

**RAJALAKSHMI ENGINEERING  
COLLEGE  
RAJALAKSHMI NAGAR, THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CB23332  
SOFTWARE ENGINEERING LAB**

**Laboratory Record Note Book**

Name : .....

Year / Branch / Section : .....

Register No. : .....

Semester : .....

Academic Year : .....

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**  
**RAJALAKSHMI NAGAR, THANDALAM – 602-105** **BONAFIDE**  
**CERTIFICATE**

**NAME:** \_\_\_\_\_ **REGISTER NO.:** \_\_\_\_\_

**ACADEMIC YEAR:** 2024-25 **SEMESTER:** III **BRANCH:** \_\_\_\_\_ B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

**CB23332-SOFTWARE ENGINEERING - Laboratory** during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on \_\_\_\_\_

Internal Examiner

Department of CSBS/CB23332

External Examiner



## INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagramt		
10.	Class Diagram		

---

Department of CSBS/CB23332



---

## 1. PREPARING PROBLEM STATEMENT

### **Aim :**

The Personal Loan Management System aims to provide a user-friendly platform for effectively managing personal loans by Centralizing loan information and details. Automating payment tracking and reminders. Offering financial insights into repayment progress and costs. Supporting budgeting and financial planning. Ensuring an intuitive experience across devices. This system will empower users to take control of their finances and improve their overall financial health.

### **Algorithm :**

#### User Management

User Registration and Login: Allow users to create accounts and log in securely.

Profile Management: Enable users to update personal information and security settings.

#### 2. Loan Management

Loan Entry: Allow users to input details of multiple loans (amount, interest rate, terms, etc.).

Loan Overview: Provide a summary of all loans with key information like balances and due dates.

#### 3. Payment Tracking

Payment Schedule: Display upcoming payment dates and amounts.

Payment Logging: Enable users to record payments made towards their loans.

Automated Reminders: Send notifications for upcoming payments.

#### 4. Financial Insights

Repayment Progress: Show visual representations of repayment timelines and remaining balances.

Interest Tracking: Provide insights on total interest paid and potential savings from extra payments.

#### 5. Budgeting and Planning

Budget Calculator: Help users create a budget considering their loan payments.

Repayment Strategies: Offer suggestions for faster repayment and debt reduction.

#### 6. Reports and Analytics

Financial Reports: Generate reports on loan status, payment history, and overall financial health.

Custom Alerts: Allow users to set personalized alerts for key financial events.

#### 7. Admin Dashboard (Optional)

User Management: Admin tools to manage user accounts and permissions.

#### 8. Security Features

Data Encryption: Ensure user data is protected and secure.

Authentication Protocols: Implement multi-factor authentication for enhanced security.

## **INPUTS FOR THE SMART LIBRARY MANAGEMENT SYSTEM:**

### **1. Borrower Information**

- **Data:** Name, address, date of birth, contact details, employment status, income details, and credit history.

- 
- **Purpose:** Used to assess eligibility, calculate loan terms, and determine repayment capacity.

## 2. Loan Application Details

- **Data:** Loan amount requested, loan tenure, type of loan, purpose of loan (e.g., personal, medical, education).
- **Purpose:** Defines the loan parameters that the borrower is applying for, which helps in loan approval and processing.

## 3. Financial Data

- **Data:** Bank account details for disbursement, monthly income, liabilities, existing loans, and expenses.
- **Purpose:** Used to assess financial stability, repayment capacity, and calculate the loan-to-income ratio.

## 4. Credit Score Data

- **Data:** Credit score fetched from external credit bureaus (e.g., Experian, Equifax).
- **Purpose:** Used to evaluate the borrower's creditworthiness and determine loan approval or rejection based on predefined criteria.

### Problem:

Current personal loan management systems face significant challenges, including fragmented information, lack of automation, and limited insights. Users often have to manage loans across multiple platforms, complicating the tracking of payments, balances, and due dates. Additionally, many systems do not provide automated reminders for upcoming payments, leading to missed deadlines and late fees. Furthermore, users typically lack meaningful analytics and visualizations to understand their repayment progress and overall financial health, making it difficult to make informed financial decisions.

### Background:

The issue with personal loan management comes from more people using loans for different financial needs, leading to multiple loans with different terms to track. Traditional methods, like spreadsheets, often don't work well and leave borrowers confused. Many existing systems are hard to use, lack payment reminders, and don't provide helpful insights, resulting in missed payments and credit problems. As borrowing becomes more common, there's a clear need for a simple, easy-to-use solution to help people manage their loans. This shows why a good personal loan management system is important.

### Relevance:

The Personal Loan Management System will help users manage their loans better by providing a central place to view all their loans and reducing late payments with automated reminders. Users will enjoy a simple interface and gain a clearer understanding of their repayment progress and interest costs. Budgeting tools will support effective financial planning, while strong security will protect their data. Educational resources will improve financial knowledge, and customizable alerts will allow for a personalized experience, leading to better overall financial health and confidence in managing loans.



---

Objectives:

1. Centralized Loan Tracking: Provide a single platform for managing all personal loans.
  2. Automated Reminders: Implement notifications for upcoming payments to avoid late fees.
  3. User-Friendly Interface: Create an intuitive design for easy navigation.
  4. Financial Insights: Offer clear analytics on repayment progress and interest paid.
  5. Budgeting Tools: Integrate features to help users plan finances around loan payments.
  6. Secure Data Management: Ensure strong security measures to protect user data.
  7. Educational Resources: Provide tools to enhance financial literacy.
  8. Customizable Alerts: Allow users to set personalized notifications based on their needs.
- These objectives aim to help users manage their loans effectively and improve their financial health.

**Result :**

The result of the **Personal Loan Management System** project is a fully automated, secure platform that streamlines the entire loan lifecycle—from application submission and credit assessment to disbursement and repayment tracking. It enables efficient management of borrower and lender data, ensuring fast loan processing, real-time transaction updates, and seamless integration with external services like payment gateways and credit scoring agencies. The system enhances user experience with intuitive interfaces, while maintaining high security and regulatory compliance standards. Overall, it improves the efficiency, accessibility, and security of personal loan management.

---

**Ex.NO. 2 : Write the software requirement specification document**

**Aim :**

The aim of the Personal Loan Management System is to empower users to take control of their financial health by providing an easy-to-use interface for managing their loans, facilitating timely payments, and offering insights into their financial status

**Algorithm :****1.1] Purpose**

The purpose of this Software Requirements Specification (SRS) document is to outline the functional and non-functional requirements for the Personal Loan Management System. This system aims to provide users with an effective way to manage their personal loans, track payments, and enhance their financial literacy.

**1.2] Scope**

The system will allow users to manage multiple personal loans from various lenders in a centralized platform. It will include features for loan tracking, payment reminders, budgeting tools, and educational resources.

**1.3] Document conventions**

help ensure consistency and clarity in project documentation. Here's a simplified summary:

**1. General Guidelines:**

- Use clear, simple language.
- Keep documents consistent in formatting, terminology, and structure.
- Include version numbers, dates, and author/approver info.

**2. Document Structure:**

- **Cover Page:** Include title, version, date, and author info.
- **Table of Contents:** For easy navigation.
- **Headings/Subheadings:** Use clear titles with consistent formatting.
- **Font & Spacing:** Use readable fonts (e.g., Arial), with 1.15 or 1.5 line spacing.

**3. Terminology:**

- Be consistent with terms like "Loan Application," "Borrower," "Lender," etc.

**4. Diagrams:**

- Label diagrams with titles and numbers (e.g., *Figure 1.1*).
- Ensure diagrams are clear and easy to understand.

**5. Version Control:**

- Keep track of changes and document history with version numbers and dates.

## 6. References:

- Use proper citations for external sources and internal references.

## 7. Code Formatting:

- Use monospaced fonts for code snippets and provide clear comments.

## 8. Review and Approval:

- Documents should be reviewed and approved by relevant stakeholders before finalization.

### 1.4]PROJECT SCOPE:

The system will allow users to manage multiple personal loans from various lenders in a centralized platform. It will include features for loan tracking, payment reminders, budgeting tools, and educational resources.

### Product Perspective

The system will be a web-based application accessible on both desktop and mobile devices, allowing users to manage their loans anytime and anywhere.

## Classes and Characteristics for the Personal Loan Management System

In the Personal Loan Management System (PLMS), different classes (or entities) represent key components of the system. Each class will have specific characteristics (or attributes) and methods (or functions) that define its behavior and interaction with other parts of the system. Below is a breakdown of the main classes and their characteristics.

### 1. Class: Borrower

Characteristics (Attributes):

BorrowerID: Unique identifier for each borrower.

Name: Full name of the borrower.

DateOfBirth: Birthdate to calculate age (for eligibility).

Email: Contact email for communication and notifications.

PhoneNumber: Contact phone number.

Address: Physical address of the borrower.

AnnualIncome: Annual income of the borrower.

CreditScore: Borrower's credit score for eligibility assessment.

LoanApplications: List of loan applications submitted by the borrower.

ActiveLoan: Reference to any active loan associated with the borrower.

KYCStatus: Indicates whether the borrower has completed Know Your Customer (KYC) verification.

Methods:

ApplyForLoan(): Allows the borrower to submit a loan application.

TrackLoanStatus(): Allows the borrower to track the current status of their loan application.

MakeRepayment(): Allows the borrower to make a repayment towards the loan.

GetLoanBalance(): Retrieves the remaining balance of the loan.

### 2. Class: Loan

Characteristics (Attributes):

LoanID: Unique identifier for each loan.

---

LoanAmount: Total amount borrowed by the borrower.

InterestRate: Interest rate applied to the loan.

LoanTerm: Duration of the loan in months or years.

StartDate: The date the loan was disbursed.

EndDate: The date the loan is expected to be fully repaid.

EMI: The equated monthly installment amount.

RepaymentSchedule: List of scheduled payments with dates and amounts.

LoanStatus: Current status of the loan (e.g., pending, approved, disbursed, closed).

OutstandingAmount: Amount left to be repaid on the loan.

Methods:

CalculateEMI(): Calculates the EMI based on loan amount, interest rate, and tenure.

UpdateLoanStatus(): Updates the loan status (e.g., from pending to approved).

GenerateRepaymentSchedule(): Generates the repayment schedule with dates and amounts.

TrackOutstandingAmount(): Tracks the remaining balance on the loan.

CloseLoan(): Marks the loan as closed when fully repaid.

### 3. Class: Lender

Characteristics (Attributes):

LenderID: Unique identifier for each lender (e.g., bank or financial institution).

LenderName: Name of the lending institution.

LoanPortfolio: List of loans issued by the lender.

LoanApprovalCriteria: Set of criteria to evaluate loan applications (e.g., credit score, income).

AvailableFunds: Total available funds the lender can lend to borrowers.

Methods:

ReviewLoanApplication(): Reviews and evaluates borrower applications based on internal approval criteria.

ApproveLoan(): Approves a loan application for disbursement.

RejectLoan(): Rejects a loan application based on evaluation.

TrackRepayments(): Tracks the repayment progress of each loan.

GenerateLoanReport(): Generates reports on loan status, repayments, and financial performance.

### 4. Class: LoanApplication

Characteristics (Attributes):

ApplicationID: Unique identifier for the loan application.

BorrowerID: Reference to the borrower applying for the loan.

LoanAmountRequested: Amount the borrower is requesting to borrow.

ApplicationDate: The date the application was submitted.

ApprovalStatus: Status of the loan application (e.g., pending, approved, rejected).

LoanType: Type of loan being applied for (e.g., personal, emergency).

Methods:

SubmitApplication(): Allows the borrower to submit a loan application.

UpdateStatus(): Updates the loan application status (e.g., approved, rejected).

SendApprovalNotification(): Sends an approval notification to the borrower once the loan is approved.

CheckEligibility(): Verifies if the borrower meets eligibility criteria for the loan.

### 5. Class: Payment

Characteristics (Attributes):

PaymentID: Unique identifier for each payment transaction.

LoanID: Reference to the loan for which the payment is being made.

PaymentAmount: The amount paid towards the loan.

PaymentDate: The date the payment was made.

PaymentMethod: Method used for the payment (e.g., credit card, bank transfer, online payment).

---

OutstandingAmountAfterPayment: The loan balance after the payment has been made.

Methods:

MakePayment(): Records the payment made by the borrower.

GenerateReceipt(): Generates a payment receipt for the borrower.

UpdateLoanBalance(): Updates the loan's outstanding balance after a payment is made.

6. Class: Admin

Characteristics (Attributes):

AdminID: Unique identifier for the system administrator.

Name: Name of the admin user.

Role: Role of the admin (e.g., Super Admin, Loan Processor).

Permissions: List of system permissions assigned to the admin.

Methods:

ManageUserAccounts(): Creates, updates, or deletes borrower and lender accounts.

GenerateSystemReports(): Generates overall system reports, including user activity and loan metrics.

ConfigureLoanSettings(): Configures loan-related settings like interest rates, fees, and approval criteria.

MonitorSecurity(): Monitors and updates security measures for the system.

7. Class: Notification

Characteristics (Attributes):

NotificationID: Unique identifier for each notification.

UserID: Reference to the user receiving the notification (borrower, lender, or admin).

Message: The content of the notification.

NotificationType: Type of notification (e.g., Loan Status Update, Payment Reminder).

Timestamp: The date and time the notification was sent.

Methods:

SendNotification(): Sends a notification to the user based on the type and content.

ViewNotifications(): Allows the user to view all their notifications.

MarkAsRead(): Marks a notification as read.

8. Class: PaymentGateway

Characteristics (Attributes):

GatewayID: Unique identifier for each payment gateway.

Name: Name of the payment gateway (e.g., Stripe, PayPal).

TransactionFee: Fee charged per transaction by the gateway.

Status: Current operational status of the payment gateway (e.g., active, inactive).

Methods:

ProcessPayment(): Processes a payment through the gateway.

RefundPayment(): Initiates a refund for a failed or reversed payment.

ValidatePayment(): Validates the payment details before proceeding with the transaction.

## **FEATURES:**

Loan Application Management

Online Application Submission: Borrowers can fill out and submit loan applications through a web interface.

Document Upload: Borrowers can upload required documents (e.g., ID proof, income proof).

Pre-Qualification Check: Automatic assessment of eligibility based on predefined criteria like credit score, income, and loan amount.

2. Loan Approval & Disbursement

Automated Loan Approval: The system auto-evaluates loan applications using defined business rules (e.g., credit score, income, and other parameters).

Manual Approval Option: Lenders can manually review and approve or reject applications.

---

Loan Disbursement: Once approved, the loan amount is transferred to the borrower's bank account via integration with payment systems.

### 3. Repayment Management

EMI Calculation: Automatic calculation of monthly repayments (EMI) based on principal, interest rate, and loan tenure.

Flexible Repayment Options: Borrowers can make full or part payments and choose the frequency of repayments.

Payment Reminders: Automated reminders via email or SMS to alert borrowers about upcoming payments.

### 4. Loan Status Tracking

Real-time Loan Status: Borrowers can track their loan status (e.g., approved, disbursed, pending).

Outstanding Balance: Borrowers can view the current outstanding balance and remaining tenure.

Repayment History: View detailed payment history including past EMIs, outstanding amounts, and interest paid.

### 5. Interest & Fees Calculation

Interest Calculation: Automated calculation of interest based on loan principal and rate, using standard or reducing balance methods.

Late Payment Fees: System tracks late payments and applies fees where applicable.

Prepayment Calculation: Borrowers can make early payments, and the system calculates any penalty or interest adjustments.

## User Management and Roles

Borrower Role: Borrowers can apply for loans, track their loan status, view repayment schedules, and make payments.

Lender Role: Lenders can review loan applications, approve loans, manage disbursements, and monitor repayments.

Admin Role: Admin users can manage the platform, configure user roles, generate reports, and monitor system activity.

## Security and Compliance

Data Encryption: Sensitive data (e.g., personal, financial details) will be encrypted to ensure privacy and security.

## OPERATING ENVIRONMENT:

The **Personal Loan Management System (PLMS)** requires the following environment for optimal performance:

### Hardware:

- **End Users:** Desktop, laptop, or mobile with at least 4 GB RAM, a dual-core processor, and 10 GB of free storage.
- **Server-Side:** Multi-core CPU, 16 GB RAM (32 GB recommended), SSD storage (500 GB+), and high-speed internet (1 Gbps).

### Software:

- **Operating System:** Linux (Ubuntu, CentOS) preferred; Windows Server is an alternative.
- **Web Server:** Apache or Nginx.
- **Database:** MySQL or PostgreSQL.
- **Application Framework:** Django (Python), Spring (Java), or Ruby on Rails.

---

**Network:**

- Stable, high-speed internet for hosting.

This setup ensures the system runs securely, efficiently, and can scale with growing usage.

**Design and Implementation Constraints**

**Security:** Must comply with data protection regulations (e.g., GDPR, PCI-DSS) to ensure sensitive financial data is securely stored and transmitted.

**Scalability:** The system should handle increasing numbers of users and transactions without significant performance degradation.

**Integration:** Must integrate with third-party services (payment gateways, credit score providers) using standardized APIs.

**Usability:** The interface must be intuitive for both borrowers and lenders, requiring minimal training.

**Performance:** The system should support fast transaction processing and real-time updates, especially for loan applications and repayments.

**Budget and Time:** Limited resources may affect feature scope, prioritizing core functionalities for initial launch.

**Key Assumptions and Dependencies for the Personal Loan Management System****Important Assumptions**

**Internet Access:** All users (borrowers, lenders, and admins) will have stable internet access to use the web-based system.

**Payment Gateway Availability:** The system will rely on third-party payment gateways (e.g., PayPal, Stripe) to process loan disbursements and repayments.

**Credit Scoring Services:** The system will use external credit bureaus (e.g., Experian) for credit score retrieval to assess loan eligibility.

**Security Protocols:** The system will implement standard security measures (SSL encryption, two-factor authentication) to ensure data privacy and transaction security.

**Cloud Hosting:** The system will be hosted on reliable cloud infrastructure (e.g., AWS, Google Cloud) to ensure scalability and availability.

**Important Dependencies**

**Payment Gateway Integration:** The system depends on third-party payment gateways to process financial transactions such as loan repayments and disbursements.

**External Credit Scoring:** The loan approval process depends on real-time credit scores from external services like Experian or Equifax.

**Cloud Infrastructure:** The system's performance and scalability depend on the cloud service provider (e.g., AWS, Azure) to handle growing traffic and data storage.

**Regulatory Compliance:** The system depends on existing financial regulations (e.g., GDPR, PCI-DSS) for data protection and transaction handling.

**SYSTEM FEATURES****Functional Requirements:**

Users can manage multiple loans, track payments, and access financial insights and educational resources.

**Non-Functional Requirements:**

The system must ensure high performance, usability, security, and scalability to accommodate a growing user base.

**USER INTERFACES:****Hardware Interface**

Purpose: Enables communication between client devices and servers.

Protocol: HTTP/HTTPS for secure web access.

Data: User and loan-related data exchanged between clients and servers.

Software Interfaces

Purpose: Integrates with external systems and services.

APIs: RESTful APIs for payment gateways, credit scoring, and banking services.

Data: Loan application, payment details, and user information passed between systems.

### Software Interfaces

Purpose: Enable interaction with external systems and services.

APIs: RESTful APIs for:

Payment Gateways: Process loan repayments and disbursements.

Credit Scoring: Fetch borrower credit scores from third-party services.

Banking Systems: Disburse loan funds and manage repayments.

Authentication: Support for OAuth, Google/Facebook login.

Data: User credentials, loan details, credit score data, transaction records.

### Distributed Database Diagram (PlantUML Code)

```
puml
Copy code
@startuml
actor Client
node "Load Balancer" {
    [Web Server]
    [Mobile Server]
}
node "Data Center 1" {
    database "Primary Database" as DB1 [App Server 1]
}
node "Data Center 2" {
    database "Secondary Database" as DB2 [App Server 2]
}
Client --> "Web Server" : Request
Client --> "Mobile Server" : Request
"Web Server" --> "App Server 1" : API Call
"Mobile Server" --> "App Server 2" : API Call
"App Server 1" --> DB1 : Read/Write
"App Server 2" --> DB2 : Read/Write
DB1 --> DB2 : Data Replication
@enduml
```

### Explanation:

- **Client** interacts with the system via web and mobile interfaces.
- **Load Balancer** directs requests to the appropriate servers.
- **App Servers** handle business logic and connect to **Primary** and **Secondary Databases**.
- **Data Replication** ensures high availability and keeps databases in sync.

### RESULT:

The SRS for personal loan management system is successfully constructed.



### 3. ENTITY RELATIONSHIP MODEL

#### **Aim :**

To Draw the Entity Relationship Diagram for personal loan management.

#### **Algorithm :**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types Step

4: Mapping of Binary 1: N Relationship Types.

Step 5: Mapping of Binary M: N Relationship Types.

Step 6: Mapping of Multivalued attributes.

#### **Input :**

# Entities

# customer

# loan officer

# bank

#loan type

#loan

#payment

#### **2. Entity Relationship Matrix**

Entities	customer	Loan officer	Loan	bank	Payment	Loan type
customer	-	1:M	1:M	1:M	-	1:M
Loan officer	-	-	1:M	1:M	-	1:M
Loan	-	M:1	M:1	M:1	M:1	M:1
Bank	-	1:M	1:M	1:M	-	1:M

payment	-	1:M	1:M	1:M	M:1	1:M
---------	---	-----	-----	-----	-----	-----

### 3. Primary Keys:

Customer\_id , officer\_id , bank\_id , loan\_typeid , loaned , paymentid.

### 4. Attributes

Customer: customer id,firstname,lastname,address,phone number, email, date of birth.

Loan officer: officerid,firstname,lastname,email,phone number, branch id

Bank: bank id,bank name,address,phonenummer

Loantype:id,name,description

Loan:loan id,loan amount,loan term,intrest rate,loan start rate,loan end date,loan status,customer id,loan officer id,loan type id

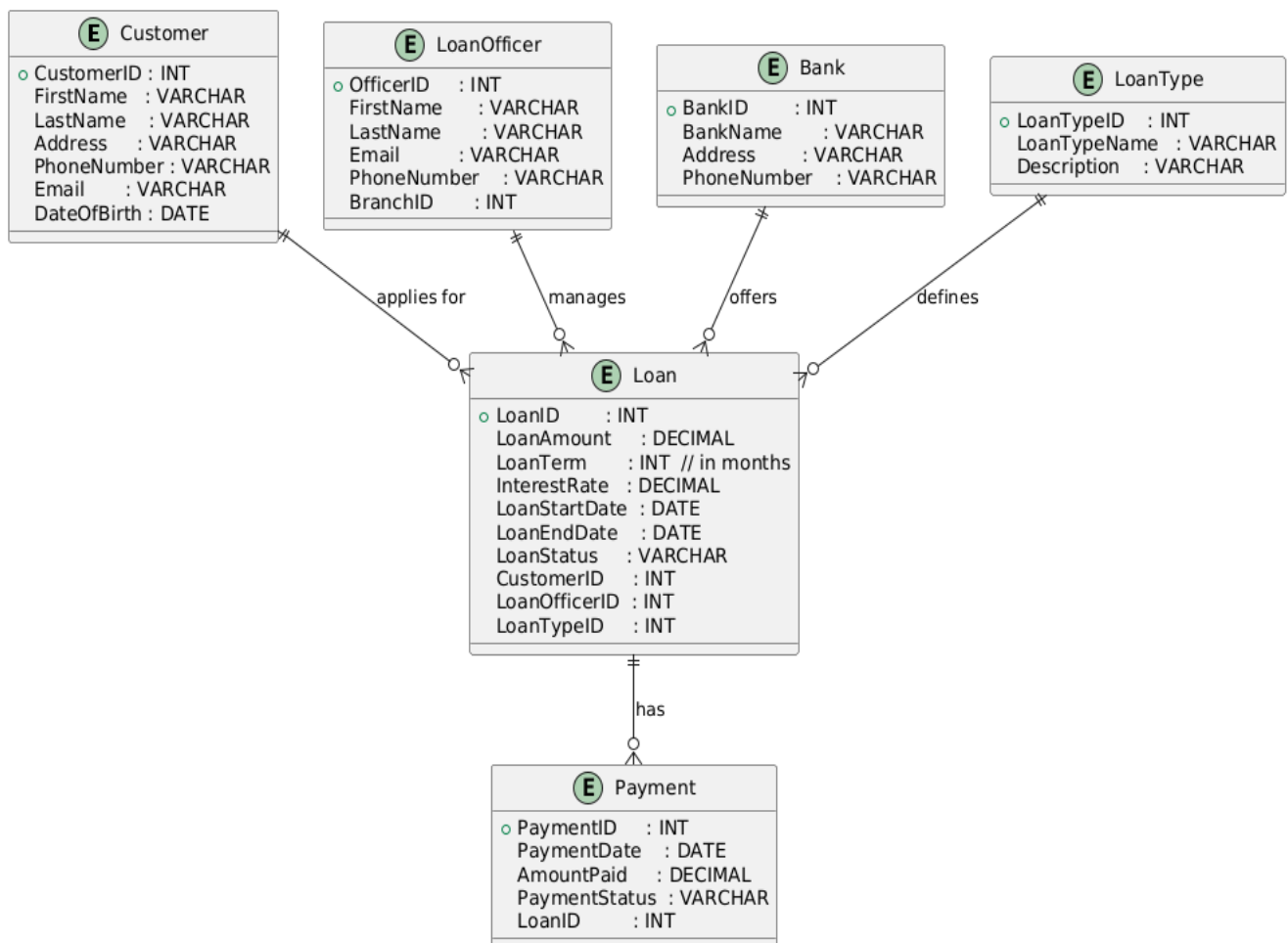
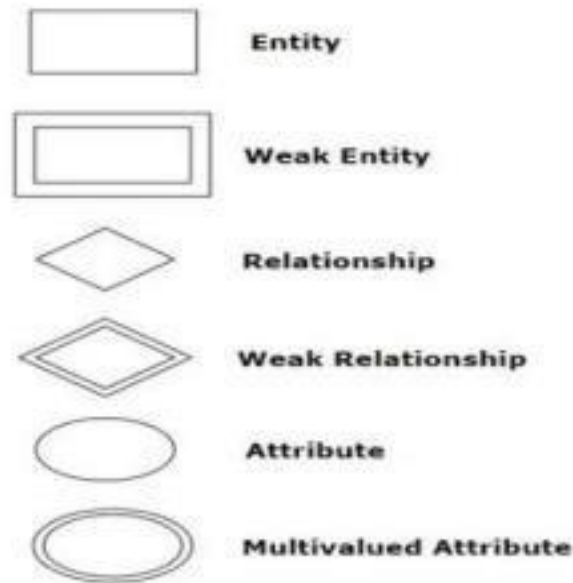
Payment:payment id,payment name,amount paid,payment status,loan id

### 5. Mapping of Attributes with Entities

Entity	Relationship
customer	customer id,firstname,lastname,address,phone number, email, date of birth
Loan officer	officerid,firstname,lastname,email,phone number, branch id
Loan	loan id,loan amount,loan term,intrest rate,loan start rate,loan end date,loan status,customer id,loan officer id,loan type id
Loan type	id,name,description
Bank	bank id,bank name,address,phonenummer
Payment	payment id,payment name,amount paid,payment status,loan id

### ER DIAGRAM:

## SYMBOLS:



**Result:** The entity relationship diagram was made successfully by following the steps described above.






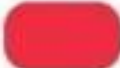










---

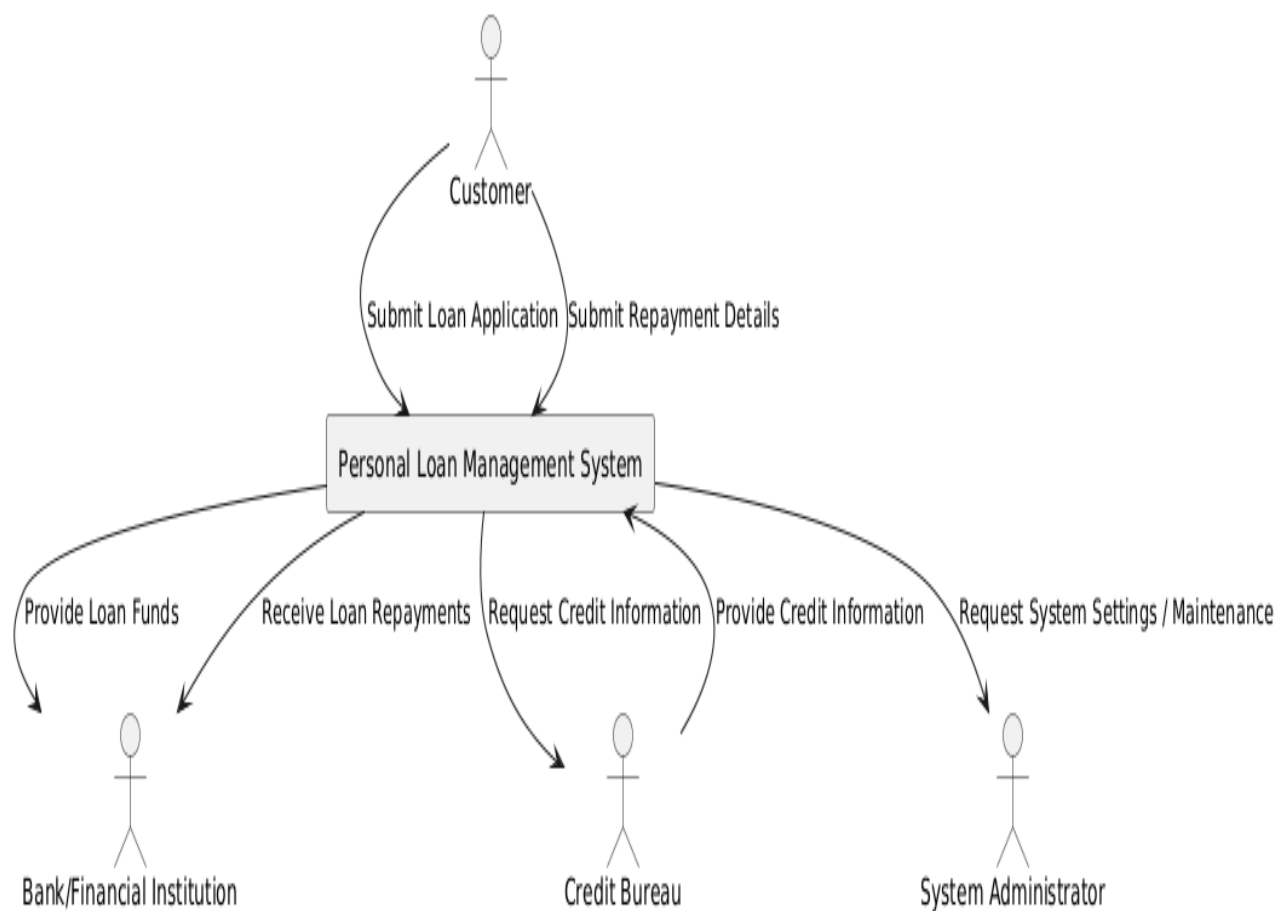
#### **4. DATA FLOW DIAGRAM**

**Aim** : To Draw the Data Flow Diagram for personal loan management system.

**Algorithm** :

1. **Customer → Loan Application Process**
  - Customer submits loan application.
2. **Loan Application Process → Credit Check Process**
  - Loan application data (customer's info) is sent for credit score evaluation.
3. **Credit Check Process → Credit Bureau**
  - Credit Check Process queries the **Credit Bureau** for customer's credit score and history.
4. **Credit Check Process → Loan Application Process**
  - Credit score and credit history returned for loan evaluation.
5. **Loan Application Process → Loan Disbursement Process**
  - If the loan is approved, details are sent to the **Loan Disbursement Process**.
6. **Loan Disbursement Process → Loan Data Store**
  - Loan data (loan amount, terms) is stored in the **Loan Data Store**.
7. **Customer → Repayment Process**
  - Customer submits repayment details.
8. **Repayment Process → Loan Data Store**
  - Repayment status is updated in the **Loan Data Store**.
9. **Repayment Process → Bank**
  - Repayment information is sent to the **Bank** for processing.
10. **Bank → Loan Data Store**
  - Bank sends confirmation of payment to update loan status in the **Loan Data Store**.
11. **Admin → Loan Data Store**
  - Admin queries and updates the **Loan Data Store** for monitoring and maintenance.

Notation	Yourdon & De Marco	Gene & Sarson	SSADM	Unified
External Entity				
Process				
Data Store				
Data Flow				



**Result:** The Data Flow diagram was made successfully by following the steps described above.





## 5. USE CASE DIAGRAM

**Aim :** To Draw the Use Case Diagram for personal loan management system.

### **Algorithm :**

- 1□ Apply for Loan: Customer applies for a loan.
2. Submit Documents: Customer submits required documents for loan approval.
3. Check Credit Score: The system checks the customer's credit score with the Credit Bureau.
4. Approve Loan: The Loan Officer approves or rejects the loan based on eligibility.
5. Disburse Loan: The Loan Officer disburses the loan to the customer.
6. Make Repayment: Customer makes a payment toward the loan balance.
7. View Loan Status: Customer can view the current status of their loan (e.g., outstanding balance, next due date).
8. Update Loan Information: Loan Officer updates loan records (e.g., balance, repayment schedules).
9. Generate Reports: System Administrator generates system reports (loan performance, user access, etc.).
10. Manage System Settings: System Administrator manages user access, system configurations, and updates.

### **Input :**

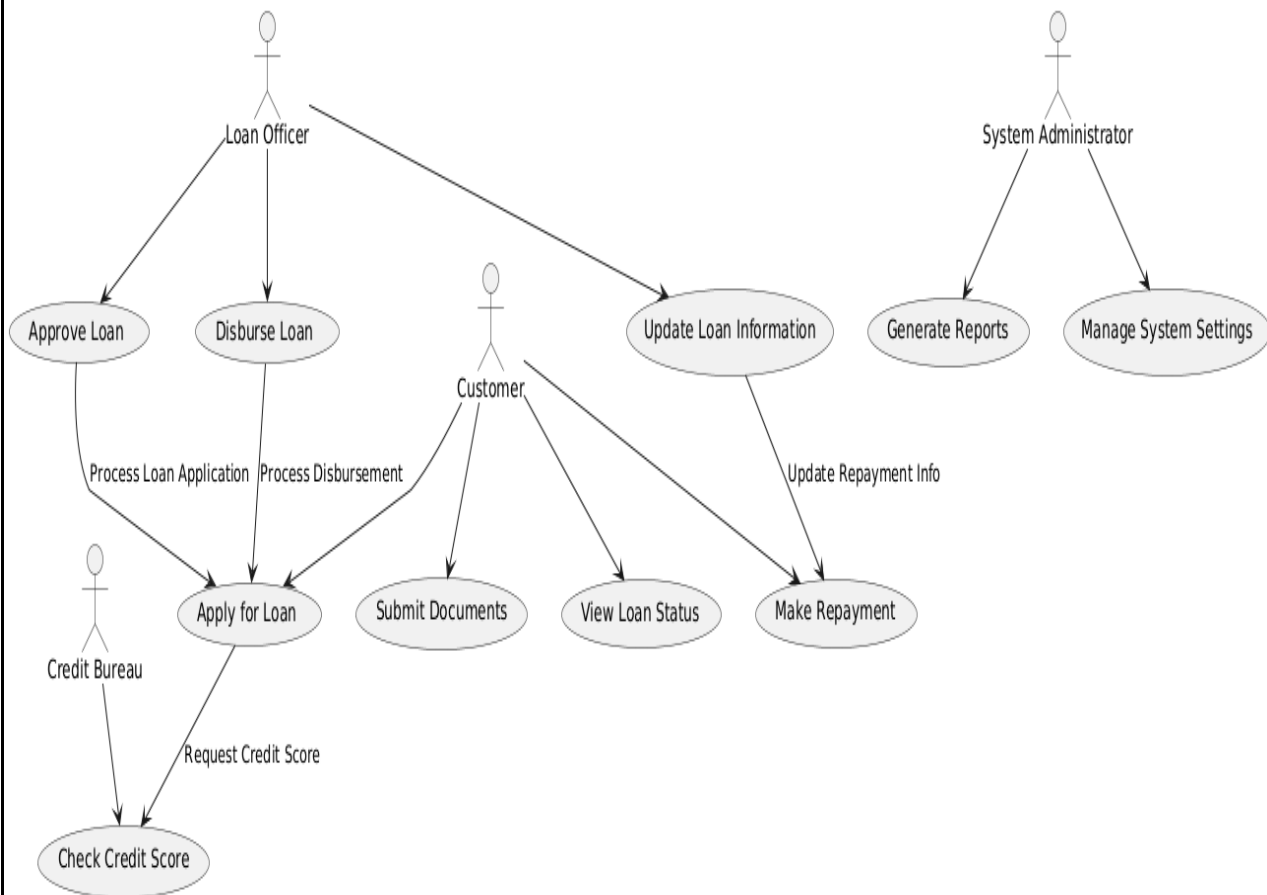
Actors

Use Cases

Relations



**Sample Output :**



**Result :** The use case diagram has been created successfully by following the steps given.

Department of CSBS/CB23332

## 6. ACTIVITY DIAGRAM

### AIM:

To Draw the activity Diagram for Personal loan management system

### ALGORITHM:

**Start** → Customer submits loan application.

2. **Validate Application** → If valid, move to **Credit Score Check**.

3. **Credit Score Check** → Query Credit Bureau.

4. **Decision Point**: If credit score is acceptable, move to **Loan Approval**. If not, **Reject Loan**.

5. **Loan Approval** → Loan Officer either approves or rejects the loan.

6. **Loan Disbursement** → If approved, loan is disbursed to customer and data is updated in the Loan Data Store.

7. **Repayment** → Customer makes a repayment.

8. **Repayment Validation** → If valid, update the Loan Data Store and notify the customer. If invalid, send an error.

9. **Monitor Loan Status** → Continuously track loan status and send reminders/alerts if necessary.

10. **End** → Loan process ends when paid off or closed.

### Inputs :

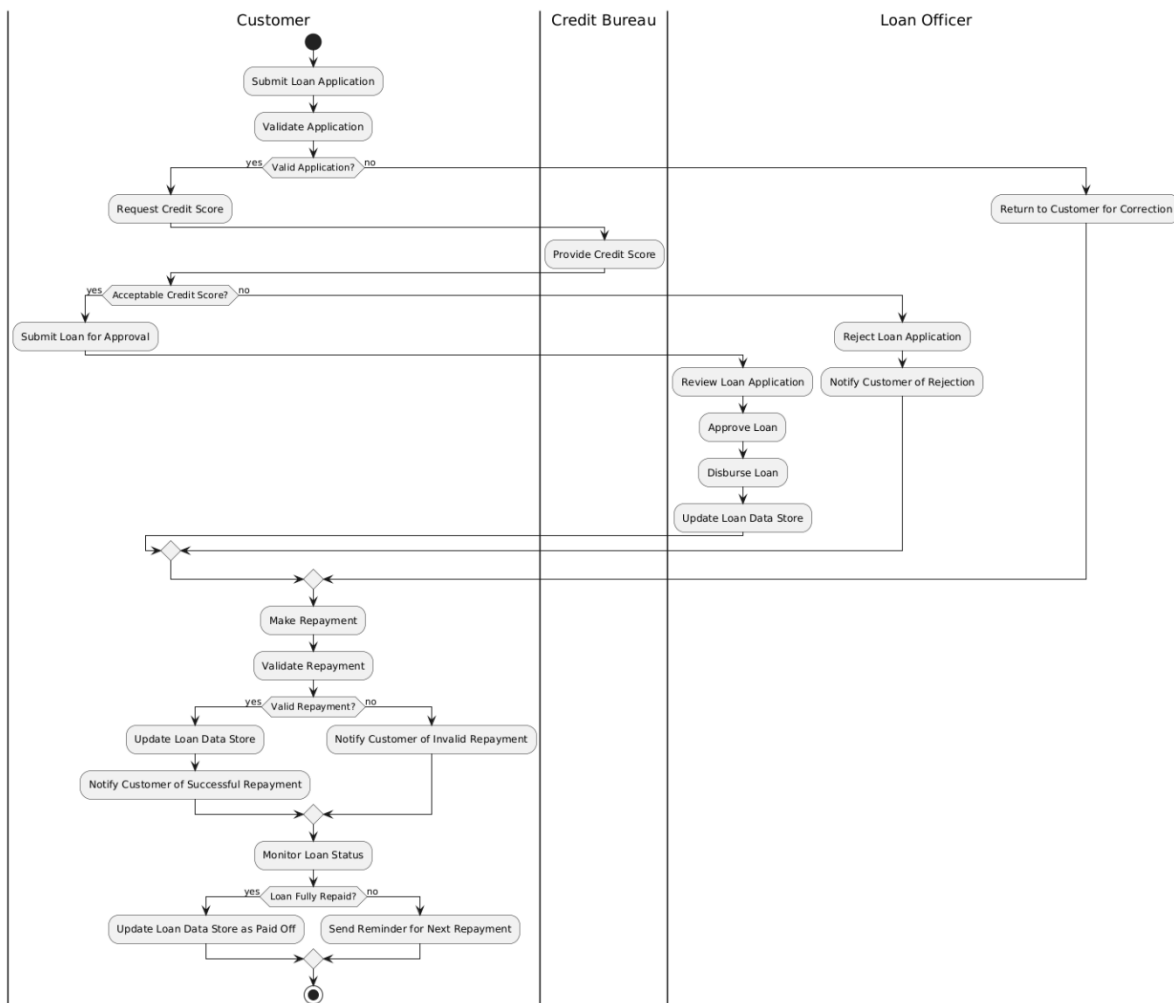
Activities

Decision Points

Guards

Parallel Activities Conditions

### Sample Output :



**Result :** The Activity diagram has been created successfully by following the steps given.

## 7. STATE CHART DIAGRAM

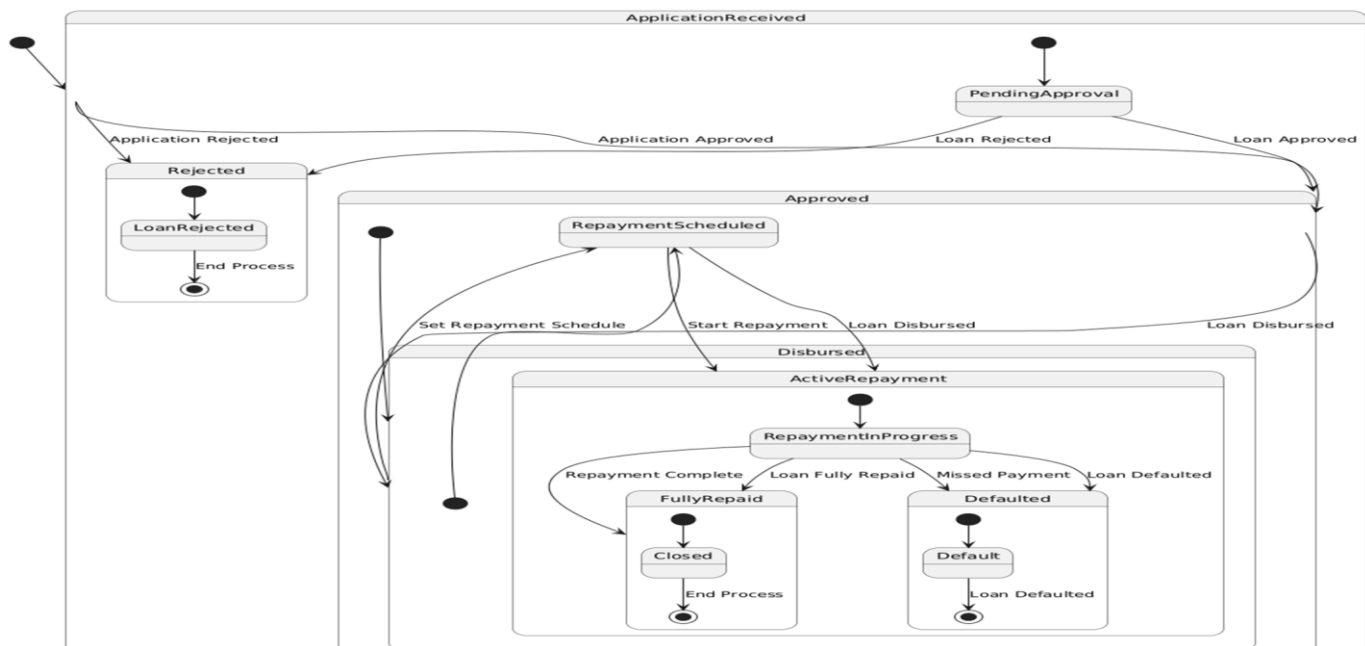
### Aim :

To Draw the State Chart Diagram for personal loan management system.

### Algorithm:

From **Application Received** → **Pending Approval**  
From **Pending Approval** → **Approved** (if the loan is approved)  
From **Pending Approval** → **Rejected** (if the loan is rejected)  
From **Approved** → **Disbursed**  
From **Disbursed** → **Repayment Scheduled**  
From **Repayment Scheduled** → **Active Repayment** (once the first payment is made)  
From **Active Repayment** → **Repayment In Progress**  
From **Repayment In Progress** → **Fully Repaid** (if the loan is paid in full)  
From **Repayment In Progress** → **Defaulted** (if the loan is defaulted)  
From **Defaulted** → **Closed** (loan closed as defaulted)  
From **Fully Repaid** → **Closed** (loan closed as fully repaid)

### Sample Output :



**Result :** The State Chart diagram has been created successfully by following the steps given.



## 8. SEQUENCE DIAGRAM

### Aim :

To Draw the Sequence Diagram for personal loan management system.

### Algorithm :

1. **Loan Application:** The user submits a loan application. The system checks whether the details are complete and valid.
2. **Loan Approval:** After validation, the loan is sent for approval. If approved, the loan moves to disbursement; if rejected, the process ends.
3. **Loan Disbursement:** Once approved, the loan amount is released, and repayment starts.
4. **Repayment:** The user starts repaying according to the scheduled plan. Payments are tracked by the system.
5. **Loan Closure:** The loan is closed either when it is fully repaid or if it defaults due to missed payments.
6. **End Process:** The loan lifecycle is complete, and the system is ready for further applications.

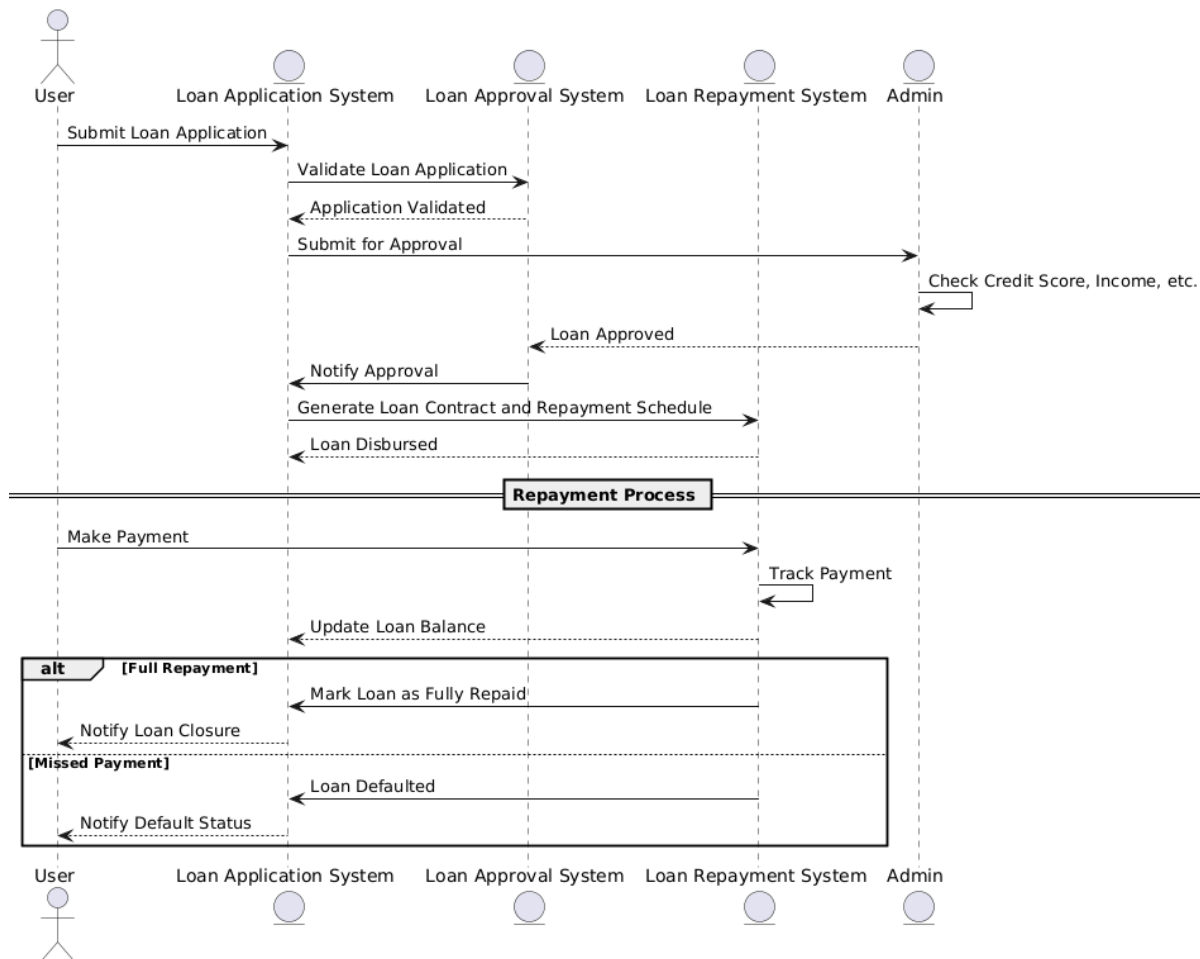
### Inputs :

Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing. Object organization.

### Sample Output :



## **Result :**

The Sequence diagram has been created successfully by following the steps given.



## 9. COLLABORATION DIAGRAM

### Aim :

To Draw the Collaboration Diagram personal loan management.

### Algorithm :

- **User:** The borrower interacting with the system.
- **Loan Application System:** Manages loan applications and validation.
- **Loan Approval System:** Responsible for loan approval and interaction with the **Admin**.
- **Admin:** Reviews the loan application for approval.
- **Loan Repayment System:** Handles loan disbursement and repayment tracking.

The **Collaboration Diagram** would show these entities and how messages are passed between them in the sequence above, focusing on the relationships and communication flow.

### Inputs :

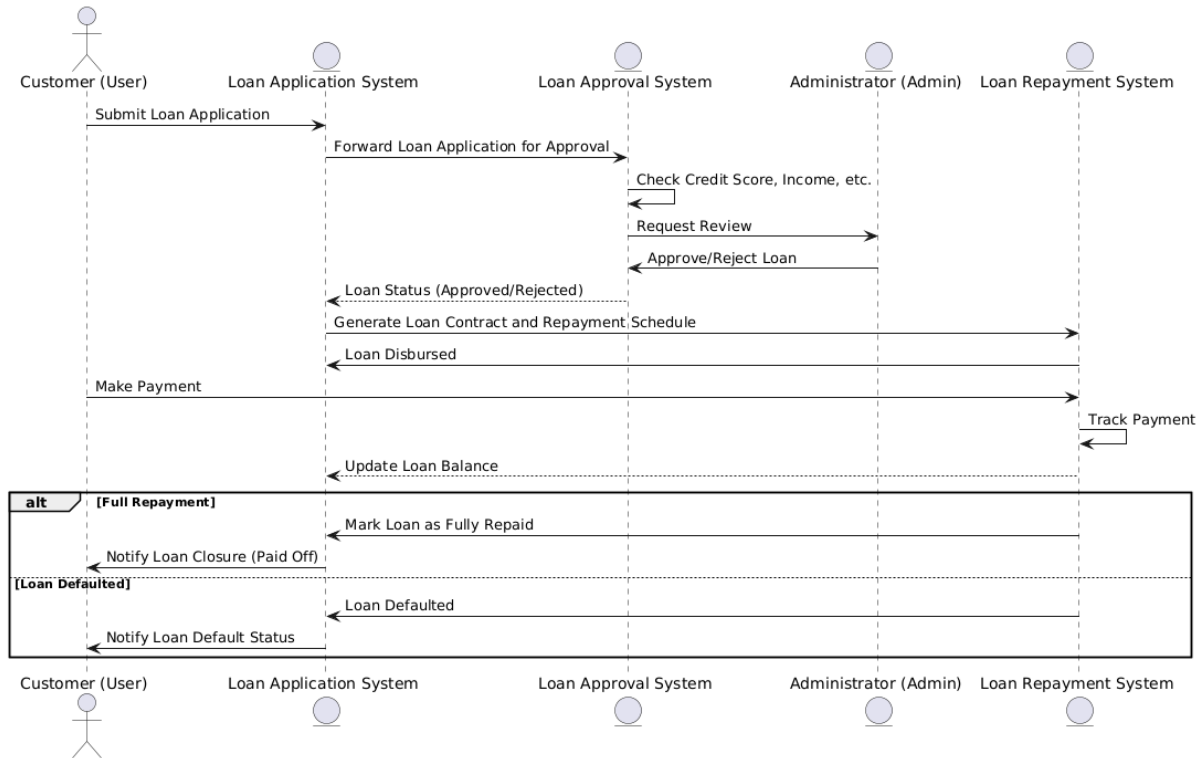
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

## Sample Output :



## RESULT:

The Collaboration diagram has been created successfully by following the steps given.



## 10. CLASS DIAGRAM

### **AIM:**

To Draw the Class Diagram for personal loan management system.

### **Algorithm :**

**Loan Validation:** Ensures that the application has all required information and follows the system's requirements.

**Credit Score Check:** Validates the user's creditworthiness before loan approval.

**Admin Approval:** Provides a manual check of the loan application.

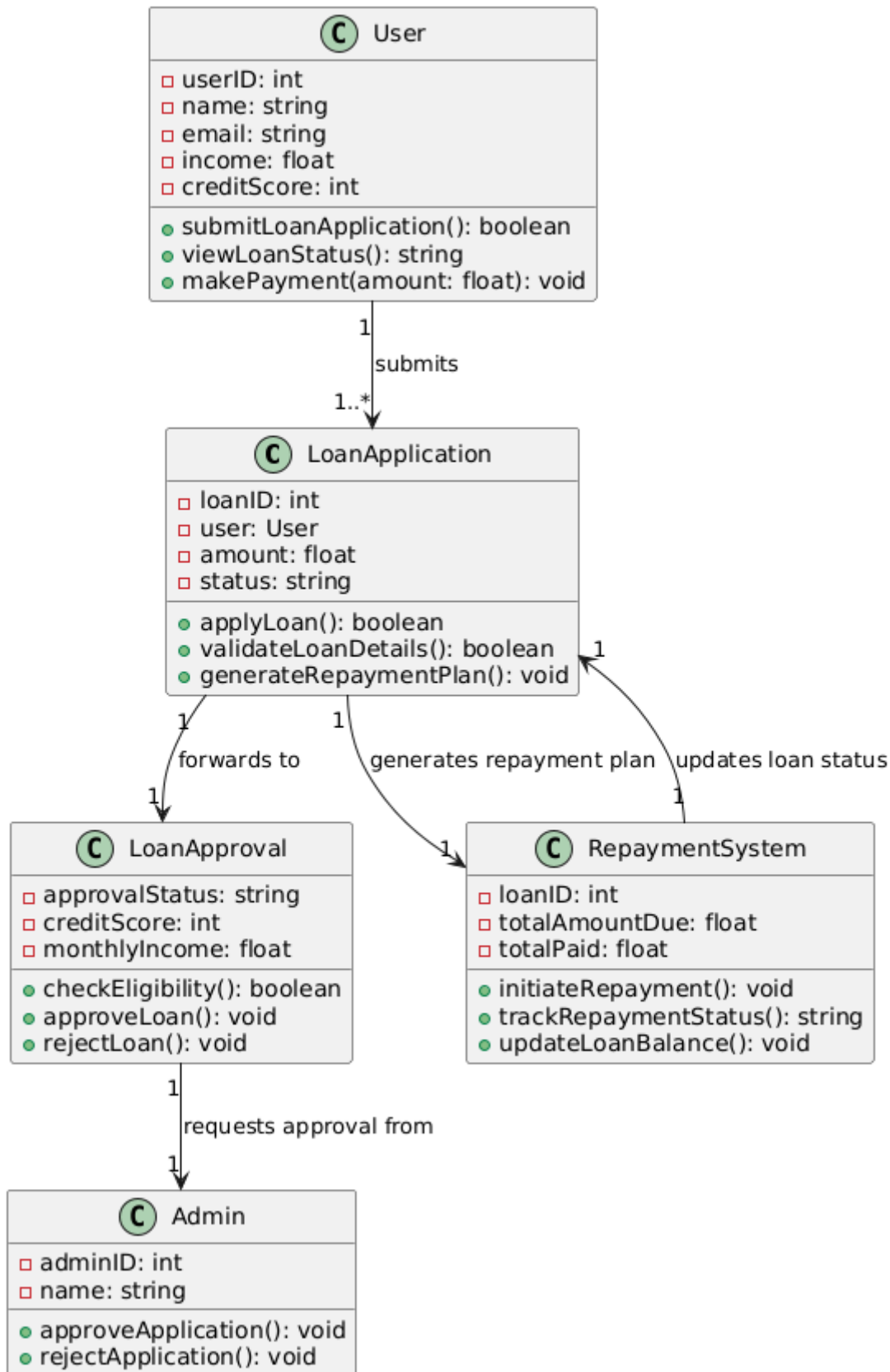
**Repayment Tracking:** Ensures the user adheres to the repayment schedule.

**Loan Closure or Default:** The system tracks whether the loan is fully repaid or defaulted.

### **Inputs :**

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

### Sample Output :



### **Result:**

The class diagram has been created successfully.

Department of CSBS/CB23332



---

<b><u>CODE FOR MINI PROJECT</u></b>
-------------------------------------

## IMPLEMENTATION CODE:

```
public class Loan {
    private double loanAmount;
    private double interestRate;
    private int loanTerm; // In months
    private double monthlyPayment;
    private double balance;

    // Constructor
    public Loan(double loanAmount, double interestRate, int loanTerm) {
        this.loanAmount = loanAmount;
        this.interestRate = interestRate;
        this.loanTerm = loanTerm;
        this.balance = loanAmount;
        this.monthlyPayment = calculateMonthlyPayment();
    }

    // Method to calculate the monthly payment based on loan details
    private double calculateMonthlyPayment() {
        double monthlyRate = interestRate / 100 / 12;
        return (loanAmount * monthlyRate) / (1 - Math.pow(1 + monthlyRate, -loanTerm));
    }

    // Method to make a payment
    public void makePayment(double paymentAmount) {
        if (paymentAmount > 0 && paymentAmount <= balance) {
            balance -= paymentAmount;
            System.out.println("Payment of " + paymentAmount + " made. Remaining balance: " +
balance);
        } else {
            System.out.println("Invalid payment amount.");
        }
    }

    // Getters
    public double getLoanAmount() {
        return loanAmount;
    }

    public double getBalance() {
        return balance;
    }
}
```



```

    public double getMonthlyPayment() {
        return monthlyPayment;
    }

    public int getLoanTerm() {
        return loanTerm;
    }

    public double getInterestRate() {
        return interestRate;
    }
}

public class LoanApplication {
    private Loan loan;

    public LoanApplication() {}

    // Method to apply for a loan
    public boolean applyForLoan(double amount, double interestRate, int loanTerm) {
        if (amount <= 0 || interestRate <= 0 || loanTerm <= 0) {
            System.out.println("Invalid loan application details.");
            return false;
        }

        loan = new Loan(amount, interestRate, loanTerm);
        System.out.println("Loan application successful! Loan details: ");
        System.out.println("Amount: " + amount);
        System.out.println("Interest Rate: " + interestRate + "%");
        System.out.println("Loan Term: " + loanTerm + " months");
        System.out.println("Monthly Payment: " + loan.getMonthlyPayment());
        return true;
    }

    // Method to get loan details
    public Loan getLoan() {
        return loan;
    }
}

import java.util.Scanner;

public class LoanManagementSystem {
    private static LoanApplication loanApplication;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        loanApplication = new LoanApplication();
    }
}

```

```

System.out.println("Welcome to the Personal Loan Management System");

while (true) {
    System.out.println("\n1. Apply for Loan");
    System.out.println("2. Make Loan Payment");
    System.out.println("3. View Loan Balance");
    System.out.println("4. Exit");
    System.out.print("Choose an option: ");

    int choice = scanner.nextInt();
    switch (choice) {
        case 1:
            applyForLoan(scanner);
            break;
        case 2:
            makeLoanPayment(scanner);
            break;
        case 3:
            viewLoanBalance();
            break;
        case 4:
            System.out.println("Exiting system. Goodbye!");
            return;
        default:
            System.out.println("Invalid option. Please try again.");
    }
}

// Apply for a new loan
private static void applyForLoan(Scanner scanner) {
    System.out.print("Enter loan amount: ");
    double amount = scanner.nextDouble();
    System.out.print("Enter interest rate (%): ");
    double interestRate = scanner.nextDouble();
    System.out.print("Enter loan term (in months): ");
    int loanTerm = scanner.nextInt();

    if (loanApplication.applyForLoan(amount, interestRate, loanTerm)) {
        System.out.println("Loan applied successfully.");
    } else {
        System.out.println("Loan application failed.");
    }
}

// Make a loan payment
private static void makeLoanPayment(Scanner scanner) {
    Loan loan = loanApplication.getLoan();

```

```

    if (loan == null) {
        System.out.println("No loan found. Please apply for a loan first.");
        return;
    }

    System.out.print("Enter payment amount: ");
    double paymentAmount = scanner.nextDouble();

    loan.makePayment(paymentAmount);
}

// View loan balance
private static void viewLoanBalance() {
    Loan loan = loanApplication.getLoan();
    if (loan == null) {
        System.out.println("No loan found. Please apply for a loan first.");
        return;
    }

    System.out.println("Remaining loan balance: " + loan.getBalance());
}
}

```

## SAMPLE OUTPUT:

Welcome to the Personal Loan Management System

1. Apply for Loan
2. Make Loan Payment
3. View Loan Balance
4. Exit

Choose an option: 1

Enter loan amount: 5000

Enter interest rate (%): 5

Enter loan term (in months): 12

Loan application successful! Loan details:

Amount: 5000.0

Interest Rate: 5.0%

Loan Term: 12 months

Monthly Payment: 428.72

1. Apply for Loan
2. Make Loan Payment
3. View Loan Balance
4. Exit

Choose an option: 2

Enter payment amount: 500

Payment of 500.0 made. Remaining balance: 4500.0

1. Apply for Loan
2. Make Loan Payment
3. View Loan Balance
4. Exit

Choose an option: 3

Remaining loan balance: 4500.0

1. Apply for Loan
2. Make Loan Payment
3. View Loan Balance
4. Exit

Choose an option: 4

Exiting system. Goodbye!