**Business Requirements Document (BRD)**

**Project Title:** CommUnity: Seamless Community Interaction and Management

**1.Project Overview**

CommUnity is a modular, multi-tenant community management system designed to simplify housing society management. The platform connects members, streamlines daily activities, and enhances community engagement. This platform provides an efficient interface for both administrators and residents, management, facilitating interaction, and engagement within the community.

Community offers tools for administrators to manage accounts, post notices, generate and track maintenance bills, and manage society profiles. Additionally, admins can handle complaints, configure emergency contacts, maintain event calendars, and review feedback. For residents, Community enables communication with fellow members, access to a digital noticeboard, online bill payments, complaint submissions, event participation, and more.

**2.Business Objectives**

* **Streamline Society Operations:** Enable administrators to manage residents, notices, and events efficiently.
* **Improve Communication:** Bridge gaps between residents and society administrators.
* **Encourage Participation:** Faster community interaction through events and discussions.
* **Transparent Financial Management:** Simplify maintenance fee collection and tracking.
* **Ensure Safety:** Provide easily accessible emergency and security contacts.
* **Enhance Resident Engagement:** Increase resident participation in community

Activities through events, polls, clubs, and social interactions.

**3. Stakeholders**

**Primary Stakeholders:**

* **Residents**: End-users who utilize the platform for communication, payments, and complaints.
* **Society Administrators**: Manage notices, events, payments, and complaints on the platform.
* **Security Personnel:** Provide and maintain security information.

**Secondary Stakeholders:**

* **Vendors/Service Providers:** Handle maintenance and service requests.
* **Payment Gateway Providers:** Process online payments securely (e.g., Razorpay)

.

**4. Functional Requirements**

**1. User Management**

* **Registration:** Admins and residents can register with unique credentials.
* **Authentication:** Secure login with JWT-based tokens.
* **Profile Management:** Users can view their personal details, Residents and admins can update their profiles.

**2. Noticeboard & Notifications**

* **Notice Creation:** Admins can post, update, and delete notices.
* **Event Management:** Plan, publish, and update events.
* **Resident Feedback:** Allow residents to provide feedback on events.

**3. Maintenance & Payment Management**

* **Bill Generation:** Admins can generate maintenance bills for residents.
* **Payment History:** Residents can view past transactions; admins can track payment status.
* **Payment Gateway Integration:** Secure payment processing via Razorpay.
* **External API:** We are using Razorpay API for making payment and transactions.

**Link:** [**https://api.razorpay.com/v1**](https://api.razorpay.com/v1)

**4. Complaint & Feedback Management**

* **Raise Complaints:** Residents can submit complaints at the community or personal level.
* **Complaint Tracking:** View real-time status and assigned personnel for resolution.
* **Complaint Resolution:** Admins can manage and resolve raised complaints.
* **Event Feedback:** Residents can submit feedback on community events.
* **Handle Feedback:** Admins review and use feedback for improvements.

**5. Event Management**

* **Event Publishing:** Admins can create events with descriptions, dates, and details.
* **Access Events:** Residents can view the event calendar and stay updated on future events.
* **External API:**

**Link:** [**https://cloudinary.com**](https://cloudinary.com)

**6. Commercial posts & Recommendations**

* **Service Advertisements:** Allow local service providers to advertise their services.
* **Resident Recommendations:** Enable residents to recommend local services or vendors.

**7.** **Emergency Contact Management:**

* **Emergency Contacts:** Store and display essential numbers (e.g., police, plumber)
* **Resident Access:** Ensure all residents have access to the emergency list.

**8. Security Service Functionality**

* **Security Details:** Record and display security staff details and schedules.
* **Visitor Management:** Optional integration for tracking visitor logs.

**5. Non- Functional Requirements**

* **Security:** Enforce secure authentication and data encryption.
* **Usability:** Intuitive and user-friendly interfaces.
* **Maintainability:** Modular architecture for scalability and easy maintenance.

**6. Project Architecture**

**Components:**

**Frontend**

* **React.js:** For building the user interface.

Utilized as the core framework for building dynamic and responsive user interfaces.

* **Tailwind CSS:** For styling with a utility-first approach.

Integrated for rapid and efficient styling with a utility-first approach.

* **Material UI:** For pre-designed components.

Ensures consistency in design elements such as buttons, modals, and navigation bars.

* **Redux**: For state management.

Used for state management, enabling seamless data flow across components.

**Backend**

* **Spring Boot:** For server-side application development.

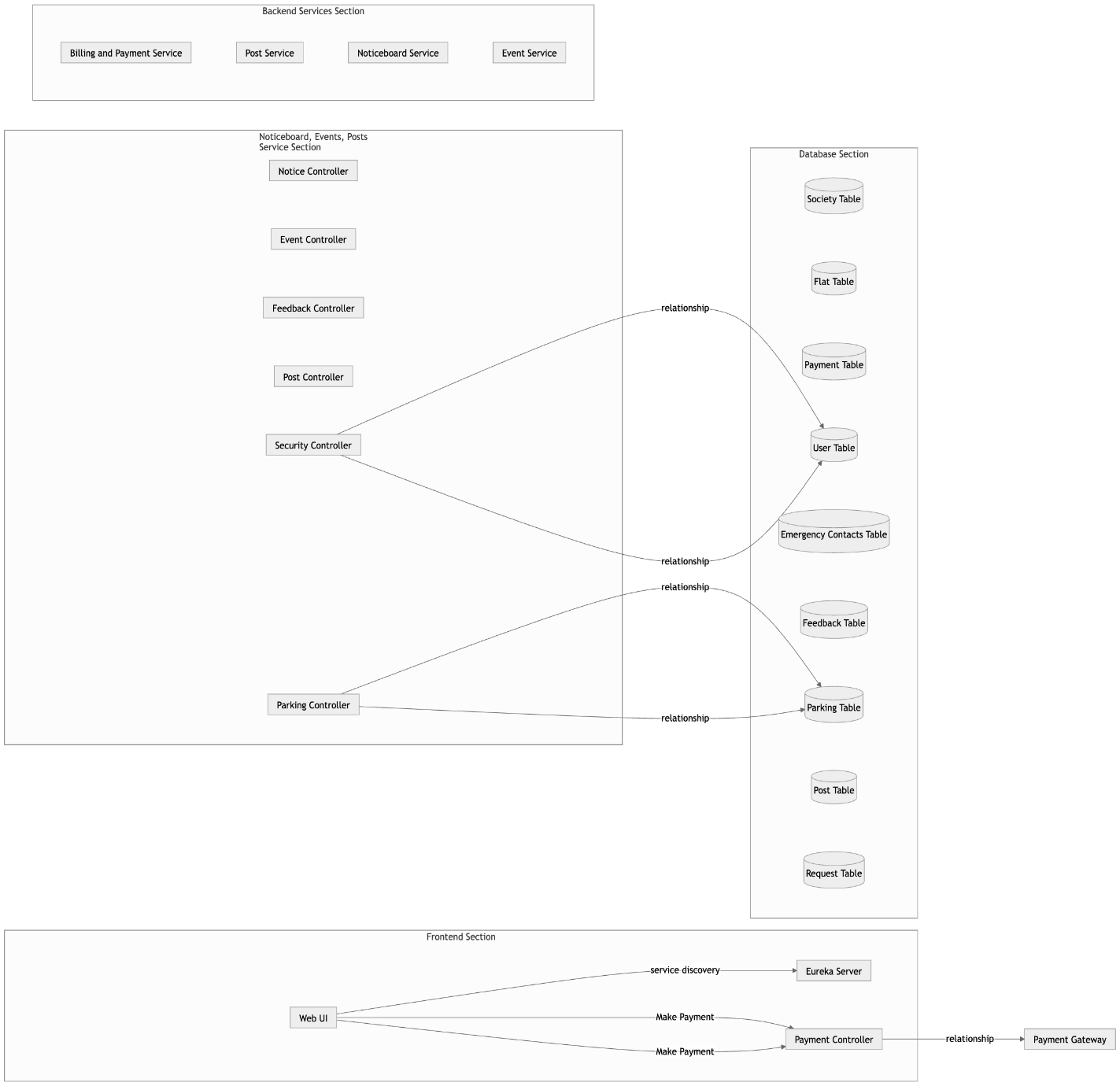
Offers microservices support, RESTful APIs, and easy integration with third-party libraries.

Simplifies backend development with pre-configured setups.

**Database**

* **MySQL**: For data storage and management.

Acts as the relational database management system for storing structured data, including resident profiles, society details, events, and payment records.



**7. Database Schema Overview**

**Key Tables:** This application will include the following databases.

1. **Society:**

* **Attributes:** societyId, name, phoneNo, societyName, societyAddress, city, district, postal, email.
* **Description:** Represents the details of a society. It serves as the primary table connecting other entities like residents, flats, security, etc.

1. **Resident:**

* **Attributes:** residentId, name, phoneNo, flatNo, postal, email, societyId, role.
* **Description:** Contains details of residents, including their contact and role in the society.

1. **Flat:**

* **Attributes:** flatId, flatNo, societyId.
* **Description:** Stores information about individual flats.

1. **Emergency Contact:**

* **Attributes:** emergencyId, personName, serviceType, phoneNo, societyId.
* **Description:** Keeps emergency contact details relevant to the society.

1. **Event:**

* **Attributes:** eventId, eventName, eventDate, eventDetails, eventImage, societyId.
* **Description:** Manages event-related information for the society.

1. **Feedback:**

* **Attributes:** feedbackId, eventId, content.
* **Description:** Collects feedback for specific events.

1. **Notice:**

* **Attributes:** noticeId, heading, content, datePosted, noticeImage, societyId.
* **Description:** Contains public notices posted within the society.

1. **Request:**

* **Attributes:** requestId, address, description, status, vendorId.
* **Description:** Tracks requests raised by residents for services.

1. **Vendor:**

* **Attributes:** vendorId, name, company, email, service, phoneNo, societyId.
* **Description:** Holds vendor information who provide services to the society.

1. **Complaint:**

* **Attributes:** complaintId, residentId, flat\_no, person\_name, title, description, status, societyId.
* **Description:** Tracks complaints lodged by residents.

1. **Parking:**

* **Attributes:** parkingId, flat\_no, parking\_no, flat\_id, societyId.
* **Description:** Manages parking information for residents and flats.

1. **Security Details:**

* **Attributes:** securityId, name, phoneNo, blockNo, societyId.
* **Description:** Tracks security personnel assigned to blocks.

1. **Post:**

* **Attributes:** postId, content, title, likeCount, postImage, societyId.
* **Description:** Captures posts shared within the society.

1. **Payment:**

* **Attributes:** paymentId, flatNo, societyId, amount, status, paymentDate.
* **Description:** Manages payments related to flats in the society.



**8. Assumptions and Constraints**

**Assumptions:**

* All residents have access to smartphones or computers.
* Internet connectivity is available to all users.
* Payment gateway services like Razorpay are pre-approved.

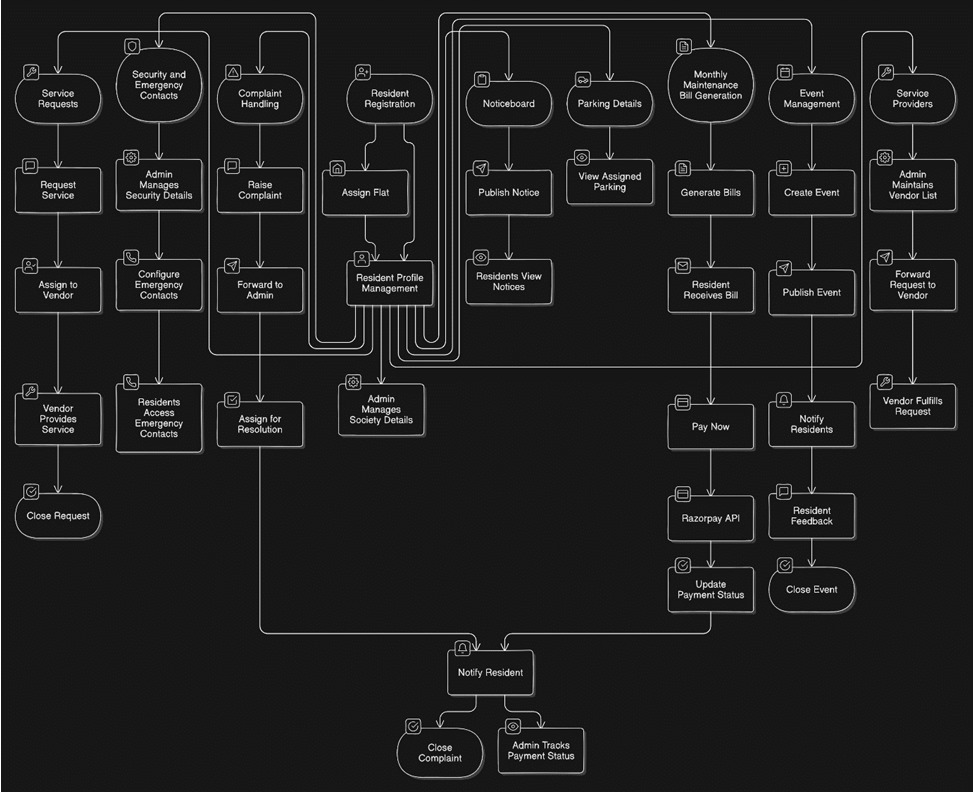
**Constraints:**

* Data privacy must adhere to local regulations.
* The platform must support multi-tenancy for different societies.

**9. Data Requirements**

* **User Data:** Includes user authentication details, profile information, contact details, and roles. This data is essential for user management, authentication, and authorization.
* **Society Data:** Information related to societies, including society details, emergency contacts, security personnel, parking information, and notices.
* **Resident Data:** Profiles of residents, including personal details, flat assignments, and roles within the society.
* **Flat Data:** Details about flats, including and their association with societies and residents.
* **Notice Data:** Notices posted by societies, including content, headings and posting dates.
* **Feedback Data:** Feedback from residents related to events and other aspects of the society.
* **Post Data:** Posts made by residents, including content and type of post.
* **Complaint Data:** Records of complaints raised by residents, including descriptions, statuses, and related societies.
* **Request Data:** Tracks service requests like maintenance, cleaning, or other society-related tasks.
* **Vendor data:** Stores details of vendors or service providers engaged by the society for maintenance and other services.
* **Bill Data:** Manages records of generated bills for residents, including maintenance fees and additional charges.
* **Payment Data:** Tracks payments made by residents for maintenance and other charges.
* **Parking Data:** Captures information about parking slots assigned to residents.

**10. Workflow Diagram:**

****