Software Requirements Specification

for

Tenant Management Website

Version 1.0 approved

Prepared by Daniel Vu

Anpha LLC.

02/05/2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Daniel Vu	03/18/25	Initial document created for Tenant Management Website	1.0

---

1. Introduction

1.1 Purpose

This SRS document outlines the functional and non-functional requirements for the Tenant Management Website, a comprehensive platform designed to facilitate rent collection, bookkeeping, tenant onboarding, lease management, property maintenance tracking, and property listing management. The system is intended to streamline operations for landlords, property managers, and tenants, providing a seamless experience in

managing rental units and ensuring timely communication and financial transactions.

1.2 Document Conventions

- **Tables** are used to detail use case steps.
- **Numbered lists** indicate sequential processes.
- **Bolded text** denotes section headers.
- **Italicized text** is used for important notes or instructions.
- Section headers are in 18-point font; subheadings in 14-point font; body text in 12-point font.
- The IEEE SRS template style is followed to align with industry standards.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers:** To understand system functionalities, constraints, and integration details.
- **Testers:** To create and execute validation tests based on defined use cases.
- **Project Managers:** To oversee progress and ensure deliverables meet defined objectives.
- **Clients and Stakeholders:** To review system capabilities and confirm that features align with business needs.

Reading Suggestions:

1. Start with the Introduction to understand system's purpose and scope.
2. Review the Overall Description for a high-level summary of system functionality and dependencies.
3. Refer to the System Features section for detailed use cases and user interactions.

4. Check the Nonfunctional Requirements for performance, security, and quality benchmarks.

1.4 Product Scope

The Tenant Management Website provides a full-featured solution for property managers and tenants by integrating the following functionalities:

- **Automated Rent Collection & Bookkeeping:** Secure payment processing and financial tracking using Stripe.
- **Tenant Onboarding & Lease Management:** Seamless documentation handling and tenant verification via Firebase Auth.
- **Property Listings & Availability:** Publicly accessible property listings showcasing rental units with real-time availability updates.
- **Maintenance Request Handling:** Digital submission and tracking of maintenance requests.
- **Tenant Communication System:** Internal messaging system enabling tenants to communicate with landlords and property managers.
- **Late Payment Tracking & Automated Reminders:** Automated notifications for overdue payments and escalation processes.

The system is designed to scale efficiently, handling multiple properties and a high volume of transactions while maintaining security and usability.

1.5 References

- IEEE SRS Template Guidelines.
- Official documentation for Angular, Express.js, MongoDB Atlas, Firebase Auth, Stripe, AWS S3, Firebase Cloud Messaging, and AWS SES.
- Compliance documents related to fair housing laws and rental regulations.

---

2. Overall Description

2.1 Product Perspective

The Tenant Management Website is a cloud-based SaaS platform that integrates front-end and back-end technologies along with cloud storage and payment processing services. The system consists of:

- **Frontend:** Angular hosted on AWS S3 with CloudFront for high-performance content delivery.
- **Backend:** Node.js with Express.js running on AWS EC2, facilitating API communication and data transactions.
- **Database:** MongoDB Atlas Free-Tier for structured tenant and property data storage.
- **Authentication:** Firebase Auth, enabling secure user authentication and authorization.
- **Payment Processing:** Stripe for rent collection, ensuring secure and seamless transactions.
- **File Storage:** AWS S3 Free-Tier for storing leases, rental agreements, and maintenance records.
- **Communication Services:** Firebase Cloud Messaging for push notifications and AWS SES for email alerts.

This architecture ensures scalability, security, and reliability for managing multi-property operations efficiently.

2.2 Product Functions

The core functionalities include:

- **Role-Based Access Control:** Different access levels for tenants, property managers, and administrators.
- **Lease Renewal & Termination:** Automated lease renewal notifications and termination workflow.

- **Financial Reporting & Bookkeeping:** Generation of rental income reports and tax-compliant statements.
- **Automated Rent Reminders:** Notifications sent via email and push alerts for upcoming rent payments.
- **Legal Compliance Tracking:** Ensuring lease agreements adhere to rental laws and regulations.

2.3 User Classes and Characteristics

- **Landlords / Property Managers:** Require full access to property management tools, lease agreements, and financial reporting.
- **Tenants:** Need access to lease details, maintenance request submission, and rent payment options.
- **Prospective Renters:** Can browse available properties and submit rental applications.
- **System Administrators:** Manage platform security, user access levels, and compliance enforcement.

2.4 Operating Environment

- The platform is accessible through modern web browsers on desktop and mobile devices.
- It is optimized for cloud hosting using AWS services, ensuring reliability and uptime.
- It supports integration with third-party accounting software for extended bookkeeping capabilities.

2.5 Design and Implementation Constraints

- The system must function within AWS Free-Tier limits where applicable.
- Authentication and payment handling must comply with industry security standards such as GDPR and PCI DSS.
- API responses should maintain a response time under 500ms for optimal user experience.

2.6 User Documentation

- **User Guide:** Step-by-step instructions for onboarding, payments, and maintenance requests.
- **Admin Guide:** Configuration and troubleshooting documentation for system administrators.
- **Developer API Documentation:** Technical details for extending or integrating with external services.

2.7 Assumptions and Dependencies

- Internet connectivity is required for real-time transactions.
- External services (Stripe, Firebase, AWS) must maintain their free-tier offerings.

---

3. External Interface Requirements

3.1 User Interfaces

The system will feature an intuitive web-based user interface designed for ease of use by landlords, tenants, and property managers. The UI will be built using Angular and will include:

- **Landlord Dashboard:** Overview of rental properties, tenant details, and rent collection status.
- **Tenant Portal:** Secure access to lease agreements, payment options, and maintenance request forms.
- **Property Listings Page:** Publicly accessible listings with search and filter functionality.
- **Mobile-Friendly Design:** Responsive UI elements optimized for both desktop and mobile devices.

3.2 Hardware Interfaces

- The system is accessible through standard web browsers on desktops, laptops, tablets, and mobile devices.
- Payment transactions will be processed securely using Stripe's API without requiring additional hardware.
- File uploads and downloads (such as lease agreements) will be stored in AWS S3, accessible via the web UI.

3.3 Software Interfaces

- **Angular Frontend:** Communicates with backend services via RESTful APIs.
- **Express.js Backend:** Handles business logic and processes requests from the frontend.
- **MongoDB Atlas:** Stores tenant, property, and transaction data.
- **Firebase Authentication:** Provides secure login functionality.
- **Stripe API:** Manages rent payments and financial transactions.
- **AWS S3:** Stores lease agreements, tenant documents, and property images.
- **Firebase Cloud Messaging:** Sends push notifications to users for payment reminders and maintenance updates.
- **AWS SES:** Sends transactional emails related to payments, lease renewals, and maintenance requests.

3.4 Communications Interfaces

- The system will communicate with third-party services using RESTful APIs over HTTPS.
- Email communications will be handled using AWS SES to ensure reliable delivery.
- Real-time notifications will be implemented using Firebase Cloud Messaging for push notifications.

---

4. System Features

4.1 UC_001 – Tenant Registration

Description: Allows new tenants to register an account.
Actors: Tenants.

Preconditions:

- Tenant must provide valid personal and contact details.
- Tenant email must not already be registered.

Postconditions:

- Tenant account is created and stored in MongoDB.
- Tenant receives a verification email.

Trigger:

- Tenant initiates the registration process from the website.

Main Course:

1. Tenant navigates to the registration page.
2. Tenant enters required details (name, email, phone, password).
3. System validates input and checks if the email is already registered.
4. If valid, system stores the new account and sends a verification email.
5. Tenant clicks the verification link in the email.
6. System marks the account as verified and active.

Alternate Course:

- **4.1A:** If tenant does not verify email within 24 hours, system sends a reminder.
- **4.1B:** If tenant requests password reset before verification, system allows reset but requires verification before login.

Exception Course:

- **4.1E:** If required fields are missing, system prompts tenant to complete them.
- **4.1F:** If email is invalid, system requests re-entry.

4.2 UC_002 – Landlord Registration

Description: Allows landlords to create an account.
Actors: Landlords.

Preconditions:

- Landlord must provide valid business information.

Postconditions:

- Landlord account is created and stored in MongoDB.
- Verification email is sent to confirm registration.

Trigger:

- Landlord initiates registration.

Main Course:

1. Landlord enters business and contact details.
2. System validates input and checks for duplicate email.
3. If valid, system creates an account and sends verification email.
4. Landlord verifies email by clicking the link.
5. System activates the account.

Alternate Course:

- **4.2A:** If business verification is required, system asks for additional documents.
- **4.2B:** If verification email expires, landlord must request a new one.

Exception Course:

- **4.2E:** If required fields are missing, system prompts landlord to complete the form.

4.3 UC_003 – Property Listing Management

Description: Allows landlords to add, edit, or remove property listings.
Actors: Landlords.

Preconditions:

- Landlord must have an active account.

Postconditions:

- Listings are updated in the database and visible to tenants.

Trigger:

- Landlord initiates listing management.

Main Course:

1. Landlord logs in and navigates to the property dashboard.
2. Landlord selects "Add New Property" and enters details.
3. System validates details and saves the listing.
4. System displays the property on the tenant portal.

Alternate Course:

- **4.3A:** If a listing needs to be edited, landlord can modify and save changes.
- **4.3B:** If a listing is removed, system archives it for record-keeping.

Exception Course:

- **4.3E:** If image upload fails, system prompts user to retry.

4.4 UC_004 – Rent Payment Processing

Description: Facilitates rent payments for tenants.
Actors: Tenants, Landlords.

Preconditions:

- Tenant must have an active lease.

Postconditions:

- Payment is recorded in the system, and the tenant receives a receipt.

Trigger:

- Tenant initiates a rent payment.

Main Course:

1. Tenant logs into the portal and selects “Make Payment.”
2. Tenant enters payment details and submits the request.
3. System processes payment via a payment gateway.
4. If successful, system updates balance and sends a receipt.
5. Landlord receives payment notification.

Alternate Course:

- **4.4A:** If auto-pay is enabled, system deducts payment automatically.

Exception Course:

- **4.4E:** If payment fails, tenant is prompted to retry.

4.5 UC_005 – Maintenance Request Handling

Description: Allows tenants to report maintenance issues.
Actors: Tenants, Landlords, Maintenance Staff.

Preconditions:

- Tenant must have an active lease.

Postconditions:

- Request is logged, and a service ticket is created.

Trigger:

- Tenant submits a maintenance request.

Main Course:

1. Tenant describes the issue and submits a request.
2. System notifies the landlord and logs the request.
3. Landlord assigns a maintenance worker.
4. Worker fixes the issue and marks the request as resolved.

Alternate Course:

- **4.5A:** If issue is urgent, system marks it as high priority.

Exception Course:

- **4.5E:** If tenant submits an incomplete request, system prompts for details.

4.6 UC_006 – Lease Agreement Generation

Description: System generates lease agreements.
Actors: Landlords, Tenants.

Preconditions:

- Landlord must have an active property listing.

Postconditions:

- Lease agreement is stored and digitally signed.

Trigger:

- Lease process is initiated by landlord or tenant.

Main Course:

1. System collects lease terms.
2. System generates lease document.
3. Tenant reviews and signs the document.
4. Landlord counter-signs.
5. System stores the signed document.

Alternate Course:

- **4.6A:** If tenant requests changes, system regenerates the agreement.

Exception Course:

- **4.6E:** If one party does not sign within 7 days, lease is voided.

4.7 UC_007 – Tenant Dashboard Access

Description: Allows tenants to view lease details, payments, and maintenance requests.
Actors: Tenants.

Preconditions:

- Tenant must be logged in.

Postconditions:

- Dashboard displays up-to-date information.

Trigger:

- Tenant logs in.

Main Course:

1. Tenant accesses dashboard.
2. System fetches and displays relevant details.

Exception Course:

- **4.7E:** If tenant session expires, system prompts re-login.

4.8 UC_008 – Lease Renewal & Termination

Description: Manages lease renewal and termination.
Actors: Landlords, Tenants.

Preconditions:

- Active lease exists.

Postconditions:

- Lease is renewed or terminated.

Trigger:

- Lease expiration date approaches.

Main Course:

1. Tenant receives renewal notification.
2. Tenant submits renewal request or termination notice.
3. Landlord reviews request and approves/rejects it.
4. System updates lease status and sends confirmation emails.

Alternate Course:

- **4.8A:** If renewal terms change, tenant must approve before finalizing.

Exception Course:

- **4.8E:** If tenant does not respond, system auto-applies lease termination.

4.9 UC_009 – Tenant Communication System

Description: In-app messaging for tenants and landlords.
Actors: Tenants, Landlords.

Preconditions:

- User must be authenticated.

Postconditions:

- Messages are stored and accessible via user dashboards.

Trigger:

- Tenant or landlord initiates a message.

Main Course:

1. Tenant sends a message to landlord.
2. Landlord receives the message and responds.
3. System logs conversation.

Alternate Course:

- **4.9A:** If tenant marks a message as urgent, system prioritizes it.

Exception Course:

- **4.9E:** If message delivery fails, system retries.

4.10 UC_010 – Late Payment Handling

Description: Manages overdue rent payments.
Actors: Tenants, Landlords.

Preconditions:

- Rent payment is past due date.

Postconditions:

- Payment is recorded, and penalties (if applicable) are applied.

Trigger:

- Rent payment becomes overdue.

Main Course:

1. System detects overdue payment.
2. Sends automated reminders.
3. Applies late fee if unpaid.
4. Landlord receives notification about overdue status.

Alternate Course:

- **4.10A:** If tenant sets up auto-payment, late fee is avoided.

Exception Course:

- **4.10E:** If tenant submits partial payment, system applies remainder.

---

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should handle at least 500 concurrent users without performance degradation.
- Payment processing should be completed within 5 seconds.
- Page load times should not exceed 2 seconds for standard interactions.

5.2 Safety Requirements

- The system should prevent duplicate rent payments.
- Error handling should ensure no loss of transaction data in case of failure.

5.3 Security Requirements

- User authentication must follow OAuth 2.0 and Firebase security best practices.
- All sensitive data (passwords, payment details) must be encrypted at rest and in transit.
- Only authorized users should have access to financial data.

5.4 Software Quality Attributes

- **Usability:** Intuitive UI for both landlords and tenants.
- **Reliability:** 99.9% uptime target for critical services.
- **Scalability:** Ability to scale storage and processing power as the user base grows.
- **Maintainability:** Well-documented API endpoints for future expansion.

5.5 Business Rules

- Late payments incur a penalty after a grace period.
- Lease termination requires a 30-day notice unless otherwise stated.
- Tenants must submit maintenance requests digitally.

5.6 Compliance and Legal Considerations

- Must adhere to fair housing regulations.

- Stripe integration must comply with PCI-DSS security standards.
- Data retention policies should align with GDPR and CCPA.

---

6. Other Requirements

- Future integration with third-party accounting systems (e.g., QuickBooks).
- Multi-language support for international tenants and landlords.

Appendix A: Glossary

- **Angular:** A frontend framework for building web applications.
- **Express.js:** A backend framework for handling API requests.
- **MongoDB Atlas:** A cloud-hosted NoSQL database.
- **Firebase Auth:** Authentication service for secure user logins.
- **Stripe:** A financial service for processing payments securely.
- **AWS S3:** A storage solution for hosting images and documents.
- **AWS SES:** An email service for sending transactional emails.
- **Firebase Cloud Messaging:** A push notification service.

Appendix B: Analysis Models

- **Use Case Diagrams:** Illustrate interactions between system users and core functionalities.
- **State Transition Diagrams:** Show how data flows from registration to payment processing.
- **Sequence Diagrams:** Detail step-by-step system processes for transactions and notifications.

Appendix C: To Be Determined List

- Final customization options for notification preferences.
- Legal agreements for digital lease signing.
- Additional features for property analytics and financial reporting.
