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

1. Introduction

1.1 Purpose

The **Community Management System (CMS)** is a web-based application designed to streamline property management, public service requests, digital voting, 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 **Next.js (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 **Next.js**.
- 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

1. **Register Property**
   As a resident, **I want to** register my property in the system, **So that** I can be recognized as a property owner and access related services.

   Pre-conditions:

   - The resident has a valid account.
   - The system allows property registration.

   Post-conditions:

   - The property is successfully registered.
   - The resident receives confirmation.

2. **Manage Resident Data**
   As an admin,**I want to** add, update, and manage resident information,**So that** records remain accurate and up to date.

   Pre-conditions:

   - The user has admin privileges.

   Post-conditions:

   - Resident data is updated in the system.

3. **Make Payment**
   As a resident,**I want to** pay society fees, utility bills, and other charges,**So that** I can fulfill my financial obligations.

   Pre-conditions:

   - The user has a valid payment method.

   Post-conditions:

- Payment is processed, and a receipt is generated.

4. **Track & Update Payments**
   As an admin,**I want to** monitor and update residents' payment records,**So that** I can ensure financial accuracy.

   Pre-conditions:

   - The user has admin privileges.

   Post-conditions:

   - Payment records are updated.

5. **Manage Bills**
   As an admin,**I want to** generate, adjust, and send utility and maintenance bills,
   So that residents receive accurate billing.

   Pre-conditions:

   - The billing module is functional.

   Post-conditions:

   - Bills are generated and sent.

6. **Book Parking & Reserve Facility**
   As a resident**I want to** book parking spots and recreation facilities,
   So that I can use shared community resources.

   Pre-conditions:

   - Availability of parking or facility is checked.

   Post-conditions:

   - Booking confirmation is provided.

7. **Submit Maintenance Request**
   As a resident,
   I want to request maintenance services,
   So that property issues can be resolved.

   Pre-conditions:

   - The user has a valid account.

   Post-conditions:

   - A maintenance request is logged.

8. **Schedule Infrastructure Maintenance**
   As an admin,
   I want to schedule and oversee maintenance for public infrastructure,
   So that the community remains well-maintained.

   Pre-conditions:

   - Maintenance teams are available.

   Post-conditions:

   - Maintenance is scheduled and tracked.

9. **RSVP & Manage Events**
   As a resident,
   I want to RSVP for events,
   So that I can participate in community activities.

   Pre-conditions:

   - Events are listed in the system.

   Post-conditions:

   - RSVP confirmation is received.

10. **Send & Receive Notifications**
    As an admin,
    I want to send important notifications,
    So that residents stay informed.

    Pre-conditions:

- Notification channels are operational.

    Post-conditions:

- Notifications are sent and logged.

11. **Request Public Service**
    As a resident,**I want to** request services like garbage collection,
    So that community needs are met.

Pre-conditions:

- The service request feature is functional.

Post-conditions:

- The request is logged and assigned.

12. **Report Crime or Incident**
    As a resident,
    I want to report crimes or incidents,
    So that authorities can respond accordingly.

Pre-conditions:

- The reporting system is available.

Post-conditions:

- Report is submitted and acknowledged.

13. **Submit Public Feedback**
    As a resident,
    I want to provide feedback on community services,
    So that improvements can be made.

Pre-conditions:

- Feedback system is accessible.

Post-conditions:

- Feedback is recorded for review.

14. **Digital Voting**
    As a resident,
    I want to participate in digital voting,
    So that I can have a say in community decisions.

Pre-conditions:

- The voting system is set up.

Post-conditions:

- Vote is submitted and counted.

15. **Manage Institutes & Healthcare**
    As an admin,
    I want to oversee educational and healthcare facilities,
    So that community members receive quality services.

Pre-conditions:

- Institutions are registered in the system.

Post-conditions:

- Management updates are reflected in the system.

5. Nonfunctional Requirements

5.1 Product Requirements

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

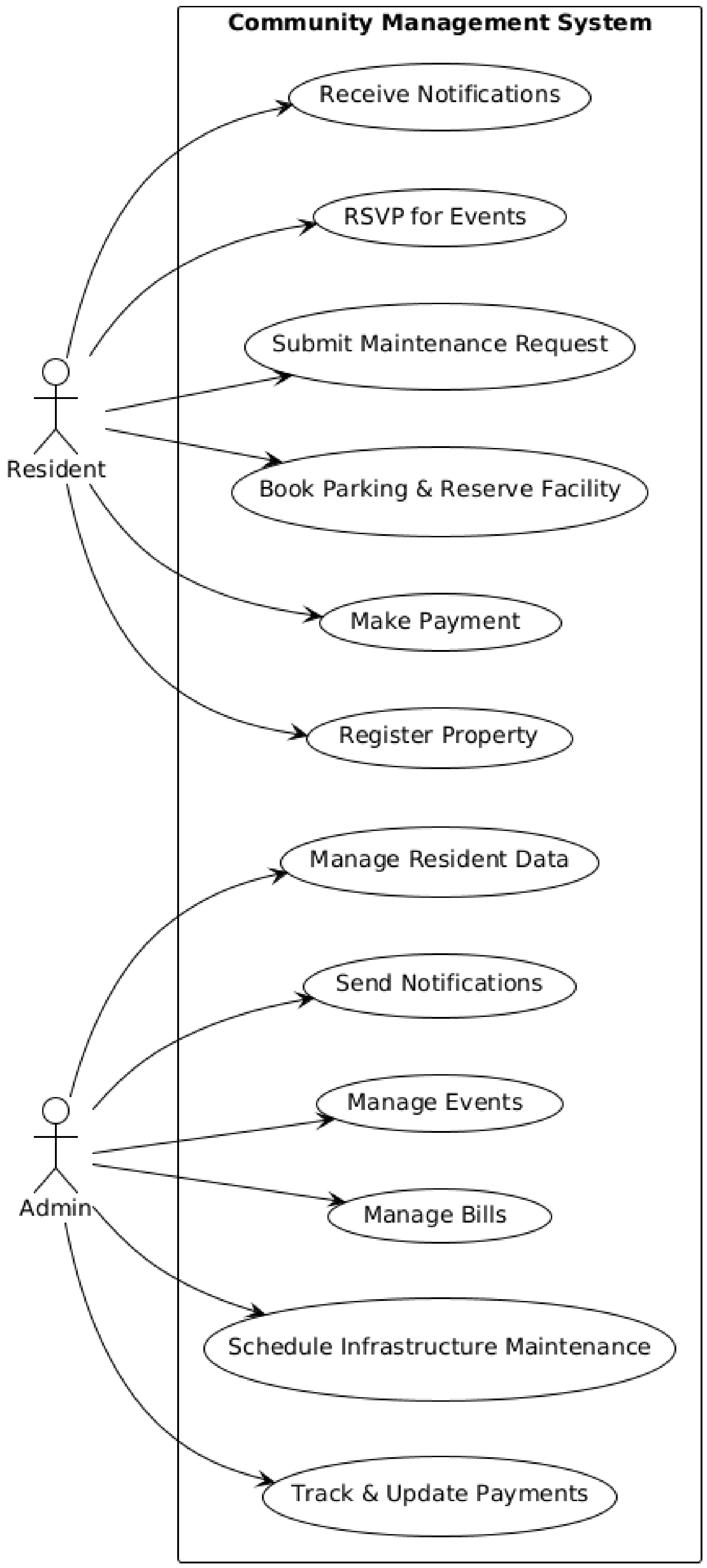
5.2 Organizational Requirements

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

5.3 External Requirements

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

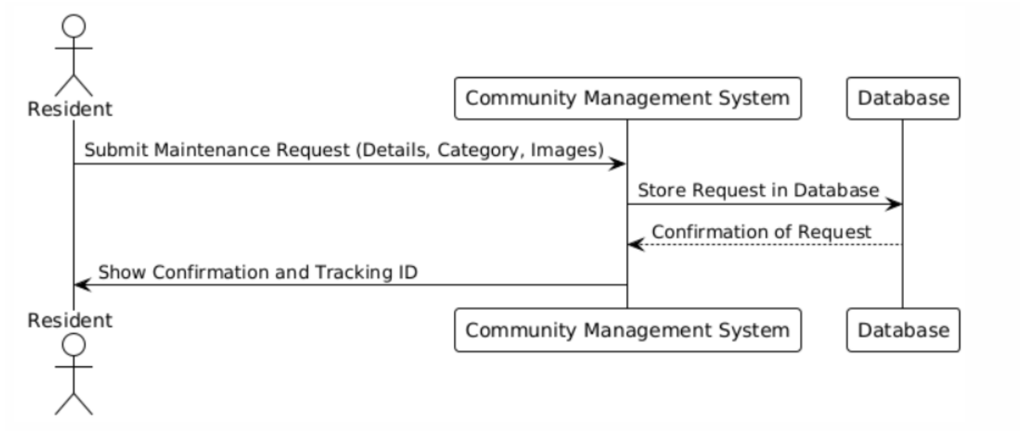
6. Use Case Diagram & User Stories

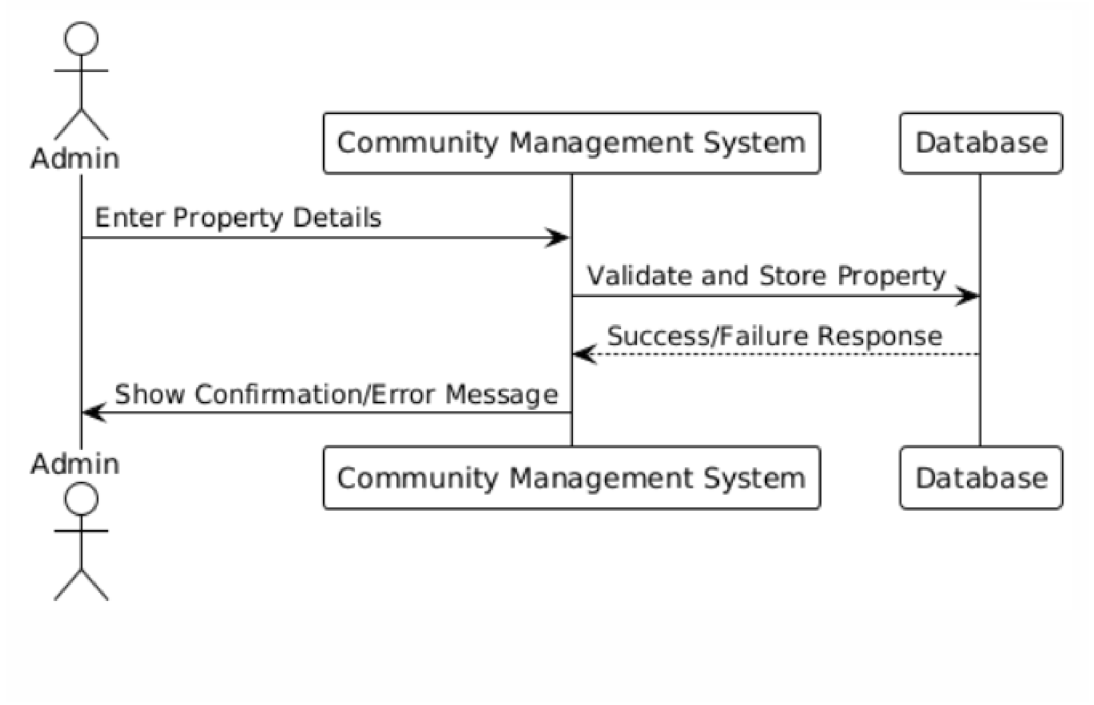

Community Management System
Receive Notifications
RSVP for Events
Submit Maintenance Request
Book Parking & Reserve Facility
Make Payment
Register Property
Manage Resident Data
Send Notifications
Manage Events
Manage Bills
Schedule Infrastructure Maintenance
Track & Update Payments
Resident
Admin

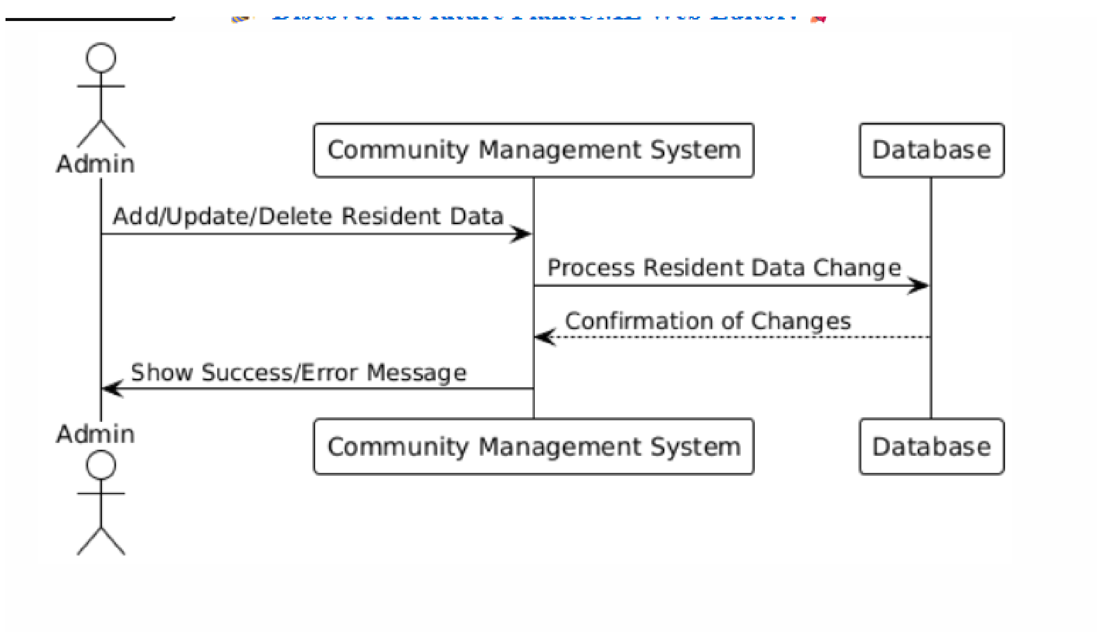
-

7. Sequence Diagrams

- **Three key activities visualized.**

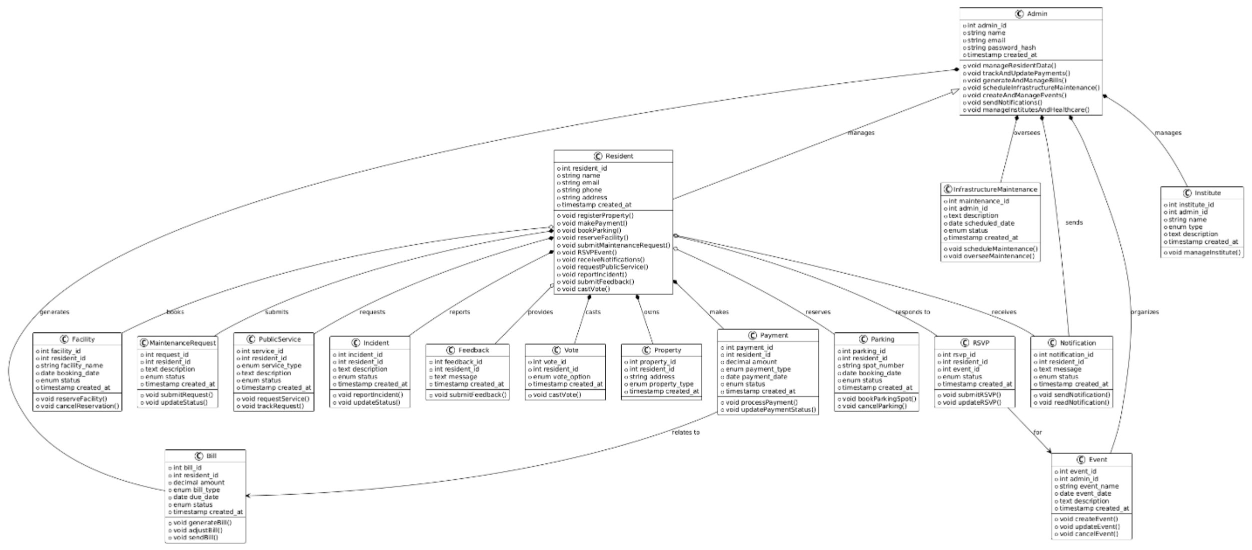

Resident
Community Management System
Database
Submit Maintenance Request (Details, Category, Images)
Store Request in Database
Confirmation of Request
Show Confirmation and Tracking ID
Resident
Community Management System
Database

- 
Admin
Community Management System
Database
Enter Property Details
Validate and Store Property
Success/Failure Response
Show Confirmation/Error Message
Admin
Community Management System
Database


Admin
Community Management System
Database
Add/Update/Delete Resident Data
Process Resident Data Change
Confirmation of Changes
Show Success/Error Message
Admin
Community Management System
Database

•

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, March 18: UI, March 20: Testing
Manage Resident Data	Muneeb	Schema, API, UI	March 15: API, March 17: UI, March 19: Testing
Make Payment	Ahmed	Payment API, UI, Security	March 16: Backend, March 18: UI March 20: Security
Track & Update Payments	Haziq	Database, Admin panel, Notifications	March 17: Backend March 19: Testing
Manage Bills	Muneeb	Billing system, API, UI	March 15: Backend March 18: UI March 20: Testing

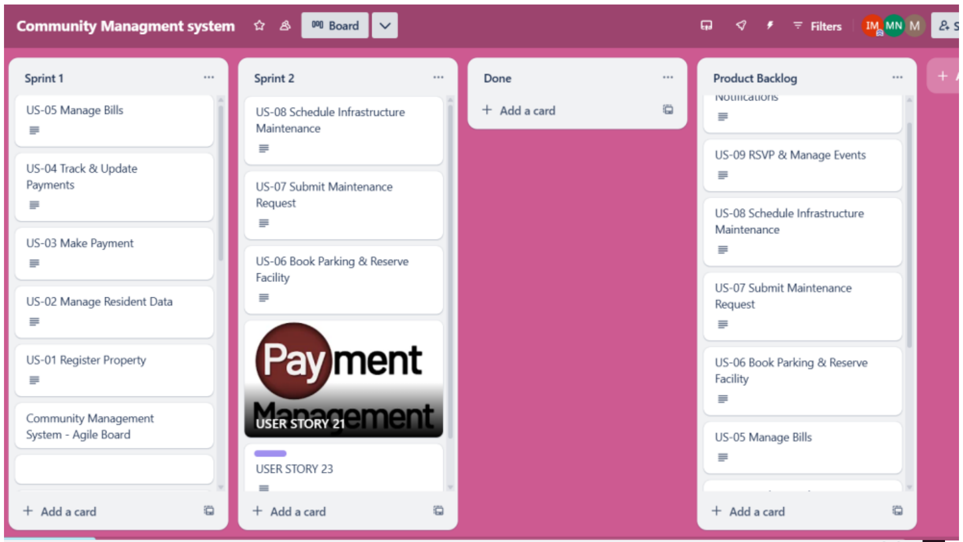
Sprint 2 Backlog (Subset of User Stories)

User Story	Assigned To	Tasks	Milestones
Book Parking & Reserve Facility	Ahmed	Reservation System, UI, API	March 19: Backend March 200: UI March 21: Testing
Submit Maintenance Request	Haziq	Request submission, File uploads, Status tracking	March 21: Backend, March 21: UI, March 21: Testing
Schedule Infrastructure Maintenance	Muneeb	Admin scheduling UI, Notifications	March 22: Backend, March 22: UI March 22: Testing

•

11. Version Control & Contribution Evidence

- https://github.com/Haziq739/SE-Assignment-01
- https://trello.com/b/CmcUFss2/community-managment-system



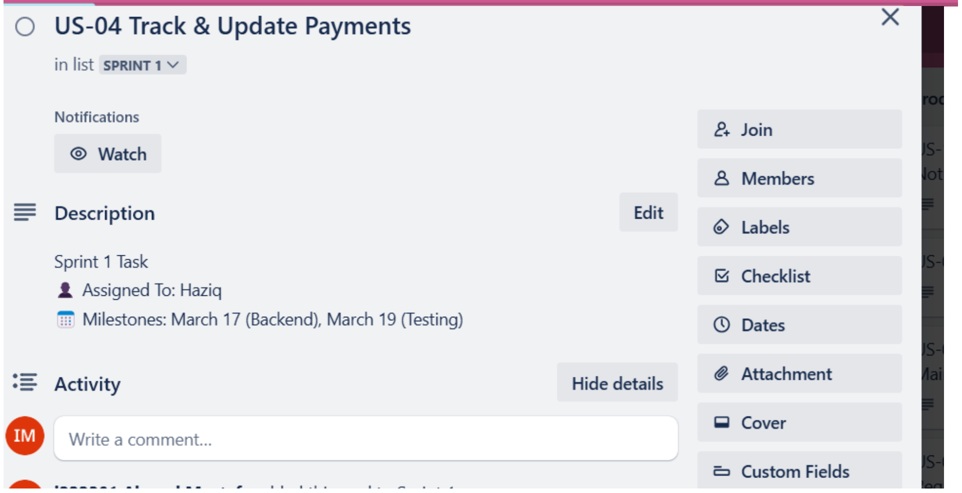
Sprint 1
- US-05 Manage Bills
- US-04 Track & Update Payments
- US-03 Make Payment
- US-02 Manage Resident Data
- US-01 Register Property
- Community Management System - Agile Board

Sprint 2
- US-08 Schedule Infrastructure Maintenance
- US-07 Submit Maintenance Request
- US-06 Book Parking & Reserve Facility
- Payment Management USER STORY 21
- USER STORY 23

Done

Product Backlog
- Notifications
- US-09 RSVP & Manage Events
- US-08 Schedule Infrastructure Maintenance
- US-07 Submit Maintenance Request
- US-06 Book Parking & Reserve Facility
- US-05 Manage Bills

•



US-04 Track & Update Payments

in list SPRINT 1

Description

Sprint 1 Task
Assigned To: Haziq
Milestones: March 17 (Backend), March 19 (Testing)

Activity

Write a comment...