**Software Requirements**

**Specification**

# for

**Real Estate Management System**

**Version 1.0**

**Prepared by**

|  |  |  |
| --- | --- | --- |
| **Amr Khaled Mohammed Kira** | **202320118** |  |
| **Amr Khaled Ahmed Hassan** | **202320012** |
| **Mohammed Mohammed Abdel Moneim** | **202220267** | |
| **Ammar Yasser Mohammed** | **202220078** |  |

**Instructor: Dr. Mohammed Reda**

**Course: Software Engineering**

**Teaching Assistant: Mohammed Karam**

**Date: 13/11/2024**

**Software Requirements Specification for Real Estate Management System**

**Page ii**

## Table of Contents

### Table of Contents ............................................................................................................................ 2 Revision History .............................................................................................................................. 3

1. **Introduction ................................................................................................................................ 1** 1.1 Purpose ................................................................................................................................................ 1 1.2 Document Conventions ....................................................................................................................... 1

1.3 Intended Audience and Reading Suggestions ...................................................................................... 1 1.4 Product Scope ...................................................................................................................................... 1

1.5 References ........................................................................................................................................... 1

### 2. Overall Description..................................................................................................................... 1

2.1 Product Perspective ............................................................................................................................. 2

2.2 Product Functions ................................................................................................................................ 2 2.3 User Classes and Characteristics ......................................................................................................... 2

2.4 Operating Environment ....................................................................................................................... 2

2.5 Design and Implementation Constraints .............................................................................................. 7

2.6 User Documentation ............................................................................................................................ 7

2.7 Assumptions and Dependencies .......................................................................................................... 9

### 3. External Interface Requirements .............................................................................................. 9

3.1 User Interfaces ................................................................................................................................... 13

3.2 Hardware Interfaces ........................................................................................................................... 13

3.3 Software Interfaces ............................................................................................................................ 14

3.4 Communications Interfaces ............................................................................................................... 14

### 4. System Features ........................................................................................................................ 15

4.1 System Feature 1 ................................................................................**Error! Bookmark not defined.**

4.2 System Feature 2 (and so on) ..............................................................**Error! Bookmark not defined.**

### 5. Other Nonfunctional Requirements ........................................................................................ 16

5.1 Performance Requirements ................................................................................................................ 19 5.2 Safety Requirements .......................................................................................................................... 19

5.3 Security Requirements ....................................................................................................................... 20 5.4 Software Quality Attributes ............................................................................................................... 20

5.5 Business Rules ................................................................................................................................... 21

### 6. Other Requirements ................................................................................................................. 21 Appendix A: Glossary .................................................................................................................. 22 Appendix B: Analysis Models ...................................................................................................... 22 Appendix C: To Be Determined List ........................................................................................... 23

**Software Requirements Specification for Real Estate Management System**

**Page ii**

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Real Estate  Management  System | <13/11/2024> | First Build | 1.0 |
|  |  |  |  |

## 1. Introduction

### 1.1 Purpose

This system aims to support property owners and managers in efficiently managing residential and/or commercial property-related tasks, including leasing, tracking rental payments, tenant information management, and maintenance scheduling.

### 1.2 Document Conventions

This document follows a standard format where high-priority requirements are bolded. Priority inheritance applies; unless otherwise noted, sub-requirements adopt the priority of their parent requirement.

### 1.3 Intended Audience and Reading Suggestions

This document is intended for the following readers:

* Managers and Property Owners: for understanding system capabilities and configuration needs.
* Tenants: to review features impacting user interaction.
* System Administrators: to understand maintenance, access control, and deployment.
* Business Stakeholders: for assessing project alignment with business goals.

### 1.4 Product Scope

This system facilitates comprehensive real estate management, enabling property owners and managers to handle rental and purchase-related tasks, such as unit leasing, rental payment tracking, tenant data management, and maintenance scheduling. By offering these features, the system aims to streamline operations, enhance tenant satisfaction, and improve property oversight.

**1.5 References** • Zillow (https://www.zillow.com/)

Provides insights into rental and property management features for a large-scale property platform, used as a benchmark for functionality and user experience.

* YouTube Video on Property Management Systems

(https://www.youtube.com/watch?v=9Jz689rY7uo)

* 1. detailed video explaining property management systems, with a focus on essential features and common challenges faced by property managers.

* Safe and Sound Security - Building Management System Blog

(https://getsafeandsound.com/blog/building-management-system/)

This blog article offers an overview of building management systems and their components, helping to guide the inclusion of features like maintenance and tenant communications.

* IEEE 830-1998 Software Requirements Specification Standard
  1. standard for creating Software Requirements Specifications, serving as a guideline for document structure, content, and level of detail in this SRS.2. Overall Description

## 2. Overall Description

### 2.1 Product Perspective

The property management software serves as an independent platform aimed at centralizing and automating property management. It will also allow integration with third-party systems (such as payment processors and accounting software) to extend its functionality.

**2.2 Product Functions** Product Functions:

* Property Listings and Management: Enables managers to create or update property listings by providing photos, descriptions, and amenities.
* Tenant and Lease Management: Maintains tenant details, lease start and end dates, and automates the lease renewal process.
* Payment Processing: Offers a secure way for tenants to make rent payments, generates receipts automatically, and allows tenants to view payment history.
* Maintenance and Service Requests: Allows tenants and managers to submit maintenance requests, track their status, and manage responses.
* Financial Management: Manages income and expense tracking, generates financial reports, and supports integration with accounting systems.

### 2.3 User Classes and Characteristics

The system will cater to different user groups, each with specific responsibilities, experience levels, and system access requirements. Below are the primary user classes:

1. Property Managers o Description: Primary users who oversee day-to-day property operations, including tenant management, rent collection, and responding to maintenance requests. o Responsibilities: o Managing tenant applications, creating and updating lease agreements, and processing renewals.
   * Overseeing financial aspects such as rent collection and expense tracking. o Assigning and tracking maintenance requests and related tasks. o Generating financial and operational reports for property owners. Characteristics:
   * Experience Level: Intermediate to Advanced, usually with basic knowledge of property management processes and computer skills. o Needs: An intuitive interface for managing properties, tenant information, and financial details. Quick access to dashboards and reporting tools.
   * Access Requirements: Full access to all system functions, including financial records, tenant and lease management, maintenance tracking, and reporting capabilities.
2. Property Owners o Description: Individuals or entities who own the properties, monitor overall performance, and ensure profitable investments.

o Responsibilities:

* Monitoring property performance metrics, such as occupancy rates and profitability.
* Reviewing and analyzing financial reports to assess ROI.
* Approving major property-related expenses or upgrades.

Characteristics:

* Experience Level: Varies from Basic to Advanced; property owners may not have detailed technical knowledge of the software.
* Needs: High-level overview reports, financial insights, and property performance data, presented in a simplified, user-friendly format.
* Access Requirements: Limited, primarily view-only access to reports, financial data, and property performance metrics. Some permissions for oversight and approval actions may be enabled.

1. Tenants o Description: Individuals renting or leasing properties. They interact with the system for lease management, payments, and maintenance requests.
   * Responsibilities:
     + Viewing lease details and important documents.
     + Making rent payments and accessing payment history.
     + Submitting maintenance requests and tracking their status.

Characteristics: o Experience Level: Basic, accustomed to straightforward, user-friendly online platforms. o Needs: A self-service portal for managing payments, viewing lease details, and submitting maintenance requests with minimal navigation. o Access Requirements: Limited to their own account, allowing only personal information access, payment history, and maintenance request submission.

1. Maintenance Staff o Description: Internal or contracted maintenance personnel responsible for addressing tenant maintenance and repair requests.
   * Responsibilities: o Viewing assigned maintenance requests and updating their status.
   * Communicating with property managers on task progress and completions. Characteristics:
   * Experience Level: Basic to Intermediate, generally focused on task-specific operations rather than system-wide functions.
   * Needs: A task-oriented interface to view and update maintenance requests efficiently, without needing to navigate other parts of the system. o Access Requirements: Restricted access to maintenance task details only, with permissions limited to updating request statuses and adding comments.
2. Financial Managers o Description: Financial professionals who manage records, transactions, and reporting for the property management company.
   * + Responsibilities:
     + Tracking income and expenses for each property. o Preparing and presenting financial statements and reports.
     + Ensuring accurate and timely rent collection processes.

Characteristics: o Experience Level: Advanced, with proficiency in financial management, accounting, and possibly familiarity with financial software.

* + - Needs: Detailed financial reporting, easy access to transaction histories, and data export capabilities to integrate with accounting software. o Access Requirements: Access to financial data, reporting tools, and tenant payment records for accurate bookkeeping, along with capabilities for exporting data.

1. System Administrators o Description: Technical staff responsible for system setup, maintenance, data security, and troubleshooting.
   * + Responsibilities: o Managing user roles, permissions, and ensuring data security.
     + Performing system maintenance, updates, and managing software backups.

Characteristics: o Experience Level: Advanced technical skills with knowledge of system administration, security protocols, and database management. o Needs: Full control over system settings, access control, user management, and security measures to ensure operational stability. o Access Requirements: Complete access to system configurations, user management, security settings, and database operations.

### 2.4 Operating Environment

Software Requirements Server-Side Software:

Operating System:

* Linux: Ubuntu 20.04 LTS or later, CentOS 7 or later.
* Windows: Windows Server 2016 or later.

Web Server:

* Apache: Version 2.4 or later, supporting high availability and secure HTTPS protocols.
* Nginx: Version 1.18 or later, suitable for handling high traffic volumes.

Database:

* MySQL: Version 8.0 or later, recommended for relational database needs and complex queries.
* MongoDB: Version 4.0 or later, for NoSQL flexibility in managing unstructured tenant and property data.
* PostgreSQL: Version 12.0 or later, for high-performance and advanced relational database features.

Programming Language and Framework:

* Backend: Node.js (14+), Python (3.8+), or PHP (7.4+) compatible with server frameworks like Express.js, Django, or Laravel.
* Frontend Framework: React (17+) or Angular (11+) for responsive and interactive user interfaces.

APIs and Middleware:

* RESTful API supports integration with third-party payment gateways, notification services, and accounting systems.
* Optional GraphQL layer for efficient querying, especially for mobile apps and complex client requests.

Security Frameworks and Tools: • SSL/TLS certificates for secure data transmission.

* OAuth2 for authentication and secure access to user accounts and sensitive data. • Firewalls and intrusion detection systems to protect against unauthorized access.

Client-Side Software:

Supported Browsers:

* The latest versions of Chrome, Firefox, Safari, and Edge are recommended for compatibility and security.
* JavaScript and cookies must be enabled in the browser.

Mobile App Compatibility:

* Android: Version 10 (API level 29) or later, supporting either native or hybrid development frameworks.
* iOS: Version 13.0 or later, with Swift or React Native for cross-platform compatibility.

Hardware Requirements

Server Requirements:

* Processor: Minimum quad-core CPU; recommended 2.5 GHz or higher for optimal performance, especially for handling multiple concurrent user sessions.
* RAM: 8 GB minimum, with 16 GB or higher recommended for systems supporting large user bases and high transaction volumes.
* Storage: Minimum 500 GB SSD, expandable depending on data retention policies, to ensure fast data access and storage for high-traffic environments.
* Network: Reliable, high-speed internet connection with a minimum of 1 Gbps network speed to support uninterrupted access and cloud services.

Client Devices:

Desktops/Laptops:

* Operating Systems: Compatible with Windows 10/11, macOS Catalina (10.15) or later, and popular Linux distributions (e.g., Ubuntu, Fedora).
* Processor: Minimum dual-core processor, ideally 2.0 GHz or higher.
* RAM: Minimum 4 GB, recommended 8 GB for multitasking and using multiple features simultaneously.
* Storage: 100 MB of free space for browser-based access, additional space if installing client-side software.

Mobile Devices:

* Android: Minimum 2 GB RAM, running Android 10 or later.
* iOS: Minimum 2 GB RAM, running iOS 13 or later.
* Additional Requirements: Device should support biometric authentication if used for secure login features.

Network and Connectivity Requirements

* Bandwidth: Minimum 1 Mbps per active user is recommended for smooth system operation, with higher bandwidth recommended for high traffic.
* Latency: Low-latency connection preferred to ensure minimal delays, especially for live data operations and real-time maintenance or payment tracking.

### 2.5 Design and Implementation Constraints

The design and implementation of the Property Management System are subject to several constraints due to corporate policies, regulatory requirements, technical limitations, and external dependencies.

1. Regulatory and Compliance Constraints

Data Privacy: The system must comply with data privacy regulations, such as GDPR (for any users in the European Union) and CCPA (for users in California), ensuring that tenant and financial data are stored, processed, and accessed securely.

Data Retention Policies: Compliance with national and international data retention laws will impact how long user data (tenant information, payment history, maintenance records) is stored and maintained.

Financial Regulations: Payment processing must adhere to PCI DSS standards for handling sensitive financial data to prevent fraud and ensure secure payment transactions.

1. Security Constraints

Authentication and Authorization: The system must implement robust authentication (e.g., OAuth2, biometric support on mobile) and role-based authorization, limiting data access based on user roles (e.g., tenant, manager, financial administrator).

Data Encryption: Sensitive information, including payment data and personal details, must be encrypted both in transit (using SSL/TLS) and at rest, particularly when stored in databases or cloud storage.

1. Hardware Limitations

Server Specifications: Server requirements, such as minimum quad-core processors and 8 GB RAM, dictate system performance and limit the number of concurrent users and database query complexity.

Mobile Device Compatibility: To ensure compatibility with older devices, the mobile app must be optimized to run on Android 10+ and iOS 13+ while considering limited resources like RAM and CPU on entry-level devices.

1. Software and Technology Constraints

Database Selection: The choice between MySQL, PostgreSQL, or MongoDB may impose constraints on data structure and scalability. Relational databases require schema consistency, while NoSQL databases offer more flexibility but might limit complex joins or transactions.

Frontend Framework: The choice of frontend frameworks, such as React or Angular, might restrict how certain features (e.g., real-time updates) are implemented, impacting the application's speed and responsiveness on slower networks.

Payment Gateway Integration: Payment processing must support third-party APIs, and any selected gateway must offer secure, seamless integration, potentially limiting choices to well-established providers like Stripe or PayPal for PCI compliance.

1. Corporate or Policy Constraints

Development Standards: The development team must adhere to internal standards, such as IEEE 830 for software requirements specifications, and follow best practices for code documentation, readability, and version control (e.g., using Git).

Maintenance and Support: The system must be designed with maintainability in mind, ensuring that future updates, bug fixes, and new features can be implemented without significant rework.

Vendor and Licensing Constraints: Certain libraries or tools may have licensing restrictions that limit commercial use. Open-source frameworks (MIT, GPL licenses) should be prioritized unless explicitly approved for purchase by the corporate budget.

1. User Interface and Accessibility Constraints

User Experience (UX): The design must account for all user classes (e.g., tenants, managers, and maintenance staff) and their varying technical skill levels. Simplicity and accessibility are priorities, with features like intuitive dashboards, easy navigation, and clear error messages.

Accessibility Standards: The UI must be compliant with WCAG 2.1 to ensure accessibility for users with disabilities, affecting design decisions, such as color contrast, font size, and alternative text for images.

1. Development Tool Constraints

Development Environment: The development team is required to use specified tools (e.g., Visual Studio Code, GitHub/GitLab for version control) to ensure consistency, collaboration, and adherence to company standards.

Testing Frameworks: Testing constraints require the use of specific testing frameworks

(e.g., JUnit for Java or Jest for JavaScript) for unit and integration testing, which may limit the choice of backend language or frameworks.

1. Network and Deployment Constraints

Bandwidth Limitations: Users in remote areas may have limited or inconsistent internet connectivity, requiring the system to be optimized for low bandwidth usage and support offline functionality where possible.

Scalability and Load Balancing: The design must accommodate the possibility of scaling up, which may require additional server resources, load balancers, or cloud-based infrastructure to ensure system performance during peak usage.

1. Dependency Constraints

Third-Party APIs: The system relies on external services for certain functionalities (e.g., payment gateways, SMS/email notifications, mapping services). These dependencies may impose constraints on uptime, transaction fees, and service limitations.

Scheduled Maintenance Windows: The PMS must accommodate planned maintenance by third-party services or infrastructure providers, which could temporarily impact system availability or limit features (e.g., payment processing).

### 2.6 User Documentation

User documentation will include a comprehensive user manual, quick-start guide, and in-app tooltips. Online tutorials and a searchable knowledge base will also be available. Documentation will be provided in both PDF and HTML formats.

### 2.7 Assumptions and Dependencies

The development and deployment of the Property Management System (PMS) are based on specific assumptions and dependencies. Changes to these factors may affect the project's feasibility, timeline, and overall functionality.

1. Assumptions

User Accessibility:

* + It is assumed that users (property managers, tenants, maintenance staff) have regular access to stable internet connections to fully utilize the system. For areas with limited connectivity, the system’s mobile app will provide offline access for some features, with data syncing once reconnected.
  + Assumes that all users, regardless of role, have basic familiarity with using online platforms, especially on mobile or desktop devices.

Hardware Availability:

* + Assumes that property managers and system administrators have access to compatible devices, such as desktops or laptops, with up-to-date operating systems (e.g., Windows 10 or macOS Catalina and above) for using the administrative functions of the system.

Mobile devices used by tenants and maintenance staff will meet the minimum hardware requirements, including Android 10+ or iOS 13+, and at least 2GB of RAM.

User Authentication and Roles:

* + Assumes that all users will comply with security practices, such as creating secure passwords and enabling two-factor authentication where available.
  + It is assumed that the role-based access control will be sufficient for user management, with roles assigned at the time of registration and reviewed periodically.

Data Volume and Scalability:

* + The system is assumed to support a moderate number of properties, tenants, and transactions per month. Future scalability needs, such as a large increase in users, will be handled by additional server resources or database optimizations if necessary.
  + Assumes that data storage requirements will be manageable within the initial storage limits and will scale with minimal impact on performance as usage grows.

Legal Compliance:

* + Assumes that the software will be compliant with current regulatory requirements, such as GDPR and CCPA, and that any changes to these regulations will be communicated to the development team to ensure compliance.
  + It is assumed that property managers and owners will handle any additional regional legal compliance needs for their specific areas, especially regarding tenant agreements and financial reporting.

Regular Maintenance and Updates:

* + Assumes that the system will undergo periodic maintenance and software updates to address security vulnerabilities and add new features.

Assumes the client or system administrators will perform these updates according to a pre-established schedule to minimize disruptions.

1. Dependencies

Third-Party Payment Gateway:

* + - The system’s payment processing relies on integration with a third-party payment gateway (e.g., Stripe, PayPal), making it dependent on the availability and performance of this service. Any downtime or API changes in the payment gateway could impact rent payments and other financial transactions.
    - Compliance with PCI DSS standards by the chosen payment gateway is necessary to maintain secure financial transactions.

Mapping and Geolocation Services:

* + - Property location and mapping features depend on external geolocation services, such as Google Maps or OpenStreetMap APIs, for accurate property address validation and display. Any interruptions or changes to these APIs could affect the mapping functionalities within the PMS.

Email and SMS Notification Services:

* + - The system’s notification features rely on third-party email and SMS services for sending lease notifications, payment reminders, and maintenance updates.

Dependencies include services such as Twilio for SMS and SendGrid for email notifications.

* + - Any downtime or cost changes in these services could impact communication between property managers and tenants.

Database Systems:

The PMS is dependent on the chosen database system (e.g., MySQL, PostgreSQL, or MongoDB) for data storage and retrieval. Any issues with database performance, maintenance, or scaling could impact the responsiveness and reliability of the system.

* + - Backup and recovery solutions rely on the database’s built-in mechanisms or external storage solutions, creating a dependency for data integrity and recovery in case of data loss or corruption.

Operating System and Browser Support:

* + - System performance and compatibility depend on certain operating systems (e.g.,

Linux for servers, Windows for clients) and modern web browsers (Chrome, Firefox, Safari, Edge). Any changes in support for these platforms could require updates or adjustments to the system’s codebase to maintain compatibility.

Server Hosting and Cloud Providers:

* + - The PMS relies on cloud or hosting providers (such as AWS, Azure, or Google Cloud) for data storage, processing, and service availability. The stability, cost, and security measures provided by these platforms directly impact the system’s performance and uptime.

Any changes to service level agreements, pricing, or features from these providers could affect system scalability, data security, and operational costs.

Legal and Regulatory Updates:

* + - Compliance with privacy and financial regulations depends on third-party legal services or internal compliance teams to monitor and interpret regulatory updates. Any change in data protection laws or financial compliance requirements could necessitate updates to the system’s data handling and reporting features.

Software Frameworks and Libraries:

The PMS depends on specific software frameworks and libraries (e.g., React for frontend development, Node.js for backend services) to provide functionality. Any security vulnerabilities or deprecations in these libraries would require timely updates.

* + - Changes in open-source licenses for libraries and frameworks may impose restrictions or additional costs, potentially impacting the overall development and maintenance strategy.

## 3. External Interface Requirements

### 3.1 User Interfaces

### *General Overview:*

### *The Property Management System (PMS) offers a user-friendly, intuitive interface for property managers, tenants, and maintenance staff. The interface mirrors Zillow’s design principles to ensure a modern, seamless, and visually appealing experience across all devices.*

### *Screen Layout:*

### *Search Bar:*

### *Prominently displayed at the top of the homepage, allowing users to search by address, city, ZIP code, or keywords.*

### *Interactive Map:*

### *A dynamic map on the right side of the screen updates in real-time as users browse listings, offering geographical context for the properties.*

### *Filters:*

### *A comprehensive set of filtering options lets users refine their search by price, number of bedrooms/bathrooms, property type, square footage, year built, and more.*

### *Users can also filter by features like swimming pools, parking, or pet-friendly properties.*

### *Listings Feed:*

### *The left side of the page shows a scrollable list of properties that match the search criteria.*

### *Each listing includes a thumbnail image, price, address, number of beds/baths, and a brief description.*

### *Property Details:*

### *Clicking on a listing opens a dedicated page featuring:*

### *High-quality images and virtual tours.*

### *Pricing history and tax data.*

### *Property descriptions and key features.*

### *School and neighborhood information.*

### *Mortgage calculator.*

### *Contact forms to connect with agents.*

### *User Profiles:*

### *Personalized accounts allow users to save favorite properties, receive alerts on price changes, and manage their searches.*

### *Navigation Bar:*

### *The top menu provides quick access to options like “Dashboard,” “Properties,” “Tenants,” “Maintenance,” “Financials,” and “Settings.”*

### *Color Scheme and Branding:*

### *Professional and visually consistent:*

### *Blue for primary actions and highlights.*

### *Gray for disabled or secondary elements.*

### *High-contrast text and icons to ensure accessibility.*

### *Consistent branding across tooltips, buttons, and notifications.*

### *Responsive Design:*

### *Optimized for mobile, tablet, and desktop views.*

### *The layout adapts dynamically, with the sidebar collapsing and content restructured for smaller screens.*

### *Interactive map and grids maintain usability across devices.*

### *Interactive Tools:*

### *Features such as “Draw Your Own Search” allow users to outline custom areas on the map for tailored property searches.*

### *Insights and Recommendations:*

### *AI-driven suggestions based on user preferences and past interactions help users discover properties they might have overlooked.*

### *Keyboard Shortcuts:*

### *Common actions feature keyboard shortcuts for improved efficiency:*

### *Ctrl+S: Save changes.*

### *Esc: Cancel or close dialogs.*

### *Error Messages and Validation:*

### *Error Messages:*

### *Displayed at the top of forms or near relevant fields in a consistent format.*

### *Color-coded for clarity: red for errors, yellow for warnings.*

### *Include brief descriptions and actionable suggestions.*

### *Validation Messages:*

### *Appear as tooltips near invalid fields.*

### *Highlight missing or incorrect inputs clearly.*

### *Sample Screens:*

### *Login Screen:*

### *Fields for username and password with options for “Forgot Password” and “Sign Up.”*

### *Modern, minimalist design.*

### *Dashboard:*

### *Overview of active leases, pending maintenance requests, and recent notifications.*

### *Interactive map showing property locations and statuses.*

### *Property Management:*

### *Allows adding or editing properties with detailed views for each property.*

### *Tenant information, maintenance assignments, and lease tracking displayed in an intuitive layout.*

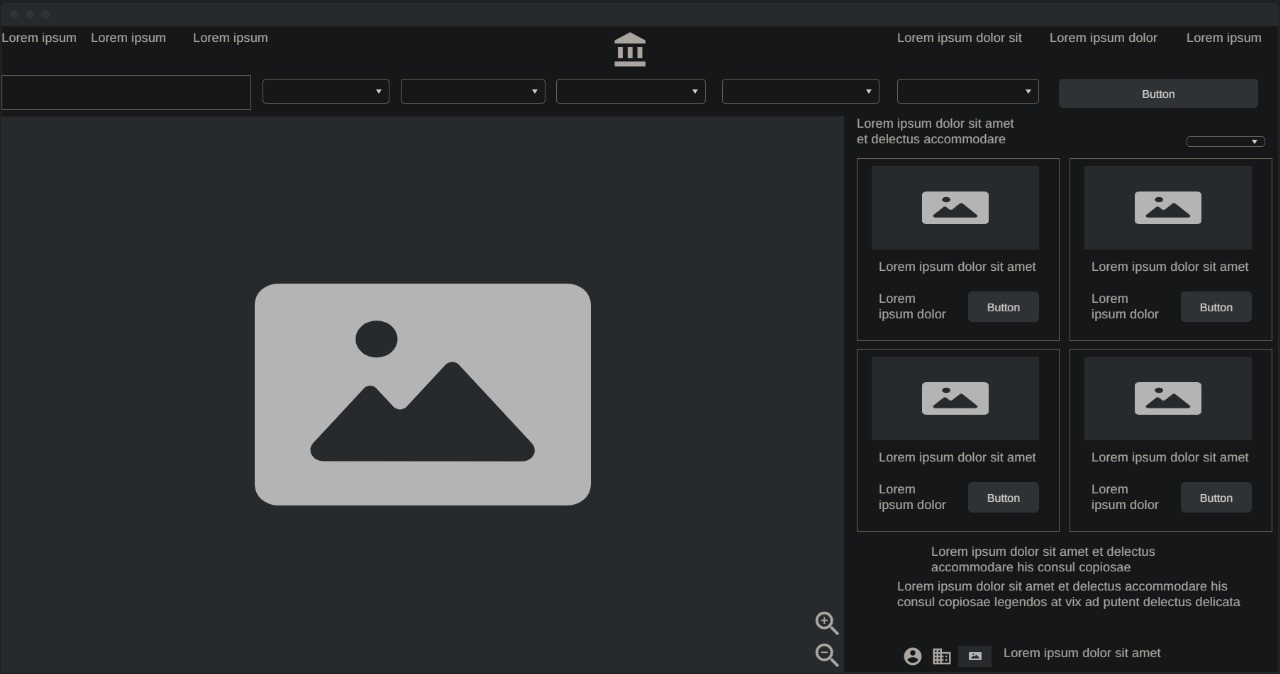
### *Financial Transactions:*

### *Displays payment history, invoice generation, and payment processing tools.*

### *Graphical reports and charts for financial analysis.*

A screenshot of a black screen

Description automatically generated



A screenshot of a phone

Description automatically generated

A screenshot of a phone

Description automatically generated

### 3.2 Hardware Interfaces

The PMS will interface with various hardware components to support both desktop and mobile platforms. Key interfaces include:

Device Compatibility:

* Desktop and laptop computers with modern operating systems, including Windows 10+, macOS Catalina+, and Linux distributions that support web applications.
* Mobile devices running Android 10+ and iOS 13+ for the mobile app.

Peripheral Devices:

* Printer Support: Users can print lease agreements, financial reports, and property details. The system will interface with standard printing protocols (e.g., PCL, PostScript) compatible with networked and local printers.
* Scanner Support: The system will support uploading scanned documents and images for lease agreements, identification, and property photos.

Hardware Requirements:

* The PMS will operate on devices with minimum hardware specifications, such as 4GB RAM and 1.5 GHz processor, to ensure acceptable performance.
* The system is optimized for touchscreen devices, providing a responsive layout and larger touch targets for mobile devices.

### 3.3 Software Interfaces

The PMS relies on several software components to support its functionality. These interfaces include:

Operating System Compatibility:

▪ The system will be compatible with Windows, macOS, and Linux on desktop environments and with Android and iOS on mobile devices.

Database Management:

* The system will use a relational database (e.g., MySQL or PostgreSQL) to store user information, property data, and transaction records. APIs will facilitate interactions between the application and the database, allowing CRUD (Create, Read, Update, Delete) operations.

External APIs:

* Payment Gateway (e.g., Stripe or PayPal): Handles all transactions, such as rent payments. The PMS will interact with this API for initiating and confirming payments.
* Geolocation and Mapping Services (e.g., Google Maps API): Used to locate properties and display them on maps.
* Notification Services:
  + Email (e.g., SendGrid API): Used for automated email notifications like payment receipts and maintenance alerts.
  + SMS (e.g., Twilio API): Used for SMS notifications to tenants and maintenance staff.

Shared Data:

* The PMS will exchange data across its components, such as user profiles, lease agreements, payment records, and maintenance logs. Shared data will be stored in the central database and accessed through APIs as needed.

### 3.4 Communications Interfaces

The PMS requires multiple communication channels to facilitate user interactions and ensure efficient data exchange.

Web Browser Requirements:

The system will be compatible with recent versions of major browsers (Chrome, Firefox, Safari, Edge), ensuring full functionality and optimal performance.

Network Protocols:

HTTP/HTTPS: The PMS will use HTTPS for secure data transmission, ensuring that user credentials, personal data, and financial information are encrypted.

WebSocket: For real-time notifications (e.g., updates on maintenance requests), the PMS may use WebSocket connections to keep clients informed.

Email and SMS Communication:

The system will send notifications and updates to users via email and SMS. Email will follow standard SMTP protocols, while SMS communication will utilize REST APIs (e.g., Twilio).

Message Formatting: Notifications will include structured templates with personalization for each user, ensuring clarity and relevance.

Encryption and Security:

Sensitive communications will be encrypted using SSL/TLS protocols, especially during payment transactions and login sessions, to prevent unauthorized access to data. Passwords and sensitive data will be hashed and encrypted to enhance security.

Data Transfer and Synchronization:

The mobile app will periodically synchronize with the main server to ensure data consistency. Data transfers between the client and server will follow secure protocols to maintain data integrity.

Data transfer rates will be optimized to accommodate variable network speeds, with priority given to critical data during synchronization.

#### **4. System Features**

4.1 Lease Management

4.1.1 Description and Priority

* Description: This feature allows property managers to create, update, view, and terminate lease agreements for tenants. It includes tracking lease terms, payment schedules, and tenant details.
* Priority: High
* Benefit: 8 - Crucial for primary functionality.
* Penalty: 7 - Without this feature, the PMS is ineffective in managing leases.
* Cost: 6 - Moderate implementation cost.
* Risk: 4 - Minimal risk if standard database functions are used.

4.1.2 Stimulus/Response Sequences

* Action: Property manager initiates “Create Lease” action.
* Response: System displays lease form fields (tenant name, property details, lease duration, etc.).
* Action: Manager submits completed lease details.
* Response: System validates input, creates lease record, and generates a lease ID.
* Action: Manager views all active leases.
* Response: System displays a list of active leases with options to view or edit details.
* Action: Manager updates lease details (e.g., rent amount).
* Response: System saves updates and confirms the successful update.

4.1.3 Functional Requirements

* REQ-1: The system shall allow managers to create new lease agreements by entering tenant information, property ID, lease start date, and end date.
* REQ-2: The system shall display an error if any required fields are left blank upon submission.
* REQ-3: The system shall save lease data and generate a unique lease ID for each record.
* REQ-4: The system shall allow managers to update lease details, such as rent amount or end date, and confirm successful updates.
* REQ-5: The system shall prevent deletion of active leases unless they are expired or terminated.

4.2 Rent Collection

4.2.1 Description and Priority

* Description: This feature enables the collection and tracking of rent payments for each tenant. Rent payments are logged, and receipts are generated. Priority: High
* Benefit: 9 - Essential for tracking income.
* Penalty: 8 - Without this, the PMS cannot manage financials effectively.
* Cost: 5 - Moderate due to integration with payment APIs.
* Risk: 5 - Potential integration issues with payment APIs.

4.2.2 Stimulus/Response Sequences

* Action: Tenant initiates rent payment.
* Response: System redirects to the payment gateway.
* Action: Tenant completes payment via external payment processor.
* Response: System receives payment confirmation and updates tenant’s rent status.
* Action: Property manager views payment history.
* Response: System displays payment logs and tenant balances.

4.2.3 Functional Requirements

* REQ-1: The system shall provide a payment interface that integrates with external payment gateways.
* REQ-2: The system shall update a tenant’s payment status to “Paid” upon successful transaction.
* REQ-3: The system shall generate a receipt for each payment and make it available to both the manager and tenant.
* REQ-4: The system shall flag overdue payments and notify the manager.
* REQ-5: The system shall allow managers to manually adjust payment records in case of corrections.

4.3 Maintenance Request Management

4.3.1 Description and Priority

* Description: Tenants can submit maintenance requests for property issues, which are tracked until resolved. Priority: Medium
* Benefit: 7 - Enhances tenant satisfaction.
* Penalty: 6 - Some inconvenience if missing, but not critical.
* Cost: 4 - Low cost to implement.
* Risk: 3 - Minimal risk as basic CRUD functions are required.

4.3.2 Stimulus/Response Sequences

* Action: Tenant submits a maintenance request.
* Response: System logs the request and assigns a unique request ID.
* Action: Property manager views all open maintenance requests.
* Response: System displays list of requests with options to assign and update statuses.
* Action: Manager assigns a request to a maintenance staff member.
* Response: System updates request status and notifies assigned staff.

4.3.3 Functional Requirements

* REQ-1: The system shall allow tenants to submit maintenance requests with details and optional photos.
* REQ-2: The system shall assign a unique ID to each request and notify the property manager.
* REQ-3: The system shall allow the property manager to assign requests to staff and track the status (e.g., “Open,” “In Progress,” “Resolved”).
* REQ-4: The system shall provide tenants with status updates on their maintenance requests.
* REQ-5: The system shall archive closed maintenance requests and allow them to be reviewed if needed.

4.4 Reporting and Analytics

4.4.1 Description and Priority

* Description: This feature provides property managers with analytical insights into finances, occupancy rates, and maintenance trends. Priority: Medium
* Benefit: 6 - Useful for operational insights and decision-making.
* Penalty: 5 - Lack of insight could reduce efficiency.
* Cost: 6 - Moderate due to data processing requirements.
* Risk: 5 - Potential data accuracy issues if not properly configured.

4.4.2 Stimulus/Response Sequences

* Action: Property manager requests a financial report.
* Response: System generates and displays report on rent payments, overdue balances, and income.
* Action: Manager views occupancy analytics.
* Response: System displays historical and current occupancy rates, including lease expirations.
* Action: Manager requests maintenance report.
* Response: System generates report on frequency and types of maintenance requests.

4.4.3 Functional Requirements

* REQ-1: The system shall generate financial reports, including total collected rent, outstanding balances, and income trends.
* REQ-2: The system shall provide occupancy analytics, showing current occupancy rate and upcoming lease expirations.
* REQ-3: The system shall generate maintenance reports summarizing the frequency and type of requests over time.
* REQ-4: The system shall allow managers to export reports in CSV and PDF formats.
* REQ-5: The system shall store historical reports for up to five years, accessible by authorized users.

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

Response Time: The system should respond quickly to user actions, such as loading pages, processing transactions, and generating reports. Response times should remain under 3 seconds for common operations and under 10 seconds for complex reports, even as the number of users or the volume of stored data increases.

Scalability: The system must support an increasing number of users and a growing database without significant degradation in performance. It should scale horizontally or vertically depending on the system architecture to maintain performance as the data size grows.

Concurrency: The system should handle multiple simultaneous users without performance loss, particularly for key functionalities like lease management, rent payment processing, and maintenance requests.

### 5.2 Safety Requirements

Data Protection: Tenant and property owner data must be protected from unauthorized access, tampering, or loss. The system should employ strong encryption for sensitive data both in transit and at rest.

Access Control: The system must implement role-based access control (RBAC), ensuring that only authorized personnel can access sensitive data (e.g., financial information, tenant details). Access levels should include admin, property managers, and tenants with appropriate restrictions.

Data Backup: The system should include an automatic backup mechanism to prevent data loss. Backups should be performed at regular intervals (e.g., daily, weekly) and stored securely in a separate location.

Audit Trails: The system should maintain an audit log of critical actions, such as creating, updating, or deleting sensitive data (e.g., leases, payments), to track and review changes for security and compliance purposes.

#### 5.3 Security Requirements

User Authentication: The system must authenticate users before granting access. Authentication should support secure login methods such as username/password, with the option for multi-factor authentication (MFA) to enhance security.

Encryption: All communication between the user’s device and the server should be encrypted using SSL/TLS protocols to prevent unauthorized interception of sensitive data.

Data Privacy: Tenant and owner data must comply with local data privacy laws (such as GDPR, CCPA, etc.). Sensitive data, such as Social Security numbers, should be masked or encrypted.

Security Vulnerability Management: The system must be regularly tested for security vulnerabilities, and patches should be applied promptly to protect against known threats (e.g., SQL injection, cross-site scripting).

### 5.4 Software Quality Attributes

Compatibility

* Device Compatibility: The system should be compatible with desktop computers, mobile phones, and tablets, ensuring a consistent experience across devices.
* Operating System Compatibility: The system should support a range of operating systems, including Windows, macOS, Linux for desktop versions, and iOS and Android for mobile apps.
* Browser Compatibility: The web version of the system must support major web browsers, including Chrome, Firefox, Safari, and Microsoft Edge, with the latest two versions of each being fully supported.

Maintainability

* Code Quality: The system’s codebase should follow industry standards and best practices for readability and modularity to simplify maintenance and future upgrades.
* Error Logging and Reporting: The system must include comprehensive logging and error reporting mechanisms. Any system failures or unexpected behaviors should be logged with sufficient details to aid debugging and resolution.
* Documentation: Comprehensive documentation should be provided for both the technical setup and user usage of the system. This will help reduce downtime and facilitate easier maintenance by support teams.

Usability

* User Interface (UI): The system should have a simple, intuitive user interface that requires minimal training. UI elements should be consistent throughout the system and adhere to common usability principles.
* User Experience (UX): The system should ensure a smooth and efficient experience for different user types (e.g., property managers, tenants). This includes fast load times, clear navigation, and easily accessible features.
* Accessibility: The system should be designed to meet accessibility standards (such as WCAG 2.1) to ensure that users with disabilities can access and use the platform effectively.

### 5.5 Business Rules

Legal Compliance for Real Estate Transactions:

* The software must comply with local rules and regulations related to the selling, buying, and renting of real estate. This includes ensuring that lease agreements are legally valid, that rental payments and fees are correctly calculated based on local laws, and that tenant rights are respected.

Tax Compliance:

* The system must adhere to local and international tax laws concerning property transactions and rental income. This includes generating reports that comply with tax authorities’ reporting standards, automating tax calculations (such as VAT or sales tax), and maintaining records of taxes paid for reporting purposes.

## 6. Other Requirements

### *6.1 Database Requirements*

* Database Type: The system will use a relational database management system (RDBMS) such as MySQL, PostgreSQL, or Microsoft SQL Server for storing tenant, property, and financial data. The choice of database should ensure ACID compliance and support for complex queries.
* Data Integrity: The system must implement data integrity constraints such as foreign key relationships, unique constraints, and checks to maintain the consistency of data across different tables.
* Database Backup and Recovery: The system must support automated database backups and provide an easy mechanism for restoring backups in the event of a system failure.
* Data Storage Capacity: The database must be able to scale and handle an increasing number of properties, tenants, and transactions without performance degradation.

### *6.2 Internationalization and Localization Requirements*

* Multi-language Support: The system should be capable of supporting multiple languages, particularly those common in the market it serves. English and Arabic should be supported, with the possibility of adding more languages as needed.
* Currency and Date Formats: The system should accommodate multiple currencies and regional date formats. For example, the system should allow the display of rental prices in

the local currency and use region-specific date formats (e.g., MM/DD/YYYY or DD/MM/YYYY).

* Time Zone Support: The system should handle different time zones for tenants and property managers. It should store dates and times in UTC format and convert them to local time zones as needed.

### *6.3 Legal Requirements*

* Compliance with Real Estate Laws: The system must comply with local real estate laws, including tenant rights, contract terms, and regulations around eviction procedures.
* Taxation Compliance: The system must comply with the taxation laws relevant to rental income and property sales, ensuring correct tax calculations and reporting.
* Data Privacy Laws: The system must adhere to applicable data privacy regulations (e.g., GDPR, CCPA), ensuring that all tenant and owner data is handled securely and stored in accordance with local laws.

## Appendix A: Glossary

Glossary of Terms

* Tenant: A person or entity who rents or leases a property.
* Owner: The individual or entity who owns the property being rented.
* Property Management System (PMS): A software application that helps property managers and landlords manage rental properties, including tenant data, lease agreements, maintenance requests, and payment processing.
* Lease Agreement: A formal contract between the owner and tenant outlining the terms and conditions of renting a property.
* Role-Based Access Control (RBAC): A method of restricting system access based on the roles of individual users within an organization.
* ACID Compliance: A set of properties (Atomicity, Consistency, Isolation, Durability) that ensure database transactions are processed reliably.
* TBD: To Be Determined—used for requirements or decisions that are yet to be finalized.
* SSL/TLS: Secure Sockets Layer / Transport Layer Security—protocols used for encrypting communication between client and server.
* Multi-factor Authentication (MFA): A security process that requires users to provide two or more verification factors to gain access to a resource.

## Appendix B: Analysis Models

* Data Flow Diagram (DFD)

[Diagram illustrating how data flows within the system, including inputs, outputs, and data transformations.]

* Class Diagram

[Class diagram showing the relationships between different objects in the system, such as tenants, owners, properties, and lease agreements.]

* Entity-Relationship Diagram (ERD)

[Diagram depicting the relationships between different entities in the database, such as tenant, owner, property, and payment tables.]

* State-Transition Diagram

[Diagram showing the states of key entities (e.g., a lease agreement) and the events that trigger transitions between these states.]

## Appendix C: To Be Determined List

TBD-1: Final decision on the payment gateway integration. Will be determined based on business and security requirements.

TBD-2: Selection of cloud hosting provider (AWS, Azure, or Google Cloud).

TBD-3: Detailed tax calculations, including local tax rates and regulations.

TBD-4: Final design of the reporting module (which reports will be included and the format). TBD-5: Localization of the system for additional languages beyond English and Arabic.