Gold Loan Management System – Project Synopsis

Project Overview

The Gold Loan Management System is a Java-based application designed to streamline the process of managing gold-backed loans. This system enables banks, NBFCs, and pawn brokers to register customers, evaluate gold collateral, issue loans, calculate interest, and track repayments.
The project starts as a **console application** using Core Java and evolves into a **GUI-based (Swing/JavaFX)** and later a **full-stack web application** using Spring Boot + React, backed by a relational database.

Project Objectives

- Build a digital platform to manage gold loan operations end-to-end.
- Demonstrate **OOP principles** (Inheritance for loan types, Encapsulation for customer data, Polymorphism for transactions).
- Implement **Collections Framework** for managing loan records.
- Ensure **data persistence** with file handling (Phase 1) and databases (Phase 2).
- Provide a **GUI interface** for loan officers and customers.
- Enable **real-world features** like interest calculation, due alerts, and penalty charges.
- Optionally deploy on **cloud platforms** for real usage simulation.

Technology Stack

Phase 1 – Core Java (Console Version)

- Language: Java (JDK 8 or higher)

- Collections: ArrayList, HashMap for customers and loans
- File Handling: CSV/JSON for storing loan details

Phase 2 – GUI/Desktop Version

- GUI Framework: Swing / JavaFX
- Database: MySQL or SQLite
- JDBC for DB connectivity

Phase 3 – Full-Stack Version

- Backend: Spring Boot (REST APIs, JPA/Hibernate)
- Frontend: React + Tailwind (Loan forms, dashboards)
- Database: MySQL/PostgreSQL
- Deployment: Vercel (frontend), Render/AWS (backend), Cloud DB

Key Features and Functionality

1. **Customer Management**
    - Register new customers with KYC details
    - Assign unique customer IDs
    - Update and delete customer records
2. **Gold Loan Processing**
    - Input gold item weight, purity, and valuation
    - Loan-to-value (LTV) calculation based on market price
    - Generate loan agreement with unique loan ID

3. **Interest & Repayment**
    - Support for monthly/quarterly interest calculations
    - EMI or lump-sum repayment options
    - Penalty calculation for late payments
    - Auto loan closure when fully repaid
4. **Collateral & Security**
    - Store collateral details (gold weight, purity, appraisal value)
    - Manage release of pledged items after settlement
    - Forfeit procedure in case of loan default
5. **Reports & Analytics**
    - Outstanding loans, interest due, and defaulters list
    - Export loan records to CSV/PDF
    - Performance dashboard (JavaFX charts or React UI)
6. **User Roles**
    - **Admin/Loan Officer**: Approve loans, update repayments, manage customers
    - **Customer (Portal in Web Version)**: View loan status, repayment history, due alerts

---

System Architecture

Console Version

- `Customer` (class: id, name, KYC details)
- `GoldCollateral` (class: weight, purity, market value)

- `Loan` (class: loanID, principal, interestRate, EMI, dueDate)
- `LoanManager` (handles loan creation, EMI, interest)
- `FileHandler` (store/retrieve loan/customer data)

GUI Version (Swing/JavaFX)

- CustomerForm, LoanForm, RepaymentPanel, ReportDashboard
- Event-driven UI with validation

Full-Stack Version

- **Backend**: REST APIs (`/customers`, `/loans`, `/repayments`, `/reports`)
- **Frontend**: React-based loan portal & dashboards
- **DB**: MySQL/PostgreSQL with relational schema (Customer ↔ Loan ↔ Repayment)

Expected Outcomes

- Mastery of **Core Java (OOP, Collections, Exception Handling, File I/O)**.
- Hands-on with **real-world financial calculations** (LTV ratio, interest, EMI, penalties).
- Strong skills in **GUI (Swing/JavaFX)** development for user-friendly dashboards.
- Knowledge of **Spring Boot + REST API + DB integration**.
- Understanding of **scalable deployment** (cloud hosting).
- A finance-focused project that mirrors **banking/NBFC use cases**.