Software Requirements Specification[Edited version]

For

Society Management System

Version 1.1

Prepared By:

Hamza Naveed	**22L-7871**
Abdul Ahad	**22L-7852**
Faraz Iftikhar	**22L-7974**
Ahmad Tahir	**22L-7869**

Table of Contents

Table of Contents

Revision History

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Functions

4. System Features

4.1 House Owner Contacts Access (Admin)
4.2 Upcoming Events (User)
4.3 Maintenance Bills Management (User)
4.4 Maintenance Bills Management (Admin)
4.5 Emergency Helplines (User)
4.6 Complaint Box (User)
4.7 Amenity Reservations (User)
4.8 Administration of Amenity Reservations
4.9 Visitor Management
4.10 Resident Voting (User)
4.11 Administration of Resident Voting
4.12 Property Listings
4.13 Administration of Resident Voting

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.5 Business Rules

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To be Determined List

1. Introduction

1.1 Purpose

The Purpose is to create a seamlessly integrated and user-friendly Society Management System that fosters transparent communication, efficient governance, and a sense of community. Our vision is to empower residents, committee members, and management personnel with a digital platform that not only simplifies daily tasks but also nurtures a vibrant and engaged living environment.

The concept is groundbreaking, especially considering the prevalent practices in housing societies where residents and management rely on fragmented chat applications like WhatsApp and Telegram for communication. Additionally, the persistence of paper-based challans for society maintenance bills and the lack of access for security guards to efficient communication platforms highlight significant inefficiencies in existing systems.

Unlike traditional housing societies websites that primarily focus on catering to potential real estate buyers and handle amenities reservations through physical visits, our innovative approach addresses these shortcomings comprehensively. By introducing a centralized web-based platform, we aim to bridge communication gaps, streamline maintenance bill processes, and empower security personnel with efficient tools. Residents will no longer have to visit the society premises for amenity reservations, enhancing convenience and accessibility. This pioneering initiative is set to revolutionize the way housing societies operate, fostering a sense of community, transparency, and security among residents.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents. This document's content is written in Arial font with a size of 11. The body's major titles are written in Times New Roman, font size 18, while the subtitles are written in Times New Roman, font size 14. Requirements have been broken down into main functional categories, with each functional area subdivided into features. Within the laid-out document format, there are functional and nonfunctional needs. This document consists of text and different diagrams to elaborate the requirements. Every functional requirement is in hierarchical order with further division to the minimum level.

1.3 Intended Audience and Reading Suggestions

The intended audience for this Society Management System (SMS) document encompasses a diverse group of stakeholders, each playing a crucial role in the project's success. This includes:

•**Project Manager**: Utilizes the document to make informed decisions regarding the development process.

•**Test Case Engineers**: Refer to the document for crafting test cases aligned with specified requirements.

•**Software Engineers**: Rely on the document to guide the development process, ensuring the software meets outlined requirements.

•**Residents**: End-users who directly engage with the SMS, relying on the document to understand and navigate the system.

•**Stakeholders and Managers**: Gain valuable insights into the forthcoming SMS, aiding their understanding and expectations.

The document is structured to provide a seamless reading experience, beginning with an overview section offering essential background knowledge. Tailored reading sequences are recommended for different user types, ensuring developers, project managers, users, and testers can efficiently access details relevant to their roles. Comprehensive explanations, supported by diagrams, detail the SMS's goals, characteristics, interfaces, functionalities, and operational constraints, fostering a holistic understanding for all involved parties.

1.4 Project Scope

To develop a cutting-edge web-based Society Management System aimed at enhancing operational efficiency and effectiveness within the community. The system's primary objectives include fostering seamless communication and collaboration among society members and management, ultimately elevating transparency and accountability in the management processes.

1.5 References

[1] IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

Revision History

Name	Date	Reason For Changes	Version
Requirements Document	25-09-2023	New Document	1.0
Requirements Document	30-11-2023	Revised Document	1.1

2. SRE Product Description

2.1 Product Perspective

The Society Management System (Resimate) is an innovative, yet-to-be-introduced product that stands as a pioneering solution in the realm of community management. It's not a mere replacement for existing systems but a new, self-contained product designed to redefine the landscape of housing society operations. Unlike conventional approaches relying on fragmented communication tools, Resimate introduces a revolutionary platform that seamlessly integrates features like House Owner Contacts, Upcoming Events, Maintenance Bills, and more. The system also interfaces with third-party service providers, ensuring smooth integration of functionalities like payment processing and vehicle tracking. Resimate not only streamlines administrative tasks but also enhances community engagement through features like Resident Voting, Campaign Contributions, and Visitor Management. This unique product, as outlined in the Software Requirements Specification, is poised to address the shortcomings of current practices by providing a holistic and user-friendly solution for transparent communication, efficient governance, and vibrant community engagement.

2.2 Product Functions

2.2.1. User Information Management: Enables residents to view and update contacts.

2.2.2. Event Management: Displays upcoming events and facilitates organization.

2.2.3. Billing and Payments: Generates electronic maintenance bills and allows secure online payments.

2.2.4. Complaints and Feedback: Provides a platform for complaint submission and feedback.

2.2.5. Amenity Reservations: Allows residents to book community facilities with a payment option.

2.2.6. Security and Monitoring: Implements a Vehicle Monitoring System and displays emergency helpline numbers.

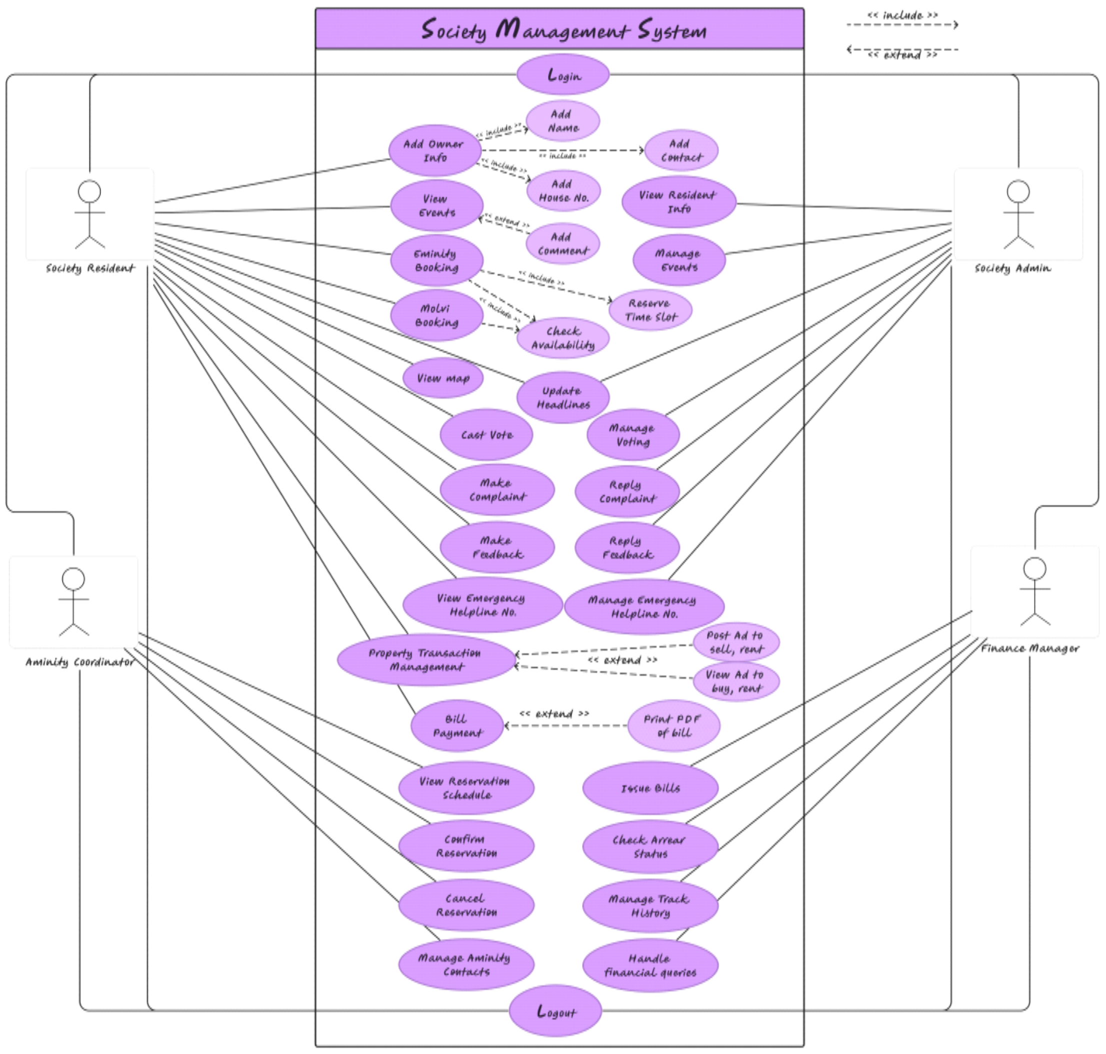
2.2.7. Community Engagement: Supports digital voting, campaign contributions, and streamlined visitor registration.

2.2.8. Property Transactions: Facilitates property listing, buying, selling, and renting with clearance checks.

2.2.9. Media and Information: Displays news headlines, community updates, and an interactive map.

2.2.10. Integration with Third-party Services: Interfaces with external providers for payment processing, vehicle tracking, and media services.

USE CASE DIAGRAM



.

4. System Feature

4.1 House Owner Contacts Access (Admin)

4.1.1 Description and Priority

This feature provides access to homeowner contact information. It is of High priority as it facilitates communication and engagement with property owners.

Rating: - Benefit: 9 Penalty: 3 Cost: 5 Risk: 4

4.1.3 Functional Requirements

4.1.3.1 The system shall provide a user interface with an option to manage homeowner contacts.

4.1.3.2 The system shall display a searchable list of homeowner contacts.

4.1.3.3 The system shall allow user to search for homeowners based on specific criteria.

4.1.3.4 The system shall enable users to view and edit existing homeowner contact details.

4.1.3.5 The system shall support the addition of new homeowner contacts, requiring necessary information such as name, address, and contact number.

4.1.3.6 The system shall include a confirmation prompt before deleting any homeowner contact.

4.1.3.7 The system shall handle errors gracefully, providing informative messages for invalid inputs or system failures.

ID & Name:	**UC-01 View and edit contact details**	
Created By:	Faraz Iftikhar	**Date created:** 30/11/2023
Primary Actors:	Residents, Administrator	**Secondary Actor:** N/A
Description:	This use case outlines the process of administrator viewing and editing existing homeowner contact details within the Society Management System.	
Trigger:	Admin selects the 'View/Edit Contacts' option in the user profile settings	
Pre-Conditions:	Pre 1: Admin must be logged in. Pre 2: Homeowner contact details must be available in the system.	
Post-Conditions:	Post 1: Changes to contact details are saved Post 2: The user's profile is updated.	
Normal Flow:	1. Administrator navigates to the admin dashboard. 2. Administrator selects 'View/Edit Contacts.' 3. System displays a list of existing homeowners' contact details. 4. Administrator selects a specific homeowner to view or edit their contact information. 5. Administrator edits the contact information as needed. 6. Administrator saves the changes. 7. System updates the user's profile with the modified contact details.	
Alternative Flow:	If there are no existing homeowner contact details, the administrator is prompted to add them.	
Exceptions:	Technical issues prevent the saving of changes.	
Priority:	Medium	
Frequency of Use:	Occasionally when administrators need to update homeowner contact information or resolve issues. Multiple times a week.	
Business Rules:	BR- **5.5.1**	
Other Information:	Ensure data privacy and secure transmission of updated contact details.	
Assumptions:	Administrators have the necessary permissions to edit homeowner contact details, and the system stores contact details accurately.	

4.2 Upcoming Events (User)

4.2.1 Description and Priority

This feature keeps residents informed about society events. It is of High priority as it enhances community engagement and cohesion.

Rating: - Benefit: 9 Penalty: 2 Cost: 6 Risk: 3

4.2.3 Functional Requirements

4.2.3.1 The system shall present an intuitive interface for residents to access information on upcoming society events.

4.2.3.2 The system shall display a comprehensive list of upcoming events with relevant details.

4.2.3.3 Residents shall be able to view additional details about each event by clicking on the event listing.

4.2.3.4 Residents shall be able to add comments on the society events.

4.2.3.5 The system shall send notifications to residents about upcoming events.

4.8 Amenity Reservations (User)

4.8.1 Description and Priority

Amenity Reservations allow residents to book community facilities. This feature is of High priority as it enhances user convenience and community engagement.

Rating: - Benefit: 8 Penalty: 2 Cost: 5 Risk: 3

4.8.3 Functional Requirements

4.8.3.1 The system shall provide a list of available community facilities for reservation

4.8.3.2 The system shall allow users to select a specific facility for reservation.

4.8.3.3 The system shall prompt users to choose a date and time for the reservation.

4.8.3.4 The system shall confirm and update the facility schedule upon user confirmation.

4.8.3.5 The system shall handle and notify users of any scheduling conflicts.

ID & Name	**UC-07 Reserve an Amenity Service**	
Created By	Hamza Naveed	**Date Created:**22/10/23
Primary Actors	Resident	**Secondary Actor:** Society Administration

Description	This use case enables residents to book community facilities. The system provides a user-friendly interface for residents to browse available facilities, select a specific one, choose a date and time for reservation, confirm the booking, and receive notifications.
Trigger	The resident selects the "Amenity Reservations" option from the menu.
Pre-Conditions	**Pre-1:** The resident is logged into the system. **Pre-2:** Community facility information is up to date in the system.
Post-Conditions	**Post-1:** The facility schedule is updated with the reservation. **Post-2:** The resident receives confirmation of the reservation.
Normal Flow	• Residents select the "Amenity Reservations" option. • The system displays a list of available community facilities. • Residents select a specific facility for reservation. • System prompts the user to choose a date and time for the reservation. • Resident confirms the reservation details. • System confirms the reservation and updates the facility schedule.
Alternative Flow	• System detects a scheduling conflict. • System notifies the user of the conflict. • User is prompted to choose an alternative date or facility.
Exceptions	**System Unavailability:** • The system is temporarily unavailable. • System displays an error message. • The user is prompted to try again later.
Priority	High
Frequency of Use	Approximately 2-3 times by a resident per month
Business Rules	BR-**5.5.2**
Other Information	In case of a scheduling conflict, the system will provide clear and timely notifications to the resident to ensure they are aware of the issue.
Assumptions	Users are familiar with community facilities. Residents are aware of reservation limitations and agree to terms.

4.9 Administration of Amenity Reservations

4.9.1 Description and Priority

Administration of Amenity Reservations allows administrators to manage bookings for community facilities. This feature is of High priority as it streamlines facility usage and ensures efficient administration.

Rating: - Benefit: 8 Penalty: 2 Cost: 5 Risk: 3

4.9.3 Functional Requirements

4.9.3.1 The system shall provide an administrative interface with an option to access amenity reservations.

4.9.3.2 The system shall display an overview of facility bookings on the administrative dashboard.

4.9.3.3 The system shall allow administrators to view detailed information about specific reservations.

4.9.3.4 The system shall enable administrators to modify existing reservations, updating dates, times, and other relevant details.

4.9.3.5 The system shall allow administrators to cancel reservations, provide confirmation prompts and adjust the facility schedule.

ID & Name	**UC 08 - Manage Amenity Services**	
Created By	Muhammad Ahmad	**Date Created:**22/10/23
Primary Actors	Admin	**Secondary Actor:** N/A
Description	This use case outlines the process by which the admin manages amenity services in the society management system.	
Trigger	Admin initiates the process of managing amenity services.	
Pre-Conditions	**Pre-1:** The admin is logged into the society management system. **Pre-2:** Society amenity parameters, such as availability, booking rules, and rates, are configured and up to date.	
Post-Conditions	**Post-1:**The system has successfully updated and reflected the changes in accordance with the amenity reservation details provided, ensuring the reservation is accurately recorded and managed.	
Normal Flow	• Admin navigates to the "Amenities" section of the admin dashboard. • System presents options related to amenity services management. • Admin selects "Manage Amenities. **For Amenity Modification:** 7. Admin selects the option to "Modify Amenity." • The system presents a list of existing amenities. • Admin selects the amenity to be modified. • The system displays current details of the selected amenity. • Admin modifies the necessary details and confirms changes. **For Amenity Deletion:** 12. Admin selects the option to "Delete Amenity." • The system presents a list of existing amenities. • Admin selects the amenity to be deleted. • System prompts admin for confirmation. • Admin confirms the deletion.	
Alternative Flow	**Alternative Flow:** • **Amenity Addition:** • If needed, the system may validate the input details to ensure they meet predefined criteria.	

	• Admin corrects any validation errors. • **Amenity Modification:** • If needed, the system may validate the modified details. • Admin corrects any validation errors.
Exceptions	• If there are issues with the input details, the system provides an error message and prompts the admin to correct the details. • Invalid Amenity Modification (Step 11b): • If there are issues with the modified details, the system provides an error message and prompts the admin to correct the details.
Priority	Medium
Frequency of Use	Approximately once a week.
Business Rules	• Only authorized admins can manage amenity services. • Amenity details should adhere to society policies and regulations.
Other Information	• The system should maintain a record of changes made to amenity details for auditing purposes. • Availability schedules should be clearly communicated to residents.
Assumptions	• Admins have a clear understanding of society's amenity policies. • The society management system is connected to a secure database to store and retrieve amenity details.

4.11 Resident Voting (User)

4.11.1 Description and Priority

Resident Voting enables community members to participate in digital voting activities. This feature is of High priority as it promotes democratic participation within the community.

Rating: - Benefit: 9 Penalty: 2 Cost: 6 Risk: 4

4.11.3 Functional Requirements

4.11.3.1 The system shall provide a user interface with an option for Resident Voting.

4.11.3.2 The system shall display a list of ongoing and upcoming community votes.

4.11.3.3 The system shall allow users to select a specific voting event to view details.

4.11.3.4 The system shall present detailed information about voting options and instructions for the selected event.

4.11.3.5 The system shall record and update votes in real-time.

ID & Name	**UC-10 Voting in Society**	
Created By	Faraz Iftikhar	**Date created:** 30/11/2023
Primary Actors	Residents, Administrator	**Secondary Actor:** N/A
Description	This use case describes the process of residents casting votes for fellow residents elected for different purposes in the community.	
Trigger	User selects the voting option in the community engagement section.	
Pre-Conditions	Pre 1: Users must be logged in Pre 2: An active voting event must be in progress.	
Post-Conditions	Post 1: The user's vote is recorded Post 2: The voting results are updated.	
Normal Flow	1. User navigates to the community engagement section. 2. User selects the active voting event. 3. User reviews the list of candidates and makes selections. 4. User submits the vote. 5. System records the vote and updates the voting results.	
Alternative Flow	- If there are no active voting events, the system notifies the user. - If the user has already voted, the system prevents multiple votes	
Exceptions	- Technical issues preventing vote submission.	
Priority	High	
Frequency of Use	Once a year.	
Business Rules	Only registered residents can participate in voting.	
Other Information	Ensure the voting process is secure and anonymous	
Assumptions	The system has accurate resident information, and there are no major disruptions during the voting process.	

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 The system shall be able to accommodate a minimum of 5,000 homeowner profiles without significant performance degradation.

5.1.1.1 The performance degradation shall not exceed 1 second when loading 10,000 homeowner profiles simultaneously.

5.1.1.2 During high traffic conditions (usage of more than 10,000 users at once), the system shall manage to maintain optimal load times for user profile data.

5.1.2 The system shall load event listings within 5 seconds of user request.

5.1.2.1 Under normal conditions (users<5,000), the execution time for loading event listings shall not exceed 0.5 seconds.

5.1.2.2 Under high traffic conditions (usage of more than 10,000 users at once), event listings shall load without exceeding 1 second.

5.1.3 The system shall update the bill status within 1 hour of bill payment.

5.1.3.1 Under normal conditions (users<5,000), the execution time to update a bill status shall not exceed 10 minutes.

5.1.3.2 Under high traffic conditions (usage of more than 10,000 users at once), the bill status shall be updated within 1 hour.

5.1.4 The system shall accommodate concurrent resident voting by at least 500 residents without performance degradation.

5.1.4.1 Under normal conditions (residents <5,000), the execution time for processing simultaneous votes shall not exceed 0.5 seconds.

5.1.4.2 During high traffic conditions (usage of more than 10,000 residents at once), the system shall process concurrent resident votes without performance degradation exceeding 1 second.

5.1.5 The system shall accommodate concurrent bill payments by at least 500 residents without performance degradation.

5.1.5.1 The execution time to process concurrent bill payments shall not exceed 0.5 seconds under normal circumstances (users<5,000).

5.1.5.2 In case under heavy traffic (usage of more than 10,000 users at once), the execution time to process concurrent bill payments shall not exceed 1 second.

5.2 Safety Requirements

5.2.1 The system shall implement robust data encryption protocols to safeguard sensitive resident information and financial transactions, ensuring protection against unauthorized access or data breaches.

5.2.2 The system shall notify if the user does not save the records.

5.2.2.1 The software shall save to previous checkpoint in case of power failure.

5.2.2.2 In case of no activity for more than 30 mins the system shall automatically log out to reduce load on server.

5.2.3 The system shall be updated once a month with a downtime of 2-3 hours at midnight.

5.2.3.1 The system shall be able to recover from a failure within an hour or less.

5.2.4 The system shall regularly undergo third-party security audits and penetration testing for validating encryption effectiveness and identifying potential vulnerabilities.

5.3 Security Requirements

5.3.1 The system shall implement Role-Based Access Control (RBAC) to restrict access to administrative features, allowing only authorized administrators to manage homeowner contacts, events, and maintenance bills.

5.3.2 The system shall enforce secure login mechanisms, including multi-factor authentication, to enhance the security of user accounts and prevent unauthorized access.

5.3.3 To prevent automated login, CAPTCHA shall be used. If there are any issues while accessing the system, error messages shall be presented.

5.3.4 In the case of downloadable files, such as PDF copies of maintenance bills, the system shall implement secure file handling to prevent unauthorized access or tampering.

5.3.5 The system shall implement secure session management practices, including automatic logout after a defined period of inactivity to reduce the risk of unauthorized access.

5.4 Software Quality Requirements

5.4.1 Robustness:

- The system shall handle unexpected inputs or scenarios without crashing, maintaining a failure rate of less than 1% in stress testing scenarios.

5.4.2 Usability:

- The system interface shall be intuitive, ensuring a task completion success rate of at least 90% for common user actions.

5.4.3 Adaptability:

- The system shall adapt to changes in society's policies or structure with minimal disruption, maintaining a documented change acceptance rate of 95%.

5.4.4 Availability:

- The system shall be available for use 99% of the time, excluding scheduled maintenance periods.

5.4.5 Correctness:

- The system shall ensure accurate representation and manipulation of data throughout its lifecycle.

5.4.6 Maintainability:

- The system codebase shall have a maintainability index of at least 80, as measured by static code analysis tools.

5.4.7 Interoperability:

- The system shall seamlessly integrate with commonly used third-party tools and platforms, achieving successful integration for at least 95% of tested cases.

5.4.8 Reliability:

- The system shall operate without critical failures for a minimum of 30 consecutive days under normal operating conditions.

5.4.9 Portability:

- The system shall be designed to run on multiple operating systems, ensuring compatibility with commonly used platforms (e.g., Windows, Linux, macOS).

5.4.10 Efficiency:

- The system shall optimize processes and workflows within each feature for efficiency, ensuring that user wait time and resource usage are minimized to less than 1 second.

Appendix A: Glossary

All terminologies used within this document were simple and self-explanatory, thus a separate glossary section is not applicable.

Appendix B: Analysis Models

The necessary analysis models, including the Class Diagram, Context Diagram, and Use Cases, have been provided at the beginning of this document to offer a comprehensive overview of the system architecture and interaction flows.

Appendix C: To be Determined List

There are currently no items that have been identified as to-be-determined. This section is not applicable and may be updated in future revisions of the document as needed.