**📘 Payment Oversight Module – Technical Documentation (Squad E3.4)**

**Backend Framework:** FastAPI

**1. System Architecture**

**1.1 Overview**

The Payment Oversight Module ensures secure handling of financial transactions by:

* Detecting and managing flagged transactions (potentially suspicious).
* Enabling payment dispute resolution through structured workflows.
* Maintaining a full audit trail for transparency and compliance.

**Core Principles:**

* Security-first design
* Regulatory compliance (AML/KYC, GDPR)
* Horizontal scalability
* End-to-end logging and monitoring

**1.2 High-Level Architecture**

* **Frontend:** Admin dashboard (Angular/React)
* **Backend Services:** FastAPI REST-based APIs
* **Database Layer:** PostgreSQL + Redis for caching flagged transactions
* **Notification Service:** Email/SMS/Push integration
* **Audit Logging Service:** Immutable append-only logs (PostgreSQL or ELK stack)
* **Monitoring:** Prometheus + Grafana dashboards

**Flow Diagram:**

flowchart LR

A[Admin User] --> B[Admin Dashboard UI]

B --> C[FastAPI Payments Oversight API]

C --> D[(Database: flagged\_transactions, payment\_disputes, audit\_logs)]

C --> E[Notification Service]

C --> F[Audit Logging Service]

**2. Data Model Specifications**

**2.1 flagged\_transactions**

| **Field** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | UUID | PK | Unique identifier |
| transaction\_id | UUID | FK → transactions | Associated transaction |
| flagged\_reason | TEXT | NOT NULL | Reason flagged (AML, fraud, anomaly) |
| flagged\_at | TIMESTAMP | DEFAULT now() | Time flagged |
| status | ENUM | {pending, reviewed, cleared} | Flag status |

**2.2 payment\_disputes**

| **Field** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | UUID | PK | Unique identifier |
| transaction\_id | UUID | FK → transactions | Disputed transaction |
| user\_id | UUID | FK → users | Disputing user |
| dispute\_reason | TEXT | NOT NULL | Dispute details |
| status | ENUM | {open, under\_review, resolved, escalated} | Lifecycle state |
| created\_at | TIMESTAMP | DEFAULT now() | Created timestamp |
| resolved\_at | TIMESTAMP | NULLABLE | Resolution timestamp |

**2.3 audit\_logs**

| **Field** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| id | UUID | PK | Unique identifier |
| admin\_id | UUID | FK → admins | Acting admin |
| action\_type | ENUM | {flagged\_view, dispute\_resolved, escalation, notification\_sent} | Admin action |
| target\_id | UUID | FK → {flagged\_transactions, payment\_disputes} | Affected entity |
| timestamp | TIMESTAMP | DEFAULT now() | Action timestamp |
| metadata | JSONB |  | Extra contextual details |

**3. API Endpoints (FastAPI Style)**

**3.1 GET /api/v1/admin/payments/flagged**

* **Description:** Retrieve all flagged transactions pending admin review.
* **Response Example:**

[

{

"id": "uuid-123",

"transaction\_id": "txn-789",

"flagged\_reason": "Unusual amount",

"status": "pending",

"flagged\_at": "2025-09-10T14:22:11Z"

}

]

* **Error Codes:**
  + 401 Unauthorized – Invalid admin token
  + 404 Not Found – No flagged transactions
  + 500 Internal Server Error

**3.2 POST /api/v1/admin/payments/disputes/{id}/resolve**

* **Description:** Resolve a payment dispute.
* **Request Body:**

{

"resolution": "Transaction refunded",

"resolved\_by": "admin-456"

}

* **Response Example:**

{

"id": "dispute-123",

"status": "resolved",

"resolved\_at": "2025-09-16T10:00:00Z"

}

* **Error Codes:**
  + 400 Bad Request – Invalid input data
  + 404 Not Found – Dispute ID not found
  + 409 Conflict – Dispute already resolved
  + 500 Internal Server Error

**4. Dispute Resolution Workflow**

sequenceDiagram

participant Admin

participant API

participant DB

participant User

participant AuditLog

Admin->>API: View flagged transactions

API->>DB: Fetch flagged records

API-->>Admin: Return flagged list

Admin->>API: Resolve dispute (POST)

API->>DB: Update dispute status

API->>User: Notify resolution outcome

API->>AuditLog: Record admin action

**5. Security & Compliance**

* **Encryption:** AES-256 (data at rest), TLS 1.3 (in transit)
* **Access Control:** JWT-based admin authentication, RBAC
* **Compliance:** GDPR, AML/KYC adherence
* **Audit Trail:** Immutable logging (append-only)
* **PII Masking:** Sensitive user info masked

**6. Error Handling**

* Structured JSON errors:

{

"error\_code": "DISPUTE\_NOT\_FOUND",

"message": "The requested dispute does not exist."

}

* Graceful degradation for partial failures
* Retry + backoff for failed notifications
* Alerting repeated error patterns via Prometheus

**7. Performance Optimization**

* Indexes on transaction\_id and status
* Database connection pooling
* Redis caching for frequently accessed flagged transactions
* Asynchronous job queues (Celery or RQ) for notifications

**8. Testing Strategy**

* **Unit Tests:** Dispute resolution logic, flagged transaction validation
* **Integration Tests:** End-to-end flagged transaction lifecycle, escalation workflow with notifications
* **Negative Tests:** Invalid IDs, unauthorized admin attempts
* **Performance Tests:** Stress tests with 10k flagged transactions, load test concurrent dispute resolution

**9. Deployment & Scaling**

* **Containerization:** Docker + Kubernetes
* **CI/CD:** GitHub Actions + Helm
* **Monitoring:** Alerts for abnormal dispute/flag rates
* **Scaling:** Horizontal scaling of FastAPI services
* **Future Enhancements:**
  + ML-based fraud detection for auto-flagging
  + Chatbot-based dispute query system
  + Blockchain-based immutable audit logs

**🔧 Troubleshooting & Best Practices**

* Check audit logs for dispute inconsistencies
* Ensure flagged reasons are clear
* Use escalation for disputes older than SLA
* Enable rate limiting to prevent brute-force attacks

✅ **Summary:**  
FastAPI backend handles all **flagged transactions, dispute resolution, notifications, and audit logs** efficiently while supporting **async operations, caching, and horizontal scaling**.

If you want, I can also **rewrite this as a FastAPI project structure** with **folders, files, Pydantic models, routers, and async endpoints** ready for implementation.