🧠Multi-Tenant Transport / Trip Management System

😖Goal

A **highly scalable multi-tenant transport system** with **Facebook-style data isolation**.

- Same database
- Same APIs
- **Data never mixes** between suppliers / companies / vehicles

  Each tenant behaves like its own private system.

❤️‍🩹Core Architecture (FINAL)

😉Multi-Tenant Golden Rule (MOST IMPORTANT)

```
accountId = tenant boundary
```

- Every **Supplier / Company / Vehicle** belongs to exactly **one accountId**
- `accountId` is generated **once at registration** and **never changes**

😌Mandatory Query Rule

Every DB query **must include**:

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

➡️This is what enables **millions of users** safely in the same database.

👤User System (WHO IS LOGGED IN?)

The **User model controls everything**.

```
User {
  accountId,
  accountType, // SUPPLIER | COMPANY | VEHICLE
  role          // ADMIN | STAFF
}
```

😾Field Meaning

Field	Meaning
accountId	Tenant isolation key
accountType	Type of system user belongs to
role	Access level (Admin / Staff)

👇Authentication Flow

👆🏾Register API

```
POST /api/v1/auth/register
```

- Supplier / Company / Vehicle registers
- New `accountId` generated
- Saved in **User + all future data**

😉Login API

```
POST /api/v1/auth/login
```

JWT Payload

```
{
  id,
  accountId,
  accountType,
  role
}
```

😸Auth Middleware (VERY IMPORTANT)

```
req.user = {
  id,
  accountId,
  accountType,
  role
};
```

➡️**All controllers depend on this**

😻Authorization Logic (403 Errors Explained)

authorizeRoles Middleware

```
authorizeRoles(["ADMIN", "STAFF"])
```

- Checks **role only**
- `accountType` is optional

❌Wrong usage:

```
authorizeRoles("ADMIN")
```

✅Correct usage:

```
authorizeRoles(["ADMIN"])
```

🙃Master Data System

Common Rule (ALL MODELS)

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

Result

- Supplier sees **only their companies**
- Company sees **only their vehicles**
- Dropdowns never leak data

😩Dropdown Flow (Practical Example)

1. Supplier logs in
2. Supplier creates Company
3. Supplier creates Vehicle
4. While adding Trip:
5. Company dropdown → filtered by accountId
6. Vehicle dropdown → filtered by accountId
7. Supplier auto-filled from login

😉Trip System (CORE BUSINESS)

Trip Design Principles

- Same vehicle → multiple trips
- Same company → multiple trips
- Same route → multiple trips

Trip Model

```
Trip {
  supplierId,
  companyId,
  vehicleId,
  from,
  to,
  date,
  totalTonLoad,
  companyRatePerTon,
  vehicleRatePerTon,
  accountId,
  createdByUserId
}
```

👂 Advance Payment System

Flexible by Design

- Multiple advances per trip
- Trip-wise OR total-wise payments

AdvancePayment Model

```
AdvancePayment {
  paidByRole,
  receivedByRole,
  amount,
  scopeType, // TRIP | TOTAL
  tripId,
  accountId
}
```

👹Excel Export System

API

```
GET /api/v1/excel/export
```

Views

View	Meaning
supplier	Profit report
company	Company payable
vehicle	Vehicle payable

Dynamic Calculation

```
profit = companyAmount - vehicleAmount
```

👉🏽Bugs Fixed in This Architecture

Issue	Fix
500 error on register	accountId + accountType added
403 forbidden	authorizeRoles array usage
createdByUserId undefined	req.user.id
JWT confusion	clean payload
ES module crashes	single module system

👉🏿Search Keywords (SAVE THIS)

```
multi tenant transport system node mongoose
accountId isolation supplier company vehicle
trip advance payment excel export
authorizeRoles admin staff
req.user.accountId pattern
```

😌End-to-End Flow (1 Line)

```
Register → Login → JWT → authMiddleware → accountId filter → Master Data → Trip
→ Advance → Excel
```

😾GOLDEN RULE (REMEMBER FOREVER)

> **accountId = Facebook Page ID**

Same database. Same APIs. But **data kabhi mix nahi hota.**

✅This document defines the **final, scalable, production-ready architecture**.