🚚Transport SaaS – Developer Documentation

1. Overview

This project is a **Multi-Tenant Transport ERP / SaaS system** built using **Node.js, Express, MongoDB (Mongoose)**.

The system is designed so that **each business runs in its own isolated tenant (Account)**. A tenant can be a **Supplier owner, Company owner, or Vehicle owner**, but technically all tenants behave the same.

> **Core Rule:** `Account (Tenant)` is the only isolation boundary in the system.

---

2. High-Level Goal

The system allows transport-related businesses to:

- Manage Suppliers, Companies, Vehicles
- Create Users (Admin / Staff)
- Record Trips involving Supplier + Company + Vehicle
- Calculate revenue, payouts, and profit
- Ensure strict data isolation between tenants

---

3. Tenant-Based Architecture

What is a Tenant?

A **Tenant** represents one independent business system.

```
Tenant (Account)
├── Users
├── Suppliers
├── Companies
├── Vehicles
└── Trips
```

- Each tenant has its own data
- Tenants never see each other's data
- Same database, different `accountId`

---

4. Account (Tenant) Model

Purpose

- Represents one business system
- Used for data isolation
- Used for future billing / subscription

Schema

```
Account
{
  name: String,
  accountId: String (unique),
  status: active | inactive,
  plan: FREE | PAID,
  timestamps
}
```

Important Notes

- No owner-specific logic stored here
- `accountId` is referenced everywhere

---

5. User Model & Access Control

Purpose

- Users log in to a tenant
- Role-based access inside a tenant

Roles

- ADMIN → Full access
- STAFF → Limited access (defined at API level)

Schema

```
User
{
  accountId: ObjectId (Account),
  accountType: SUPPLIER | COMPANY | VEHICLE,
  name,
  email,
  password,
  role: ADMIN | STAFF,
  resetPasswordToken,
  resetPasswordExpires,
```

```
  timestamps
}
```

Business Rules

- Same email allowed in different tenants
- Email must be unique within a tenant

---

6. Business Master Records

These are **data records**, not system owners.

6.1 Supplier

```
Supplier
{
  accountId,
  name,
  email,
  mobile,
  address,
  gstNumber,
  timestamps
}
```

6.2 Company

```
Company
{
  accountId,
  name,
  address,
  mobile,
  gstNumber,
  timestamps
}
```

6.3 Vehicle

```
Vehicle
{
  accountId,
  vehicleNumber,
  ownerName,
  mobile,
  driverName,
```

```
  driverPhone,
  capacity,
  timestamps
}
```

Key Rule

> Any tenant can create **Suppliers, Companies, and Vehicles**, regardless of who owns the tenant.

---

7. Trip (Core Business Logic)

Purpose

Trips represent actual transport operations.

Each trip involves: - 1 Supplier - 1 Company - 1 Vehicle - All belonging to the same tenant

Schema

```
Trip
{
  accountId,
  supplierId,
  companyId,
  vehicleId,
  loadingPoint,
  unloadingPoint,
  tripDate,
  totalTonLoad,
  companyRatePerTon,
  vehicleRatePerTon,
  status: pending | running | completed,
  createdByUserId,
  timestamps
}
```

---

8. Business Logic (Important)

Trip Financial Calculation

```
Company Amount = totalTonLoad × companyRatePerTon
Vehicle Payment = totalTonLoad × vehicleRatePerTon
Profit = Company Amount − Vehicle Payment
```

Status Flow

```
pending → running → completed
```

---

9. Data Isolation Rule (MOST IMPORTANT)

Every query MUST include:

```
{ accountId: req.user.accountId }
```

Example

```
Supplier.find({ accountId: req.user.accountId })
Trip.find({ accountId: req.user.accountId })
```

This guarantees: - No data leakage - Secure multi-tenant system

---

10. What This System Can Do

- Multi-tenant SaaS
- Supplier / Company / Vehicle management
- Role-based user access
- Trip tracking
- Profit calculation
- Scalable for thousands of tenants

---

11. What This System Intentionally Does NOT Do

- No cross-tenant sharing
- No owner-based hard restrictions
- No business logic inside database

(All permissions handled at API / UI level)

---

12. Future Enhancements (Planned)

- Invoice generation
- GST / TDS handling
- Dashboard analytics
- Subscription billing
- Mobile application

- Advanced permissions

---

13. Summary for New Developers

> This is a **tenant-first transport ERP system**.

- `Account` = Tenant
- `accountId` = Security boundary
- All business records belong to tenant
- Trips connect Supplier + Company + Vehicle
- Simple schema, powerful business logic

✅If you understand this document, you can work on any part of the system confidently.