Backend Developer Assignment

Objective

Design and build a **Loan Origination System (LOS)** that safely and concurrently processes multiple loan applications, simulates system approval delays, supports agent-manager hierarchies, and sends mock notifications to relevant parties. The focus is on clean architecture, database design, thread-safety, and deployment readiness.

Functional Requirements:-

1. Submit Loan Application

Create an API to submit a new loan application.

Fields(e.g):

- loan_id
- customer_name
- customer_phone
- loan_amount
- loan_type (ENUM: PERSONAL, HOME, AUTO, BUSINESS)
- application_status
- created_at

Endpoint:
POST /api/v1/loans

2. Automated Loan Processing (Multithreaded Simulation)

Implement a background job using a thread pool to simulate system approval.

Each thread:

- Picks a loan with status APPLIED
- Waits for a random delay (e.g., 25 seconds) to simulate system checks
- Then applies simple rules to decide:
    - APPROVED_BY_SYSTEM
    - REJECTED_BY_SYSTEM
    - Or UNDER_REVIEW (if needs human input)

3. Agent Assignment and Review

If a loan goes to UNDER_REVIEW:

- Automatically assign it to an available agent
- Each agent belongs to exactly one **manager**.
- An Agent can be the manager of another agent.
- Upon assignment, send a **push notification to the agent and their manager**.
- Allow agents to review and decide the outcome

Agent Decision Endpoint:
PUT /api/v1/agents/{agent_id}/loans/{loan_id}/decision
Request Body:

{ "decision": "APPROVE" | "REJECT" }

4. Notification (Mocked)

Mock the notification service:

- When a loan is assigned to an agent → push notification to the agent
- When a loan is approved, → SMS to the customer

Use a NotificationService interface with a logger/mock implementation.

5. Loan Status Monitoring

Provide real-time counts of loans in each status.

Endpoint:
GET /api/v1/loans/status-count

6. Top Customers API

Return the **top 3 customers** with the most approved loans (APPROVED_BY_SYSTEM + APPROVED_BY_AGENT).

Endpoint:
GET /api/v1/customers/top

7. **Fetch Loans by Status with Pagination**

Endpoint:
GET /api/v1/loans?status={status}&page={n}&size={m}

Evaluation Criteria

Area	What We Look For
Scalability	Can handle a large number of loan applications safely
Thread-Safety	Proper locking to avoid race conditions
Clean Architecture	Modular and maintainable codebase
Mock Integration	Decoupled notification logic
DB Design	Normalized, indexed where necessary
Testing	Unit coverage for core services
Code Quality	Readable, SOLID
Documentation	Clear README and comments

Deployment Requirements

- Clean README.md with setup instructions
- Instructions to run the app locally with PostgreSQL/MySQL
- **Docker is optional** — focus on runnable instructions and code organization.
- Export all tested APIs (success + failure cases)
- Add Postman collection to the GitHub repo

Submission: GitHub repository link with instructions in `README.md`.