ChatGPT

Loan Origination & Loan Management System – Comprehensive Requirements Prompt

Overview

Design an **integrated Loan Origination System (LOS) and Loan Management System (LMS)** that handles the end-to-end lifecycle of loans on a single platform – from initial application and underwriting through loan servicing and closure [1]. The solution should support **multiple lending products and scenarios** (retail consumer loans, commercial/wholesale loans, co-lending partnerships, supply chain finance, and securitization deals like Pass-Through Certificates (PTC) and Direct Assignments (DA)). It must be deployable both **on-premises and as a cloud-native solution**, ensuring scalability to handle a **large number of accounts** (high-volume retail lending) with high performance and reliability. The system's backend will be built with **Python** (leveraging appropriate frameworks for web services and data handling), following best practices for robust, maintainable code. All components should emphasize security, compliance, and configurability to adapt to various financial products and regulatory requirements.

Functional Requirements

Loan Origination System (LOS) Features

The LOS module will facilitate **loan initiation and underwriting** for all supported product types, streamlining the process from application to disbursement. Key functionalities include:

- **Multi-Channel Application Intake:** Accept loan applications through multiple channels – online web portals, mobile apps, branch offices, third-party partner platforms, etc. The system should aggregate applications from all channels and perform initial validation (format checks, required fields, document uploads) automatically [2]. Borrower onboarding should include **KYC (Know Your Customer) verification**, identity checks, and credit bureau checks via API integrations.

- **Loan Product Configuration:** Support a variety of loan products with configurable terms. Administrators can define product parameters such as loan type (personal, mortgage, SME, invoice finance, etc.), interest rate type (fixed or floating), repayment tenure, repayment schedule type (e.g. equated monthly installments, interest-only with balloon payment), fees (processing fees, origination charges), **prepayment rules**, and **penal charges** for late payment. This allows the LOS to handle retail loans, corporate loans, co-lending agreements, and specialized products (supply chain finance, etc.) within one system.

- **Intelligent Workflow & Underwriting:** Implement an automated workflow engine to route applications through the required stages of processing [2]. For example: assign to loan officers or underwriters based on product or workload, then to credit managers for approval, etc. The workflow should be **configurable for multi-level approvals** (e.g. analyst -> manager -> credit committee for wholesale loans). The system sends notifications or tasks to relevant staff at each stage. It should track application status in real time (e.g. **Pending docs**, **In underwriting**, **Approved**, **Declined**).

- **Automated Credit Decisioning:** Integrate **credit scoring and risk analysis** tools. The LOS should automatically pull in credit data (credit scores, credit history) and financial information about the borrower to assess eligibility [2]. Implement business rules or AI/ML models for **automated decisioning** on simpler cases (based on credit score, debt-to-income, etc.), while flagging complex cases for manual review [3]. Ensure all decisions follow the institution's credit policy – the system should perform compliance checks to enforce policy rules (e.g. loan amount limits, borrower credit grade thresholds) [4].

- **Document Management and E-Signature:** Provide a secure document upload and management system for all required documents (income proof, IDs, collateral documents, etc.). Customers and staff can upload documents through the portal; the system should index and store them with the application. Generate **templated loan documents** (term sheets, sanction letters, loan agreements) populated with applicant and loan details [4]. Include built-in support for **electronic signatures** to streamline agreement signing. All documents and agreements must be stored and associated with the loan record for audit trail.

- **Co-Lending & Partnership Handling:** Support scenarios where a loan is funded by multiple lending partners. During origination, the system should allow defining multiple participants (e.g. a bank and an NBFC co-lending in a fixed ratio). It must route the application through each partner's credit approval if needed and consolidate the final decision. Upon approval, allocate the loan amount and interest sharing according to the partnership agreement. The system should record each partner's share of the principal and interest for downstream servicing.

- **Supply Chain Finance Workflows:** For supply chain finance (e.g. invoice financing, factoring), enable specialized origination flows. This includes capturing invoice or purchase order data, linking borrowers (suppliers) and payers (buyers), and performing risk assessment on both. The LOS should allow setting up short-term credit facilities that are repaid by future receivables. It may require integration with corporate ERP systems or invoice registries to pull invoice data and validate transactions.

- **Securitization & Asset Assignment:** If loans are to be securitized or assigned (PTC or DA transactions), the system should support tagging certain loans or pools of loans for securitization during origination. This includes capturing additional data (like pool ID, tranche details) and due diligence checks for assets being securitized. The LOS should facilitate creating a pool of loans and preparing necessary reports/data tapes for investors or purchasing institutions. (The actual securitization process and SPV management is mostly outside the LOS, but the system should provide data and flag those accounts appropriately for the LMS to manage their cash flows separately.)

- **Approval, Communication, and Disbursement:** Once credit checks and underwriting are completed, the LOS will generate an **approval or rejection outcome**. If rejected, it should record reasons and possibly communicate the decision to the applicant. If approved, the system should support final **offer acceptance** from the borrower. Communicate approval and next steps instantly to the borrower (e.g. through the portal or email/SMS) [4]. After acceptance, **disbursement** of the loan is initiated: the LOS should create a disbursement instruction (with amount, bank account details of borrower, etc.) for the finance team or core banking system. Where integrated, the system can trigger an automated funds transfer to the borrower's account [4]. Upon disbursement, all loan data and repayment schedule are handed off to the Loan Management module for servicing. Every step (from application to funding) should be logged for audit purposes.

Loan Management System (LMS) Features

The LMS module is responsible for **servicing and post-disbursement management** of loans until closure. It must handle a wide range of repayment scenarios, interest calculations, and delinquency management for the various loan types. Key LMS functionalities include:

- **Loan Account Setup & Amortization Schedule:** When a loan is activated (post-origination), the system generates a **repayment schedule** based on the loan terms. This includes all installment due dates and amounts over the loan tenure. Support flexible amortization models – **Equal Monthly Installments (EMI)**, **interest-only periods** with principal balloon payment, **bullet repayments**, **step-up or step-down plans**, etc. The schedule should detail the split of each installment into principal and interest components, and any fees if applicable. The system must accurately calculate due principal and interest amounts for each period according to the agreed interest rate and compounding convention [5]. It should handle various interest calculation conventions (e.g. daily accrual with 30/360 vs ACT/365 day count basis, compounding frequency) and multiple repayment frequencies (monthly, quarterly, weekly, annual, etc.) [6]. Changes to loan terms (interest rate changes for floating-rate loans, extensions, restructures) should trigger automatic regeneration of the schedule. The amortization schedule should be accessible to borrowers (e.g., via portal) and internal users, and configurable to be shared or emailed to borrowers [5].

- **Payment Processing & Allocation:** The LMS must track all loan repayment transactions. When a borrower makes a payment, the system should accept inputs from various channels – online payments, direct debits, bank transfers, checks, etc. (integration with payment gateways and banking networks). Each **incoming payment** is matched to the loan and the due items. The system then allocates the funds according to a defined **waterfall order** (for example: fees first, then interest, then principal, then any residual/overpayment) [7]. Partial payments should be handled gracefully – the system allocates what was received and leaves the remainder as outstanding. It should also support **prepayments** (extra payments made before the due date or paying off the loan early): the system must allow both full pre-closure of loans and partial prepayments. Upon prepayment, it should recalculate the outstanding principal and optionally adjust the future schedule (either reducing the tenor or lowering installment amounts as per business rules). Any **prepayment penalties or fees** configured for the product should be automatically applied. The payment processing engine must be **atomic and accurate**, updating balances in real-time and creating a transaction log for each payment (with timestamp, method, etc.). After applying a payment, the system updates the loan's status (next due amount, paid-to-date, etc.) and can generate receipts or updated statements for the borrower. All payment events need to be recorded for audit, and any discrepancies (like an amount that doesn't match an expected installment) should flag an exception for review [7]. The LMS should also support **automated payment reconciliation** for electronic payments – matching incoming transactions to the correct loans and installments [5].

- **Interest, Fees, and Charges Handling:** The system should accrue interest on a daily basis (or appropriate frequency) to track earned and due interest. It must handle **different interest rate schemes**: fixed interest loans (constant rate), floating rates (with periodic resets based on index like LIBOR/SOFR or central bank rates, plus spread), and **penalty interest** for overdue amounts. If a loan has adjustable rates, the system should update schedules and accruals whenever the rate changes. It should also manage **grace periods** (no interest or payment due in an initial period if applicable) and **interest capitalization** if interest can be added to principal (for example, Payment-In-Kind interest or deferred interest). Aside from interest, the LMS should apply any **fees and charges** associated with the loan during servicing. This includes late payment fees, periodic charges, or foreclosure charges. The business rules for fees (flat fee vs. interest on overdue amount, grace days allowed, etc.) should be configurable per product. For example, the system might impose a late fee after 5 days past due or charge a **penal interest rate** (an increased interest rate on the overdue amount) for the period of delay. These charges should be applied automatically as per configuration and reflected in the loan account.

- **Delinquency Management (DPD Tracking & Collections):** For each loan, the LMS must monitor **delinquencies**. When a payment due date passes without full payment, the system should calculate **Days Past Due (DPD)** – essentially how many days have elapsed since the installment due date without payment. It should categorize accounts into delinquency buckets (e.g., 1-30 days overdue, 31-60 days, 61-90 days, etc.) as per standard or custom criteria. The system should **flag overdue loans** and generate alerts/tasks for the collections team when certain DPD thresholds are crossed [8]. For example, an alert to a loan officer or collections agent when a loan becomes 10 days overdue [8]. The LMS should maintain a log of missed payments and DPD for each account, and this data will feed into collections workflows. **Collections workflow:** integrate features for collections management such as scheduling reminder notifications to borrowers (emails/SMS for upcoming or missed payments), creating promises-to-pay records, and escalating to legal remedies if loans become highly delinquent. The system can prioritize collection efforts based on rules (higher outstanding or higher DPD accounts get priority) [8]. If an account crosses into non-performing status (e.g., 90+ DPD), the system should mark it and possibly trigger suspension of interest accrual or other accounting treatments as per policy.

- **Co-Lending/Syndication Servicing:** For loans that involve multiple lenders or participants (co-lending or syndicated loans), the LMS must accurately manage **distribution of cash flows**. This means each repayment collected from the borrower needs to be **allocated to each lending party** in proportion to their ownership share or according to the syndication agreement [9]. The system should maintain sub-accounts or ledgers for each partner's portion of the loan. When a borrower pays an installment, the system calculates how much of the principal and interest (and fees) belong to each partner and credits their accounts accordingly [9]. It should also handle scenarios where one partner might have different interest rates or fee arrangements (if applicable in some co-lending models). All reports and ledgers should reflect the breakdown by partner. Additionally, if the loan servicing involves forwarding payments to a co-lender (e.g., a platform collecting payments then remitting to participants), the system should generate the payout instructions or schedules. In case of delinquency, the system may also need to notify all partners of missed payments or send co-branded notices as needed [10].

- **Supply Chain Finance Servicing:** For invoice financing loans, the LMS should link loan accounts to specific invoices or receivables. It needs to track when the financed invoice is paid by the buyer, so that those funds go towards loan repayment. This might involve integration with accounts receivable systems or manual input when an invoice is paid. The LMS should accommodate short loan tenures (sometimes 30-90 days) and possibly dynamic repayment amounts (the invoice could be paid in parts or with adjustments). It should also handle any factoring fees or discount charges applicable to these loans. Essentially, ensure the loan is closed when the underlying invoice payment is received (or handle extensions if the invoice isn't paid on time, which then becomes a collections issue).

- **Securitization Management:** If loans (or portions of loans) are sold or securitized, the LMS should reflect that in its records. For PTCs (Pass-Through Certificates), the system should mark those loans as part of a pool and track that the **beneficial owner of the cashflows** (principal and interest) is now the investors. The system might need to split the accounting: for securitized loans, continue to service the borrower (collect payments), but then channel those payments (minus servicing fees) to an **investor account or trust**. This requires tracking pool performance: e.g., generating reports of collections on securitized pools, remaining balance, prepayments, etc., for investors or trustees. In the case of Direct Assignment (DA) deals (whole loan transfers to another institution), the system should support transferring ownership of the loan to another entity. That might mean the loan is boarded onto the buyer's system and possibly taken off from the originator's portfolio. The LMS could facilitate this by exporting all loan data and providing a mechanism to **partially or fully derecognize** the loan from the originator's books once sold. (If the system serves as a platform for both originator and purchaser, it could simply reassign the account to the new owner's portfolio within the system). In summary, the LMS should be flexible to accommodate **asset transfers**, keeping clear records of any portion of loans sold, and ensure that servicing continues correctly for the end borrower while payments get routed as per the new ownership structure.

- **Loan Accounting & Statements:** Maintain accurate accounting for each loan account. The system should track outstanding principal, accrued interest, and fees at all times. Each financial transaction (disbursement, payment, fee charge, interest accrual, write-off) should generate ledger entries that can be integrated with the general ledger or core accounting system. Support **multi-currency accounting** if loans can be in different currencies [11] (with proper currency conversion and FX rate tracking for reporting). The LMS should be able to generate periodic **loan statements** for borrowers (showing transactions, balances, next due, etc.) and support inquiries like amortization tables, payoff quotes (how much to fully pay off today), etc.

- **Reports and Analytics:** Provide comprehensive reporting on the loan portfolio. Key metrics include: portfolio outstanding balances, interest income earned, delinquency reports (number of accounts in each DPD bucket), NPA/non-performing loan lists, prepayment statistics, etc. Include the ability to slice by product, region, partner, etc. The system should also support **regulatory reporting** needs (e.g., reports for credit bureaus or central bank, if applicable) and have an **audit trail** of all changes. Built-in analytics can help portfolio management – for example, monitoring concentration risks, calculating portfolio yield, or identifying trends in loan performance [12] [13]. Real-time dashboards for business users (showing originations, approvals, collections, etc.) are a plus.

- **Borrower Self-Service Portal:** In addition to back-office functions, include a customer-facing portal (web and mobile accessible) where borrowers can log in to **view their loan details**, upcoming due dates, outstanding amounts, etc. Borrowers should be able to make payments through this portal (integrate payment gateway), download statements, and even apply for new loans or submit service requests. The portal should send automated reminders for due payments and notifications for important events (like loan approval, or if a payment is missed). This enhances customer experience and reduces manual work for the lender [14] [15].

Product and Use-Case Coverage

Ensure the LOS/LMS platform can cater to the following lending domains, with any specialized functionality as noted:

- **Retail Loans (Consumer & Small Business):** Handle high-volume, standardized loans such as personal loans, home loans, auto loans, credit card loans, and small business loans. Emphasize automation and speedy processing in origination (with features like instant credit decisions for eligible applicants) and efficient mass-processing in servicing (like automated ACH payment collection, mass email reminders). The system should manage large numbers of relatively smaller accounts efficiently, with strong scalability.

- **Wholesale/Commercial Loans:** Support corporate loans, SME loans, project finance, and other complex facilities. These often require more complex origination workflows (detailed financial analysis, custom covenants, multiple decision checkpoints) and flexible servicing (interest-only periods, revolving credit lines, re-drawing facilities). The system should allow manual intervention and override at various stages (since big loans might need custom handling). Additionally, incorporate covenant tracking (monitor financial covenants, trigger alerts if a borrower breaches conditions) and facility management (multiple drawdowns, tranche management).

- **Co-Lending & Partnership Lending:** As described, accommodate loans jointly funded by multiple institutions. The platform should support both **joint origination** (sharing application data and credit decisions between partners) and **joint servicing** (splitting repayments). This is essential for partnership models like bank+fintech co-lending, where a certain percentage of the loan is contributed by each party. The system should make it transparent what each partner's stake is and ensure accurate accounting for each.

- **Supply Chain Finance:** Incorporate the needs of invoice factoring, bill discounting, purchase order finance, and dealer financing. These products often involve short-term credit to businesses secured by their receivables or payables. The system should link loans to specific assets (invoices, bills) and events (like goods delivery, invoice due date). Origination might be triggered by invoice uploads or data feeds; servicing might involve syncing with payments from buyers. Provide the ability to manage credit limits for repeat corporate clients and handle multiple short loans that roll over.

- **Securitization (PTC/DA transactions):** The software should be versatile to handle when the lender packages loans into securities or sells them. This involves tracking which loans (or what portion of loan portfolios) have been securitized or sold. The LOS/LMS should facilitate data export for due diligence and then mark those accounts accordingly. Post-securitization, the LMS can continue to service the loans but must **route cash flows to investors or new owners**. It should produce reports like pool collection reports, and track any credit enhancement or tranching details if needed. Essentially, it should not be a black box – provide transparency and controls to manage these structured finance actions within the platform, so the lender can manage retail, partnership, and securitized assets all in one system.

Technical & Non-Functional Requirements

- **Architecture & Deployment:** The application must be designed for both **on-premise and cloud-native deployments**. This implies a modular, containerizable architecture (e.g. microservices or a well-structured monolith) that can run on traditional servers or on cloud platforms with auto-scaling. Use of containerization (Docker/Kubernetes) for cloud is recommended, while also allowing installation on standalone servers for on-prem. Design the system to be **scalable horizontally** to handle a growing number of loans and transactions – for example, high-volume retail lending should be supported by scaling application servers and databases. Ensure the system can handle concurrent users (staff and customers) and high transaction throughput (e.g., processing thousands of payments per hour, etc.) without performance degradation.

- **Technology Stack:** Utilize **Python for the backend** business logic and services. Leverage appropriate frameworks (such as Django or Flask/FastAPI for web APIs, and possibly task queues like Celery for asynchronous jobs like batch interest accruals or report generation). The database should be a robust **relational DBMS** (e.g., PostgreSQL or MySQL) to ensure ACID compliance for financial transactions, with an ORM for data modeling. For full text search or large-scale analytics, additional stores (like ElasticSearch or a data warehouse) can be integrated as needed. The frontend (if custom-built) can be implemented with modern web frameworks (React, Angular, or Vue) for an admin interface and borrower portal, or the system can expose a REST/GraphQL API for integration with other front-ends or mobile apps. Ensure the design follows a **layered architecture** separating concerns (UI, API, business logic, data), making it maintainable and extensible.

- **Integration Capabilities:** Provide a rich set of **APIs and integration hooks**. The system should integrate with external services such as: credit bureaus (for credit reports), KYC/AML providers (for identity verification and sanctions screening), payment gateways and banking networks (for payment processing), accounting systems (to sync loan accounting entries), and CRM or customer support systems. Use RESTful APIs or webservices for real-time integrations, and support batch data import/export (e.g., CSV, Excel) for bulk operations or migration. Webhooks or callback mechanisms are useful for real-time updates (e.g., notify external systems when a loan is funded or a payment is received). For cloud deployment, all services should be stateless where possible to allow scaling, with session management handled via tokens or a distributed cache if needed.

- **Security & Compliance:** Implement strong **security controls** across the application. This includes authentication (with support for multi-factor authentication for users), role-based access control (different permissions for loan officers, underwriters, admins, etc.), and encryption of sensitive data at rest and in transit. All personal and financial customer data must be protected (using protocols like HTTPS, TLS encryption for data transport, and encryption of sensitive fields in the database). **Audit trails** are essential: log every important action (application approvals, data changes, payment postings) with timestamp and user info for compliance [16] [17]. The system must comply with relevant regulations in lending – e.g., GDPR for data privacy, and local financial regulations for data retention, borrower communication, and reporting. Include configurable business rules to ensure regulatory compliance (for example, interest calculations as per local law, or handling of customer consents). For cloud deployments, ensure isolation of data between tenants if offered as a SaaS to multiple lenders, and provide tools for data backup and recovery.

- **Performance & Reliability:** The system should be designed for **high availability**. Use clustering or failover strategies for critical components (e.g., a primary-secondary database or a clustered database setup; multiple app servers behind a load balancer). Target low latency for user actions – e.g., credit decisioning queries should return quickly (possibly via precomputed credit scoring or async processing for heavy tasks). Batch processes (like daily interest accrual, generation of bills, reports) should be optimized and scheduled in off-peak times or distributed if large. Provide monitoring and logging for all services (to track performance, errors, usage). The solution should also handle **large data volumes** – optimizing queries and using indexing to manage millions of loan records if needed. Archival strategies might be needed for very old closed loans to keep the live database performant (while still retaining history as required by law).

- **Modularity & Configurability:** The LOS and LMS should be modular in design but operate on a unified data model. This allows the same software to “double up” as both LOS and LMS, simply by enabling or disabling modules. Lenders who need only origination or only servicing should be able to use the relevant parts. The system should allow extensive configuration without code changes – e.g., defining new loan products, setting up approval workflows, configuring interest rates or fee rules, setting credit policy parameters – all through admin UI or config files. This makes the platform flexible to adapt to new products (such as new types of loans or partnership models) and new regulations.

- **User Experience:** Ensure the user interface is intuitive for all user types. Loan officers, underwriters, and operations staff should have a clear dashboard of their tasks (e.g., applications to review, loans needing funding, delinquent accounts to follow up). The system should support searching and filtering of loans, and viewing comprehensive loan details on one screen (including all application info, documents, payment history, etc.). For borrowers, the self-service portal should be mobile-responsive and easy to navigate, providing transparency into their loan status and enabling self-service wherever possible (payments, updates).

- **Testing & Quality:** Provide a complete test suite and support UAT for the platform. Given the financial nature, the system must be **accurate** – all calculations (interest, amortization, fees) need to be correct to the cent. Implement thorough unit tests for calculation functions (e.g., interest accrual, amortization generation), and integration tests for end-to-end flows (apply -> approve -> disburse -> make payments -> close loan). Also consider edge cases: leap year interest calculations, variances in day-count conventions, early payoff scenarios, etc., to ensure no scenario breaks the calculations. The system should also be tested for performance (simulate high load for retail volumes) and security (penetration testing, vulnerability scanning) before production use.

By addressing all the above requirements, the resulting LOS/LMS platform will be capable of supporting diverse lending products (from personal loans to complex co-lending and securitized portfolios) and managing the full loan lifecycle efficiently and accurately. The prompt above provides a blueprint for coding agents (like OpenAI Codex or similar) to implement the system with a **Python-based backend**, ensuring that all critical functionalities – from **repayment schedule generation, interest and principal tracking, prepayment and penalty handling, flexible repayment frequencies, to DPD (days past due) tracking and collections** – are covered in the software design and development [5] [8].

---

[1] Steps Involved in a Loan Origination System | Abrigo
https://www.abrigo.com/blog/basic-functions-loan-origination-system/

[2] [3] [4] [5] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] Loan Management Software: Features, Costs, Integrations
https://www.scnsoft.com/lending/loan-management-software

[6] [7] Hypercore
https://www.hypercore.ai/blog-article/how-does-a-loan-management-system-work