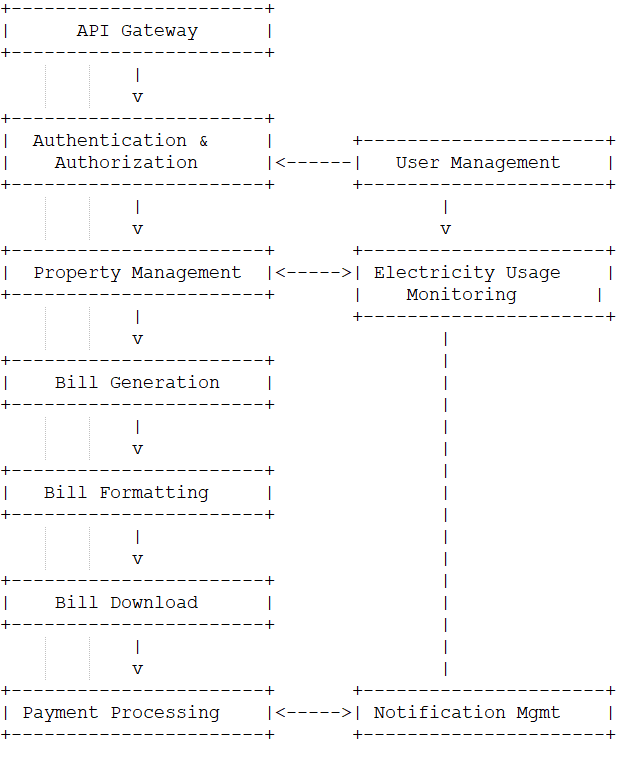
**PayMaytrix --- High-Level Design (HLD)**

**Microservices Architecture:  
  
**

**Components:**

1. **API Gateway (Zuul):**
   * Acts as an entry point for all incoming requests.
   * Handles authentication and routing to appropriate microservices.
   * Provides a single entry point for the client-side applications.
2. **Authentication & Authorization:**
   * Handles user registration, login, and authentication using JWT.
   * Defines user roles and permissions using an AuthorizationService.
   * Ensures secure access to various features based on user roles.
3. **User Management:**
   * Manages user information and user profiles.
   * Allows users to update their personal details and manage properties.
4. **Property Management:**
   * Provides functionalities to add, view, update, and delete properties associated with users.
5. **Electricity Usage Monitoring:**
   * Monitors real-time electricity usage using smart meter integration or manual entry.
   * Stores historical usage data for analysis.
6. **Bill Generation:**
   * Generates electricity bills based on usage data and tariff rates.
   * Stores bill information in the database.
7. **Bill Formatting:**
   * Formats bills in PDF format for user convenience.
8. **Bill Download:**
   * Enables users to download their bills in PDF format.
9. **Payment Processing:**
   * Handles payment processing for bill payments using various payment gateways.
   * Ensures secure and reliable transactions.
10. **Notification Management:**
    * Manages notifications for upcoming bill due dates, pending payments, etc.
    * Allows users to mark notifications as read.

**Communication:**

* Microservices communicate with each other via RESTful APIs.
* Microservices register themselves with Eureka for service discovery.
* The API Gateway (Zuul) routes requests to the appropriate microservices.

**Database:**

* Each microservice will have its own database or schema, based on their specific responsibilities.
* PostgreSQL database will be used for storage.

**Deployment:**

* Each microservice will be containerized using Docker for easy deployment.
* Kubernetes can be used for container orchestration and management.

**Security:**

* JWT (JSON Web Tokens) will be used for authentication and authorization.
* Communication between microservices will be secured using HTTPS.
* User passwords will be securely hashed using BCryptPasswordEncoder.

**Scalability:**

* Each microservice can be scaled independently based on demand.
* Spring Cloud Eureka will handle load balancing and service discovery.

**Error Handling and Logging:**

* Global exception handlers will be implemented to handle errors gracefully.
* Application logs will be generated and stored centrally for easy monitoring and debugging.

**Frontend:**

* The frontend can be developed using ReactJS.
* The frontend communicates with microservices through the API Gateway.