**CommUnity: Seamless Community Interaction and Management**

PROJECT-1

Team Members

* Atul Kumar Nayak
* Swathi Pyla

**Abstract Problem Statement**

CommUnity is a modular, API-centric, multi-tenant Community Management System designed to simplify housing society management by connecting members and streamlining daily activities. It provides an efficient platform for both administrators and residents to interact and manage community affairs.

For administrators, CommUnity offers tools for account management, approving new residents, posting notices, generating and tracking maintenance bills, and managing society profiles. Admins can handle complaints, configure emergency contact lists, maintain an event calendar, and review feedback. They can send out weekly digests, organize interest-based clubs, conduct polls or surveys, update vacant flat information, and even post commercial content. Admins also track resident engagement through the 'I am interested' poll for commercial posts.

For residents, CommUnity facilitates communication with fellow members, providing access to a digital noticeboard for important announcements, and enables online payment of maintenance bills. Residents can raise complaints, view emergency contacts, manage their profiles, check event calendars, give feedback on events, read weekly digests, participate in polls, join interest-based clubs. Residents can also post personal recommendations or share interesting content using the tag-along feature.

**Functional Requirements**

**1. User Management**

User Registration: New users (residents, non-residents, and administrators) can register on the platform.

Admin Approval: Registered residents require admin approval to join a community.

Profile Management: Users can view and update their personal details.

**2. Noticeboard and Notifications**

Noticeboard Access: Residents can view all past and current notices, circulars, and announcements on a digital noticeboard.

WhatsApp Notifications: Residents receive WhatsApp notifications for important notices and announcements.

**3. Maintenance and Payment Management**

Bill Generation: Admins can generate maintenance bills for residents.

Online Payments: Residents can view their bills and make payments via Credit or Debit cards.

Payment Tracking: Admins can track bill payments and identify overdue accounts.

Payment Receipts: Residents receive instant payment receipts upon successful transactions.

**4. Complaint and Feedback Management**

Raise Complaints: Residents can submit complaints at the community or personal level.

Complaint Tracking: Residents can track the status of their raised complaints.

Complaint Resolution: Admins can manage and resolve raised complaints.

Event Feedback: Residents can submit feedback on events organized by the community.

Handle Feedback: Admins can review feedback from residents and use it for improvements.

**5. Event Management**

Event Calendar: Admins can create and update an event calendar for residents to track upcoming events.

Access Event Calendar: Residents can view the event calendar and stay updated on future events.

Weekly Digest: Admins can send weekly summaries of the past week’s events and upcoming activities to residents.

**6. Voting and Polls**

Conduct Polls/Surveys: Admins can create and conduct polls or surveys to gather residents’ opinions.

Poll Participation: Residents can vote in polls and view the results once available.

**7. Club and Sub-group Management**

Interest-Based Clubs: Admins can create and manage interest-based clubs (e.g., gardening, yoga).

Join Clubs: Residents can join clubs and participate in club-specific discussions.

**8. Commercial Posts and Recommendations**

Admin Commercial Posts: Admins can post commercial content visible to all residents.

Resident Recommendations: Residents can post personal recommendations or share interesting content using the tag-along feature.

'I Am Interested' Poll: Residents can engage with commercial posts by marking interest via a poll.

**9. Emergency Contact Management**

Configure Emergency Contacts: Admins can maintain and update a list of local emergency contacts (e.g., police, ambulance, pharmacy).

Access Emergency Contacts: Residents can view and access emergency contact details at any time.

**10. Security Service Functionality**

The system shall provide a module where admins can allocate security personnel to specific blocks within the community.

Residents can view the details (name, contact) of security personnel assigned to their block.

**Non-Functional Requirements**

**1.Security**

Basic Encryption: Sensitive data must be encrypted in transit using HTTPS to ensure privacy and data protection.

Role-Based Access: The system should implement role-based access control, allowing only authorized users to perform certain actions or access specific features.

**2. Usability**

User Interface: The application should be intuitive and easy to navigate, with a clean and simple design optimized for desktop use.

Responsiveness: The application should be fully responsive, ensuring a consistent and seamless user experience across various devices and screen sizes.

Basic Accessibility: The application should cater to users with varying levels of technical ability, incorporating clear layouts and accessible features.

**3. Maintainability**

Code Simplicity: Code should be modular, well-organized, and thoroughly documented to facilitate easy updates, troubleshooting, and future enhancements.

Automated Testing: Implement basic automated tests to ensure the stability of core functionalities and ease of future code modifications.

**4. Scalability**

Basic Scalability: The system should be designed to scale horizontally, allowing for the addition of more users and communities without significant performance degradation. The infrastructure should be easily expandable to handle increased loads.

Elasticity: The system should be capable of dynamically allocating resources to meet demand, ensuring consistent performance during peak usage.

**5. Interoperability**

API Integration: The system should expose RESTful APIs for integration with other services, enabling easy communication and data exchange with external systems.

Modular Design: The architecture should support modular components, allowing for the integration of new features or third-party services with minimal disruption.

**Technology Stack**

**Frontend**: React

Develop a simple and responsive user interface using React.

Focus on building a few key pages (e.g., Login, Dashboard, Announcements, Payments).

**Backend**: Java with Spring Boot

Implement RESTful APIs using Spring Boot to handle the core business logic.

Integrate with MongoDB for storing user data, announcements, and payment records.

**Database**: MongoDB

Set up MongoDB to store user profiles, announcements, discussions, and payment records.

Keep the database schema simple and easy to manage.

**Payment Gateway**: Stripe API

Integrate Stripe for handling payments.

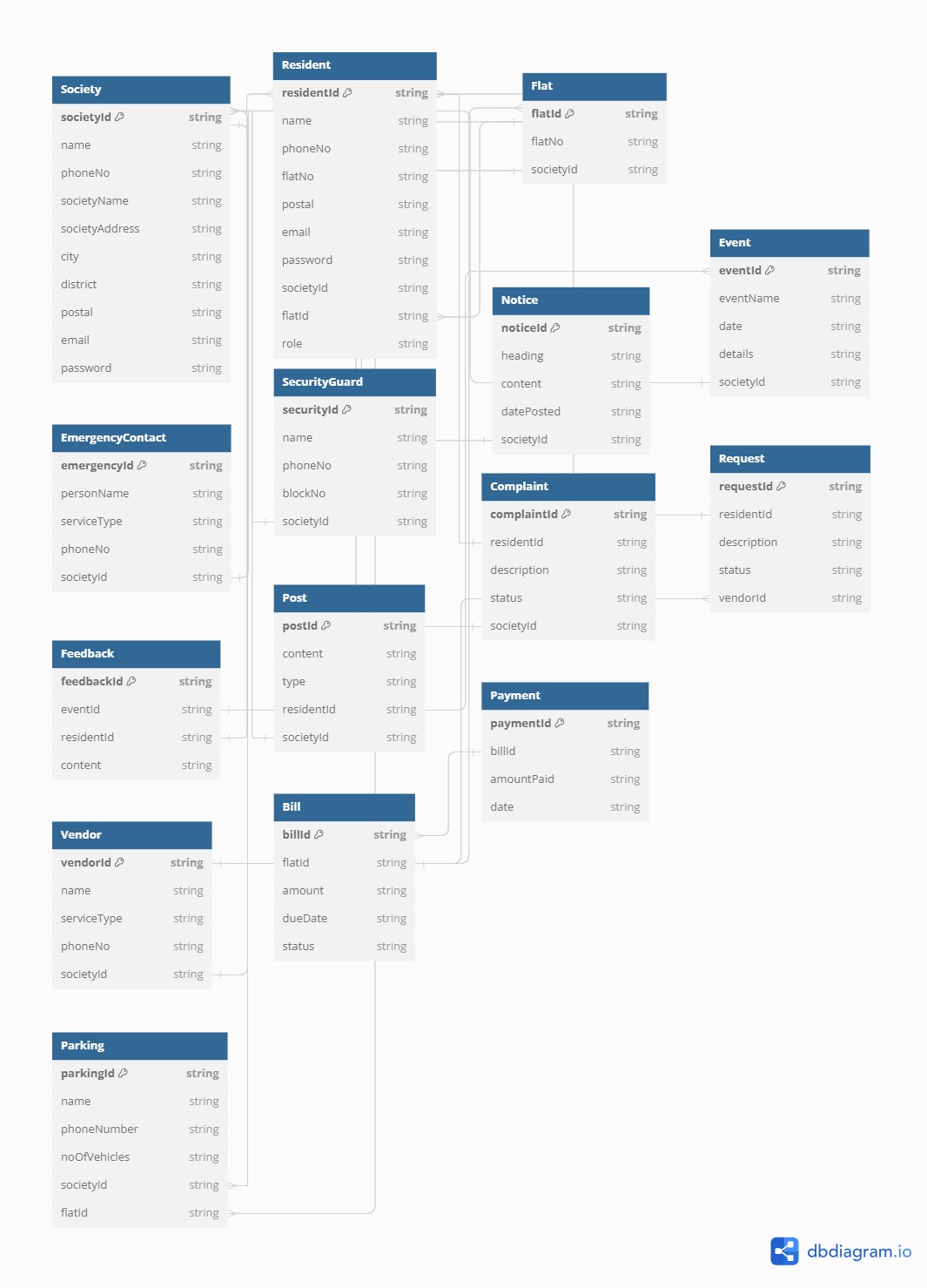
Implement basic payment processing and store transaction details in MongoDB.

**Notification Service**: SNS

A managed service that enables you to send notifications to users via multiple protocols,

including SMS, email, and mobile push notifications.

**Database Schema**



**Microservices**

**1.** **Society Management Service**

Entities: Society, Flat, Resident, SecurityGuard, EmergencyContact, Parking

Responsibilities:

Manage societies, flats, and resident profiles.

Handle authentication and authorization.

Manage security personnel, emergency contacts, and parking assignments for societies.

**2. Noticeboard, Event & Post Service**

Entities: Notice, Event, Feedback, Post

Responsibilities:

Manage notices, events, and posts.

Collect and manage feedback related to events.

**3. Complaint & Request Service**

Entities: Complaint, Request, Vendor

Responsibilities:

Manage resident complaints, requests, and vendor assignments.

**4. Billing & Payment Service**

Entities: Bill, Payment

Responsibilities:

Manage billing and payment processing for flats.