# Loan Management System Overview

## 1.Introduction

The Loan Management System (LMS) is a comprehensive platform designed to streamline the entire lifecycle of loan processing, from origination to repayment. Its primary purpose is to enhance efficiency and accuracy in managing loans, ensuring that both lenders and borrowers have a seamless experience. The LMS offers a range of features that cater to the unique needs of financial institutions, including automated workflows, real-time data analytics, and customizable reporting tools.

One of the key features of the Loan Management System is its ability to simplify loan management. By automating routine tasks such as application processing, document verification, and payment tracking, the system reduces the administrative burden on staff and minimizes the chances of human error. Borrowers benefit from a user-friendly interface that allows them to apply for loans online, track their application status, and manage their repayment schedules effortlessly.

Security is a critical aspect of the Loan Management System. It employs robust encryption and access control mechanisms to ensure that customer data remains confidential and protected from unauthorized access. Additionally, the system regularly undergoes security audits and updates to address potential vulnerabilities, thus fostering trust between lenders and their clients.

## 

## 2. System Architecture

The Loan Management System (LMS) is built upon a robust and scalable architecture that effectively integrates various components to deliver a seamless user experience. The architecture can be broadly divided into three main layers: the backend, the database, and the authorization part.

### 2.1 Architectural Diagram

### A screen shot of a cell phone Description automatically generated

### 2.2 Backend Implementation

The backend is implemented with Spring Boot, a powerful framework for building Java applications. It provides a robust server-side environment that handles business logic, data processing, and integration with external services. Spring Boot's microservices architecture allows for modular development, making it easier to maintain and scale the LMS as user demands evolve. The backend is responsible for processing loan applications, managing user accounts, and ensuring compliance with regulatory requirements.

### 

### 2.3 Database

H2 database serves as the relational database management system (RDBMS) for the LMS. H2 is very fast, open source, JDBC API and also embedded and server modes; disk-based or in-memory database. Additionally, it employs advanced indexing techniques to ensure efficient query performance, even with large datasets.

### 2.4 Authorization

To secure user access and protect sensitive data, the LMS utilizes spring security and basic authentication for authorization and identity management allowing users to authenticate once and gain access to multiple services within the system. With Spring security, the LMS maintains stringent access control policies, enabling only authorized personnel to manage sensitive loan information.

## 3. Modules

The Loan Management System (LMS) comprises several key modules that work synergistically to streamline loan operations. Among these, the Loan Management, Installment Management, and Authorization modules play critical roles in facilitating efficient loan processing and ensuring compliance with regulatory standards.

### 3.1 Loan Management Module

The Loan Management module serves as the core component of the LMS, responsible for overseeing the entire lifecycle of a loan. It encompasses functionalities such as application processing, approval workflows, and document management. This module allows lenders to track loan requests from submission to funding, ensuring that all necessary documentation is collected and verified. Additionally, it supports various loan products, allowing institutions to configure terms, interest rates, and repayment schedules tailored to different borrower profiles. By automating these processes, the Loan Management module reduces the time required for loan approvals and enhances overall operational efficiency.

### 3.2 Installment Management Module

The Installment Management module focuses specifically on the repayment aspect of loans. It allows users to configure repayment schedules based on the loan terms defined in the Loan Management module. This module calculates installment amounts, tracks payment due dates, and manages any late fees or penalties associated with missed payments. It also generates reminders for borrowers, ensuring timely repayments and reducing the risk of defaults. By providing a clear overview of repayment obligations, this module not only supports borrowers in managing their finances but also aids lenders in maintaining healthy loan portfolios.

### 3.3 Authorization Module

The Authorization module is essential for maintaining the security and integrity of the LMS. It manages user roles and permissions, ensuring that only authorized personnel can access sensitive information. By implementing a robust authentication mechanism, this module safeguards customer data and complies with regulatory requirements. It also supports audit trails, allowing institutions to monitor user activities and track changes made within the system. Through strict access controls and comprehensive reporting, the Authorization module plays a vital role in protecting the institution's assets and maintaining trust with customers.

## Technologies Used

The development of the Loan Management System (LMS) incorporates a variety of technologies that ensure a robust, secure, and efficient platform for managing loans. Each component of the system is carefully selected to optimize performance and enhance user experience.

### Backend Technologies

The backend of the LMS is powered by **Spring Boot**, a framework that simplifies Java application development. Spring Boot offers a range of features including dependency injection, security, and data access that are essential for building scalable applications. It also supports microservices architecture, allowing different components of the system to function independently while facilitating seamless integration.

### Database Choice

The system employs **H2** as its relational database management system (RDBMS). H2 is known its embedded and server modes; disk-based or in-memory databases and also very fast to integrate, open source. It also has browser based console application.

### Security Tools

For security, the LMS integrates **Spring Security** for authentication and authorization. Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements

### Build Tool

The development process is streamlined using **Maven**, a build automation tool that manages project dependencies and facilitates the build process for Java applications. Maven’s standardized project structure and extensive repository of plugins allow for efficient compilation, packaging, and deployment of the LMS.

### Testing Frameworks

To ensure the reliability and stability of the LMS, testing frameworks such as **JUnit** and **Mockito** are employed. JUnit is a widely used framework for unit testing Java applications, while Mockito is utilized for creating mock objects in tests. Together, these frameworks help verify that the system operates as intended and meets quality standards throughout the development lifecycle.

## Database Design

The database design for the Loan Management System (LMS) is structured to support the complex relationships between various entities involved in loan processing. A key component of this design is the Entity Relationship Diagram (ERD), which visually represents the relationships among the different tables in the database. The primary entities in the LMS include Customer, Loan, and LoanInstallment.

### Entity Relationship Diagram (ERD)

### 

A screenshot of a computer

Description automatically generated

### Key Tables

#### Customer Table

The Customer table is foundational to the LMS, storing vital information about borrowers. Key attributes include:

* **Id**: A unique identifier for each customer.
* **Name**: The full name of the customer.
* **Surname**: The email address used for communication and account verification.
* **CreditLimit**: The maximum amount of credit the customer can use.
* **UserCreditLimit**: The amount of credit the customer has already used.

This table enables the system to associate loans with specific customers and track their loan history.

#### Loan Table

The Loan table captures essential details about each loan issued. Important attributes include:

* **Id**: A unique identifier for each loan.
* **Customer**: A foreign key linking to the Customer table to identify the borrower.(ManyToOne)
* **LoanAmount**: The total amount of the loan.
* **NumberOfInstallment**: The number of installments the loan will be paid in.
* **CreateDate**: The date when the loan was issued.
* **isPaid**: Indicates whether the loan has been paid back.

This table facilitates tracking the lifecycle of each loan, including its terms and repayment status.

#### LoanInstallment Table

The LoanInstallment table is crucial for managing the repayment details of each loan. Key attributes include:

* **Id**: A unique identifier for each installment.
* **Loan**: The loan this installment belongs to. (ManyToOne)
* **Amount**: The amount due for each installment, which may vary based on payment schedules.
* **PaidAmount**: The amount of money the customer has already paid for this installment.
* **DueDate**: The date by which the installment should be paid.
* **PaymentDate**: The date when the installment was paid.
* **IsPaid:** Indicates whether the installment has been paid.

This table allows the LMS to generate reminders, track payments, and manage late fees, thereby promoting responsible borrowing and timely repayments.

## API Documentation

The Loan Management System (LMS) provides a set of API endpoints that facilitate interaction with its core functionalities. The following sections detail the available endpoints for creating loans, listing loans, listing installments, and paying loans, along with their request and response formats.

### Create Loan Endpoint

**Endpoint:** POST /api/loans

**Request Format:**

{  
 "customerId": "Long",  
 "amount": "BigDecimal",  
 "interestRate": "Double",  
 "numberOfInstallments": "Integer"  
}

**Response Format:**

{  
 "loanId": "Long",  
 "totalAmount": " BigDecimal ",  
 "isPaid": "Boolean"  
}

**Description:** This endpoint allows users to create a new loan. The request body must include details such as the customer ID, loan amount, interest rate, and loan duration. Upon successful creation, a loan ID is returned along with the status and a message indicating the result.

### List Loans Endpoint

**Endpoint:** GET /api/loans

**Request Parameters:**

* "customerId": "Long"
* "isPaid": "Boolean"
* "numberOfInstallments": "Integer"

**Response Format:**

[  
 {  
 " loanId ": "Long",  
 "loanAmount": "BigDecimal",  
 "isPaid": "Boolean",  
 "numberOfInstallments": "Integer"  
 }  
]

**Description:** This endpoint retrieves a list of all loans in the LMS. The response provides an array of loan objects, each containing essential details such as loan ID, customer ID, loan amount, interest rate, and current status.

### List Installments Endpoint

**Endpoint:** GET /api/loans/{loanId}/installments

**Response Format:**

[  
 {  
 "installmentId": "Long",  
 "amount": "BigDecimal",  
 "paidAmount": "BigDecimal ",  
 "dueDate": " LocalDate (YYYY-MM-DD)”  
 "paymentDate": " LocalDate (YYYY-MM-DD)”,  
 "isPaid": "Boolean"  
 }  
]

**Description:** This endpoint allows users to fetch all installments associated with a specific loan, identified by the loan ID. The response includes due dates, amounts, and payment statuses for each installment.

### Pay Loan Endpoint

**Endpoint:** POST /api/loans/{loanId}/pay

**Request Format:**

{  
 "amount": "number"  
}

**Response Format:**

{  
 "installmentsPaid": "Integer",  
 "totalPaid": "BİgDecimal",  
 "isLoanFullyPaid": "Boolean"  
}

**Description:** This endpoint processes payments for a specific loan. Users must specify the installment ID and the payment amount. The response indicates the success or failure of the payment, along with a corresponding message.

## Aspects and Authorization

In the Loan Management System (LMS), authorization is a vital aspect that ensures the security and integrity of sensitive data. It governs who can access specific functionalities within the system, thereby safeguarding customer information and maintaining compliance with regulatory requirements. The framework for authorization primarily revolves around the AuthorizationAspect, which encapsulates the logic for determining access rights based on user roles and customer IDs.

The **AuthorizationAspect** acts as an interceptor that checks if a user has the necessary permissions to execute certain actions. This aspect is critical for implementing security policies throughout the system, and it operates by leveraging annotations to define access control at various levels. By employing aspect-oriented programming principles, the AuthorizationAspect can cleanly separate authorization logic from business logic, thus enhancing code maintainability and readability.

One of the key elements of this authorization mechanism is the use of the **@AuthorizeAction** annotation. This custom annotation is applied to methods within the service layer to indicate that specific authorization checks should be performed before method execution. When a method marked with @AuthorizeAction is called, the AuthorizationAspect intercepts the call and retrieves the necessary context, such as the authenticated user's role and the relevant customer ID. This enables the system to determine whether the user has the right to perform the requested action based on their role and the associated customer data.

The logic for determining access based on customer IDs is particularly significant in ensuring that users can only access information pertinent to their roles. For instance, a loan officer may be authorized to view and manage loans for customers within a specific branch, while a customer service representative might only have access to their assigned clients. By incorporating customer IDs into the authorization checks, the LMS enforces strict access controls that prevent unauthorized access to sensitive data, fostering a secure environment for both lenders and borrowers.

## Unit Testing

Unit testing is a crucial aspect of the development process for the Loan Management System (LMS), ensuring that individual components function correctly and meet quality standards. The primary strategy for unit testing the LMS involves testing the smallest testable parts of the application, typically methods or functions, in isolation from the rest of the system. This approach helps identify issues early in the development cycle, reducing the cost and effort associated with fixing defects later.

### Types of Tests Conducted

The types of tests conducted in the LMS include:

1. **Unit Tests**: These tests focus on individual methods or classes, verifying that they perform as expected. For instance, a unit test might check whether the Loan Management module correctly calculates interest based on a given loan amount and interest rate.
2. **Integration Tests**: While unit tests verify small components, integration tests check how different modules interact with each other. For example, testing the interaction between the Loan Management and Installment Management modules emphasizes that data flows correctly between them.

### Tools Used for Coverage Analysis

To ensure comprehensive test coverage, the LMS utilizes several tools:

* **JUnit**: This framework is widely used for writing unit tests in Java. It provides annotations and assertions that facilitate the creation and execution of tests.
* **Mockito**: This mocking framework allows developers to create mock objects for testing interactions between components without relying on their actual implementations.

### Goals for Test Coverage

The primary goals for test coverage in the LMS are:

* **Achieve High Coverage**: Aim for at least 80% code coverage to ensure that most of the application logic is tested. This target helps minimize the risk of undetected bugs in production.
* **Identify Critical Paths**: Focus on testing critical paths, particularly in the Loan Management and Payment processing functionalities, as these are central to the system’s operation.
* **Continuous Integration**: Integrate unit tests into the continuous integration pipeline, enabling automated test execution whenever changes are made, thus maintaining code quality throughout development.

## Deployment

Deploying the Loan Management System (LMS) involves several steps to ensure both the backend and frontend applications are built, configured, and deployed correctly. Below are the deployment steps along with the necessary environment setup requirements.

### Environment Setup

1. **Prerequisites**: Ensure that the following software is installed on your machine:
   * **Java Development Kit (JDK)**: Version 17 or higher for building the backend.
   * **Maven**: For managing the backend build process.
   * **H2**: Installed and running for the database.
2. **Database Configuration**:
   * No further configuration required for H2 database, it is embedded and very fast to integrate.

### Building the Backend

1. **Clone Repository**: Clone the LMS backend repository from the version control system (e.g., GitHub).

* git clone https://github.com/yourusername/lms-backend.git  
  cd lms-backend

1. **Package the Application**: Use Maven to build the application.

* mvn clean package

1. **Run the Application**: Start the backend service.

* java -jar target/lms-backend.jar

Testing the API

Postman:



Please see the attached postman collection above.

Open API / Swagger UI

URL: http://localhost:8090/swagger-ui/index.html

A screenshot of a computer

Description automatically generated

Final Steps

1. **Integration**: Ensure that the frontend is correctly configured to communicate with the backend API endpoints. Update the API URL in the frontend environment configuration as necessary.
2. **Testing**: After deployment, conduct thorough testing to ensure that both applications are functioning as expected and that all integrations are seamless.

By following these steps, you will successfully deploy the Loan Management System, making it available for use by lenders and borrowers.

## Future Enhancements

As we look ahead, there are several potential enhancements that could be implemented in the Loan Management System (LMS) to further improve its functionality, user experience, and overall efficiency. These enhancements aim to address evolving user needs and industry trends, ensuring that the LMS remains competitive and valuable to financial institutions and their clients.

### Automated Payment Scheduling

One significant enhancement could be the integration of an automated payment scheduling feature. This would allow borrowers to set up recurring payments based on their preferred schedule, reducing the risk of late payments. By automating the payment process, the LMS can send reminders and notifications to borrowers, ensuring they are aware of upcoming payments and thus promoting timely repayments.

### Advanced Loan Filtering Options

Another valuable enhancement is the introduction of advanced loan filtering options. This feature would enable lenders to apply multiple criteria when searching for loans, such as loan amount, interest rate, term length, and borrower credit score. Enhanced filtering capabilities would streamline the decision-making process for lenders, allowing them to quickly identify suitable loan candidates and tailor their offerings to meet specific borrower needs.

### Financial Reporting Capabilities

Enhancing the financial reporting capabilities of the LMS could provide stakeholders with deeper insights into loan performance and risk management. Implementing customizable reporting tools would allow users to generate comprehensive reports on various metrics, including default rates, repayment patterns, and portfolio performance. This data-driven approach can aid in strategic planning and compliance with regulatory requirements.

### Multi-Currency Support

With the increasing globalization of financial services, adding multi-currency support to the LMS would be a significant advantage. This feature would allow users to manage loans in different currencies, catering to international clients and expanding the market reach of financial institutions. Multi-currency support would also include real-time exchange rate updates and conversion tools, facilitating seamless transactions for borrowers and lenders alike.

By considering these potential future enhancements, the Loan Management System can continue to evolve and meet the dynamic needs of its users, ultimately driving better financial outcomes for both lenders and borrowers.