**Documentation for Building Shipping Package**

**Overview**

The goal is to implement a shipping package for your multi-tenancy web application. This package will allow vendors to manage their shipping operations through two key features:

1. **Self Shipping**:
   * Vendors can define shipping rates (flat or custom).
   * Vendors can provide manual tracking information.
2. **Automatic Shipping**:
   * Automates shipping operations using third-party integrations:
     + **Interstate/Intercity Deliveries** via **Shiprocket**.
     + **Hyperlocal Deliveries** via **Borzo**.

**Feature Requirements**

**Self Shipping**

* **Flat Rate Option**:
  + Vendors can set a universal flat rate for all their shipments.
* **Custom Rate Rules**:
  + Vendors can define rates based on parameters such as location, weight, or product type.
* **Manual Tracking**:
  + Vendors manually input tracking IDs for each shipment.

**Automatic Shipping**

* **Shiprocket Integration**:
  + **Rate Calculation**: Fetch rates based on delivery location, weight, and dimensions.
  + **AWB (Air Waybill) Generation**: Generate AWB numbers for tracking.
  + **Tracking**: Provide real-time shipment tracking information.
  + Use [Shiprocket API Documentation](https://apidocs.shiprocket.in/).
* **Borzo Integration**:
  + **Delivery Cost Calculation**: Fetch costs for hyperlocal deliveries.
  + **Shipment Request**: Create and schedule same-day or next-day deliveries.
  + **Tracking**: Provide real-time updates for hyperlocal shipments.
  + Use [Borzo API Documentation](https://borzodelivery.docs.apiary.io/).

**Functional Modules**

**1. Admin Panel**

* Manage vendors’ access to shipping features.
* Define subscription tiers (e.g., include self-shipping, automatic shipping, or both).

**2. Vendor Dashboard**

* Configure shipping options:
  + Self Shipping: Set rates and tracking IDs.
  + Automatic Shipping: Integrate with Shiprocket/Borzo.
* View and manage shipments.

**3. Customer Portal**

* Display shipping options (self-shipping and automatic shipping) during checkout.
* Show estimated delivery costs and times.
* Provide tracking information for orders.

**System Architecture**

**Database Design**

**Tables**

1. **ShippingConfig**:
   * id: Primary key.
   * vendor\_id: Foreign key linking to vendors.
   * flat\_rate: Fixed rate for shipments.
   * custom\_rate\_rules: JSON field defining custom rules (e.g., location, weight).
   * use\_shiprocket: Boolean (enable/disable Shiprocket integration).
   * use\_borzo: Boolean (enable/disable Borzo integration).
2. **Orders**:
   * id: Primary key.
   * order\_id: Unique order identifier.
   * shipping\_method: Enum (self, shiprocket, borzo).
   * tracking\_id: Tracking ID for shipment.
   * status: Enum (pending, shipped, delivered, canceled).
3. **Shipments**:
   * id: Primary key.
   * order\_id: Foreign key linking to orders.
   * provider: Enum (self, shiprocket, borzo).
   * awb: Air Waybill for automatic shipping.
   * tracking\_url: URL for tracking the shipment.
   * status: Enum (in transit, delivered, failed).

**Implementation Plan**

**1. Backend Development**

**1.1 API for Self Shipping**

* **Endpoints**:
  + POST /api/self-shipping/rates: Add/update flat or custom rates.
  + POST /api/self-shipping/tracking: Add tracking ID for an order.
  + GET /api/self-shipping/orders: Fetch all self-shipping orders.

**1.2 Integration with Shiprocket**

* **Endpoints**:
  + POST /api/shiprocket/rates: Fetch rates from Shiprocket API.
  + POST /api/shiprocket/awb: Generate AWB for an order.
  + GET /api/shiprocket/track: Fetch tracking status.
* **Steps**:
  + Authenticate using Shiprocket API credentials.
  + Fetch rates based on delivery parameters.
  + Generate AWB and save it to the database.
  + Provide tracking URLs to customers.

**1.3 Integration with Borzo**

* **Endpoints**:
  + POST /api/borzo/cost: Fetch delivery cost from Borzo API.
  + POST /api/borzo/schedule: Schedule a delivery.
  + GET /api/borzo/track: Fetch tracking status.
* **Steps**:
  + Authenticate using Borzo API credentials.
  + Fetch delivery cost for hyperlocal shipments.
  + Schedule deliveries and store tracking information.
  + Provide real-time updates to customers.

**2. Frontend Development**

**Vendor Dashboard**

1. **Shipping Configuration**:
   * Page to configure self-shipping and automatic shipping settings.
   * Form inputs for flat/custom rates and API credentials for third-party integrations.
2. **Manage Shipments**:
   * Table view to list all orders with shipment details (method, tracking ID, status).

**Customer Portal**

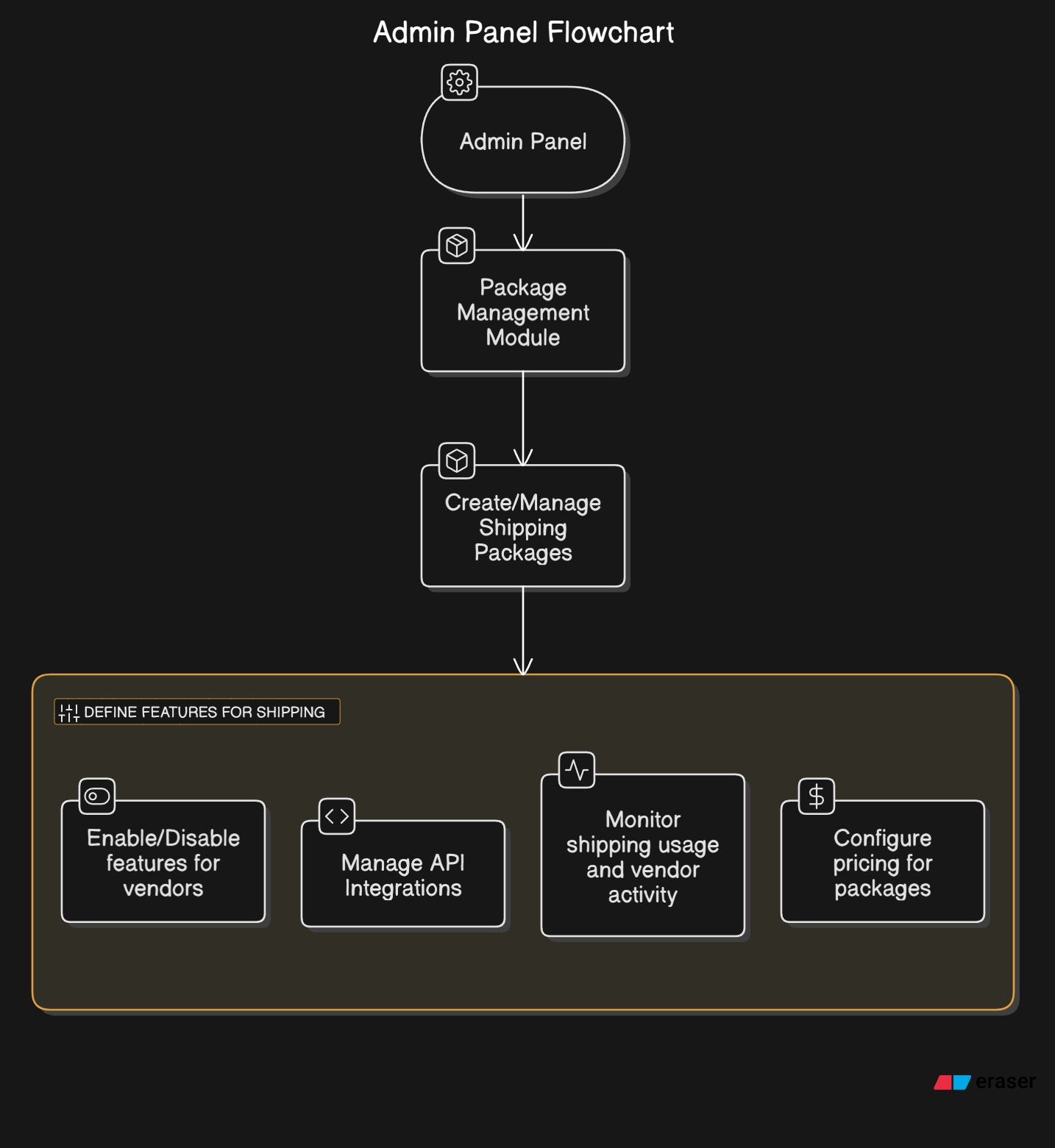
1. **Checkout Page**:
   * Display available shipping methods (self-shipping, Shiprocket, Borzo).
   * Show estimated costs and delivery times.
2. **Order Tracking**:
   * Provide tracking information with URLs for automatic shipping.

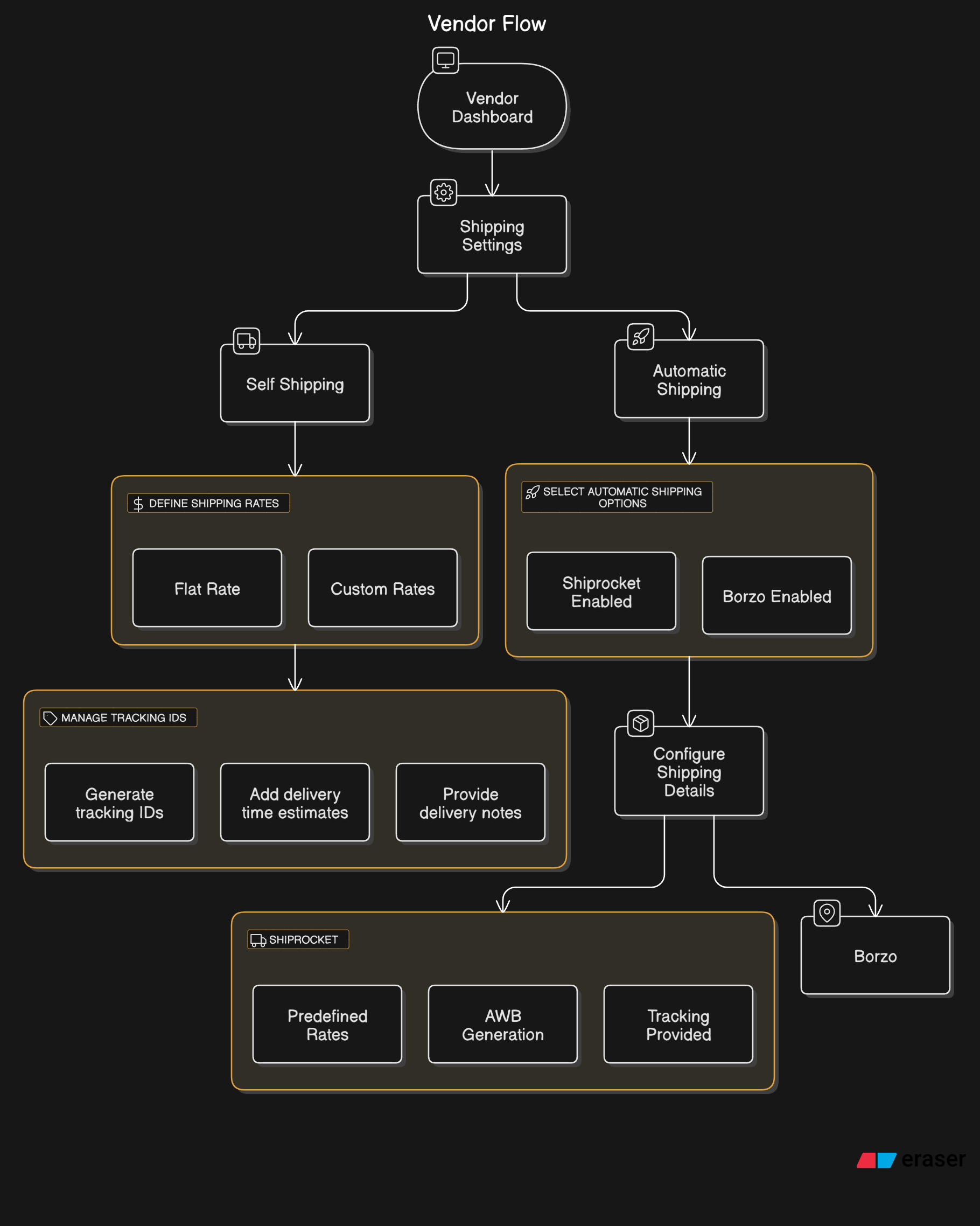
**3. Admin Panel**

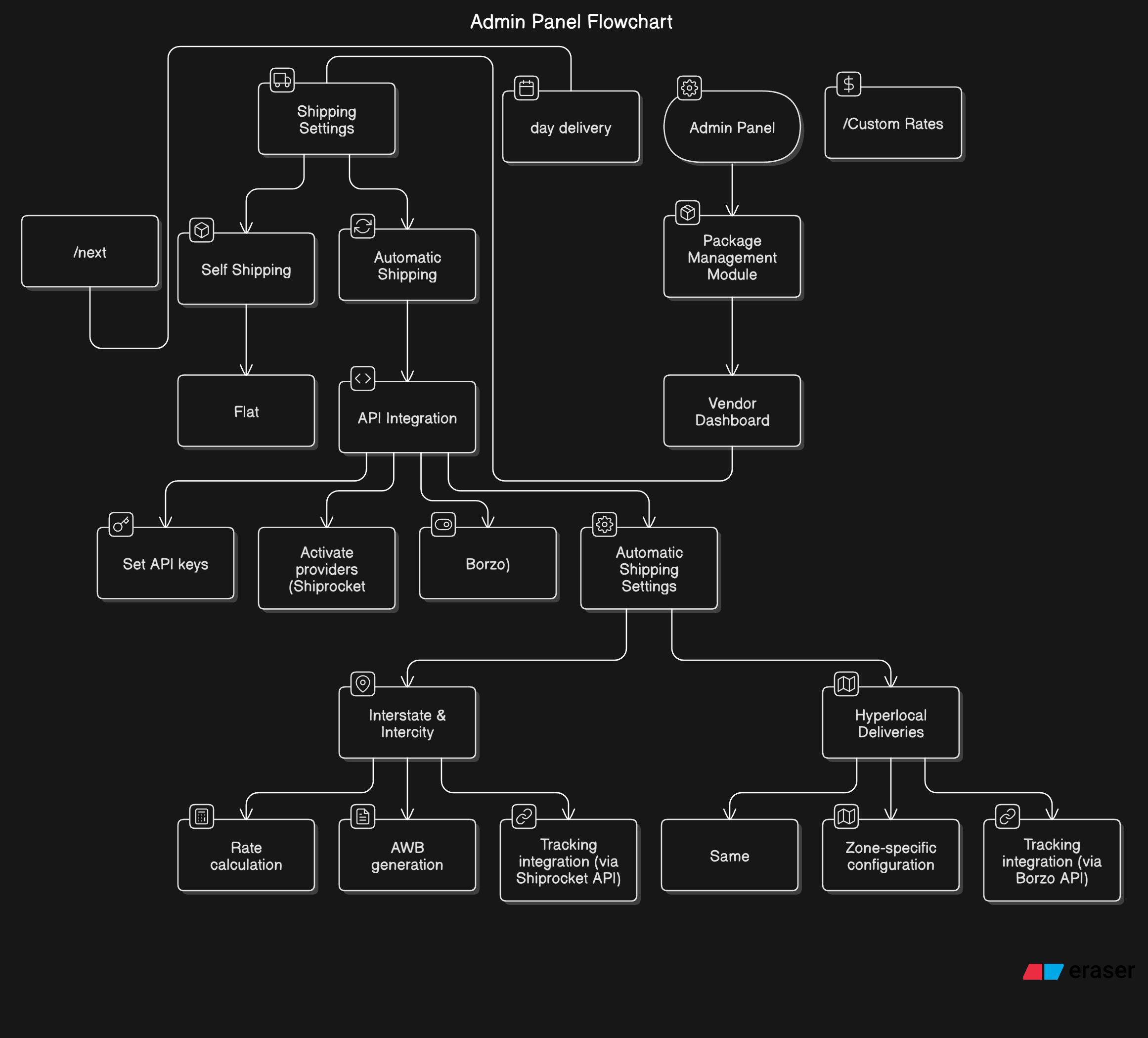
**Features**

* Manage vendor-specific shipping configurations.
* View total shipping activity and generate reports.

**Flowchart**

****

****

****

Total work flow

**Tasks and Timeline**

| **Task** | **Duration** |
| --- | --- |
| Requirement Analysis | 2 days |
| Project Setup | 1 day |
| Database Design | 6 days |
| API Integration | 6 days |
| Backend Logic Implementation | 6 days |
| Frontend Development | 6 days |
| Testing | 2 days |
| Deployment | 1 day |
| **Total Time** | **30 days** |

**Tools and Technologies**

* **Backend:** Laravel 9 (PHP), integrated with Shiprocket and Borzo APIs for shipping automation.
* **Frontend:** Blade Templating Engine,Bootstrap for responsive and clean UI design.
* **Database:** MySQL for managing vendor, customer, order, and shipping data.
* **HTTP Client**: Guzzle (for API requests)
* **Third-Party APIs:** Shiprocket (Intercity/Interstate deliveries) and Borzo (Hyperlocal deliveries).
* **Testing Tools:**
  + **Postman:** For API endpoint validation.
  + **PHPUnit:** For backend unit and integration testing.

**Deployment Plan**

1. Deploy the backend API to a cloud server (e.g., AWS, DigitalOcean).
2. Host the frontend application on a scalable service (e.g., Netlify, Vercel).
3. Secure API endpoints using HTTPS and API keys.
4. Monitor system performance and error logs post-deployment.