Loan Management System - Project Report

# 1. Problem Statement

The Loan Management System is a GUI-based application developed in Java, backed by a MySQL database, aimed at simplifying the loan application and approval process for users and bank administrators (admins). The system allows users to apply for various types of loans—Home Loan, Car Loan, and Education Loan—and allows admins to manage user data, loan applications, and bank offers. The system enforces an eligibility condition that the monthly EMI must not exceed 50% of the applicant's salary. Interest rates and loan terms are fetched from the database based on loan type.

# 2. Steps / Flow of Work

## 2.1. GUI Design and Pages

- Landing Page: Login form with a link to the Signup page.

- Signup Page: Accepts:  
 - Username  
 - First Name  
 - Last Name  
 - Email  
 - Phone Number  
 - Address  
 - Password  
 - Two buttons: Sign up as User or Admin

## 2.2. Roles and Functionalities

* Admin:
* - Login using admin credentials.
* - View all loan applications.
* - Approve or reject loans based on eligibility.
* - Manage users (view, update or delete).
* - Add/update bank offers for different loan types.
* User:
* - Login with registered credentials.
* - View profile with all personal details.
* - Apply for a loan (Home, Car, or Education).
* - Select best bank offer from available options.
* - Apply only if EMI is less than 50% of salary.
* - Use EMI Calculator tab to estimate monthly payments.
* - View status of submitted loan applications.

## 2.3. Database (MySQL)

Tables:

* - Users (user\_id, username, password ,role,first name, last name, email, phone\_number, address, credit\_score, salary,account\_status,created\_at,last\_login)
* - Loans (loan\_id, user\_id,bank\_id, loan\_type, amount,interest\_rate, tenure,status,emi,application\_date,processed\_date)
* - Bank(bank\_id,bank\_name,home\_loan,car\_loan\_rate,education\_loan\_rate,created\_at)

# 3. Results

The system works as intended with the following features:  
- GUI-based login and signup flow.  
- Role-based access (Admin/User).  
- Dynamic loan application system.  
- Real-time EMI calculation.  
- Admin management for users and offers.  
- Data persistence and retrieval using MySQL.

# 3.1 System Architecture

#### Technology Stack

* **Programming Language**: Java (Swing for UI)
* **Database**: MySQL
* **Frameworks & Tools**: Maven for dependency management and flatlaf for UI.

#### Backend Components:

* **LoanController.java**: Manages loan-related operations.
* **AdminController.java**: Handles admin operations.
* **LoanDAO.java**: Manages database interactions for loans.
* **BankController.java**: Handles bank-related business logic.
* **BankDAO.java**: Manages database interactions for banks.
* **DBConnection.java**: Handles MySQL database connection.

#### Frontend Components:

* **LandingPage.java**: Main UI for users.
* **AdminDashboard.java**: Admin control panel with multiple management features.
* **BankManagementFrame.java**: UI for managing bank-related operations.
* **ViewAllLoansDialog.java**: Displays all loans in a single window.
* **ManageLoan.java**: Provides options for loan updates and processing.
* Various forms for loan application and admin controls.

# 4. Discussion

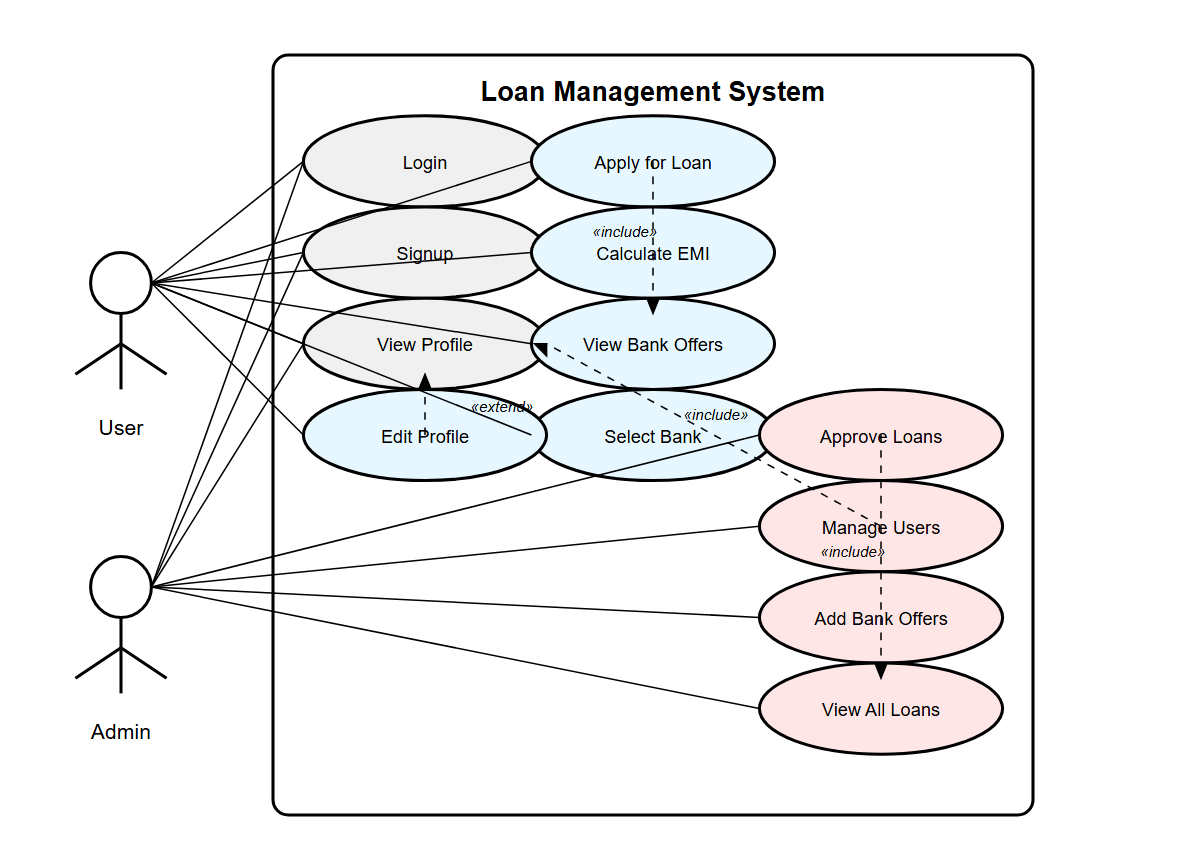
## 4.1. Limitations

* - No real-time document verification or KYC integration.
* - No email/SMS notifications
* -No Automated Credit Scoring: The system lacks credit score integration, making risk assessment less efficient.
* - Limited to three loan types.
* - Admin decisions are manual—no auto-evaluation algorithms beyond EMI check.
* - Basic security (e.g., no encryption for stored passwords)

## 4.2. Future Scope

* - Integrate secure user authentication with encryption (e.g., using bcrypt).
* - Add auto-approval using AI/machine learning based on user data and risk profile.
* - Support for additional loan types (business, personal).
* - Add digital document upload and KYC validation.
* - Generate loan agreement PDFs and payment schedule.
* - Implement notification system (email/SMS alerts).
* - Add data analytics for loan trends and user behavior.

# 5. Use Case Diagram



# 6. Database Schema

CREATE DATABASE IF NOT EXISTS LoanManagementSystem;

USE LoanManagementSystem;

-- Users Table with all enhancements

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(50) NOT NULL,

role ENUM('user', 'admin') NOT NULL,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

phone\_number VARCHAR(15),

address VARCHAR(255),

credit\_score INT DEFAULT 600,

salary DECIMAL(12, 2) DEFAULT NULL COMMENT 'Monthly salary in local currency',

account\_status ENUM('active', 'suspended', 'blocked') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login TIMESTAMP NULL

);

-- Banks Table (new)

CREATE TABLE Banks (

bank\_id INT AUTO\_INCREMENT PRIMARY KEY,

bank\_name VARCHAR(100) NOT NULL UNIQUE,

home\_loan\_rate DECIMAL(5, 2) NOT NULL,

car\_loan\_rate DECIMAL(5, 2) NOT NULL,

education\_loan\_rate DECIMAL(5, 2) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Loans Table with all enhancements

CREATE TABLE Loans (

loan\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

bank\_id INT,

loan\_type ENUM('home', 'car', 'education') NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

interest\_rate DECIMAL(5, 2) NOT NULL,

tenure INT NOT NULL COMMENT 'Loan tenure in months',

status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',

emi DECIMAL(12, 2) COMMENT 'Calculated EMI amount',

application\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

processed\_date TIMESTAMP NULL,

processed\_by INT NULL COMMENT 'Admin user ID who processed this loan',

remarks VARCHAR(255) COMMENT 'Admin remarks for approval/rejection',

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (bank\_id) REFERENCES Banks(bank\_id),

FOREIGN KEY (processed\_by) REFERENCES Users(user\_id)

);

-- Insert sample admin user

INSERT INTO Users (username, password, role, first\_name, last\_name, email, account\_status)

VALUES ('admin', 'admin123', 'admin', 'System', 'Administrator', 'admin@loansystem.com', 'active');

-- Insert sample banks

INSERT INTO Banks (bank\_name, home\_loan\_rate, car\_loan\_rate, education\_loan\_rate)

VALUES

('State Bank of India', 8.4, 9.2, 10.1),

('HDFC Bank', 8.6, 9.5, 10.3),

('ICICI Bank', 8.7, 9.6, 10.5);

-- Insert sample user

INSERT INTO Users (username, password, role, first\_name, last\_name, email, credit\_score, salary)

VALUES ('john\_doe', 'password123', 'user', 'John', 'Doe', 'john@example.com', 720, 75000.00);

# Project Created by :

**Name : Abhinav Singh | Roll No : CSE/23004/1052**

**Name : Abhishek Kumar Singh | Roll No : CSE/23008/1056**

**Github :** https://github.com/Abhinav359810/Loan-Mangement.git