Module: Backend APIs & DynamoDB Integration

1. Overview

The Backend APIs and DynamoDB Integration module is responsible for storing, processing, and retrieving financial transaction and order data for the multi-vendor analytics dashboard.

This module provides a serverless backend architecture using AWS services to handle vendor transactions, order management, and dashboard data retrieval in real time.

The backend system ensures scalability, security, and high performance for multi-vendor financial data processing.

---

2. DynamoDB Database Design

Purpose

The VendorSales table is used to store all vendor transaction and order-related data in a scalable NoSQL database. DynamoDB is chosen to support high throughput and low-latency access for real-time analytics.

Table Name : VendorSales

Data Fields Stored

Field Name	Description
vendor_id	Unique identifier for each vendor
order_id	Unique order reference number
product_id	Product identifier
gross_amount	Total order value before commission
platform_commission	Commission charged by the platform
net_amount	Final payout amount to the vendor

payment_status	Payment state (Paid, Pending, Refunded)
timestamp	Transaction date and time

This schema supports transaction logging, order tracking, and financial analytics.

---

3. Global Secondary Indexes (GSIs)

To support efficient querying, two Global Secondary Indexes were configured:

Vendor-Time Index

- Used to retrieve vendor transactions based on time range
- Supports dashboard analytics such as daily, weekly, and monthly revenue
- Enables fast time-series data retrieval

Vendor-Order Index

- Used to retrieve specific order details
- Supports order tracking and order-level queries
- Reduces latency for order lookup operations

These indexes improve query performance and scalability.

---

4. Backend APIs Architecture

The backend exposes three main REST APIs through API Gateway, processed by AWS Lambda, and backed by DynamoDB.

4.1 Dashboard API

Purpose

The Dashboard API provides aggregated financial data for vendors to display in the analytics dashboard.

Functionality

- Retrieves vendor-wise financial summaries
- Supports filtering by date range
- Returns aggregated metrics such as revenue and earnings

This API is consumed by the frontend to display charts and financial summaries.

4.2 Transactions API

Purpose

The Transactions API is used to store and retrieve vendor transaction records.

Functionality

- Inserts new transaction records into DynamoDB
- Retrieves historical transaction data for vendors
- Supports analytics and reporting features

This API acts as the core transaction logging service.

4.3 Orders API

Purpose

The Orders API provides order-level details for vendors.

Functionality

- Retrieves specific order information

- Supports vendor-specific order lookup
- Used in the order management section of the dashboard

---

5. Lambda and DynamoDB Processing Flow

AWS Lambda functions act as the business logic layer between API Gateway and DynamoDB.

Processing Steps:

1. API Gateway receives HTTP requests from the frontend
2. Lambda validates and processes the request
3. Lambda reads or writes data to DynamoDB
4. Response is returned to the frontend dashboard

This serverless architecture ensures automatic scaling and minimal operational overhead.

---

6. Vendor Data Security and Isolation

To maintain multi-tenant security:

- Each vendor is assigned a unique vendor_id
- Authentication is handled using AWS Cognito
- Vendor identity is validated through secure tokens
- Vendors can only access their own transactions, orders, and dashboard data

This ensures strict data isolation and privacy between vendors.

---

7. System Integration Architecture

The backend module integrates with the following components:

- **AWS Cognito** – User authentication and vendor identity
- **API Gateway** – Secure REST API interface
- **AWS Lambda** – Backend processing logic
- **DynamoDB** – Transaction and order data storage
- **React Frontend Dashboard** – Data visualization and user interface

The system follows a fully serverless cloud-native architecture.

---

8. Performance and Scalability

The backend system is designed for high performance and scalability:

- DynamoDB provides horizontal scalability and low latency
- Global Secondary Indexes optimize query operations
- Lambda automatically scales based on request volume
- API Gateway supports secure and scalable API management

The system can handle large volumes of vendor transactions efficiently.

---

9. Module Summary

The Backend APIs and DynamoDB Integration module provides the core infrastructure for storing, retrieving, and processing financial transaction and order data. It enables real-time dashboard analytics, secure vendor data access, and scalable cloud-based processing using serverless AWS services.