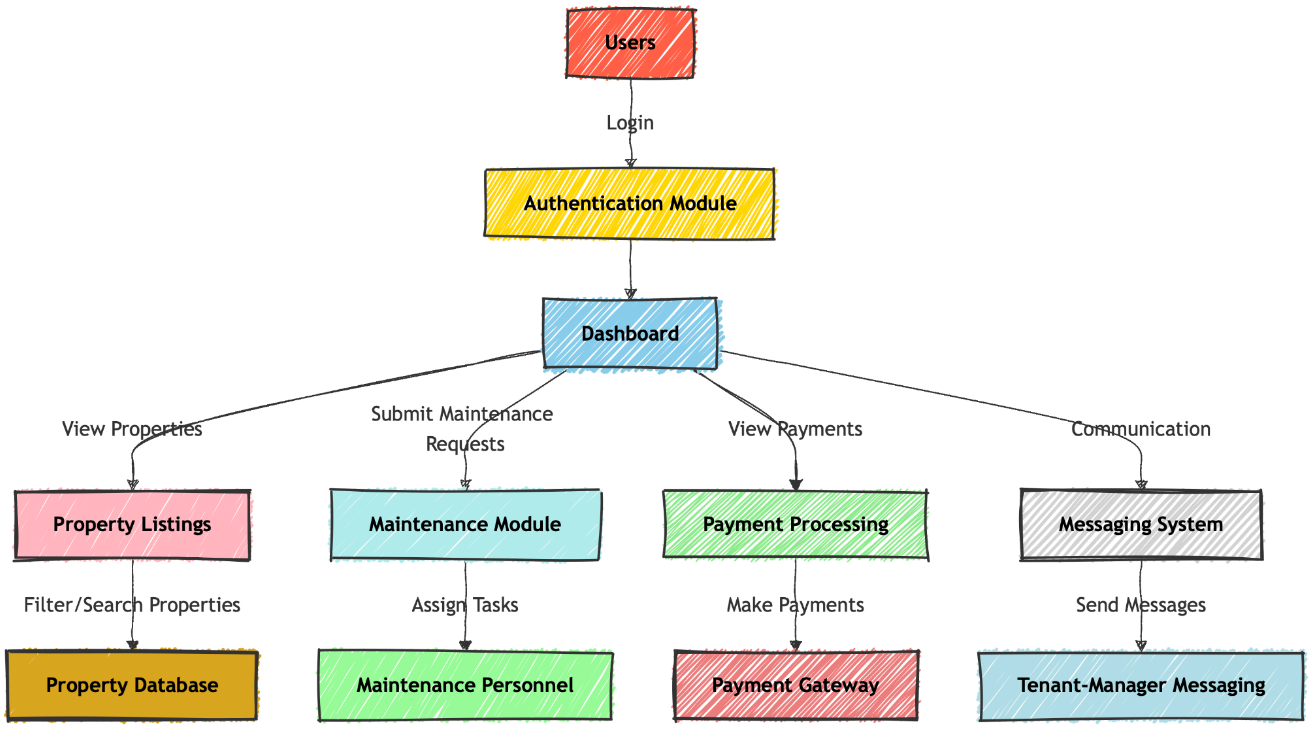
3.1 User Interfaces

The user interface (UI) is a critical component of the system, acting as the primary point of interaction between users and the REMS. It must provide an intuitive, responsive, and consistent experience across various devices, ensuring accessibility and ease of use for different user roles. The following details outline the characteristics and structure of the UI.

- **General UI Design Principles**:
  - The UI will be designed following **Material Design Guidelines** to ensure a modern, clean, and intuitive interface.
  - **Responsive Design**: The UI will dynamically adapt to various screen sizes and resolutions, offering seamless functionality on desktops, laptops, tablets, and smartphones. The design will prioritize usability on small screens, with collapsible menus, large touch targets, and minimized scrolling.
  - **Consistency**: Across all sections and modules, a consistent look and feel will be maintained to reduce user confusion. Consistent use of icons, color schemes, typography, and layout principles will enhance familiarity and ease of navigation.
  - **Color Contrast & Accessibility**: High contrast between background and text will ensure readability, especially for users with visual impairments. Adherence to **WCAG 2.1 AA** guidelines will be mandatory for accessibility compliance.
- **Detailed Layout**:

    - **Login/Authentication Pages**: These will include fields for username and password, with the option to log in using **OAuth** providers (Google, Microsoft, etc.). Error messages will be prominently displayed for invalid credentials or failed authentication attempts.
    - **Dashboard**:
        - Property managers will have access to a multi-widget dashboard showing real-time property listings, maintenance requests, financial summaries, and notifications.
        - Tenants will see an interface focused on their lease agreements, rent payments, and submitted maintenance requests.
        - The dashboard will include customizable widgets, allowing users to personalize their experience by adding, removing, or resizing modules (e.g., seeing rent status as a priority).
        - Each dashboard component will display its status (e.g., **Rent Due**, **Overdue Maintenance**, **Upcoming Lease Expiration**) using **colored indicators**: green for completed, yellow for in-progress, and red for overdue.
- **Interaction Elements**:
    - **Forms and Input Fields**:
        - All input fields (e.g., user registration, property listing forms, payment details) will have real-time validation using **JavaScript** and **server-side validation** for added security. Fields will display inline error messages next to invalid inputs.
        - Auto-fill and **field masking** (e.g., for dates, phone numbers, and credit card inputs) will improve user experience and minimize input errors.
    - **Navigation**:
        - A **side navigation bar** will be used for primary navigation (e.g., access to properties, tenants, payments, reports). The navigation will collapse into icons on smaller screens for improved usability.
        - **Breadcrumb navigation** will be included for users to easily trace their steps within the system, helping them backtrack to previous screens.
    - **Buttons and Controls**:
        - Standard button types will be used across the platform: **primary** buttons (e.g., for submitting forms), **secondary** buttons (e.g., cancel actions), and **icon buttons** (e.g., for editing or deleting items).
        - Buttons will trigger immediate feedback, such as changing color or displaying a loading animation during server interactions, to prevent double-clicks and ensure users know the system is responding.
- **Accessibility Features**:
    - **Keyboard Navigation**: The system will support full keyboard navigation (e.g., tabbing through form fields, pressing "Enter" to submit) for users with mobility impairments. Shortcuts will allow users to quickly access key system features (e.g., opening the payments page with Alt+P).

    - **Screen Readers**: HTML5 ARIA attributes will be integrated to ensure compatibility with screen readers, providing detailed descriptions of interactive elements.
    - **Color Blindness Support**: Custom color palettes will be available for users with color vision deficiencies, offering different visual modes to enhance accessibility.
- **Error Handling and Alerts**:
    - **Error Modals**: For critical errors (e.g., inability to submit a payment), the UI will display modal dialogs that prevent further interaction until the error is addressed.
    - **Warning Banners**: Non-critical warnings, such as incomplete form fields, will be displayed using banners or tooltips, providing users with real-time guidance on how to resolve the issue.
    - **Success Notifications**: Upon successful completion of actions (e.g., rent payment submission), the system will display a confirmation message at the top of the screen or as a toast notification.



3.2 Hardware Interfaces

The hardware interfaces for REMS ensure that the system can interact with both client devices (laptops, desktops, tablets, smartphones) and specialized hardware (e.g., sensors, IoT devices). These interactions ensure that users can access the system from various devices and that data from physical hardware is transmitted reliably to the system.

- **User Devices**:
    - The REMS will support **cross-platform compatibility**. Specifically:

        - **Desktops & Laptops**: The system will be fully compatible with devices running the latest versions of Windows, macOS, and Linux distributions. The minimum hardware requirements will be 4GB of RAM and a dual-core processor to handle real-time updates and large datasets.
        - **Tablets & Smartphones**: The mobile interface will be optimized for iOS (version 13 and above) and Android (version 8.0 and above). Mobile devices with smaller screens will use adaptive layouts to maintain usability, such as larger buttons and vertical scrolling pages.
        - **Printers**: The system will support **printing** through standard **network printers** or local printers connected to the user's device. Printed documents (e.g., lease agreements, financial reports) will be formatted for A4 or letter-sized paper, depending on regional standards.
- **IoT Sensors & Smart Devices**:
    - REMS will integrate with **IoT-based sensors** and smart devices installed within properties. Examples include:
        - **Smart Thermostats**: REMS will read data from IoT-enabled thermostats, allowing property managers to track and optimize heating/cooling expenses.
        - **Occupancy Sensors**: These sensors will be integrated to track room usage, automatically adjusting HVAC settings and providing insights into energy usage.
    - Data from IoT devices will be collected using the **MQTT protocol**, which is lightweight and optimized for machine-to-machine communication. **CoAP (Constrained Application Protocol)** will be used for devices with constrained resources.
- **Integration with Cameras**:
    - For security and monitoring purposes, the system will support integration with **IP cameras** installed in common areas. These cameras will stream live video to the property manager's dashboard, using **RTSP (Real-Time Streaming Protocol)** for reliable video transmission.
    - **Motion detection sensors** integrated with the cameras will send alerts in case of unusual activity, allowing property managers to respond swiftly.
- **Hardware-Specific Security**:
    - REMS will support integration with **smart locks** and **keyless entry systems** for secure access to properties. **Zigbee** or **Z-Wave** communication protocols will be used to allow seamless control of locks via the system's interface.
    - The system will maintain logs of all access attempts, providing an audit trail for property managers.

3.3 Software Interfaces

REMS must interface with various external and internal software components to deliver comprehensive functionality. These interfaces will ensure smooth data transfer between REMS and other systems, allowing for efficient property management, tenant interactions, payment processing, and regulatory compliance.

- **Database Interface**:
    - The system will interface with **MongoDB** to store and retrieve data, including tenant information, property listings, maintenance requests, and transaction histories.
        - **CRUD Operations**: All Create, Read, Update, and Delete operations will be performed securely via MongoDB's native API. To enhance performance, **database indexing** will be employed for frequently queried fields (e.g., tenant names, property addresses).
        - **Connection Pooling**: The backend will implement connection pooling to handle high volumes of database queries efficiently, maintaining optimal performance even during peak usage.
    - **Backup and Recovery**: The database will interface with an external cloud storage solution (e.g., AWS S3) for daily backups. The backup system will use **incremental backups** to reduce storage costs and ensure quick recovery times in case of data loss.
- **Third-Party API Integrations**:
    - **Payment Gateways**:
        - REMS will integrate with third-party payment gateways such as **Stripe**, **PayPal**, and **UPI** to process rent payments and security deposits. Each payment gateway's API will be accessed using **RESTful services** over **HTTPS** to ensure secure transactions.
        - **Transaction Logging**: Every payment transaction, whether successful or failed, will be logged in the system for audit purposes, including timestamps, amounts, payment methods, and transaction IDs.
        - **PCI Compliance**: All payment processes will adhere to **PCI DSS** (Payment Card Industry Data Security Standards) to ensure that sensitive payment details (e.g., credit card numbers) are never stored on REMS servers.
    - **Government Verification APIs**:
        - REMS will integrate with government services to perform **tenant background checks** and **property owner verification**. These services will require OAuth 2.0 authentication for secure access.
        - **Data Handling**: Sensitive tenant data, such as personal identification numbers, will be transferred securely using **AES-256 encryption** and logged in compliance with privacy laws (e.g., GDPR).
    - **Document Management Systems (DMS)**:
        - REMS will interface with external **document storage and management systems** (e.g., **Dropbox**, **Google Drive**, **OneDrive**) to store and retrieve lease agreements, legal documents, and property images.
        - Documents will be tagged and indexed automatically for easy retrieval, and all communications with the DMS will use **RESTful APIs** over **SSL/TLS** connections to ensure data security.
- **Authentication**:

    - REMS will offer both **traditional username/password-based authentication** and **Single Sign-On (SSO)**using **OAuth 2.0** providers (Google, Microsoft).
    - **Session Management**: JWT tokens will be used to manage session states, with tokens stored in the browser's **localStorage** for web users. Tokens will expire after a set period (e.g., 60 minutes) to prevent session hijacking.
    - **Role-Based Access Control (RBAC)**: Each user will have specific permissions assigned based on their role (e.g., tenant, property manager, owner). The system will enforce RBAC for sensitive actions like editing financial reports or adding new properties.

3.4 Communications Interfaces

The REMS platform requires robust communication capabilities to interact with users and external systems. This includes email notifications, SMS alerts, and real-time communication channels that ensure timely interactions between tenants, property managers, and maintenance personnel.

- **Email Notifications**:
    - REMS will use **SMTP** for sending emails, integrating with external providers such as **SendGrid** or **Mailgun**to handle email distribution. Common use cases for email notifications include:
        - **Rent Payment Reminders**: Automated emails will remind tenants of upcoming rent deadlines and overdue payments.
        - **Lease Expiration Alerts**: Notifications will be sent 30 days before a lease expiration date, allowing tenants and managers to renew or make alternative arrangements.
        - **Maintenance Request Updates**: Tenants will receive email updates regarding the status of their submitted maintenance requests (e.g., approved, in-progress, completed).
    - **Security**: Emails will be sent over **TLS-secured** channels, ensuring that sensitive information (e.g., payment confirmations) is not intercepted during transmission. All emails will include an **unsubscribe link** to comply with anti-spam regulations.
- **SMS Alerts**:
    - The system will integrate with an SMS provider (e.g., **Twilio**) to send urgent messages via text, ensuring that users are immediately notified in case of critical events:
        - **Emergency Maintenance**: Property managers can trigger SMS alerts to notify tenants of emergency repairs that may affect utilities or safety.
        - **Late Rent Notifications**: Tenants will receive SMS reminders when rent is overdue, with a link to the online payment portal.
    - SMS delivery will be tracked, and the system will resend failed messages after a configurable time delay. SMS messages will be sent in plain text but secured using **2FA (Two-Factor Authentication)** where appropriate.
- **Web Communication**:

    - **WebSockets** will be used for real-time communication between tenants and property managers within the system's messaging platform. This allows users to send and receive messages without needing to refresh the page.
        - The **tenant-manager messaging system** will allow tenants to report issues (e.g., broken appliances) and receive immediate responses from property managers.
        - All messages exchanged will be **logged** and **encrypted**, ensuring that a permanent record is maintained for future reference.
    - **RESTful APIs** will be used for asynchronous communication, such as fetching new data without reloading the entire page (e.g., new payment records, updated property listings).
- **Security Protocols for Communication**:
    - All communication (email, SMS, web) will occur over **HTTPS**, ensuring the data is encrypted during transit. **SSL/TLS certificates** will be renewed automatically to ensure ongoing security.
    - **End-to-End Encryption (E2EE)** will be used for sensitive data transmitted via WebSockets and messaging systems, ensuring that no unauthorized third party can intercept communication between tenants and property managers.
    - **Rate Limiting**: To prevent abuse (e.g., spam emails or excessive SMS notifications), rate-limiting algorithms will be applied to ensure that users cannot trigger an excessive number of notifications in a short time frame.

4. System Features

This section outlines the key features of the **Real Estate Management System (REMS)**, detailing their behavior, associated functional requirements, user interactions, and how they integrate with the broader system.

4.1 System Feature 1: User Management

4.1.1 Description and Priority

The **User Management** module is a core feature that handles the registration, authentication, profile management, and role-based access control for all users. The system supports different user roles such as **tenants**, **property managers**, **owners**, and **administrators**. User Management is a **High Priority** feature, as it controls system access and protects sensitive data through authentication mechanisms.

4.1.2 Stimulus/Response Sequences

- **Registration Process**:
    1. User clicks on the "Register" button on the login page.
    2. The system prompts the user to input personal details (e.g., name, email, government ID).

3. After completing the form, the user submits it.
4. The system validates the data (e.g., checking email format, ensuring required fields are filled).
5. If valid, the system stores the data in the database and sends a verification email.
6. The user clicks the verification link, activating the account.
7. The user can now log into the system.

- **Login Process**:
    1. User enters their email and password on the login page.
    2. The system checks the credentials against the database.
    3. If correct, the system generates a **JWT token** and creates a session for the user.
    4. The user is redirected to their dashboard based on their role (tenant, manager, etc.).
    5. If incorrect, the system prompts an error message.
- **Role Management**:
    1. An admin assigns roles to users during registration (e.g., tenant, manager).
    2. Each role is associated with a set of permissions (e.g., tenants can submit maintenance requests, managers can manage property listings).
    3. The system restricts actions based on the user's role.

4.1.3 Functional Requirements

1. **REQ-1**: The system must allow users to register using an email address and a strong password.
2. **REQ-2**: Passwords must be hashed and stored securely using industry-standard hashing algorithms such as **bcrypt**.
3. **REQ-3**: Users must be able to reset their password by requesting a password reset link, sent via email.
4. **REQ-4**: The system must support **OAuth 2.0** for third-party authentication via Google, Facebook, and Microsoft.
5. **REQ-5**: The system must enforce **role-based access control (RBAC)**, ensuring that each user has access only to features relevant to their role.
6. **REQ-6**: Admin users must be able to view, edit, and remove other users.
7. **REQ-7**: The system must maintain a **log of user activities**, including login times, changes made to the profile, and system actions.

4.2 System Feature 2: Property Listings and Search

4.2.1 Description and Priority

The **Property Listings** module allows users to browse and filter available properties. Property managers can add, update, and archive listings, while tenants can search for and view detailed property information. The listings module supports **CRUD (Create, Read, Update, Delete)** operations and advanced filtering capabilities. This is a **High Priority** feature, as it forms the core of tenant interaction with the system.

4.2.2 Stimulus/Response Sequences

- **Property Manager Actions**:
  1. Manager logs in and navigates to the property management section.
  2. Manager clicks “Add Property” and fills out details such as property type, size, rent price, and available amenities.
  3. The system verifies the data and creates a new listing in the database.
  4. Manager can update or archive properties as needed.
- **Tenant Search Process**:
  1. Tenant enters search criteria (e.g., location, price range, number of bedrooms) in the search bar.
  2. The system retrieves properties that match the criteria.
  3. The tenant clicks on a property to view detailed information such as photos, rent terms, and contact information.
  4. Tenant can express interest by submitting a contact form or booking a viewing.

4.2.3 Functional Requirements

1. **REQ-8**: The system must allow property managers to add, edit, and delete property listings.
2. **REQ-9**: The system must provide tenants with the ability to filter properties by location, rent price, size, and property type (e.g., commercial, residential).
3. **REQ-10**: Each property listing must include details such as the property’s address, rent amount, available amenities, and photos.
4. **REQ-11**: The system must allow tenants to save properties to a **favorites list** for later reference.
5. **REQ-12**: The system must support **real-time search** with auto-suggestions as tenants type their search criteria.
6. **REQ-13**: Properties that are no longer available must be marked as “archived” but retained in the system for historical and reporting purposes.
7. **REQ-14**: Property managers must be able to upload images, floor plans, and documents related to each property.

4.3 System Feature 3: Payment Processing

4.3.1 Description and Priority

The **Payment Processing** module facilitates rent payments, security deposits, and other property-related fees. Tenants can view their outstanding payments and submit payments through multiple gateways (e.g., **Stripe**, **PayPal**, **UPI**). This feature is critical for ensuring timely rent collection and is a **High Priority** feature.

4.3.2 Stimulus/Response Sequences

- **Rent Payment**:
  1. Tenant logs into the system and navigates to the “Payments” section.
  2. The system displays any outstanding rent payments, including due dates and the total amount.
  3. Tenant clicks on “Pay Now” and selects their preferred payment method.
  4. The system redirects the tenant to the chosen payment gateway (e.g., Stripe).
  5. After completing the payment, the system sends a confirmation email and updates the payment status in the tenant’s account.
- **Payment History**:
  1. Tenant navigates to the “Payment History” section.
  2. The system retrieves a record of all past payments, including dates, amounts, and transaction IDs.
  3. The tenant can download payment receipts as PDF files.

4.3.3 Functional Requirements

1. **REQ-15**: The system must support integration with **Stripe**, **PayPal**, and **UPI** for processing payments.
2. **REQ-16**: Tenants must be able to view their payment history, including transaction details and payment status.
3. **REQ-17**: The system must support **automatic rent reminders** via email and SMS, sent 5 days before the rent is due.
4. **REQ-18**: The system must allow tenants to set up **recurring payments** for rent through their preferred payment method.
5. **REQ-19**: Payment data must be stored securely, following **PCI DSS** (Payment Card Industry Data Security Standard) guidelines.
6. **REQ-20**: The system must allow property managers to view payment records and generate **financial reports**.
7. **REQ-21**: Failed payments must trigger a retry mechanism, allowing tenants to attempt payment again.

4.4 System Feature 4: Maintenance Requests

4.4.1 Description and Priority

The **Maintenance Requests** feature allows tenants to report issues (e.g., plumbing, electrical problems) and track the progress of repairs. Property managers can assign maintenance personnel to tasks and monitor their status. This is a **Medium Priority** feature, as it supports tenant satisfaction and ensures properties are well-maintained.

4.4.2 Stimulus/Response Sequences

- **Submitting a Maintenance Request**:
    1. Tenant logs in and navigates to the “Maintenance” section.
    2. Tenant clicks “Submit Request” and fills out a form describing the issue (e.g., broken faucet).
    3. The system validates the form and creates a maintenance ticket.
    4. Property manager receives a notification about the new request.
- **Maintenance Task Assignment**:
    1. Property manager views open maintenance requests.
    2. Manager assigns the task to available maintenance personnel, setting a priority level (e.g., high, medium, low).
    3. Maintenance personnel receives the task in their dashboard.
    4. Task status is updated as the issue is addressed, and the tenant is notified upon completion.

4.4.3 Functional Requirements

1. **REQ-22**: Tenants must be able to submit maintenance requests via a form that captures issue details, location, and urgency.
2. **REQ-23**: Property managers must be able to assign tasks to maintenance personnel based on availability and priority.
3. **REQ-24**: The system must send notifications to tenants when a maintenance request is submitted, accepted, and completed.
4. **REQ-25**: Maintenance personnel must have access to a dashboard showing their assigned tasks, deadlines, and priority levels.
5. **REQ-26**: The system must allow property managers to generate **maintenance reports**, showing request frequency, time-to-completion, and costs.
6. **REQ-27**: The system must support the **escalation** of maintenance requests if they remain unresolved after a set time.

4.5 System Feature 5: Tenant-Owner Communication

4.5.1 Description and Priority

The **Tenant-Owner Communication** module provides a secure, real-time messaging platform that allows tenants to communicate with property managers and owners. Tenants can inquire about property-related issues, and property managers can send announcements, such as maintenance schedules or rent payment reminders. This feature ensures open communication between both parties, contributing to tenant satisfaction. The priority of this feature is **Medium**, as it is critical for day-to-day interactions but not essential for system functionality.

4.5.2 Stimulus/Response Sequences

- **Sending a Message**:
    1. Tenant navigates to the "Messages" section in the dashboard.
    2. Tenant selects the recipient (e.g., property manager or owner) and types a message regarding an issue or inquiry.

    3. Tenant clicks "Send," and the system sends the message in real-time.
    4. The property manager receives a notification of the new message and can respond directly in the system.

- **Viewing System Announcements**:
    1. Property manager posts an announcement (e.g., scheduled maintenance) that is visible to all tenants in the property.
    2. Tenants receive an in-app notification and email regarding the announcement.
    3. Tenants can acknowledge the announcement or respond with questions if permitted.

4.5.3 Functional Requirements

1. **REQ-28**: The system must support **real-time messaging** between tenants and property managers.
2. **REQ-29**: Messages must be stored securely, ensuring **end-to-end encryption** for sensitive communications.
3. **REQ-30**: The system must allow tenants to view a history of their conversations with property managers.
4. **REQ-31**: The system must allow property managers to send bulk announcements to tenants within a property or across multiple properties.
5. **REQ-32**: The system must support **notifications** via email and SMS when new messages are received.
6. **REQ-33**: The system must display **read receipts** for messages, indicating when a message has been viewed by the recipient.
7. **REQ-34**: The system must prevent message spam by implementing **rate limits**, ensuring that users cannot send more than a set number of messages in a short time.

4.6 System Feature 6: Verification and Security

4.6.1 Description and Priority

The **Verification and Security** module is responsible for ensuring the integrity of user data by validating user identities, performing background checks, and securing sensitive tenant and property owner information. It integrates with government APIs for background checks and police verification of tenants. Additionally, this feature handles **two-factor authentication (2FA)** and data encryption protocols. The priority of this feature is **High**, as it safeguards the system's data and ensures legal compliance with property management laws.

4.6.2 Stimulus/Response Sequences

- **Tenant Verification**:
    1. Tenant submits required documents (e.g., government-issued ID, proof of income) during the registration process.

    2. The system interfaces with external government verification services via API to validate the documents.
    3. The system notifies the tenant and property manager once the verification is complete.
- **Police Verification**:
    1. After tenant registration, property managers can initiate a police verification request via the system.
    2. The system sends a verification request to local law enforcement agencies via a secure API.
    3. Once the verification process is completed, the results are automatically updated in the tenant's profile.

4.6.3 Functional Requirements

1. **REQ-35**: The system must integrate with government APIs to verify tenant identities and background information.
2. **REQ-36**: The system must allow tenants to upload documents for verification purposes, supporting common formats such as PDF and JPEG.
3. **REQ-37**: The system must implement **two-factor authentication (2FA)** using SMS or email verification.
4. **REQ-38**: The system must notify property managers when tenant verification or police verification is complete.
5. **REQ-39**: All sensitive user data, including documents, must be stored securely using **AES-256 encryption**.
6. **REQ-40**: The system must allow property managers to view and download verification reports for auditing purposes.
7. **REQ-41**: The system must support configurable verification workflows based on regional requirements (e.g., police verification may only be required in certain regions).
8. **REQ-42**: The system must comply with **GDPR** and local data protection laws regarding user data retention and deletion.

4.7 System Feature 7: Reporting and Analytics

4.7.1 Description and Priority

The **Reporting and Analytics** module provides users (primarily property managers and owners) with real-time insights into financial performance, tenant behavior, and property occupancy. It generates detailed reports on rent payments, maintenance costs, and tenant satisfaction. The system supports customizable dashboards and automated report generation. This is a **Medium Priority** feature, essential for long-term business decision-making but not for day-to-day operations.

4.7.2 Stimulus/Response Sequences

- **Generating Financial Reports**:
    1. Property manager navigates to the "Reports" section.
    2. Manager selects the type of report (e.g., rent collection, expenses) and specifies the date range.
    3. The system retrieves the relevant data and generates a detailed report, displaying it on the dashboard.
    4. Manager can export the report in PDF or Excel format.
- **Viewing Real-Time Analytics**:
    1. Manager opens the "Analytics Dashboard."
    2. The system displays a summary of key performance indicators (KPIs) such as rent collection rate, occupancy rate, and pending maintenance requests.
    3. Manager can click on any KPI to view detailed data and trends over time.

4.7.3 Functional Requirements

1. **REQ-43**: The system must provide property managers with the ability to generate financial reports, including rent collection, maintenance costs, and utility expenses.
2. **REQ-44**: The system must support real-time analytics, displaying KPIs such as rent collection rate, tenant occupancy, and pending maintenance requests.
3. **REQ-45**: Users must be able to customize their **dashboard widgets**, adding or removing reports based on their preferences.
4. **REQ-46**: The system must allow users to export reports in multiple formats, including PDF and Excel.
5. **REQ-47**: The system must generate **automated reports** based on a set schedule (e.g., monthly financial summaries), sent via email to property managers and owners.
6. **REQ-48**: The system must maintain a **historical record** of all generated reports for future reference.
7. **REQ-49**: Reports must be generated based on real-time data, ensuring accuracy and up-to-date information.

4.8 System Feature 8: Portal Integration for Property Buying and Selling

4.8.1 Description and Priority

The **Portal Integration** feature allows property managers to list properties for sale and enables prospective buyers to search for and purchase properties directly through the system. It provides a seamless experience for managing both rental and sales listings. This feature is critical for property management firms that handle both rental and sales transactions, making it a **High Priority** feature for those businesses.

4.8.2 Stimulus/Response Sequences

- **Listing a Property for Sale**:
    1. Property manager navigates to the "Property Sales" section.

2. Manager clicks "List Property for Sale" and fills out the necessary details (e.g., asking price, property condition).
3. The system verifies the listing details and posts the property for sale.
4. Buyers can now search and view the property listing.

- **Buyer Purchase Process**:
    1. Buyer navigates to the property portal and searches for available properties.
    2. Buyer clicks on a property listing to view details such as asking price, property photos, and contact information.
    3. Buyer submits a purchase inquiry through the system, which notifies the property manager.
    4. The system allows property managers to track purchase inquiries and manage the sales pipeline.

4.8.3 Functional Requirements

1. **REQ-50**: The system must allow property managers to list properties for sale, providing details such as asking price, property condition, and sale terms.
2. **REQ-51**: Buyers must be able to search for properties for sale using filters such as price, location, and property type.
3. **REQ-52**: The system must allow buyers to submit inquiries or requests for viewings directly through the portal.
4. **REQ-53**: Property managers must be able to manage purchase inquiries, tracking them through the sales pipeline.
5. **REQ-54**: The system must support integration with external real estate listing services, allowing properties to be syndicated across multiple platforms.
6. **REQ-55**: The system must allow property managers to mark properties as “sold” once a transaction is complete, removing them from the active listings.
7. **REQ-56**: The system must support **contract management**, allowing property managers to upload and store sales contracts related to property transactions.

4.9 System Feature 9: Document Management

4.9.1 Description and Priority

The **Document Management** feature enables property managers, tenants, and owners to securely upload, store, and share documents related to property transactions, leases, maintenance records, and other important paperwork. The system provides a centralized location for managing documents with version control, audit logs, and permission-based access. This is a **High Priority** feature, especially for large property management firms handling multiple properties.

4.9.2 Stimulus/Response Sequences

- **Uploading Documents**:

    1. Tenant or property manager navigates to the "Documents" section.
    2. User clicks "Upload Document" and selects the file from their device.
    3. The system validates the file format (e.g., PDF, DOCX) and uploads it to the server.
    4. The document is tagged and categorized for easy retrieval later.
- **Viewing and Sharing Documents**:
    1. User searches for a document using filters such as document type, property name, or date.
    2. The system retrieves and displays the document.
    3. User can download or share the document with other users (e.g., sending a lease agreement to a tenant).

4.9.3 Functional Requirements

1. **REQ-57**: The system must allow users to upload and store documents in common formats, including PDF, DOCX, and JPEG.
2. **REQ-58**: Documents must be stored securely with **AES-256 encryption** to protect sensitive information.
3. **REQ-59**: The system must allow users to categorize and tag documents for easy retrieval.
4. **REQ-60**: Users must be able to share documents with other users within the system, with permission-based access control.
5. **REQ-61**: The system must provide version control for documents, allowing users to revert to previous versions when necessary.
6. **REQ-62**: The system must maintain an **audit log** of all document-related actions, including uploads, downloads, and modifications.
7. **REQ-63**: The system must support the **bulk upload** of documents for property managers handling multiple properties.

4.10 System Feature 10: Notifications and Alerts

4.10.1 Description and Priority

The **Notifications and Alerts** feature ensures that all users are kept up to date with important system events. This includes rent due reminders, maintenance request updates, system announcements, and payment confirmations. The system provides in-app notifications, as well as notifications via email and SMS. The priority of this feature is **Medium**, as it improves user engagement and ensures timely communication but is not central to core system functionality.

4.10.2 Stimulus/Response Sequences

- **Rent Due Reminder**:
    1. The system checks the tenant’s account daily to see if the rent due date is approaching (e.g., within 5 days).
    2. If the rent is due, the system sends a reminder via email and SMS to the tenant.

    3. A notification banner also appears in the tenant's dashboard with a "Pay Now" button.

- **Maintenance Request Update**:
    1. The tenant submits a maintenance request, and the property manager assigns the request to maintenance personnel.
    2. The system sends a notification to the tenant when the request status changes (e.g., in progress, completed).
    3. The tenant receives an in-app notification and email with the updated status.
- **System Announcements**:
    1. The property manager creates a system-wide announcement (e.g., scheduled maintenance affecting all tenants).
    2. The system sends the announcement as a notification to all tenants via email and in-app alerts.
    3. Tenants can acknowledge the announcement or ask questions directly from the notification.

4.10.3 Functional Requirements

1. **REQ-64**: The system must allow property managers to create and send system-wide announcements that appear in both email and in-app notifications.
2. **REQ-65**: Tenants must receive automatic reminders for upcoming rent payments, overdue rent, and lease expiration dates.
3. **REQ-66**: The system must send **real-time notifications** to tenants when their maintenance request status changes.
4. **REQ-67**: The system must support **SMS notifications**, particularly for high-priority events such as late rent payments or emergency maintenance alerts.
5. **REQ-68**: Users must be able to manage their notification preferences, opting in or out of specific types of notifications (e.g., email only, SMS only, or in-app notifications only).
6. **REQ-69**: Notifications must include clickable links to relevant system actions (e.g., a "Pay Now" button for rent due notifications or a "View Request" button for maintenance updates).
7. **REQ-70**: The system must log all notifications, keeping a **history** of what was sent, when, and to whom, for audit purposes.

4.11 System Feature 11: Lease Management

4.11.1 Description and Priority

The **Lease Management** module facilitates the creation, storage, and management of lease agreements between tenants and property managers. It automates the process of generating lease contracts, supports electronic signatures, and tracks lease start and end dates. This feature is essential for maintaining legal agreements and tracking tenant occupancy, making it a **High Priority** feature.

4.11.2 Stimulus/Response Sequences

- **Creating a New Lease Agreement**:
  1. Property manager navigates to the "Lease Management" section.
  2. Manager clicks “Create New Lease” and inputs details such as tenant name, property address, lease duration, and rent amount.
  3. The system generates a pre-formatted lease agreement using a customizable template.
  4. The tenant receives an email with a link to review and sign the lease electronically.
  5. Once signed, the system finalizes the agreement and updates the tenant's profile with the lease information.
- **Lease Expiration Notification**:
  1. The system monitors lease expiration dates for all tenants.
  2. One month before the lease end date, the system sends a notification to both the tenant and the property manager.
  3. If the lease is renewed, the system updates the lease term and notifies both parties.

4.11.3 Functional Requirements

1. **REQ-71**: The system must allow property managers to generate lease agreements using predefined templates.
2. **REQ-72**: The system must support **electronic signatures** for tenants to sign lease agreements digitally.
3. **REQ-73**: Lease agreements must include essential details such as property address, rent amount, lease duration, tenant obligations, and manager obligations.
4. **REQ-74**: The system must send automatic reminders to both tenants and property managers when a lease is about to expire (e.g., 30 days before expiration).
5. **REQ-75**: Lease agreements must be stored securely, with access limited to authorized users (tenant, property manager, legal personnel).
6. **REQ-76**: The system must allow tenants and property managers to download signed lease agreements in **PDF format**.
7. **REQ-77**: The system must maintain a **version history** of lease agreements, tracking any changes made and keeping a record of all versions.
8. **REQ-78**: The system must support **lease renewal**, allowing property managers to extend a lease with updated terms without having to create a new agreement from scratch.
9. **REQ-79**: The system must allow property managers to terminate a lease early, providing the reason for termination (e.g., breach of contract, tenant vacating).

4.12 System Feature 12: Document E-Signing Integration

4.12.1 Description and Priority

The **Document E-Signing Integration** feature enables electronic signing of documents, including lease agreements, contracts, and other important legal documents, directly within the system. This feature improves the efficiency of handling legal documents and ensures that transactions are completed quickly and securely. The priority of this feature is **High**, as it facilitates legal compliance and reduces manual paperwork.

4.12.2 Stimulus/Response Sequences

- **Signing a Lease Agreement**:
    1. The property manager creates a new lease agreement and sends it to the tenant for review.
    2. The tenant receives an email with a secure link to the document.
    3. The tenant reviews the document and clicks "Sign Electronically."
    4. The system verifies the tenant's identity using **two-factor authentication (2FA)** (e.g., SMS verification code).
    5. After identity verification, the tenant signs the document, and the system stores the signed version securely.
- **Requesting Signatures from Multiple Parties**:
    1. Property manager creates a document requiring signatures from multiple parties (e.g., tenant, co-signer).
    2. The system sends secure links to each party, allowing them to sign the document sequentially.
    3. Each party receives a notification once all signatures are complete, and the signed document is made available for download.

4.12.3 Functional Requirements

1. **REQ-80**: The system must support **electronic signatures** for legally binding documents such as lease agreements, sale contracts, and maintenance agreements.
2. **REQ-81**: The system must verify the identity of signatories using **two-factor authentication (2FA)** before allowing them to sign documents.
3. **REQ-82**: The system must provide a **document audit trail**, recording when the document was sent, viewed, signed, and finalized.
4. **REQ-83**: The system must allow for **multi-party signing**, where multiple parties can sign a document sequentially or simultaneously.
5. **REQ-84**: Signed documents must be stored securely, with access restricted to authorized users.
6. **REQ-85**: The system must send automatic reminders to signatories who have not yet signed the document.
7. **REQ-86**: Signed documents must be available for download in **PDF format**, with digital signature validation included.
8. **REQ-87**: The system must support the revocation of signatures if a mistake is found before the document is finalized, allowing the document to be re-signed.

4.13 System Feature 13: Data Backup and Recovery

4.13.1 Description and Priority

The **Data Backup and Recovery** feature ensures that all system data is backed up regularly and can be restored in case of a system failure, data corruption, or disaster. The feature is crucial for maintaining data integrity and preventing data loss. The priority of this feature is **High**, as it ensures the reliability and availability of the system.

4.13.2 Stimulus/Response Sequences

- **Scheduled Data Backup**:
    1. The system automatically creates daily backups of all data, including user accounts, property listings, payments, and documents.
    2. Backups are securely stored in a cloud storage service (e.g., AWS S3).
    3. The system retains backups for a configurable period (e.g., 30 days) before older backups are deleted to save storage space.
- **Data Recovery**:
    1. In the event of a system failure or data corruption, the system administrator accesses the "Data Recovery" section of the admin dashboard.
    2. The administrator selects the most recent backup to restore.
    3. The system initiates the recovery process, restoring data from the backup.
    4. Once complete, the system sends a notification to all users informing them that the system is back online.

4.13.3 Functional Requirements

1. **REQ-88**: The system must automatically perform **daily backups** of all data, including user information, property listings, payments, documents, and logs.
2. **REQ-89**: Backup data must be stored securely using **AES-256 encryption** in a remote cloud storage service (e.g., AWS S3, Google Cloud).
3. **REQ-90**: The system must allow administrators to **restore data** from backups in the event of a system failure, data corruption, or user error.
4. **REQ-91**: The system must maintain a **backup retention policy**, automatically deleting backups older than a configurable time period (e.g., 30 days).
5. **REQ-92**: The system must support **incremental backups** to reduce storage costs by only backing up data that has changed since the last backup.
6. **REQ-93**: The system must notify administrators if a scheduled backup fails, allowing them to take corrective action.
7. **REQ-94**: The system must provide a **backup audit log**, showing when backups were created, where they were stored, and who accessed them.
8. **REQ-95**: Users must be notified if a data recovery operation is in progress, ensuring transparency during system downtimes.

5. Non-Functional Requirements

The non-functional requirements define the performance, security, usability, and operational standards that the **Real Estate Management System (REMS)** must meet. These requirements are as critical as functional requirements because they ensure the system performs efficiently, is secure, easy to use, and scalable. In this section, we provide a detailed overview of each key non-functional aspect, focusing on how it will shape the user experience, system resilience, and compliance with industry standards.

5.1 Performance Requirements

5.1.1 Response Time

Description: The system must ensure quick response times for all operations, as delays can frustrate users, especially during peak times (e.g., when many tenants make rent payments simultaneously).

Details:

- **Web Application Response**: The REMS should aim for a **95th percentile** response time of **less than 2 seconds** for standard operations such as loading property listings, managing tenants, and submitting payments. For administrative tasks such as generating reports, the response time can be up to **5 seconds** but no more.
- **Form Submissions**:
  - When tenants submit maintenance requests or payments, the system should process these submissions within **3 seconds**. If any part of the process takes longer than 3 seconds, an **"in-progress"** indicator must appear to keep users informed that their action is being processed.
- **Database Query Performance**:
  - Queries accessing tenant, payment, and property records should execute in **less than 500 milliseconds**. For complex queries such as generating year-end financial reports, the execution time should remain **below 2 seconds** for up to **100,000 records**.
- **Simultaneous User Access**:
  - The system should maintain performance standards with up to **5,000 concurrent users** performing actions such as property searches, payment submissions, and document uploads.
  - Even during **peak usage periods**, such as the beginning of a new month (when many rent payments are due), the system should not exceed a **3-second response time** for core functionality like payment processing.

5.1.2 Scalability

Description: The system must scale both horizontally and vertically to handle increased loads as the user base grows over time or during peak periods of activity (e.g., rent due dates).

Details:

- **Horizontal Scalability**:
  - The system must be designed to scale horizontally by adding more instances of web servers and database replicas to balance the load. **Load balancers** must be in place to distribute user requests evenly across servers.
- **Database Scalability**:
  - As the number of tenants, properties, and records grows, the system's database must scale to handle millions of records without significantly increasing response times. **Sharding** and **partitioning** strategies will be employed to distribute data across multiple servers if the dataset exceeds a certain threshold (e.g., 10 million records).
- **Media Handling**:
  - The system should handle **large media files** (e.g., property images, floor plans, lease documents) through an **asynchronous upload system** to prevent timeouts. Media files must be stored in a scalable storage solution like **Amazon S3** or **Google Cloud Storage**, and served through a **Content Delivery Network (CDN)** to minimize latency.

5.1.3 Throughput

Description: Throughput measures the number of transactions the system can handle per second, particularly during peak usage hours.

Details:

- **Tenant Interactions**:
  - The system should process at least **100 database write operations per second** for high-activity interactions such as rent payments, maintenance requests, or document uploads.
- **Real-Time Notifications**:
  - The notification system should handle the dispatch of **500 email or SMS notifications per minute** to ensure timely delivery. These include reminders, maintenance updates, and system alerts.
- **Payment Gateway Integration**:
  - Payment processing throughput must support **at least 50 concurrent payment transactions per second**, distributed across multiple payment gateways (e.g., Stripe, PayPal, UPI) without causing delays or system crashes.

5.2 Security Requirements

Security is of utmost importance in REMS, given the system's responsibility for managing sensitive tenant and financial data. This section defines how the system will secure user information and transactions, ensuring compliance with industry standards such as **PCI DSS** and **GDPR**.

5.2.1 Authentication and Authorization

Description: REMS must ensure that only authorized users can access specific functionalities. This will be managed through robust authentication and role-based authorization.

Details:

- **Role-Based Access Control (RBAC)**:
    - Users must be assigned specific roles (e.g., tenant, property manager, owner, admin), and permissions must be restricted based on these roles. Property managers, for instance, will have access to sensitive tenant information, while tenants will only have access to their own records.
    - The system should support **granular access control**, allowing administrators to configure custom roles and permissions as needed.
- **Authentication Protocols**:
    - **Two-Factor Authentication (2FA)**: For users with administrative access (e.g., property managers, financial officers), **2FA** should be enforced, adding an additional layer of security. This will be done using either SMS-based verification or an email verification link.
    - The system will support **OAuth 2.0** for login, enabling third-party authentication via trusted platforms like Google and Microsoft, reducing the risk of weak passwords.

5.2.2 Encryption Standards

Description: REMS must protect all sensitive data—both in transit and at rest—using the highest encryption standards.

Details:

- **Data In Transit**:
    - All communication between the client (browser or mobile app) and the server must be encrypted using **TLS 1.2** or higher. This ensures protection against man-in-the-middle attacks.
- **Data At Rest**:
    - Sensitive data such as personally identifiable information (PII) and financial records must be encrypted using **AES-256 encryption**. This applies to both the database and any backup storage.
    - **Password Storage**: User passwords must never be stored in plaintext. They must be salted and hashed using **bcrypt** or **Argon2** to prevent unauthorized access even in the event of a data breach.

5.2.3 PCI DSS Compliance

Description: As REMS handles rent payments and other financial transactions, it must adhere to **PCI DSS** (Payment Card Industry Data Security Standard) to protect cardholder data.

Details:

- **Tokenization**:
    - REMS will use a **tokenization service** (e.g., Stripe, PayPal) to handle all payment information, ensuring that no credit card data is stored directly in the system. Only tokens (i.e., encrypted representations of the payment information) will be stored and used for transactions.
- **Network Segmentation**:
    - The system's payment processing modules will be segmented from the core application to minimize security risks. Access to payment servers will be restricted, ensuring that only authorized components and personnel can interact with them.

5.2.4 Data Protection and Privacy Compliance (GDPR)

Description: REMS must comply with the **General Data Protection Regulation (GDPR)** to protect user privacy and ensure that tenants' rights are respected.

Details:

- **Right to Access**:
    - Users must be able to request a copy of all their personal data stored in the system. REMS must provide this data in a machine-readable format within **30 days** of the request.
- **Right to Erasure (Right to be Forgotten)**:
    - Users must have the option to request the deletion of their personal data. REMS must process such requests within **30 days** and notify users once the data has been permanently deleted.
- **Data Breach Notifications**:
    - In the event of a data breach, REMS must notify affected users and regulatory authorities within **72 hours**, outlining the nature of the breach, the potential consequences, and any measures taken to mitigate further risk.

5.2.5 Audit Trails and Logging

Description: A comprehensive logging and auditing system must be in place to track user actions and system changes. These logs will be essential for both operational troubleshooting and security audits.

Details:

- **User Activity Logs**:
    - All actions performed by users (e.g., logging in, updating a profile, making a payment) must be logged with a timestamp, user ID, and IP address. These logs will be stored securely and be accessible only to authorized personnel.
- **System Changes**:

    - Any changes to critical system settings (e.g., security policies, role permissions) must also be logged, with detailed information about the changes made, the responsible user, and the time of the change.
- **Log Retention**:
    - Audit logs must be retained for a minimum of **12 months**, after which they can be archived securely or deleted, depending on regulatory requirements.

5.3 Usability Requirements

Usability is a key factor for REMS, as the system will be used by a diverse audience, ranging from property managers and tenants to legal and financial professionals. The usability requirements focus on ensuring that the system is intuitive, easy to navigate, and accessible for all users.

5.3.1 User Interface Consistency and Ease of Use

Description: The system must provide a consistent and intuitive user experience across all devices, ensuring minimal training is required for new users.

Details:

- **UI Consistency**:
    - The system must maintain a consistent look and feel across all modules, ensuring that elements such as buttons, form fields, and navigation menus behave uniformly. This reduces cognitive load and makes it easier for users to navigate between different parts of the system.
- **Intuitive Layouts**:
    - The design should adhere to best practices for user interface design, such as **Material Design Guidelines**, ensuring that users can easily understand how to complete tasks (e.g., submitting a payment, uploading a document).
- **Interactive Feedback**:
    - Every user action, such as clicking a button or submitting a form, must provide real-time feedback (e.g., success notifications, error messages) to ensure that users are aware of the outcome of their actions. Delays in response must be communicated using progress indicators or loading animations.

5.3.2 Mobile Usability

Description: Given that many users will access the system via mobile devices, REMS must offer a responsive and mobile-friendly interface.

Details:

- **Mobile-Optimized Design**:
    - The system must be fully responsive, adapting to screen sizes ranging from small smartphones to large desktop monitors. Key interactions (e.g., submitting

payments, viewing property listings) must remain usable even on smaller screens.
- **Touchscreen Compatibility**:
    - Buttons and interactive elements should be large enough to be easily tapped on mobile devices, with touch-friendly navigation replacing hover-based interactions typically used on desktops.

5.3.3 Accessibility Standards

Description: The system must be accessible to users with disabilities, following international accessibility standards such as **WCAG 2.1 AA**.

Details:

- **Screen Reader Compatibility**:
    - All interactive elements (e.g., forms, buttons, dropdowns) must be compatible with screen readers. Text alternatives must be provided for images and icons, ensuring that visually impaired users can navigate the system.
- **Keyboard Navigation**:
    - The system must support full keyboard navigation, allowing users to complete tasks without using a mouse. Users should be able to navigate between form fields, submit forms, and open menus using the **Tab**, **Enter**, and **Arrow** keys.
- **Accessible Color Schemes**:
    - The system must provide high-contrast color schemes to aid users with visual impairments. Additionally, color alone must not be used as the sole means of conveying important information (e.g., error messages should be accompanied by text, not just a red highlight).

5.4 Reliability and Availability Requirements

5.4.1 Uptime and Service Availability

Description: The system must be available and operational **24/7**, with minimal downtime and a high availability guarantee. This is critical for ensuring tenant satisfaction and smooth property management operations.

Details:

- **Availability Guarantee**:
    - REMS must maintain an uptime of **99.9%**, ensuring that the system is available for users nearly all the time. This translates to no more than **43 minutes of downtime** per month.
- **Scheduled Maintenance**:
    - Maintenance windows should be scheduled during **off-peak hours**, such as late nights or weekends. Users must be notified at least **48 hours** in advance of any maintenance that could impact system availability.

5.4.2 Disaster Recovery and Failover

Description: REMS must have a comprehensive disaster recovery plan in place, ensuring minimal data loss and quick recovery in the event of a catastrophic failure.

Details:

- **Disaster Recovery Plan**:
    - The system must maintain **redundant data centers** in geographically distinct regions, ensuring that a failure in one data center does not bring the entire system down.
- **Automated Failover**:
    - In case of a primary data center failure, the system must automatically failover to a backup data center within **15 minutes**. This ensures minimal disruption to users during emergencies.
- **Data Loss Prevention**:
    - The system must ensure that no more than **15 minutes of data is lost** in the event of a catastrophic failure (i.e., **Recovery Point Objective - RPO**). All critical transactions must be logged in real-time to ensure data integrity during recovery.

5.4.3 Backup Frequency and Retention

Description: Backups must be performed regularly to ensure that data can be restored in case of data corruption, user error, or a cyber-attack.

Details:

- **Backup Frequency**:
    - The system must perform **hourly incremental backups** of tenant and property data, with **full backups**performed once a day.
- **Retention Policy**:
    - Backups must be retained for a minimum of **90 days**, after which they can be deleted or archived. Archived backups should be stored in a **long-term storage system** for compliance with financial and legal requirements.
- **Backup Security**:
    - All backup data must be encrypted before being transferred to off-site storage. Access to backups must be restricted to authorized personnel only, ensuring that backup data cannot be accessed or tampered with.

6. Other Requirements

This section outlines additional requirements and constraints that the **Real Estate Management System (REMS)** must adhere to. These include environmental constraints, legal and regulatory considerations, operational constraints, and compatibility with external systems. These requirements ensure that the system operates smoothly within its intended ecosystem and complies with relevant standards.

6.1 Environmental and Operational Constraints

6.1.1 Hosting and Infrastructure

Description: REMS will be deployed in a cloud-based environment to ensure high availability, scalability, and security. This section outlines the infrastructure-related constraints and requirements for hosting the system.

Details:

- **Cloud Provider**:
  - REMS must be hosted on a **reliable cloud infrastructure** such as **AWS**, **Google Cloud**, or **Microsoft Azure**. The infrastructure should provide auto-scaling, secure storage, and high availability.
- **Geographic Redundancy**:
  - The system must leverage multiple data centers in geographically distinct locations to ensure redundancy. This will help mitigate the risks of localized outages or natural disasters affecting system availability.
- **Server Specifications**:
  - Each server instance hosting the system should have a minimum configuration of **8 CPU cores**, **16 GB of RAM**, and **SSD storage** to handle the application load efficiently.
  - Server load must be monitored in real-time, and additional instances should be spun up during periods of high demand (e.g., the first of the month when tenants are paying rent).
- **Database Infrastructure**:
  - REMS must utilize a **NoSQL** database such as **MongoDB** for flexibility and scalability in handling large datasets (e.g., property listings, tenant information, payments). The database should support **replication** and **sharding** to ensure continuous availability and performance under load.
  - A **relational database** (e.g., **PostgreSQL** or **MySQL**) may be used for certain structured data and reporting needs, with appropriate indexing to optimize query performance.

6.1.2 Third-Party Service Integrations

Description: REMS relies on various third-party services for critical functionalities such as payment processing, document storage, and notifications. The system must ensure compatibility with these external services and adhere to their respective standards.

Details:

- **Payment Gateways**:
  - The system must integrate with popular payment gateways such as **Stripe**, **PayPal**, and **UPI** to process rent payments and security deposits. These integrations must adhere to each service's **API standards** and must be updated regularly to accommodate changes in these services' offerings or APIs.
- **Email and SMS Services**:
  - For email notifications, REMS will use services such as **SendGrid** or **Mailgun**. For SMS notifications, services like **Twilio** will be used. Both services must be integrated using **REST APIs** to send real-time notifications (e.g., rent reminders, maintenance alerts).
- **Document Storage**:
  - REMS must store large documents such as lease agreements and property images using a secure, scalable storage service like **AWS S3** or **Google Cloud Storage**. These services must support encryption of data at rest and in transit.

6.1.3 Hardware Requirements

Description: The system must be compatible with the hardware available to both tenants and property managers. These requirements define the minimum and recommended hardware configurations for running the system effectively.

Details:

- **Minimum System Requirements for Users**:
  - **Desktops**: The system should run on desktops with at least **4GB of RAM** and a **dual-core processor**. The system should support modern browsers such as **Chrome**, **Firefox**, and **Edge**.
  - **Mobile Devices**: The system should be compatible with both **Android** (version 8.0 and above) and **iOS**(version 12 and above). The mobile version must be optimized for both smartphones and tablets.
- **Internet Connectivity**:
  - A minimum **broadband connection** with speeds of at least **5 Mbps** is recommended for accessing the system. The system should still be accessible at lower speeds but may experience longer loading times when large documents or media files are being accessed.

6.2 Compliance Requirements

6.2.1 Legal Compliance

Description: REMS must comply with local, national, and international laws that govern property management, tenant rights, data protection, and financial transactions.

Details:

- **Tenant-Landlord Regulations**:
  - The system must adhere to all relevant **tenant-landlord laws** in each jurisdiction it operates. For example, local laws may dictate the legal process for **evictions**, **rent control**, and **lease agreements**. These regulations must be supported in the system's legal templates and workflows.
- **Data Privacy Laws**:
  - REMS must comply with **GDPR** (General Data Protection Regulation) in the European Union and **CCPA**(California Consumer Privacy Act) in California, as well as other local data privacy regulations. This includes allowing users to request data access, rectification, or deletion and ensuring that their data is processed in compliance with these laws.
- **Financial Compliance**:
  - The system must follow relevant financial regulations, including **PCI DSS** for handling credit card transactions and **AML (Anti-Money Laundering)** laws. It must also support regional taxation requirements, including generating appropriate **tax reports** for property managers and owners.

6.2.2 Industry Standards

Description: REMS must comply with industry best practices for software development, security, and operational resilience. These include standards for software quality, testing, and cybersecurity.

Details:

- **ISO/IEC 27001**:
  - REMS should align with the **ISO/IEC 27001** standards for **Information Security Management Systems (ISMS)**. This ensures that the system's security policies, data management processes, and user authentication protocols are following best practices for minimizing security risks.
- **ISO 9001**:
  - The system should adhere to **ISO 9001** for **Quality Management Systems (QMS)**. This standard emphasizes process efficiency, user satisfaction, and continuous improvement in product quality.
- **OWASP Top 10**:
  - All web application security should comply with the **OWASP Top 10** security risks, ensuring protection against common vulnerabilities such as **SQL Injection**, **Cross-Site Scripting (XSS)**, and **Cross-Site Request Forgery (CSRF)**.

6.3 Software Quality Attributes

6.3.1 Maintainability

Description: The system must be maintainable to allow for future updates, bug fixes, and feature enhancements without disrupting normal operations.

Details:

- **Modular Codebase**:
    - The system must be built using a **modular architecture**, where each module (e.g., payment processing, tenant management, property listings) can be developed, tested, and maintained independently. This allows developers to isolate and address bugs or add features without affecting other parts of the system.
- **Code Documentation**:
    - Detailed documentation must be provided for each module, including the **API documentation**, **data models**, and **configuration files**. All code changes must follow clear guidelines for naming conventions and code structure to ensure long-term maintainability.
- **Version Control**:
    - The system must be version-controlled using **Git** or a similar version control system. All code must be committed with detailed messages, and **branching strategies** such as **Git Flow** should be used to manage features, bug fixes, and releases.
- **Continuous Integration (CI)**:
    - The system must support **continuous integration (CI)**, with automated testing performed for every change pushed to the repository. Any failed tests should block deployment, ensuring that only stable and tested code reaches production.
- **Automated Unit Testing**:
    - Each component must have **unit tests** to validate its functionality. Test coverage must aim for **80% or higher**to ensure that the code is reliable and easily maintainable. Automated testing should run in a **CI/CD pipeline**, ensuring that all new features or patches are automatically tested before deployment.

6.3.2 Flexibility and Extensibility

Description: REMS must be flexible and extendable, allowing new features to be added as business requirements change.

Details:

- **API-First Design**:
    - The system should be designed with an **API-first** approach, allowing for easy integration with future external systems, mobile apps, or third-party tools. RESTful APIs must be well-documented, scalable, and versioned to support backward compatibility.
- **Plug-in Architecture**:

    - The system should support a **plug-in architecture** where new features (e.g., integration with a new payment gateway or document management system) can be added as separate modules without requiring a complete system overhaul.
- **Feature Toggles**:
    - **Feature toggles** (also known as **feature flags**) must be implemented to allow administrators to enable or disable new features or experimental functionality without deploying new code. This ensures that beta features can be tested with minimal risk.

6.4 Interoperability Requirements

Description: The system must be able to integrate with other systems, platforms, and third-party services, allowing seamless communication and data sharing across multiple platforms.

6.4.1 Integration with Accounting Systems

Description: REMS must support integration with external accounting systems to automate the process of financial tracking, rent collection, and tax calculation.

Details:

- **Supported Systems**:
    - The system must integrate with popular accounting systems such as **QuickBooks**, **Xero**, or **Sage**. This allows property managers to automatically sync rent payments and expenses with their accounting software.
- **Data Exchange Format**:
    - Data exchanged between REMS and external accounting systems must use standard formats such as **CSV**, **JSON**, or **XML** to ensure compatibility. APIs for exporting data must be provided so that property managers can pull financial records directly into their accounting software.
- **Reconciliation**:
    - The system must support **financial reconciliation**, allowing property managers to match incoming payments with their internal accounting records. If discrepancies are found, they must be flagged for review.

6.4.2 Integration with IoT Devices

Description: REMS should support the integration of **IoT devices** (Internet of Things) for monitoring and automating property management tasks such as security, lighting, and energy management.

Details:

- **Smart Thermostats and Sensors**:

    - Property managers should be able to integrate **smart thermostats** and **energy meters** into REMS, allowing them to monitor and optimize energy usage in real-time. Alerts should be sent when thresholds are crossed (e.g., when energy consumption exceeds a predefined limit).
- **Smart Security Systems**:
    - The system must support integration with smart locks and surveillance cameras, allowing property managers to remotely manage access to buildings and monitor live video feeds. Alerts must be generated for suspicious activity (e.g., unauthorized access to a property).
- **Data Analytics**:
    - Data from IoT devices must be stored and analyzed within REMS to provide insights on property usage, energy efficiency, and security. Reports generated from these devices should be viewable by property managers in real-time.

Appendix A: Glossary

The glossary contains definitions of key terms and acronyms used in this document, ensuring that all stakeholders have a shared understanding of the terminology.

Terms

- **API (Application Programming Interface)**: A set of tools, definitions, and protocols used for building software and allowing different applications to communicate with each other.
- **Authentication**: The process of verifying the identity of a user, typically by checking login credentials like username and password.
- **Authorization**: The process of verifying whether a user has the correct permissions to access a particular resource or perform an action.
- **CRUD**: Stands for **Create, Read, Update, Delete** – the four basic operations that can be performed on a database.
- **GDPR (General Data Protection Regulation)**: A legal framework that sets guidelines for collecting and processing personal information of individuals within the European Union (EU).
- **ISO 9001**: A global standard for quality management systems that demonstrates an organization's ability to consistently meet customer and regulatory requirements.
- **MongoDB**: A NoSQL database program, used for large-scale data storage, offering flexibility through document-based storage.
- **OAuth 2.0**: An authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Google or Facebook.
- **PCI DSS (Payment Card Industry Data Security Standard)**: A set of security standards designed to ensure that all companies that accept, process, store, or transmit credit card information maintain a secure environment.
-

Appendix B: Analysis Models

The analysis models section presents various diagrams, charts, or models that help visualize the system's structure, behavior, or workflow.

Use Case Diagram

A use case diagram represents the interaction between users (actors) and the system. Below is a simple description of the primary use cases for the **Real Estate Management System (REMS)**.

1. **Tenant Role**:
    - Search properties.
    - Submit rent payments.
    - Submit maintenance requests.
    - Review lease agreements.
2. **Property Manager Role**:
    - Manage property listings.
    - Review and respond to maintenance requests.
    - Manage tenant accounts.
    - Generate financial reports.
3. **Administrator Role**:
    - Manage user accounts.
    - Configure system settings.
    - View system logs and audit trails.

Sequence Diagrams

Sequence diagrams represent the sequence of interactions between different entities in the system over time. For example, the following could illustrate the payment process:

1. **Tenant initiates payment**.
2. **System validates payment details**.
3. **System interacts with external payment gateway (Stripe/PayPal)**.
4. **Payment confirmation is received**.
5. **Payment record is updated in the database**.
6. **Tenant receives payment confirmation**.

Appendix C: To Be Determined (TBD) List

The **TBD List** includes areas where further clarification is needed or decisions that are yet to be finalized. These items will require attention during the later phases of development.

TBD Item	Description	Status
Payment Gateway Selection	Final selection of third-party payment gateways (e.g., Stripe, PayPal)	Pending
Mobile App Platform	Whether the mobile app will be native or cross-platform (e.g., Flutter)	In Discussion
Backup Retention Policy	Final decision on how long backups will be stored before archiving	Pending
Language Support	Additional languages to be supported in future versions	TBD
Third-Party Integration API	Final API to be selected for document signing (e.g., DocuSign)	In Review
User Training Documentation	Final outline and structure of user training guides	In Progress