Microservice development

Rest Api   ⟶   Transactions and Payments

Tools I used:
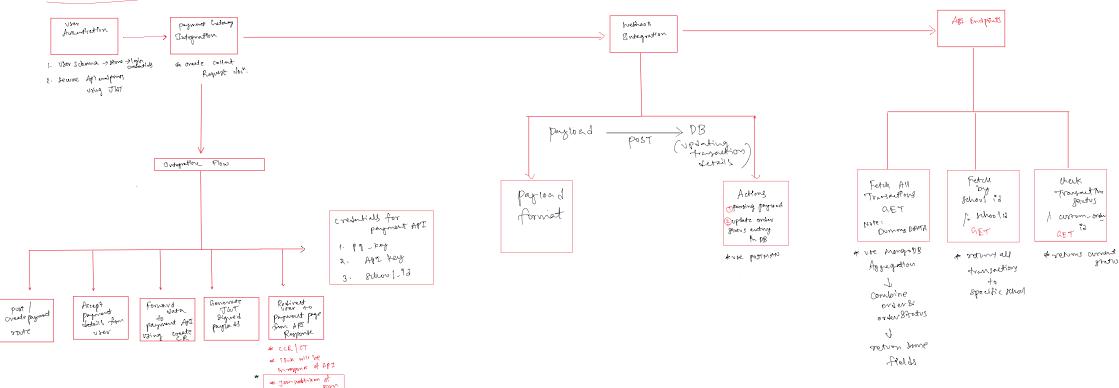   NodeJs + Express Js
    Mongo DB

DB schemas
    MongoDB

Order Schema	Order Status Schema	Webhook Logs Schema
Order-related Info	payment Trans- Info	Webhook Logs

Flow of Integration & Auth:

User Authentication	→	payment Gateway Integration	⟶	Webhook Integration	⟶	API Endpoints

1. User Schema → stores → login credentials
2. Secure API endpoints using JWT

& create collect Request Noti.

Integration Flow

Payload ⟶ POST ⟶ DB (updating transaction details)

Payload format

Actions
1️⃣ parsing payload
2️⃣ update order status entry in DB

* use POSTMAN

credentials for payment API
1. PG - key
2.   API key
3.   School - Id

Post / create payment route	Accept payment details from user	Forward Data to payment API using create CR	Generate JWT Signed payloads	Redirect user to payment page from API Response

* CCR | CT
* Link will be in response of API
* json web Token at sign

API Endpoints

Fetch All Transactions GET	Fetch by school id |: school id GET	check Transaction status |: custom-order id GET
Note: Dummy DATA		

* use MongoDB Aggregation
    ↓
    Combine order & orderStatus
    ↓
    return some fields

* returns all transactions to specific school

* returns current status

Key Notes:

```
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│ 1. Error Handling│ ──▶ │ Environment     │ ──▶ │ Read me   and   │ ──▶ │ Scalability &   │ ──▶ │ Security        │ ──▶ │ Robust          │
│    of API's      │     │   Configuration │     │ Documentation   │     │ Performance     │     │                 │     │ Logging         │
│ 2. Incoming Data │     │ - env files     │     │                 │     │                 │     │                 │     │                 │
│    validation    │     │                 │     │                 │     │                 │     │                 │     │                 │
│                  │     │  sensitive keys │     │                 │     │                 │     │                 │     │                 │
└─────────────────┘     └─────────────────┘     └─────────────────┘     └─────────────────┘     └─────────────────┘     └─────────────────┘
```

○ MongoDB Atlas connection string
○ Payment API credentials (API key, PG key)
○ JWT secret and expiry time

○ Setup and installation instructions
○ API usage examples
○ Environment variable configuration
○ Postman collection for testing

Pagination: Implement pagination for all list endpoints (e.g., /transactions) using limit and page query parameters.

○ Sorting: Support sorting by fields such as payment_time, status, or transaction_amount via query parameters (e.g., sort=payment_time&order=desc).

○ Indexing: Ensure important fields like school_id, custom_order_id, and collect_id are indexed in MongoDB to speed up queries.

Use **JWT Authentication** for all protected routes.

○ Sanitize and validate incoming requests to prevent injection attacks.

○ Use HTTPS in production and set appropriate CORS policies.

○ Log incoming webhook events and failed transactions for audit and debugging.