RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

HARDWORK DISCIPLINE

HEC

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332
SOFTWARE ENGINEERING LAB

Laboratory Record Note Book

Name : . . . .

Year / Branch / Section : . . . .

Register No. : . . . .

Semester : . . . .

Academic Year : . . . .

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   External Examiner

INDEX

EX NO:1	**WRITE THE COMPLETE PROBLEM STATEMENT**
DATE:	

AIM:

To prepare PROBLEM STATEMENT for Loan Management System.

ALGORITHM:

1. Identify the Problem:

○ Analyze the current loan management system to pinpoint inefficiencies (e.g., slow processing, manual data handling, poor inter-department communication).

2. Requirements Gathering:

○ Conduct interviews and surveys with stakeholders (bank employees, customers) to gather their needs and expectations for the new system.

3. Define High-Level Goals:

○ Clearly outline what the new system needs to achieve, such as faster loan processing, automation, real-time customer updates, and scalability.

4. Document Objectives:

○ List specific objectives based on stakeholder input (e.g., reduce processing time by 50%, improve system integration, enable real-time loan status tracking).

5. Write the Problem Statement:

○ Create a non-technical problem statement that clearly explains the issues without suggesting solutions.

6. Finalize and Agree:

○ Review and get agreement from all stakeholders on the problem statement before moving to the requirements engineering phase.

7. Start Requirements Engineering:

○ Use the problem statement as input to define detailed system requirements, guiding the design and development of the new Loan Management System.

INPUT:

● Problem Statement:

The input is the problem statement prepared by the customer (XYZ Bank), detailing issues such as slow loan processing, lack of automation, and poor customer communication.

● Overview of Existing System:

The current system is manual, fragmented, and inefficient, with long processing times and lack of integration between departments.

● Customer Expectations:

The customer expects improvements in speed (reduce processing time by 50%), automation of workflows, real-time updates for customers, and better inter-departmental integration.

● Stakeholder Feedback:

Feedback from bank staff (slow approval, data entry errors) and customers (lack of loan status visibility) helps shape the problem statement.

● Requirements Elicitation:

Gather detailed requirements through interviews and surveys with stakeholders to understand pain points and desired features.

● System Analysis:

Review the existing loan processing workflow to identify bottlenecks and areas for improvement.

Problem:

The current loan management system relies heavily on manual processes, leading to delays and inefficiencies in loan processing. A significant portion of loan applications experience extended approval times due to slow data entry, lack of automation, and fragmented communication between departments. As a result, customers often face uncertainty regarding the status of their applications, contributing to dissatisfaction. The system also struggles to handle increasing volumes of applications, which can affect its performance and scalability. This inefficiency limits the ability to provide timely responses and hampers overall customer experience.

Background:

The existing loan management system is outdated and lacks integration between key departments such as loan origination, credit assessment, and approval. Data entry is largely manual, and many tasks are duplicated across various teams, leading to errors and delays. Additionally, the system is not scalable, making it difficult to handle the growing number of loan applications. As the volume of loans increases, the system struggles to process requests in a timely manner, which impacts both operational efficiency and customer satisfaction. Customers have limited visibility into the progress of their applications, which contributes to frustration and a poor user experience.

Relevance:

Efficient loan processing is critical for maintaining competitive advantage and customer satisfaction in the financial industry. As customer expectations for faster, more transparent services increase, the limitations of the current system become more apparent. By modernizing the loan management system, organizations can streamline operations, reduce approval times, and automate key processes. Improved inter-departmental communication and real-time updates for customers will enhance transparency, reduce errors, and improve overall satisfaction.

Additionally, the ability to scale with increasing loan volumes will ensure that the system remains effective and efficient as the business grows. Addressing these issues will not only improve operational performance but also position the organization to better meet future customer demands and stay competitive in the market.

Objectives for the Loan Management System:

1. **Automation of Loan Processing**:

○ Automate key steps in the loan application process, such as data entry, credit risk assessments, and loan approval decisions, to reduce manual effort and increase processing speed.

2. **Reduce Loan Processing Time**:

○ Decrease the overall time required to process loan applications by at least 50%, improving efficiency and meeting customer expectations for faster service.

3. **Improve Inter-Departmental Integration**:

○ Create a centralized system that enables seamless communication and data sharing between departments involved in loan processing (e.g., credit risk, loan origination, customer service) to reduce delays and errors.

4. **Enhance Customer Communication and Transparency**: ○ Implement

a customer-facing portal or tracking system that allows applicants to view the status of their loan applications in real-time, receive notifications on updates, and communicate with relevant

departments.

5. **Increase Scalability**:

○ Design the system to handle a growing number of loan applications, ensuring it can accommodate increased demand without

compromising performance or user experience.

RESULT:

EX NO:2	**WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT**
DATE:	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet (SRS) for Loan Management System.

ALGORITHM:

Functionality

● Define key features: Loan application submission, risk assessment, approval workflow, status tracking, customer communication, reporting. ● Roles: Outline tasks for customers, loan officers, and admins.

External Interfaces

● User Interface: Customer portal, loan officer dashboard. ● System Integration: Connect to external services (e.g., credit scoring APIs, notification services).

● Hardware: Web-based, cloud-hosted infrastructure.

Performance

● Response Times: Loan application submission ≤ 5 sec, approval notifications ≤ 10 sec.

● Availability: 99.9% uptime, recovery within 30 minutes. ● Scalability: Handle increased application volume without performance loss.

Attributes

● Portability: Web and mobile access.

● Correctness: Accurate loan calculations and data handling. ● Security: Strong encryption, secure logins (multi-factor). ● Maintainability: Modular and easy to update.

● Scalability: Accommodate growing loan volumes.

Design Constraints

● Tech Stack: Java (backend), React (frontend), MySQL (database). ● Compliance: Adhere to PCI-DSS, GDPR.

● Hosting: Cloud-based infrastructure (AWS).

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to define the requirements for a Loan Management System (LMS) to streamline and automate the processing of loan applications. The system will simplify the loan approval

process, improve decision-making accuracy, enhance customer satisfaction, and increase operational efficiency.

1.2 Document Conventions

The following conventions are used throughout this document:

● DB: Database

● API: Application Programming Interface

● LMS: Loan Management System

● UI: User Interface

● CRUD: Create, Read, Update, Delete

● ERP: Enterprise Resource Planning

1.3 Intended Audience and Reading Suggestions

This document is intended for the development team, project managers, system administrators, and stakeholders involved in the loan management process. The primary audience includes:

● Bank Management: To understand the system’s capabilities and performance.

● Development Team: For building the system based on the specified requirements.

● End Users (Loan Officers & Customers): To guide their interaction with the system.

It is recommended that readers familiarize themselves with the basic concepts of loan management systems, automation, and financial regulations.

1.4 Project Scope

The Loan Management System aims to automate and improve the efficiency of processing loan applications. It will handle tasks such as data entry, loan approval, credit risk assessment, customer notifications, and reporting. The system will integrate with existing banking infrastructure and be scalable to handle increased loan volumes. The LMS will offer a user-friendly interface for

both internal users (loan officers, managers) and external users (customers) for real-time updates on loan statuses.

The key features of the LMS include:

● Loan Application Management: Automates the process of loan application submission, validation, and approval.

● Credit Risk Assessment: Integrates with automated credit scoring tools to evaluate applicants' risk.

● Customer Interaction: Allows customers to track loan status, communicate with loan officers, and receive notifications.

● Reports & Analytics: Provides real-time reporting on loan processing metrics, application statuses, and financial health.

● Compliance & Security: Ensures the system meets financial industry regulations and secures sensitive data.

1.5 References

● "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe

● Loan Management: A Comprehensive Guide by BankingTech ● https://www.techradar.com/banking-automation

2. OVERALL DESCRIPTION

2.1 Product Perspective

The Loan Management System (LMS) is an integrated application for managing the entire loan lifecycle, from loan application submission to approval, processing, and disbursement. The system will store and manage the following information:

● Loan Details: This includes the type of loan, amount, interest rate, tenure, loan status (approved, rejected, pending), and disbursement details. ● Customer Information: This includes the customer's personal details (name, address, contact number), financial details (income, credit score), and loan history.

● Loan Officer/Employee Information: This includes details about the loan officers managing the applications, including their employee ID, department, and responsibilities.

2.2 Product Features

The Loan Management System will have the following major features:

● Loan Application Management: Customers can submit loan applications and provide supporting documents for evaluation.

● Credit Risk Assessment: The system will integrate with third-party credit scoring APIs to assess the creditworthiness of applicants.

● Loan Approval/Disbursement Workflow: Loan officers can approve or reject applications, and the system will track approval, disbursement, and repayment stages.

● Customer Communication: Automatic notifications (email/SMS) will be sent to customers regarding loan status updates, approvals, and payment reminders.

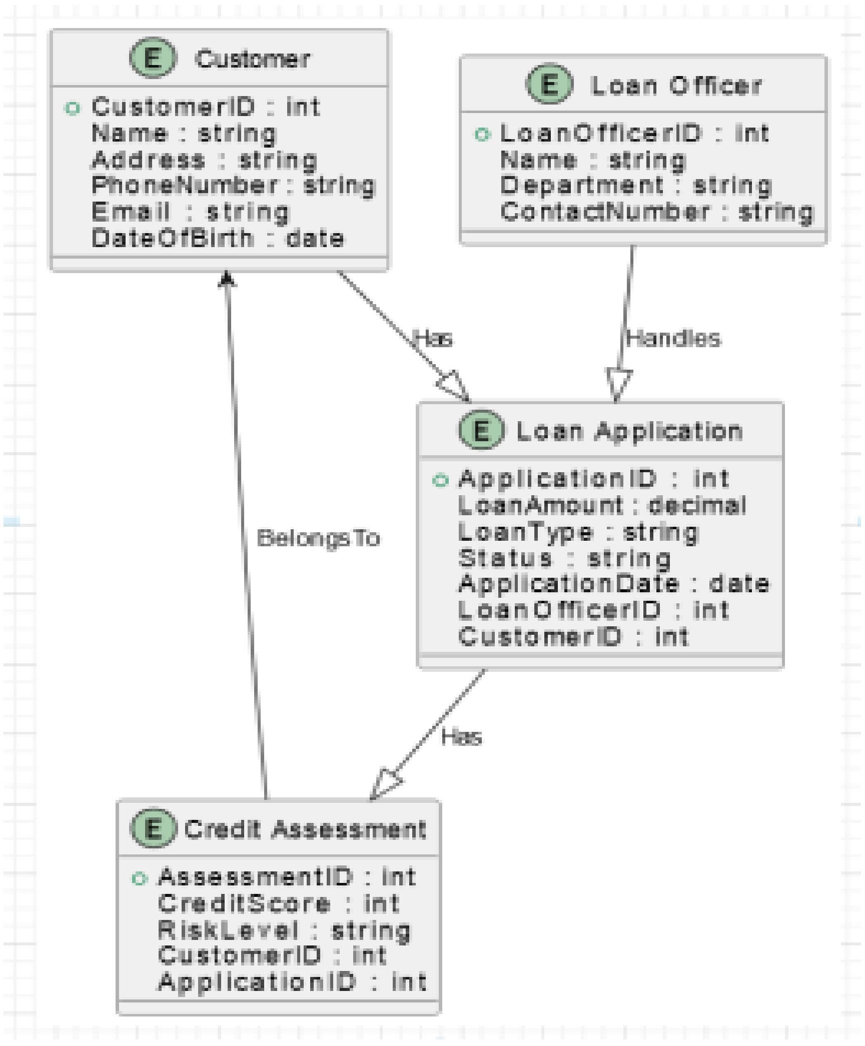
● Reports and Analytics: The system will provide analytical reports for loan officers and management on loan status, outstanding payments, and application volumes.

ER Model: The system will include the following key entities: ○ Loan Application

○ Customer

○ Loan Officer

○ Credit Risk Assessment


E Customer
o CustomerID : int
Name : string
Address : string
PhoneNumber : string
Email : string
DateOfBirth : date
E Loan Officer
o LoanOfficerID : int
Name : string
Department : string
ContactNumber : string
Has
Handles
E Loan Application
o ApplicationID : int
LoanAmount : decimal
LoanType : string
Status : string
ApplicationDate : date
LoanOfficerID : int
CustomerID : int
BelongsTo
Has
E Credit Assessment
o AssessmentID : int
CreditScore : int
RiskLevel : string
CustomerID : int
ApplicationID : int

● **Customers**: Can submit loan applications, track the status of their loan, and receive updates.

● **Loan Officers/Employees**: Can process and approve/reject loan applications, manage customer queries, and generate reports. The system will allow customers to view loan product details, apply for loans, and check the status of their application. Loan officers will be responsible for reviewing and processing loan applications, evaluating risk, and ensuring the correct documentation is attached.

2.4 Operating Environment

The LMS will operate in the following environment:

● **Distributed Database**: The system will use a centralized or distributed database to store customer and loan details securely.

● **Client/Server System**: The application will be built using a client-server architecture, where users interact with the system through a web interface, and the backend processes requests on the server.

● **Operating System**: The LMS will be designed to work on a cloud infrastructure, accessible via web browsers on both desktop and mobile devices.

● **Database**: The LMS will use a relational database management system (RDBMS) like MySQL or PostgreSQL to store loan and customer data. ● **Frontend Technology**: The system's user interface will be developed using HTML, CSS, JavaScript, and React.js to ensure responsiveness and easy access.

2.5 Design and Implementation Constraints

1. **Database Design**:

○ The system must support a distributed or centralized database, ensuring that customer and loan data is stored securely and is easily accessible to authorized users.

○ It must maintain ACID properties (Atomicity, Consistency, Isolation, Durability) for all transactions to ensure data integrity.

2. **Tech Stack**:

○ The system will be implemented using Java for the backend, React.js for the frontend, and MySQL/PostgreSQL for database management.

3. **Query Handling**:

○ SQL queries will be used to fetch loan data, customer details, and

transaction histories. The database should be optimized to handle complex queries efficiently.

4. **Security**:

○ All sensitive customer data, including financial and personal information, must be encrypted using SSL/TLS for secure

transmission.

○ Multi-factor authentication (MFA) will be required for both customers and employees to access the system.

2.6 Assumptions and Dependencies

● The system assumes that applicants have an internet connection and can access the LMS online through a web browser.

● The system depends on an external credit scoring service (e.g., Experian, Equifax) to assess the creditworthiness of loan applicants.

● The LMS will support geolocation features to ensure customers are located within the country/region where the bank operates and complies with regional regulations.

Geographical Distribution: If the system operates in multiple regions or countries, the database will be geographically distributed to ensure data locality and compliance with regional data protection laws.

3. SYSTEM FEATURES

3.1 Functionality

The core functionalities of the **Loan Management System (LMS)** include:

1. **Loan Application**: Customers can apply for different types of loans (e.g., personal, home, vehicle) by providing necessary details like loan amount, term, and purpose.

2. **Loan Approval/Disapproval**: Loan officers can review applications, approve or reject loans based on eligibility criteria, and update loan status. 3. **Loan Repayment**: Customers can view their repayment schedules, make payments online, and track outstanding balances.

4. **Notifications**: Automatic notifications (via email/SMS) will be sent to users about loan approval status, payment reminders, and upcoming due dates. 5. **Admin Features**: Administrators can monitor loan applications, approve or reject loans, generate reports, and manage user accounts.

6. **Loan History**: Customers can view their past loan transactions, including disbursed amounts, repayments, and current balances.

3.2 External Interface Requirements

1. **User Interfaces**:

○ **Web Interface**: The primary interface of the system will be a user-friendly web application, allowing customers to apply for loans,

check their loan status, and make payments online. Loan officers and administrators will use this interface for managing loan applications and customer information.

○ **Mobile Interface**: A mobile-optimized version of the system will be available, allowing users to access their loan details and make repayments from smartphones and tablets.

2. **Hardware Interfaces**:

○ **Client Devices**: The system will be accessible from desktops, laptops, tablets, and smartphones. The application should be responsive and adapt to different screen sizes.

○ **Server**: The platform will require a robust server capable of handling concurrent user requests, loan applications, transactions, and generating reports. The server should also have sufficient storage for user data, loan history, and transaction records.

3. **Software Interfaces**:

○ **Operating Systems**: The system will support Windows, macOS, Linux for the web interface and iOS and Android for the mobile application.

○ **Database**: The system will utilize an SQL-based database (e.g., MySQL, PostgreSQL) to store user data, loan information,

repayment records, and application statuses.

○ **Web Technologies**: The front-end of the web application will be developed using HTML5, CSS3, and JavaScript, while the back-end will be built using technologies such as Python, PHP, or Node.js. 4. **Communication Interfaces**:

○ **Internet Protocols**: The system will use HTTPS (Hypertext Transfer Protocol Secure) for secure communication between users, the web server, and the database. This ensures encrypted data transmission to protect sensitive information such as personal details and

payment information.

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Performance Requirements

● **Response Time**: The system should respond to user queries (such as loan application submission, loan status check, etc.) within **2 seconds** to provide an efficient user experience.

● **Availability**: The system should maintain **99.9% uptime**, ensuring that users (both customers and administrators) can access the system at all times, even during peak loan processing periods.

4.2 Safety Requirements

● **Data Backup**: Regular backups of the loan data, user information, and transaction records should be performed automatically to prevent data loss in case of system failure. Backups should be stored securely and should be easily retrievable.

● **Error Handling**: The system must be designed to handle errors without affecting ongoing loan processing. In case of an error, the system should log the issue, provide error messages to users, and allow administrators to resolve the issue without interrupting service.

4.3 Security Requirements

● **User Authentication**: The system must implement secure login mechanisms to ensure only authorized users (customers, loan officers, administrators) can access the relevant features of the system. Multi-factor authentication (MFA) should be used for sensitive operations like loan approval or modification.

● **Data Encryption**: All sensitive data, such as personal customer information, loan details, and payment records, must be encrypted using **industry-standard encryption algorithms** (e.g., AES-256) both in transit (via HTTPS) and at rest in the database to protect the confidentiality and integrity of the data.

● **Role-based Access Control**: Access to different parts of the system

should be restricted based on the role of the user. For instance, loan officers and customers should have different levels of access to loan information, application processes, and transaction history.

4.4 Software Quality Attributes

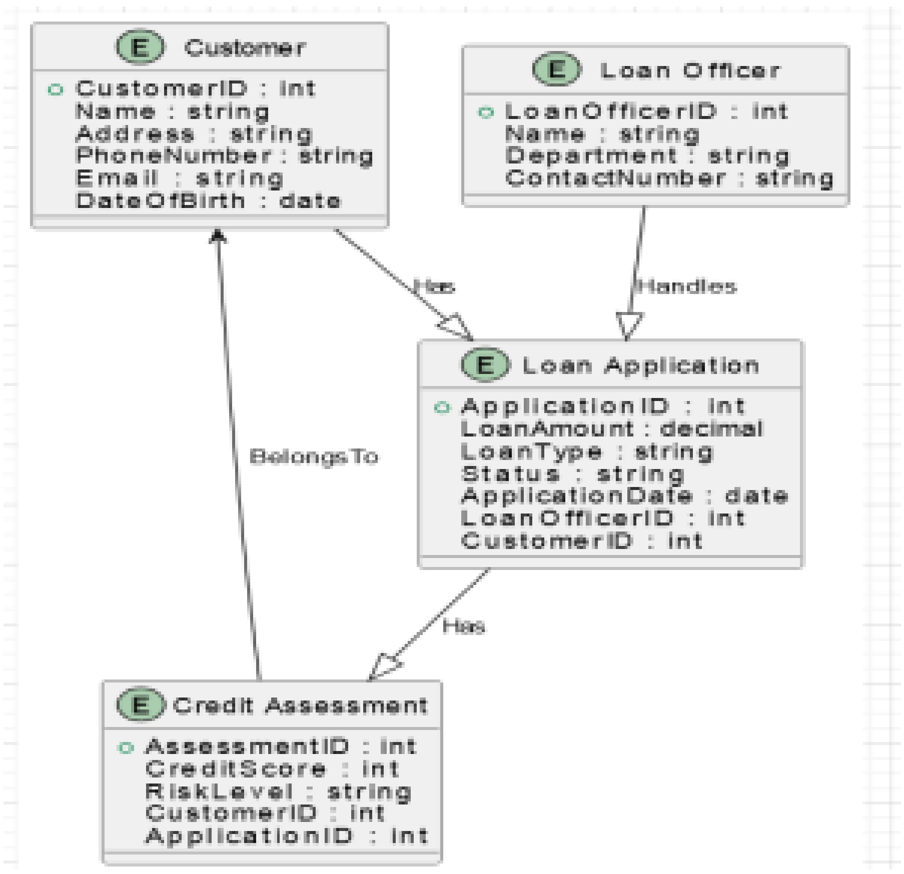
● **Usability**: The Loan Management System should be easy to navigate for all user types (customers, loan officers, administrators). It should have an intuitive interface, with clear instructions, and minimal training required for new users. Customers should be able to apply for loans, track application statuses, and make repayments with ease.

● **Maintainability**: The system should be designed to allow easy updates, bug fixes, and integration of new features. It should adhere to best practices in software engineering to ensure that it is maintainable in the long term and can be upgraded without major disruptions.

● **Scalability**: The system should be scalable to accommodate increasing numbers of loan applications, customers, and loan transactions. As the number of users grows or new features are added (e.g., new loan types, repayment methods), the system should handle the increased load without performance degradation.

RESULT:

OUTPUT:


E Customer
CustomerID : int
Name : string
Address : string
PhoneNumber : string
Email : string
DateOfBirth : date
E Loan Officer
LoanOfficerID : int
Name : string
Department : string
ContactNumber : string
Has
Handles
E Loan Application
ApplicationID : int
LoanAmount : decimal
LoanType : string
Status : string
ApplicationDate : date
LoanOfficerID : int
CustomerID : int
BelongsTo
Has
E Credit Assessment
AssessmentID : int
CreditScore : int
RiskLevel : string
CustomerID : int
ApplicationID : int

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE:	

AIM:

To Draw the Entity Relationship Diagram for Loan Management System.

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

Entity Relationship Matrix

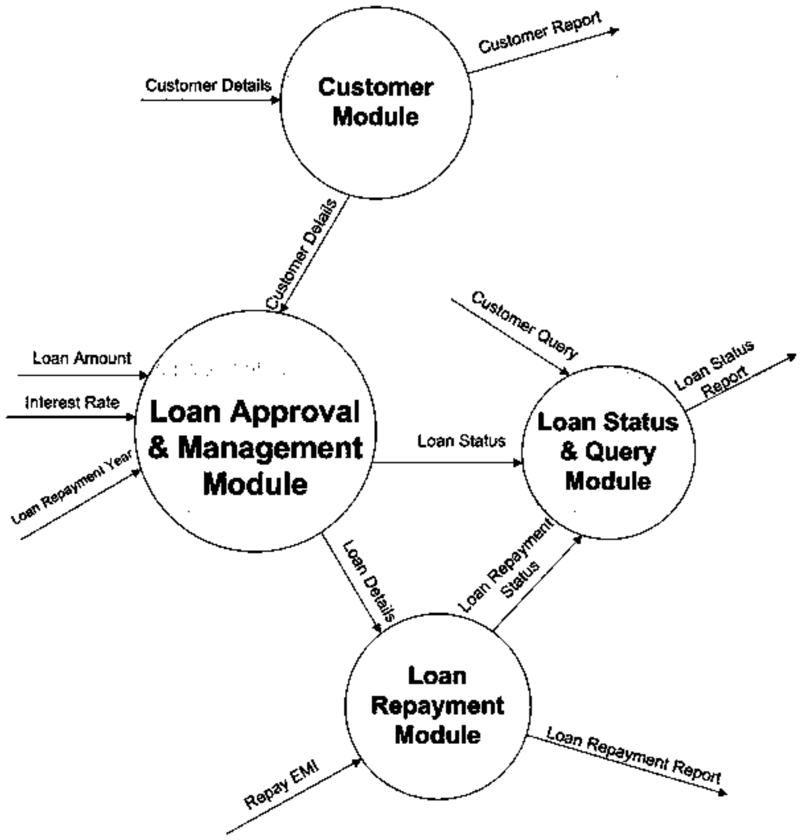
Primary Keys

Attributes

Mapping of Attributes with Entities

RESULT:

OUTPUT:



Level 2 DFD for Loan Management System

EX NO:4	**DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1**
DATE:	

AIM:

To Draw the Data Flow Diagram for Loan Management System and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

INPUT:

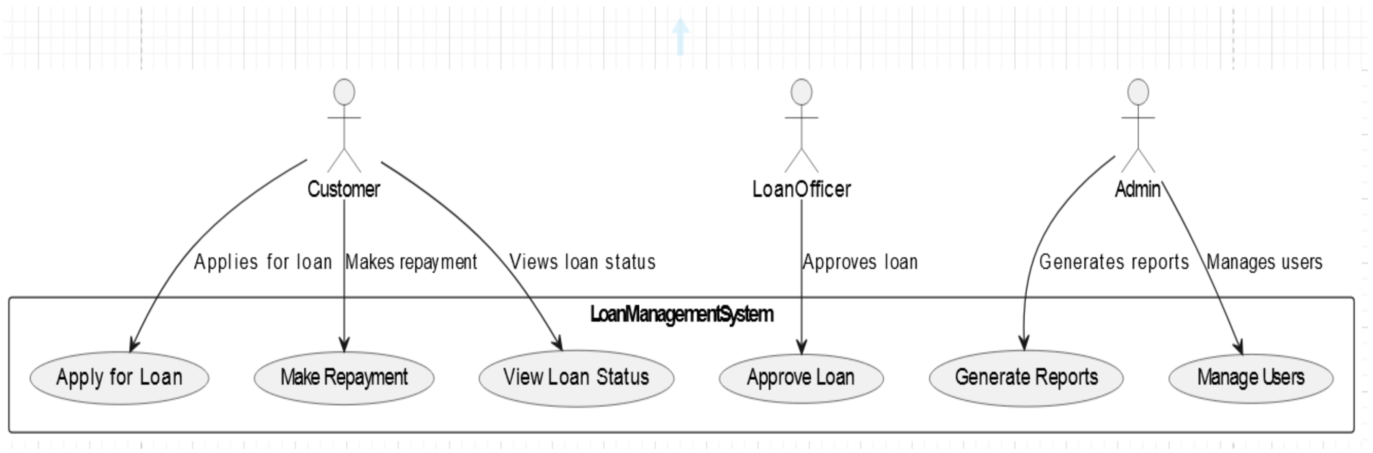
Processes

Datastores

External Entities

RESULT:

OUTPUT:


Customer
LoanOfficer
Admin
Applies for loan
Makes repayment
Views loan status
Approves loan
Generates reports
Manages users
LoanManagementSystem
Apply for Loan
Make Repayment
View Loan Status
Approve Loan
Generate Reports
Manage Users

EX NO:5	**DRAW USE CASE DIAGRAM**
DATE:	

AIM:

To Draw the Use Case Diagram for Loan Management System.

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

INPUTS:

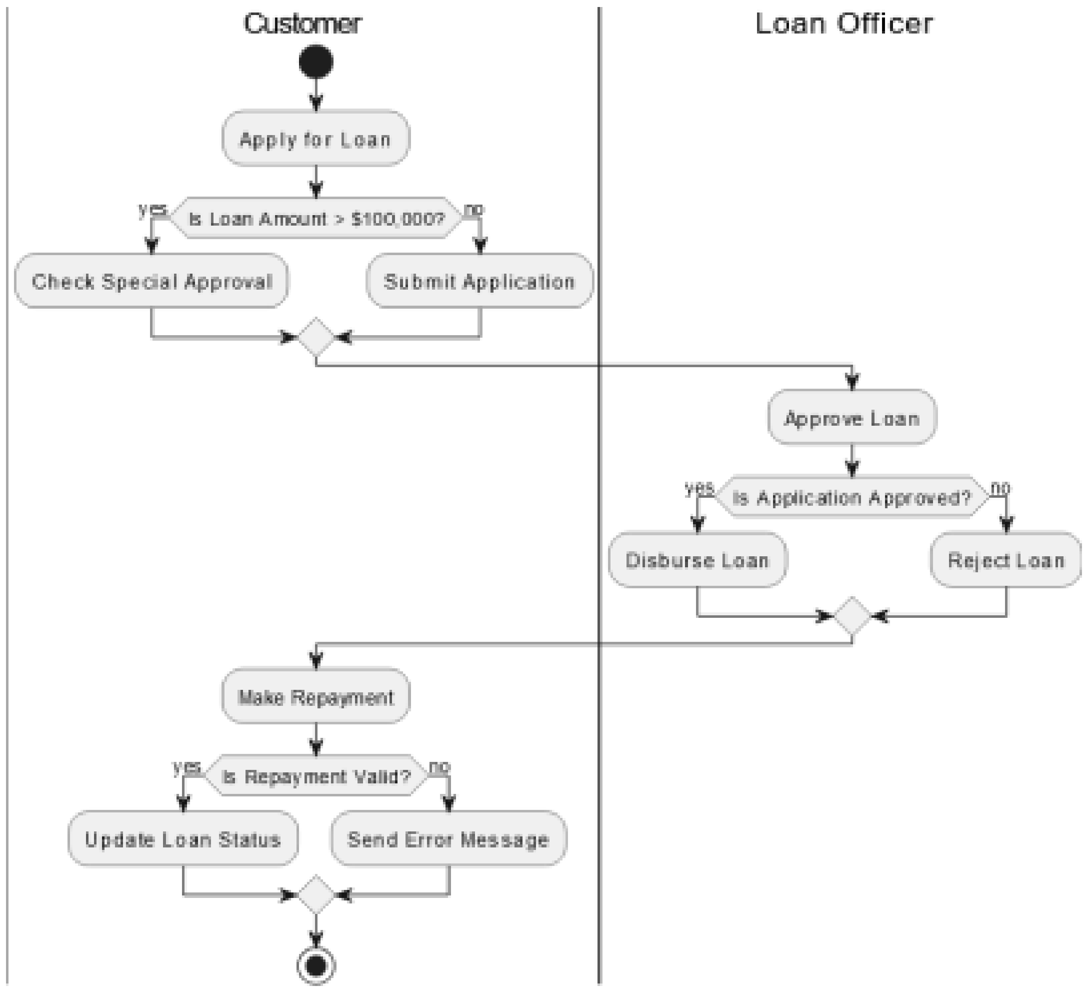
Actors

Use Cases

Relations

RESULT:

OUTPUT:


Customer
Loan Officer
Apply for Loan
Is Loan Amount > $100,000?
yes
no
Check Special Approval
Submit Application
Approve Loan
Is Application Approved?
yes
no
Disburse Loan
Reject Loan
Make Repayment
Is Repayment Valid?
yes
no
Update Loan Status
Send Error Message

EX NO:6	**DRAW ACTIVITY DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the activity Diagram for Loan Management System.

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

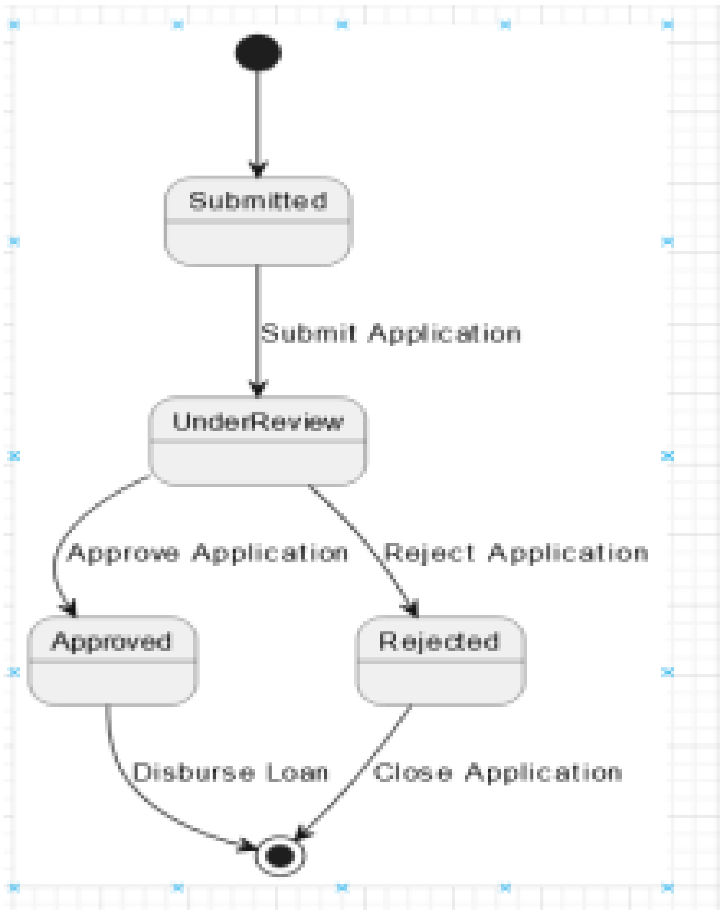
Guards

Parallel Activities

Conditions

RESULT:

OUTPUT:


Submitted
Submit Application
UnderReview
Approve Application
Reject Application
Approved
Rejected
Disburse Loan
Close Application

EX NO:7	**DRAW STATE CHART DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the State Chart Diagram for Loan Management System.

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

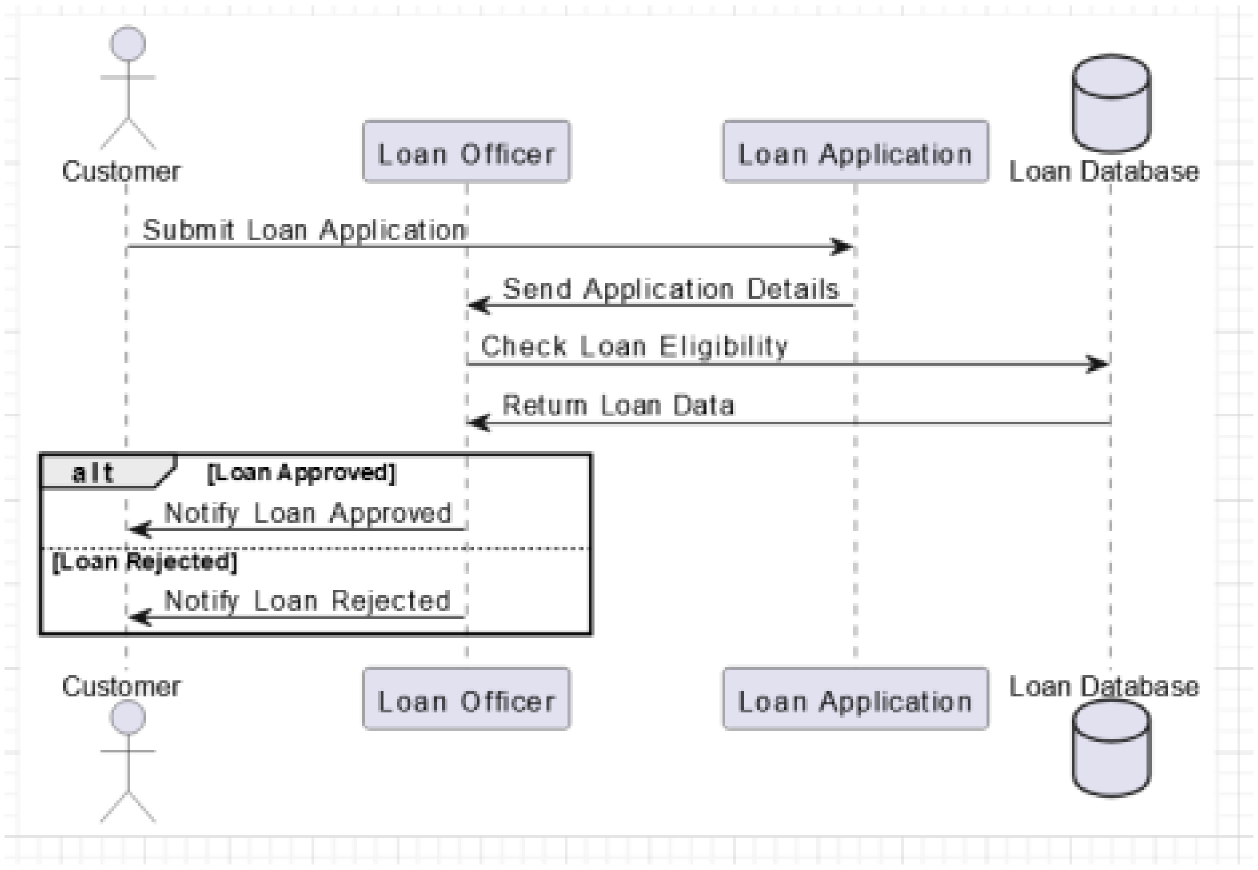
Objects

States

Events

RESULT:

OUTPUT:


Customer
Loan Officer
Loan Application
Loan Database
Submit Loan Application
Send Application Details
Check Loan Eligibility
Return Loan Data
alt
[Loan Approved]
Notify Loan Approved
[Loan Rejected]
Notify Loan Rejected
Customer
Loan Officer
Loan Application
Loan Database

EX NO:8	**DRAW SEQUENCE DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the Sequence Diagram for Loan Management System.

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

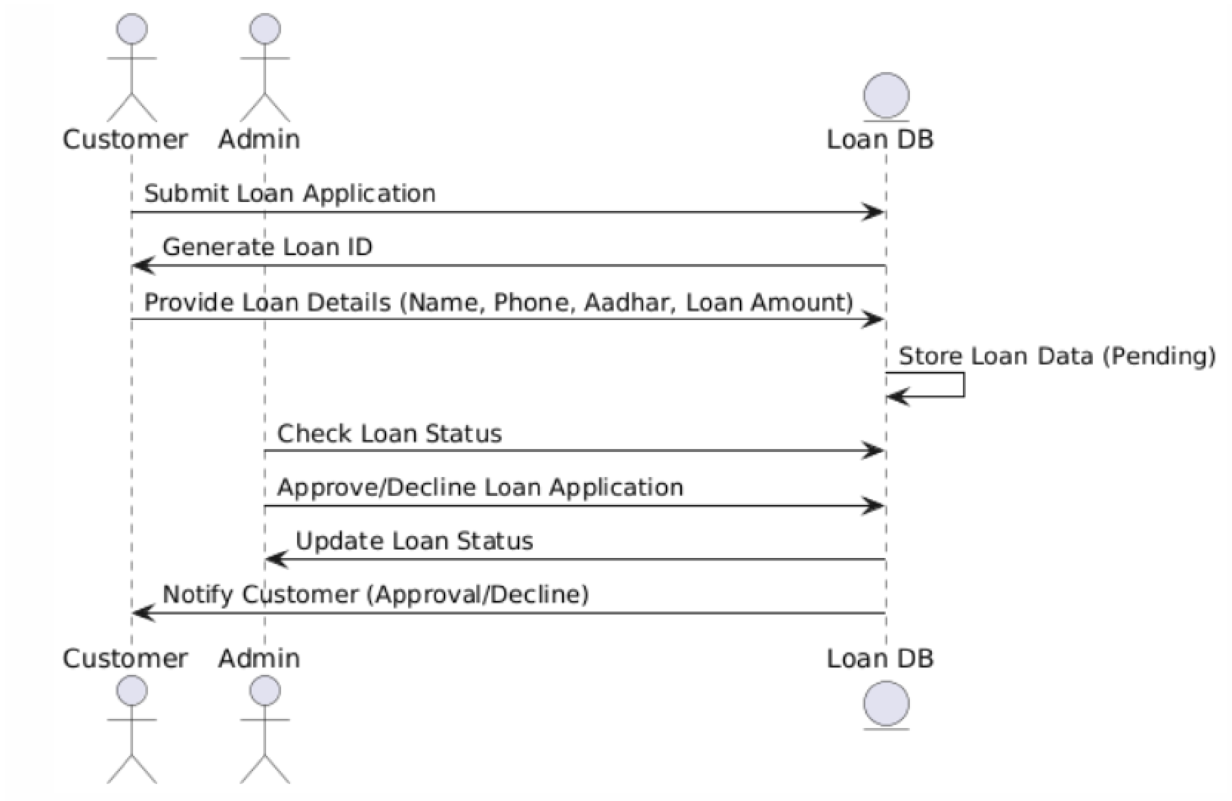
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

RESULT:

OUTPUT:


Customer
Admin
Loan DB
Submit Loan Application
Generate Loan ID
Provide Loan Details (Name, Phone, Aadhar, Loan Amount)
Store Loan Data (Pending)
Check Loan Status
Approve/Decline Loan Application
Update Loan Status
Notify Customer (Approval/Decline)
Customer
Admin
Loan DB

EX NO:9	**DRAW COLLABORATION DIAGRAM OF ALL USE CASES**
DATE:	

AIM:

To Draw the Collaboration Diagram for Loan Management System.

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

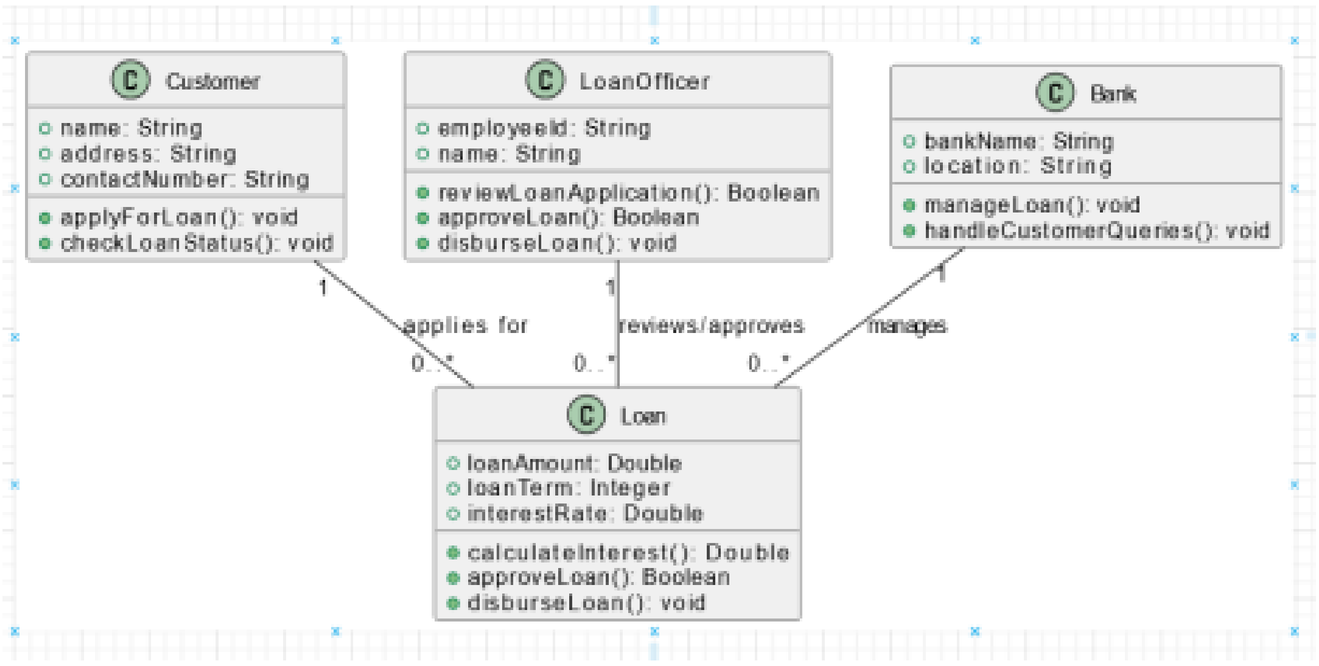
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

RESULT:

OUTPUT:


Customer
name: String
address: String
contactNumber: String
applyForLoan(): void
checkLoanStatus(): void
LoanOfficer
employeeId: String
name: String
reviewLoanApplication(): Boolean
approveLoan(): Boolean
disburseLoan(): void
Bank
bankName: String
location: String
manageLoan(): void
handleCustomerQueries(): void
1
applies for
0..*
1
reviews/approves
0..*
1
manages
0..*
Loan
loanAmount: Double
loanTerm: Integer
interestRate: Double
calculateInterest(): Double
approveLoan(): Boolean
disburseLoan(): void

EX NO:10	**ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.**
DATE:	

AIM:

To Draw the Class Diagram for Loan Management System.

ALGORITHM:

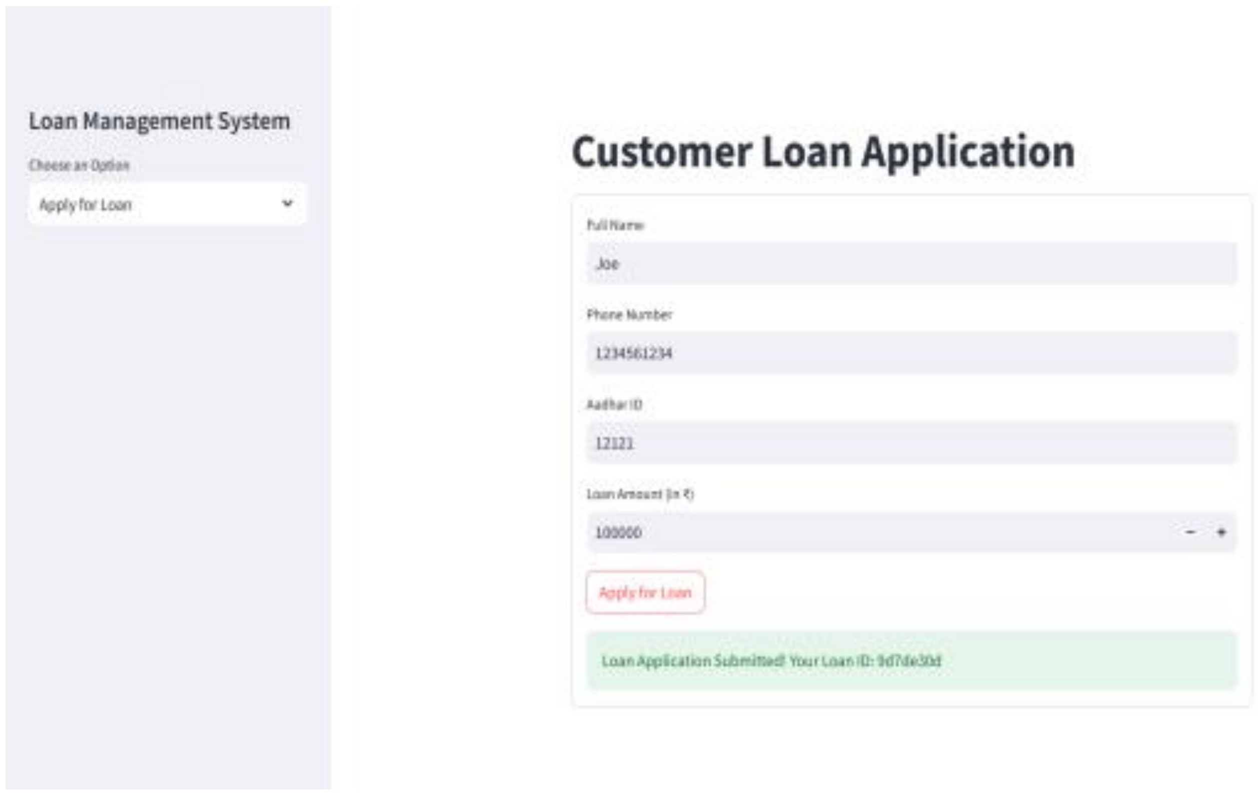
1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

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

RESULT:

OUTPUT:



Loan Management System

Choose an Option

Apply for Loan

Customer Loan Application

Full Name

Joe

Phone Number

1234561234

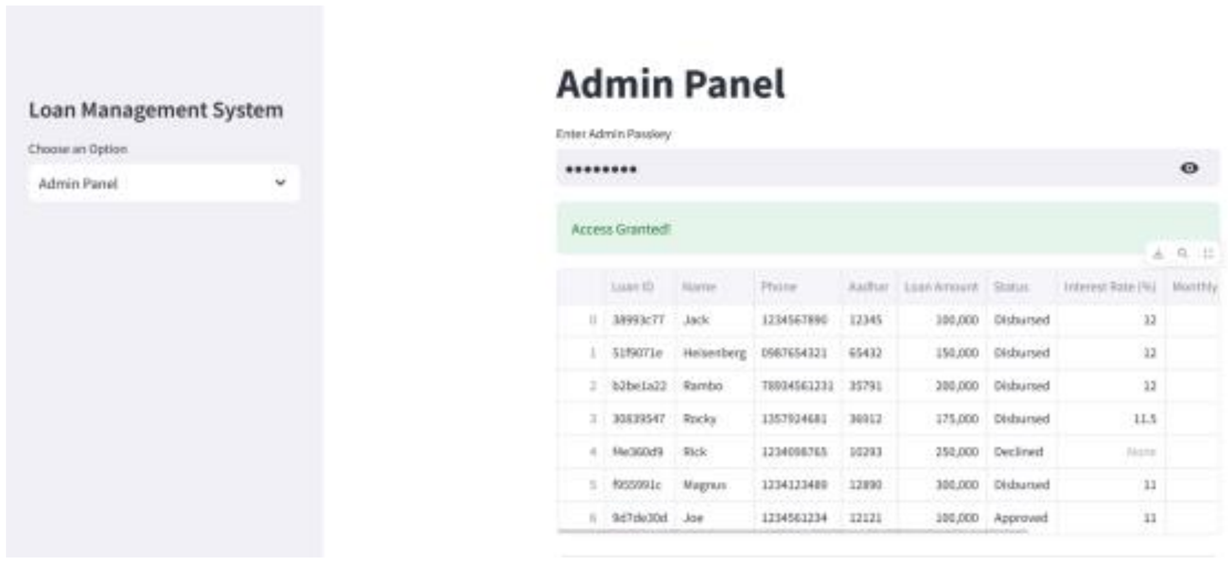
Aadhar ID

12121

Loan Amount (in ₹)

100000

Loan Application Submitted! Your Loan ID: 9d7de30d



Loan Management System

Choose an Option

Admin Panel

Admin Panel

Enter Admin Passkey

••••••••

Access Granted!

	Loan ID	Name	Phone	Aadhar	Loan Amount	Status	Interest Rate (%)	Monthly
0	38993c77	Jack	1234567890	12345	100,000	Disbursed	12	
1	51f9071e	Heisenberg	0987654321	65432	150,000	Disbursed	12	
2	b2be1a22	Rambo	78934561231	35791	200,000	Disbursed	12	
3	30839547	Rocky	1357924681	36912	175,000	Disbursed	11.5	
4	f4e360d9	Rick	1234098765	10293	250,000	Declined	None	
5	f955091c	Magnus	1234123489	12890	300,000	Disbursed	11	
6	9d7de30d	Joe	1234561234	12121	100,000	Approved	11	

EX NO:11	**MINI PROJECT- LOAN MANAGEMENT SYSTEM**
DATE:	

AIM:

The provided code aims to create a robust Loan Management System using Streamlit, incorporating data persistence through a CSV file. The system offers functionalities for both customers and administrators:

For Customers:

- **Apply for Loans:** Customers can submit loan applications by providing personal details and the desired loan amount.
- **Check Loan Status:** Customers can log in using their unique Loan ID to view the status of their application (pending, approved, declined, or disbursed). For approved or disbursed loans, they can view detailed information such as interest rate, monthly repayment, and pending balance.

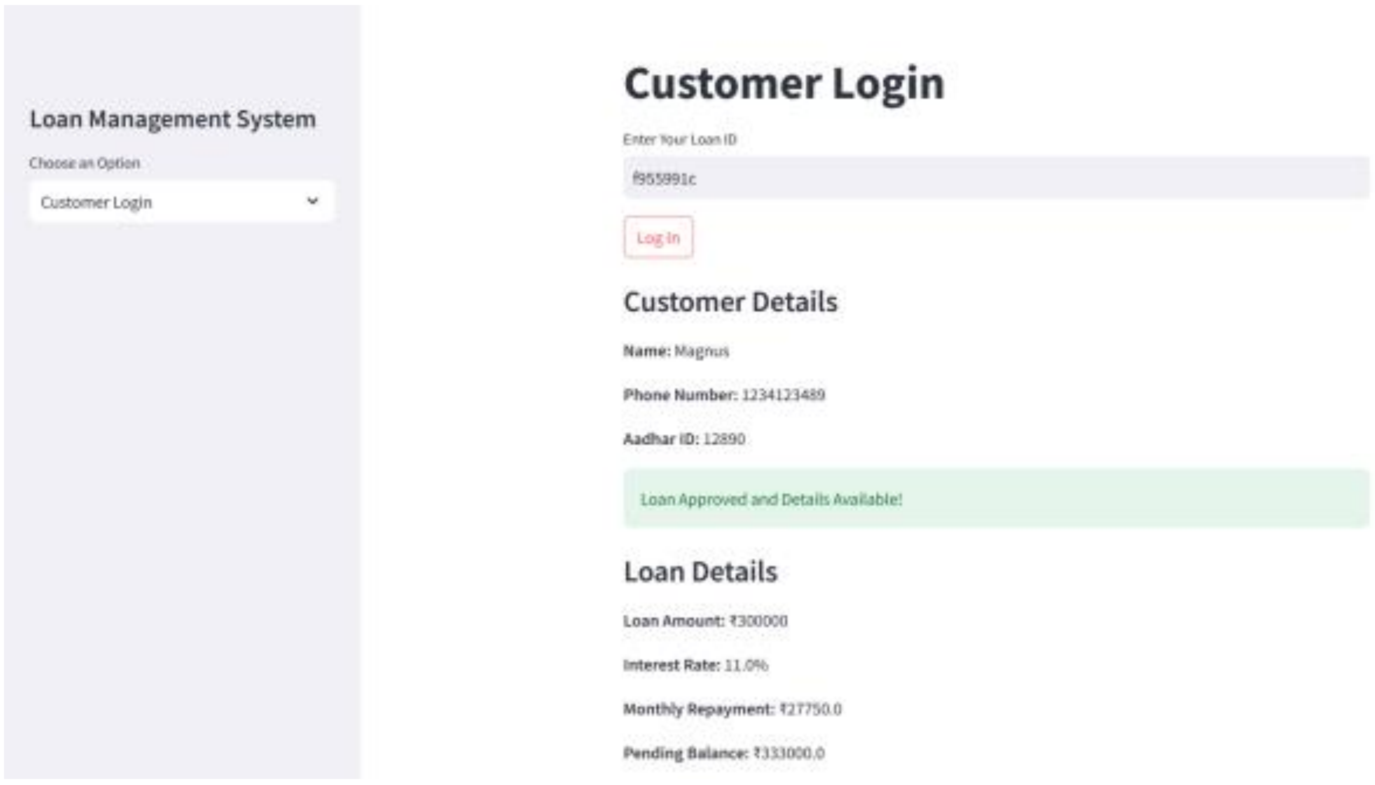
For Administrators:

- **Manage Loan Applications:** Administrators can review pending loan applications, approve or decline them, and set interest rates for approved loans.
- **Disburse Loans:** Once a loan is approved, administrators can disburse it, updating the loan status to "Disbursed."
- **View Loan Database:** Administrators can view the entire loan database, including details of all loans.

ALGORITHM:

1. **Data Persistence:**
   - The system utilizes a CSV file (loan_db.csv) to store loan information, ensuring data is preserved even after the Streamlit session ends.
   - The load_loan_db() function loads the data from the CSV file at the beginning of the application, while the save_loan_db() function saves any changes made to the database back to the CSV file.
2. **Loan Application:**
   - The apply_loan() function handles new loan applications.
   - It takes customer information (name, phone number, Aadhar ID, and loan amount) as input.
   - It generates a unique Loan ID and creates a new entry in the loan_db with the provided information and an initial "Pending" status.
   - The updated database is saved to the CSV file.
3. **Admin Panel:**
   - The admin_panel() function provides an interface for administrators.
   - It requires a passkey for authentication.
   - The admin can view the entire loan database, including pending, approved, and disbursed loans.
   - They can approve or decline pending loan applications, setting the interest rate for approved loans.
   - They can disburse approved loans by changing the loan status to "Disbursed."
   - Any changes made to the database are saved to the CSV file.
4. **Customer Login:**
   - The customer_login() function allows customers to log in using their Loan ID.
   - It retrieves the corresponding loan information from the database.
   - Based on the loan status, it displays appropriate information to the customer:

OUTPUT:



Customer Login

Enter Your Loan ID

f955991c

Customer Details

Name: Magnus

Phone Number: 1234123489

Aadhar ID: 12890

Loan Approved and Details Available!

Loan Details

Loan Amount: ₹300000

Interest Rate: 11.0%

Monthly Repayment: ₹27750.0

Pending Balance: ₹333000.0

        - **Pending:** Displays a message indicating the application is pending.
        - **Approved/Disbursed:** Displays detailed loan information, including interest rate, monthly repayment, and pending balance.
        - **Declined:** Displays a message indicating the application was declined.

5. **Main Application:**
    - The main Streamlit application provides a user interface with options for customers to apply for loans and for administrators to access the admin panel.
    - The appropriate functions are called based on the user's choice.

PROGRAM:

```
import streamlit as st

import pandas as pd

import uuid

import os

# Define the file path for storing the loan data

LOAN_DB_PATH = "loan_db.csv"

# Load the loan database from CSV (if it exists)

def load_loan_db():

    if os.path.exists(LOAN_DB_PATH):

        return pd.read_csv(LOAN_DB_PATH)

    else:

        # If no file exists, return an empty DataFrame with columns

        return pd.DataFrame(columns=[

            'Loan ID', 'Name', 'Phone', 'Aadhar', 'Loan Amount',

            'Status', 'Interest Rate (%)', 'Monthly Repayment (₹)',

            'Pending Balance (₹)'

        ])



# Save the loan database to a CSV file

def save_loan_db():
```

```
 st.session_state.loan_db.to_csv(LOAN_DB_PATH, index=False)
# Initialize the loan database
if 'loan_db' not in st.session_state:
    st.session_state.loan_db = load_loan_db()
# Helper function to calculate loan details
def calculate_loan_details(loan_amount, interest_rate, tenure_months=12):
    total_interest = loan_amount * (interest_rate / 100)
    total_amount = loan_amount + total_interest
    monthly_repayment = total_amount / tenure_months
    return round(total_interest, 2), round(monthly_repayment, 2), round(total_amount, 2)
# Function to handle new loan application
def apply_loan():
    st.title("Customer Loan Application")
    with st.form(key='loan_form'):
        name = st.text_input("Full Name")
        phone = st.text_input("Phone Number")
        aadhar = st.text_input("Aadhar ID")
        loan_amount = st.number_input("Loan Amount (in ₹)", min_value=1, step=1)
        submit = st.form_submit_button("Apply for Loan")
        if submit:
            if not name or not phone or not aadhar or loan_amount <= 0:
                st.error("Please fill out all fields correctly!")
            else:
                loan_id = str(uuid.uuid4())[:8]
                new_entry = {
                    "Loan ID": loan_id,
                    "Name": name,
                    "Phone": phone,
```

```
                "Aadhar": aadhar,

                "Loan Amount": loan_amount,

                "Status": "Pending",

                "Interest Rate (%)": None,

                "Monthly Repayment (₹)": None,

                "Pending Balance (₹)": None

            }

            # Append the new entry to the loan database

            st.session_state.loan_db = pd.concat([st.session_state.loan_db, pd.DataFrame([new_entry])],
ignore_index=True)

            save_loan_db()  # Save the updated database to CSV

            st.success(f"Loan Application Submitted! Your Loan ID: {loan_id}")

# Function for admin access

def admin_panel():

    st.title("Admin Panel")

    passkey = st.text_input("Enter Admin Passkey", type="password")

    if passkey == "admin123":  # Replace with your desired passkey

        st.success("Access Granted!")

        st.dataframe(st.session_state.loan_db)



        # Approve/Decline Loan

        with st.form(key='manage_loans'):

            loan_id = st.text_input("Enter Loan ID to Approve/Decline")

            action = st.radio("Action", ["Approve", "Decline"])

            interest_rate = st.number_input("Set Interest Rate (%)", min_value=0.0, max_value=100.0, step=0.1)
if action == "Approve" else None

            submit_action = st.form_submit_button("Submit")

            if submit_action:

                if loan_id in st.session_state.loan_db['Loan ID'].values:
```

```
            idx = st.session_state.loan_db.index[st.session_state.loan_db['Loan ID'] == loan_id].tolist()[0]

                if action == "Approve":

                    loan_amount = st.session_state.loan_db.at[idx, 'Loan Amount']

                    total_interest, monthly_repayment, total_amount = calculate_loan_details(loan_amount, 
interest_rate)

                    st.session_state.loan_db.at[idx, 'Status'] = "Approved"

                    st.session_state.loan_db.at[idx, 'Interest Rate (%)'] = interest_rate

                    st.session_state.loan_db.at[idx, 'Monthly Repayment (₹)'] = monthly_repayment

                    st.session_state.loan_db.at[idx, 'Pending Balance (₹)'] = total_amount

                    st.success(f"Loan ID {loan_id} has been Approved!")

                else:

                    st.session_state.loan_db.at[idx, 'Status'] = "Declined"

                    st.success(f"Loan ID {loan_id} has been Declined!")

                save_loan_db()  # Save the updated database to CSV

            else:

                st.error("Invalid Loan ID!")

        # Loan Disbursement

        with st.form(key='disburse_loan'):

            disburse_id = st.text_input("Enter Loan ID for Disbursement")

            disburse_action = st.form_submit_button("Disburse Loan")

            if disburse_action:

                if disburse_id in st.session_state.loan_db['Loan ID'].values:

                    idx = st.session_state.loan_db.index[st.session_state.loan_db['Loan ID'] == 
disburse_id].tolist()[0]

                    if st.session_state.loan_db.at[idx, 'Status'] == "Approved":

                        st.session_state.loan_db.at[idx, 'Status'] = "Disbursed"

                        st.success(f"Loan ID {disburse_id} has been Disbursed!")

                    else:

                        st.error("Loan must be Approved before Disbursement!")
```

```
                else:
                    st.error("Invalid Loan ID!")
                save_loan_db()  # Save the updated database to CSV
    else:
        st.error("Invalid Passkey!" if passkey else "Enter the passkey to access the admin panel.")
# Function for customer login
def customer_login():
    st.title("Customer Login")
    loan_id = st.text_input("Enter Your Loan ID")
    login_action = st.button("Log In")
    if login_action:
        if loan_id in st.session_state.loan_db['Loan ID'].values:
            idx = st.session_state.loan_db.index[st.session_state.loan_db['Loan ID'] == loan_id].tolist()[0]
            status = st.session_state.loan_db.at[idx, 'Status']

            # Display customer details
            st.write("### Customer Details")
            st.write(f"*Name:* {st.session_state.loan_db.at[idx, 'Name']}")
            st.write(f"*Phone Number:* {st.session_state.loan_db.at[idx, 'Phone']}")
            st.write(f"*Aadhar ID:* {st.session_state.loan_db.at[idx, 'Aadhar']}")

            # Display loan status and details
            if status in ["Approved", "Disbursed"]:  # Consider both "Approved" and "Disbursed"
                st.success("Loan Approved and Details Available!")
                st.write("### Loan Details")
                st.write(f"*Loan Amount:* ₹{st.session_state.loan_db.at[idx, 'Loan Amount']}")
                st.write(f"*Interest Rate:* {st.session_state.loan_db.at[idx, 'Interest Rate (%)']}%")
```

```
            st.write(f"*Monthly Repayment:* ₹{st.session_state.loan_db.at[idx, 'Monthly Repayment (₹)']}")

            st.write(f"*Pending Balance:* ₹{st.session_state.loan_db.at[idx, 'Pending Balance (₹)']}")

        elif status == "Declined":

            st.error("Your Loan Application was Declined.")

        else:

            st.info("Your Loan Application is still Pending.")

    else:

        st.error("Invalid Loan ID!")

# Main Streamlit Application

st.sidebar.title("Loan Management System")

user_choice = st.sidebar.selectbox("Choose an Option", ["Apply for Loan", "Customer Login", "Admin Panel"])



if user_choice == "Apply for Loan":

    apply_loan()

elif user_choice == "Customer Login":

    customer_login()

elif user_choice == "Admin Panel":

    admin_panel()
```

CONCLUSION:

The provided code effectively implements a Loan Management System, offering a user-friendly interface for both customers and administrators. Key features include:

- **Customer-centric:** Customers can easily apply for loans and track their application status.
- **Admin-friendly:** Administrators can efficiently manage loan applications, approve/decline loans, set interest rates, and disburse funds.
- **Data Persistence:** The system utilizes a CSV file to store loan data, ensuring data integrity and availability.

This system provides a solid foundation for a loan management solution, with potential for further enhancements such as payment tracking, reminders, and more advanced security measures.