

---

# **Software Requirements Specification**

**for**

# **Community Management System**

**Prepared by**

**Ahmed Mustafa, Haziq Naeem, Muneeb-ul-Islam**

**NUCES ISLAMABAD**

**20-3-2025**

# **1. Table of Contents**

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

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

3. External Interface Requirements

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

4. System Features

5. Nonfunctional Requirements

5.1 Product Requirements

5.2 Organizational Requirements

5.3 External Requirements

6. Use Case Diagram & User Stories

6.1 Use Case Diagram

6.2 User Stories with Pre/Post Conditions

7. Sequence Diagrams

8. Class Diagram

9. Product Backlog

10. Sprint Backlog

11. Meeting Minutes

12. Version Control & Contribution Evidence

---

# 1. Introduction

## 1.1 Purpose

The **Communi3y** is Community Management System (CMS) that is a web-based application designed to streamline property management, public service requests, digital booking, facility reservations, and incident reporting within a community.

## 1.2 Document Conventions

This document follows the IEEE SRS template format.

## 1.3 Intended Audience and Reading Suggestions

- **Developers & Database Administrators** – Understand system functionalities.
- **Project Managers & Scrum Masters** – Track development milestones.
- **Residents & Admins** – Comprehend system capabilities.

## 1.4 Product Scope

The system provides **digital solutions** to manage properties, request public services, participate in digital voting, reserve community facilities, and report incidents. The backend will be implemented using **Next.js, Node.js, and MySQL**.

## 1.5 References

- IEEE 830-1998 Software Requirements Specification Standard.
  - Project Diagrams: Use Case Diagram, Sequence Diagram, Class Diagram.
- 

# 2. Overall Description

## 2.1 Product Perspective

This system is a standalone web-based application that digitalizes community management operations, replacing manual processes.

## 2.2 Product Functions

- **Property Registration**
- **Public Service Requests**
- **Digital Voting System**
- **Recreation Facility Reservation**
- **Crime & Incident Reporting**

## 2.3 User Classes and Characteristics

- **Residents** – Property owners or tenants accessing services.
- **Admins** – Management personnel handling operations.

## 2.4 Operating Environment

- Web-based system accessible via modern browsers.
- Hosted on a **local server infrastructure**.
- Uses **Html, CSS and React(if applicable at the end of sprint 3) (Frontend), Node.js (Backend), MySQL (Database)**.

## 2.5 Design and Implementation Constraints

- The system should be deployed on a **MySQL database**.
- Must follow **security protocols** for data encryption and access control.

## 2.6 User Documentation

- **User Manual** (for residents & admins)
- **API Documentation** (for developers)

## 2.7 Assumptions and Dependencies

- Reliable **internet connection** for cloud-based functionality.
- Compliance with **local property laws** for registration validation.

---

# 3. External Interface Requirements

## 3.1 User Interfaces

- **Web-based UI** developed using **HTML, CSS and REACT.js (applicable at the end of sprint 3)**.
- Interactive forms for **property registration, voting, and service requests**.

## 3.2 Hardware Interfaces

- Deployable on **standard server hardware**.
- Must support **cloud hosting** for scalability.

## 3.3 Software Interfaces

- Integration with **MySQL database**.

- Node.js for backend operations.

### 3.4 Communications Interfaces

- Email & SMS notifications for critical updates.
  - Secure HTTPS for data transmissions.
- 

## 4. System Features

- 4.1 User Stories & Acceptance Criteria

### GWT-Formatted User Stories with Pre/Postconditions

---

#### 1. Register Property

**Scenario: Register a property in the system**

Given the resident is logged in and on the property registration page,  
And the resident has valid property documents ready,  
When the resident fills in property details and uploads the documents,  
Then the system should validate the data and documents,  
And the registration request should be accepted,  
And the resident should receive a confirmation notification.

**Preconditions:**

- Resident has an active user account.
- Property is not already registered.

**Postconditions:**

- Property is linked to the resident's account.
  - Confirmation is recorded in the resident's notification history.
- 

#### 2. Manage Resident Data

**Scenario: Add, update, and manage resident data**

Given the admin is logged in with appropriate access rights,  
When the admin views, edits, or deletes a resident profile,  
Then the changes should be reflected immediately in the database,  
And a confirmation message should be displayed.

**Preconditions:**

- Admin is authenticated and authorized.

**Postconditions:**

- Database reflects updated resident information.
  - Action is logged with a timestamp.
- 

**3. Make Payment****Scenario: Pay society fees and bills**

Given the resident is logged in and has outstanding bills,  
When the resident selects a bill and completes a payment through a valid method,  
Then the payment should be processed securely,  
And a digital receipt should be generated,  
And the payment status should update in the dashboard.

**Preconditions:**

- Resident has valid payment method linked.
- Bill is generated and pending.

**Postconditions:**

- Payment is recorded in the system.
  - Dashboard shows updated payment status.
- 

**4. Track & Update Payments****Scenario: Admin monitors and updates payment records**

Given the admin is authenticated and viewing a resident's payment history,  
When the admin updates the payment status manually,  
Then the changes should be saved,  
And a timestamped log should be created.

**Preconditions:**

- Admin has access to financial records.

**Postconditions:**

- Updated payment status visible in resident record.
  - Audit trail log is updated.
- 

**5. Manage Bills****Scenario: Admin generates and sends bills**

Given the admin is on the bill generation module,

When the admin enters or adjusts bill details (amount, due date, etc.),  
Then bills should be assigned to the appropriate residents,  
And a notification should be sent.

**Preconditions:**

- Admin has access to billing module.

**Postconditions:**

- Bill is stored in system.
  - Notification reaches residents via email/dashboard.
- 

**6. Book Parking & Reserve Facility****Scenario: Reserve community resources**

Given the resident is logged in and on the reservation portal,  
When they select an available parking or facility slot,  
Then the resource should be marked as reserved,  
And the resident should receive a confirmation.

**Preconditions:**

- Slots are available.
- Resident has no conflicting reservations.

**Postconditions:**

- Resource is booked.
  - Reservation log is updated.
- 

**7. Submit Maintenance Request****Scenario: Request property maintenance**

Given the resident is logged in and has identified an issue,  
When the resident fills out a request and uploads supporting media,  
Then the system should log the request and assign a ticket number,  
And the resident should see the request in their history and track the status.

**Preconditions:**

- Resident account is active.
- Issue is within maintenance scope.

**Postconditions:**

- Request is logged in system.
  - Resident receives ticket and status tracking.
- 

## 8. Schedule Infrastructure Maintenance

### Scenario: Schedule community infrastructure maintenance

Given the admin is logged in and on the maintenance module,  
When the admin schedules a new task with date, time, and location,  
Then the task should be assigned to the relevant maintenance team,  
And progress should be tracked until completion.

#### Preconditions:

- Admin has maintenance scheduling permissions.
- Teams are available.

#### Postconditions:

- Maintenance task created.
  - Status updates reflected in the system.
- 

## 9. Manage Event RSVPs

### Scenario: RSVP to community events

Given the resident is logged in and on the events page,  
When they select an event and confirm attendance,  
Then the RSVP should be saved,  
And the resident should receive a reminder or event pass.

#### Preconditions:

- Event is upcoming.
- RSVP deadline not passed.

#### Postconditions:

- Resident marked as attendee.
  - Reminder/event pass is sent.
- 

## 10. Send & Receive Notifications

### Scenario: Admin sends notifications to residents

Given the admin is logged in and on the notification panel,  
When they draft and send a notification,



Then it should appear in residents' dashboards/inboxes,  
And the system should log the timestamp and audience.

**Preconditions:**

- Admin has messaging privileges.

**Postconditions:**

- Notification is delivered.
  - Message log is updated.
- 

**11. Request Public Service****Scenario: Submit a public service request**

Given the resident is logged in and needs a public service,  
When they submit a request with description and location,  
Then a request ticket should be generated,  
And the resident should be able to track the status.

**Preconditions:**

- Service is available for request.

**Postconditions:**

- Ticket is generated.
  - Request is tracked in the system.
- 

**12. Report Crime or Incident****Scenario: Report a community crime or incident**

Given the resident is logged in and has witnessed an incident,  
When they report the details and location,  
Then the system should assign it to the concerned authority,  
And send an acknowledgment.

**Preconditions:**

- Reporting feature is available to residents.

**Postconditions:**

- Report is saved and sent to authorities.
  - Acknowledgment sent to resident.
-

### 13. Submit Public Feedback

**Scenario: Submit community feedback**

Given the resident is logged in and on the feedback module,  
When they write and submit feedback,  
Then the system should acknowledge the submission,  
And the feedback should reflect in the admin analytics.

**Preconditions:**

- Resident is allowed to give feedback.

**Postconditions:**

- Feedback is logged.
  - Analytics updated.
- 

### 14. Digital Voting

**Scenario: Participate in digital voting**

Given the resident is logged in and there is an active poll,  
When they view and submit their vote,  
Then the system should confirm submission,  
And the vote should remain anonymous.

**Preconditions:**

- Poll is active.
- Resident is eligible to vote.

**Postconditions:**

- Vote is cast anonymously.
  - Confirmation message displayed.
- 

### 15. Manage Institutes & Healthcare

**Scenario: Manage education and healthcare facilities**

Given the admin is logged in and on the institute/healthcare module,  
When the admin adds or edits profiles and schedules,  
Then the updates should reflect in the community portal.

**Preconditions:**

- Admin has relevant access.

**Postconditions:**

- Profiles and schedules are updated.
- Portal shows current information.

---

## 5. Nonfunctional Requirements

### 5.1 Product Requirements

- The system should support **1000+ concurrent users**.
- Should process **real-time transactions**.

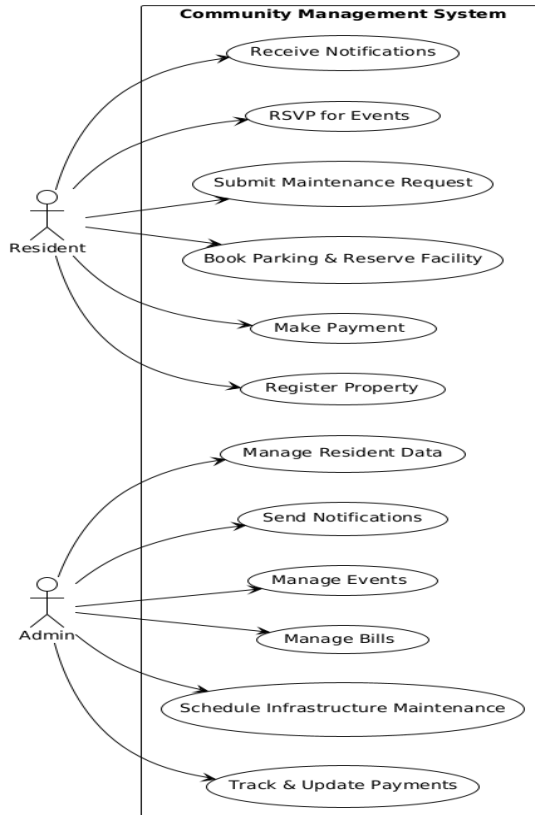
### 5.2 Organizational Requirements

- Uses **Agile methodology** for iterative development.

### 5.3 External Requirements

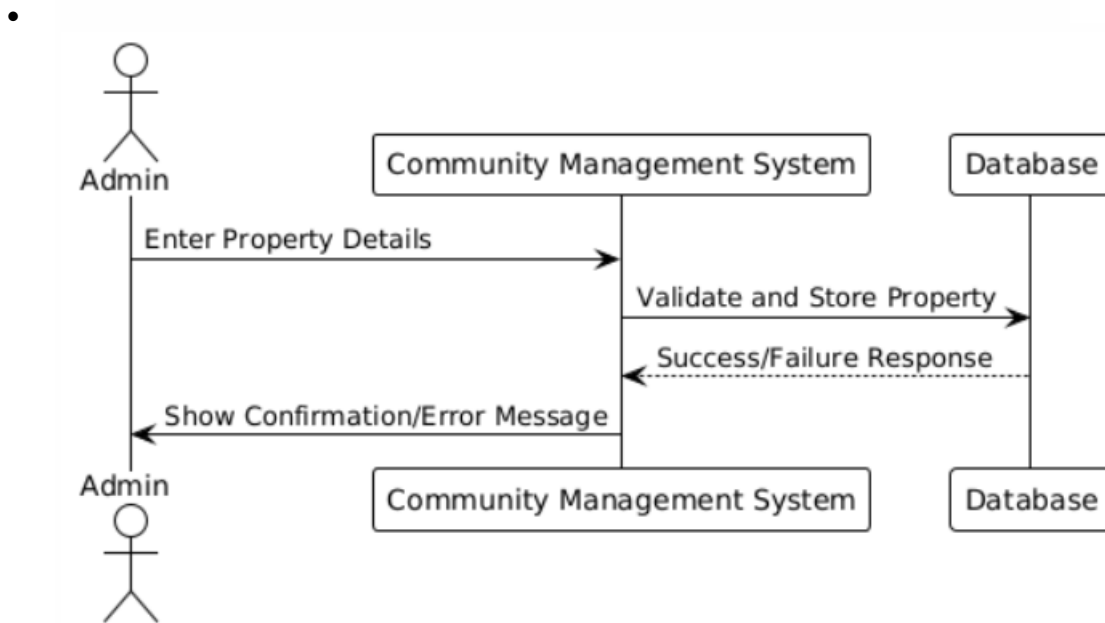
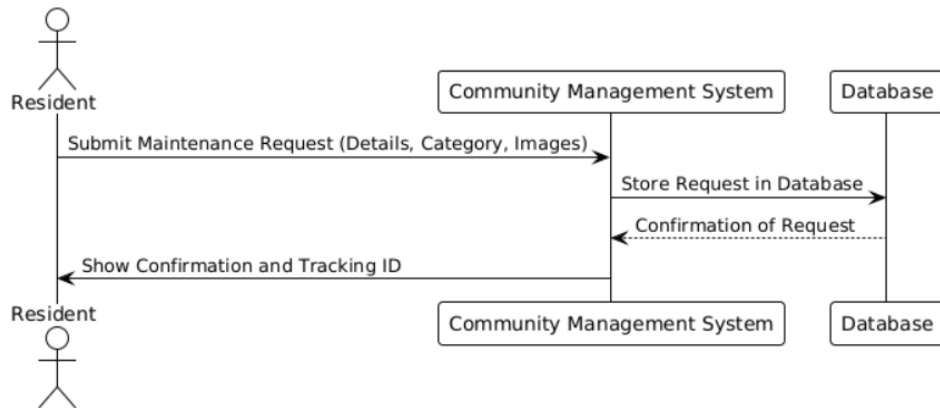
- Should comply with **GDPR & local property laws**.
- 

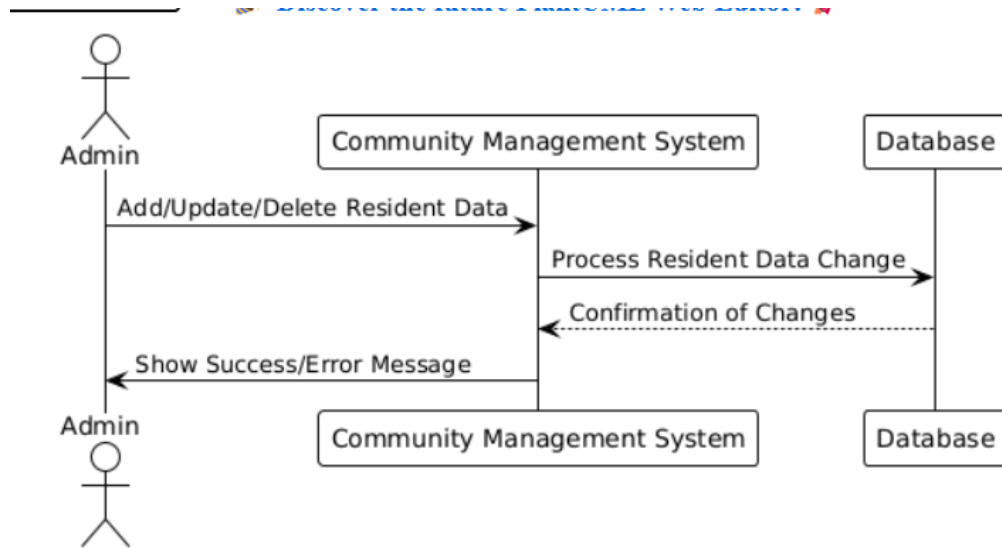
## 6. Use Case Diagram & User Stories



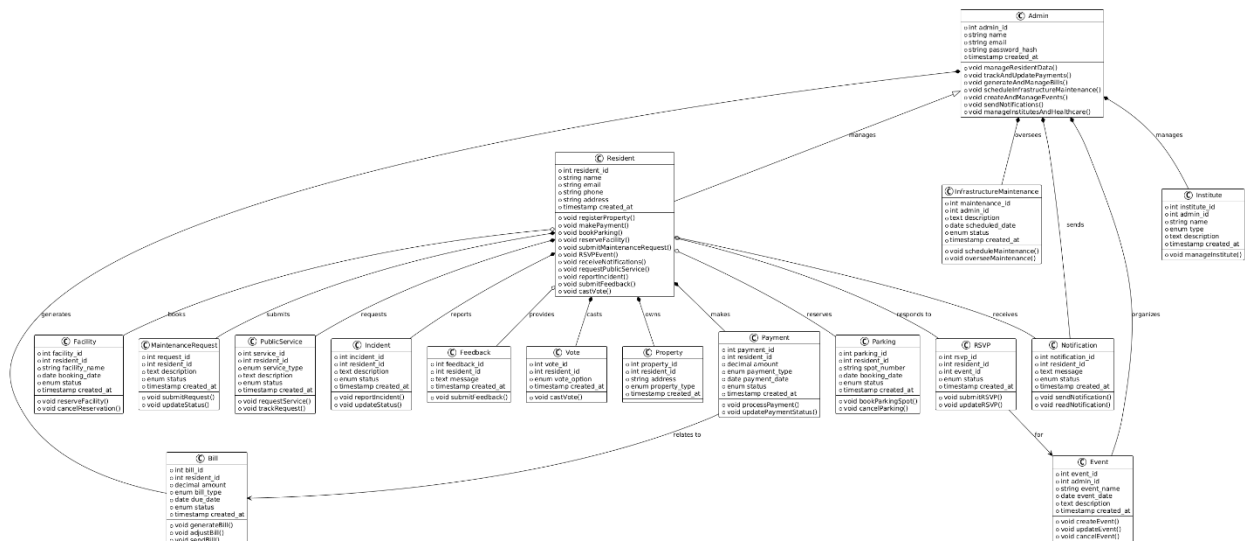
## 7. Sequence Diagrams

- Three key activities visualized.





## 8. Class Diagram



## 9. Product Backlog

The product backlog includes all user stories prioritized based on their importance.

| User Story                          | Priority |
|-------------------------------------|----------|
| Register Property                   | High     |
| Manage Resident Data                | High     |
| Make Payment                        | High     |
| Track & Update Payments             | High     |
| Manage Bills                        | High     |
| Book Parking & Reserve Facility     | Medium   |
| Submit Maintenance Request          | Medium   |
| Schedule Infrastructure Maintenance | Medium   |
| RSVP & Manage Events                | Low      |
| Send & Receive Notifications        | Low      |

## 10. Sprint Backlog

### Sprint 1 Backlog

| User Story              | Assigned To | Tasks                                | Milestones   |
|-------------------------|-------------|--------------------------------------|--|
| Register Property       | Haziq       | Database schema, API, UI             | March 16: Backend,<br>March 18: UI,<br>March 20: Testing |
| Manage Resident Data    | Muneeb      | Schema, API, UI                      | March 15: API,<br>March 17: UI,<br>March 19: Testing     |
| Make Payment            | Ahmed       | Payment API, UI, Security            | March 16: Backend,<br>March 18: UI<br>March 20: Security |
| Track & Update Payments | Haziq       | Database, Admin panel, Notifications | March 17: Backend<br>March 19: Testing                   |
| Manage Bills            | Muneeb      | Billing system, API, UI              | March 15: Backend<br>March 18: UI<br>March 20: Testing   |

**Sprint 2 Backlog (Subset of User Stories)**

| User Story                          | Assigned To | Tasks   | Milestones   |
|-------------------------------------|-------------|---|--|
| Book Parking & Reserve Facility     | Ahmed       | Reservation System, UI, API                       | March 19: Backend<br>March 20: UI<br>March 21: Testing   |
| Submit Maintenance Request          | Haziq       | Request submission, File uploads, Status tracking | March 21: Backend,<br>March 21: UI,<br>March 21: Testing |
| Schedule Infrastructure Maintenance | Muneeb      | Admin scheduling UI, Notifications                | March 22: Backend,<br>March 22: UI<br>March 22: Testing  |

## 11. Meeting Minutes

---

**Project:** Communi3y – Community Management System (CMS)

**Meeting Title:** Functional and Technical Requirements Review

**Date:** 15 MARCH 2025

**Time:** 8:30 pm

**Location:** Virtual

**Chairperson:** Muneeb ul Islam, Product Owner

**Recorder:** Ahmed Mustafa, Scrum Master

**Attendees:**

- 1) Ahmed (Developer, Scrum Master)
  - 2) Muneeb (Developer, Product Owner)
  - 3) Haziq (Developer, Scrum Team)
- 

### a. Agenda

1. Review of SRS Document
  2. Clarification of Roles & Responsibilities
  3. Discussion on Sprint 1 and Sprint 2 Backlog
  4. Confirmation of Design Artifacts (Use Case, Sequence, Class Diagrams)
  5. Roadmap Review and Action Items
- 

### b. Meeting Notes:

- Reviewed the document and project purpose: A web-based CMS designed to facilitate digital community management (property registration, service requests, incident reporting, etc.).

- Reiterated the use of IEEE SRS format and audience scope (Developers, Admins, Residents).
- Confirmed product scope: Includes digital voting, payment, reservations, and feedback systems.
- Technologies finalized:
  - **Frontend:** HTML, CSS, React.js (planned post-Sprint 3)
  - **Backend:** Node.js
  - **Database:** MySQL
- Two main user classes: **Residents** and **Admins**
- System will be hosted locally but must support cloud scalability.

### C. Sprint Planning

- **Sprint 1 Stories Reviewed:**
  - *Register Property, Manage Resident Data, Make Payment, Track Payments, Manage Bills*
  - Tasks assigned with clear milestones. Developers confirmed task deadlines.
- **Sprint 2 Stories Reviewed:**
  - *Book Parking, Submit Maintenance Request, Schedule Infrastructure Maintenance*
  - Emphasis placed on testing and UI readiness by March 22.

### d. Non-Functional & Compliance Requirements

- Agreed on:
  - System must support **1000+ concurrent users**
  - Real-time transaction capability

---

### e. Decisions Made

- React.js integration will be postponed until end of Sprint 3, meanwhile use HTML and CSS for testing the frontend purposes.
- Documentation (User Manual & API Docs) to be created alongside system features.



- Working on Backend using Node.js and MySQL for database.
- Digital voting and feedback systems and other user stories marked for Sprint 3 due to lower priority.

---

#### 4) Action Items

| Action Item                        | Responsible   | Deadline |
|------------------------------------|---------------|----------|
| Update Class and Sequence Diagrams | Haziq, Ahmed  | March 15 |
| Begin Sprint 2 Development         | All Devs      | March 19 |
| Finalize UI/UX designs             | Ahmed         | March 22 |
| Backend and DB setup               | Muneeb. Haziq | March 16 |

---

**Next Meeting:** 15 April 2025

**Objective:** Review Sprint 3 progress and initial testing feedback.

---

## 12. Version Control & Contribution Evidence

- [https://github.com/Mustafaahmed10/SE\\_Project](https://github.com/Mustafaahmed10/SE_Project)
- <https://trello.com/b/CmcUFss2/community-managment-system>
- ScreenSHots:

