**High-Level Design (HLD) document**

**I. Introduction**

This document outlines the High-Level Design for building Payment approval workflow API for Employee Benefits(EB) Line of business. The system implements a tiered, sequential approval process where each tier requires only one approval from any authorized approver within that tier.

**II. Requirements**

* **Tiered Approvals:** Payments are routed through three tiers based on the payment amount and Disbursement Type.
  + Tier 1: Amount < Tier 1 Max Approval Limit for given Disbursement type
  + Tier 2: $50,000 <= Amount < Tier 2 Max Approval Limit for given Disbursement type
  + Tier 3: $1,000,000 <= Amount <= Tier 3 Max Approval Limit for given Disbursement type
* **Sequential Approvals:** Approvals must occur in sequence (Tier 1, then Tier 2, then Tier 3 if applicable).
* **Single Approval per Tier:** Only one approval is required per tier. Any authorized approver within a tier can provide the approval.
* **Any Approver Can Approve:** If multiple approvers are assigned to a tier, the first approval received is sufficient. Subsequent approval attempts for the same tier are ignored.
* **Approver cannot be same as Requestor:**

**III. System Architecture**

**A. Architecture Diagram:**

+-----------------+ +-----------------+ +-----------------+ +-----------------+  
| Payment Service | --> | Approval Service | --> | User Service | --> | Notification |  
+-----------------+ +-----------------+ +-----------------+ +-----------------+  
 ^ | |  
 | v v  
 +-----------------------+-----------------------+  
 |  
 v  
 +-----------------+  
 | Database (e.g.,|  
 | MySQL) |  
 +-----------------+

**B. Components:**

1. **Payment Service:**
   * Handles the creation of payment requests.
   * Determines the required approval tiers based on the payment amount.
   * Manages the overall payment status (PENDING, APPROVED, REJECTED, PROCESSED).
   * Initiates the approval workflow by creating the initial Tier 1 approval step.
2. **Approval Service:**
   * Handles individual approval submissions.
   * Validates the approver's authorization (checks if the user is an approver for the current tier).
   * Updates the approval status of a tier.
   * Transitions the workflow to the next approval tier (if needed) by creating the corresponding approval\_step record.
   * Marks the payment as POSTED when all required tiers have approved. Handles duplicate approval attempts within a tier.
3. **User Service:**
   * Manages user authentication and authorization.
   * Provides a method to check if a user is authorized to approve a specific tier.
4. **Notification Service:**
   * Sends notifications (email, SMS, in-app) to approvers when a payment requires their attention. (Asynchronous notifications are recommended using a message queue).
5. **Database:**
   * Stores payment requests, approval steps, user information, and the mapping of users to approval tiers.

**IV. Data Model (MySQL Example)**

* payment\_request: id (UUID), amount, currency, status (PENDING, APPROVED, REJECTED, PROCESSED), created\_by (user\_id), created\_at, updated\_at
* approval\_step: id (UUID), payment\_id (FK), tier (1, 2, 3), status (PENDING, APPROVED, REJECTED), approver\_user\_id (FK, nullable), comments, created\_at, updated\_at
* user: id (UUID), username, email, ...
* approver\_tier: user\_id (FK), tier (1, 2, 3) (Maps users to the tiers they can approve)

**V. Workflow**

**A. Payment Creation:**

1. User submits a payment request with the amount.
2. Payment Service creates a payment\_request record (status: PENDING).
3. Payment Service determines the highest required tier based on the amount.
4. Payment Service creates the initial approval\_step record for Tier 1 (status: PENDING).

**B. Approval Process:**

1. Approver submits an approval for a payment.
2. Approval Service retrieves the current approval\_step (status: PENDING).
3. Approval Service validates the approver's authorization (using User Service).
4. Approval Service updates the current approval\_step record (status: APPROVED, approver\_user\_id updated).
5. Approval Service checks if there are more tiers to approve.
6. If yes, Approval Service checks if a PENDING approval\_step record for the next tier *already exists*. If not, it creates one. This prevents duplicate approvals.
7. If no more tiers to approve, Approval Service updates the payment\_request record (status: APPROVED).

**VI. API Endpoints (RESTful)**

* POST /payments: Create a new payment request.
* GET /payments/{id}: Retrieve payment details and approval status.
* POST /payments/{id}/approvals: Submit an approval for a payment.
* GET /payments/{id}/approvals: Retrieve the approval history for a payment.

**VII. Technology Stack**

* **Spring Boot:** Core framework.
* **Spring Data JPA:** Database access.
* **Spring Security:** Authentication and authorization.
* **Lombok:** Code simplification.
* **MapStruct/ModelMapper:** DTO mapping.
* **MySQL (or other relational database):** Data persistence.
* **RabbitMQ/Kafka (recommended):** Asynchronous notifications.
* **Swagger/OpenAPI:** API documentation.

**VIII. Non-Functional Requirements**

* **Security:** Protect API endpoints, validate user roles and permissions.
* **Scalability:** Design for a large number of payments and users.
* **Reliability:** Ensure the system is fault-tolerant.
* **Performance:** Optimize database queries and API responses.
* **Maintainability:** Write clean, well-documented code.

**IX. Key Design Considerations**

* **Idempotency:** Implement idempotency for the approval submission endpoint to handle duplicate requests.
* **Transactions:** Use @Transactional to ensure data consistency.
* **Error Handling:** Implement robust error handling and logging.
* **Notifications:** Design a flexible notification system.
* **Auditing:** Consider adding auditing to track changes to payment requests and approvals.

**X. Conclusion**

This HLD provides a comprehensive overview of the design for the payment disbursement application's approval workflow API. By adhering to this design, the development team can build a secure, scalable, and maintainable system. The focus on sequential approvals and the single approval per tier simplifies the implementation and ensures a clear and efficient approval process.