MINI BANK LOAN

Project report submitted in partial fulfilment of the requirements for the award of Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

P. Ramya 24KB1A05EW

S.Jarina 24KB1A05HS

S.Roshnara 24KB1A05JF

T.Sravya 24KB1A05LE

Under the Guidance of

Smt.Sruthi.

A logo of a science and technology company

AI-generated content may be incorrect.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

N.B.K.R.IST

N.B.K.R. INSTITUTE OF SCIENCE AND TECHNOLOGY

(AUTONOMOUS)

A logo of a science and technology company

AI-generated content may be incorrect.

**CERTIFICATE**

**This is to certify that the project report entitled MINI BANK LOAN being submitted by**

P. Ramya 24KB1A05EW

S. Jarina 24KB1A05HS

S. Roshnara 24KB1A05JF

T. Sravya 24KB1A05LE

In partial fulfilment for the award of the Degree of Bachelor of technology in computer science and engineering to the N.B.K.R.IST is a record of Bonafede work carried out under my guidance and supervision

Smt. Sruthi Dr. A. Raja Sekhar Reddy

Designation M. Tech, Ph. D

Head of the Department.

DECLARATION

I hereby declare that the dissertation entitled

MINI BANK LOAN submitted for the B. Tech

Degree is my original work and the dissertation has not formed the basis award of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar

Date:5 may 2025

P. Ramya 24KB1A05EW

S. Jarina 24KB1A05HS

S. Roshnara 24KB1A05JF

T. Sravya 24KB1A05LE

Acknowledgement:

I would like to thank my project guide [Instructor's Name] for their invaluable guidance and encouragement throughout the development of this project. I also thank my institution for providing the infrastructure and learning resources that made this work possible. Lastly, I extend my gratitude to friends and family for their constant support and feedback.

Abstract:

The Mini Bank Loan System is a software application developed to digitize and streamline the loan management process for banks. It automates user registration, loan application, approval, and repayment. Admins can monitor and manage applications with ease. This system improves efficiency, reduces errors, and enhances customer experience by offering an easy-to-use digital platform for loan-related activities.

INTRODUCTION:

Banks handle numerous loan applications daily, making manual processing time-consuming and error-prone. This project aims to simplify that by developing a computerized system. The system allows customers to apply for loans and check status, while admins can approve, reject, or manage loans. It increases speed, accuracy, and accountability in loan processing.

Objective of the Project:

* Automate the loan process
* Allow users to apply for loans online
* Enable administrators to manage loan data
* Reduce paperwork and processing time
* Improve transparency and reliability
* Make data retrieval faster and safer
* Ensure secure access for both users and admins

Scope of the Project:

* Designed for small or medium banks
* Provides secure login for users and admins
* Users can apply for and track loan applications
* Admins can manage, approve, or reject loans
* Includes repayment tracking and loan history
* Scalable for future integration with other banking modules
* Can be extended to include mobile apps

System Requirements:

* Frontend: HTML, CSS, JavaScript
* Backend: Python (Flask/Django) or PHP
* Database: MySQL or SQLite
* OS: Windows/Linux
* Server: Apache/Nginx
* Browser: Chrome/Firefox
* Hardware Requirements:
* 2+ GHz processor
* 4 GB RAM or more250 GB hard disk

System Design:

Data Flow Diagrams (DFD):

* Level 0: Shows overall user-admin-system interaction
* Level 1: Details user registration, loan request, admin approval

Entity-Relationship Diagram (ERD):

* User (user\_id, name, email)
* Admin (admin\_id, name, email)
* Loan (loan\_id, amount, status, interest)
* Repayment (payment\_id, date, amount, loan\_id)
* Relationships: User → Loan → Repayment

System Implementation:

Modules:

* User Module:

Register, login, apply for loan, view status

* Admin Module:

Login, view applications, approve/reject, track repayments

* Loan Module: Stores loan data, status, terms
* User Interface:

Simple web UI for easy access

Dashboard for users/admins

Forms for loan applications and status checking

Alerts and status notifications

Program:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CUSTOMERS 5

#define NAME\_LENGTH 50

// Customer structure

typedef struct {

int id;

char name[NAME\_LENGTH];

float income;

int eligible; // 1 if eligible, 0 otherwise

} Customer;

// Linked list node for active loans

typedef struct LoanNode {

int customerId;

float loanAmount;

float emi;

struct LoanNode\* next;

} LoanNode;

// Function to check eligibility

int checkEligibility(float income) {

return income >= 30000; // Eligible if income >= 30,000

}

// Function to add a loan to the linked list

LoanNode\* addLoan(LoanNode\* head, int customerId, float loanAmount, float emi) {

LoanNode\* newNode = (LoanNode\*)malloc(sizeof(LoanNode));

newNode->customerId = customerId;

newNode->loanAmount = loanAmount;

newNode->emi = emi;

newNode->next = head;

return newNode;

}

// Function to display all loans

void displayLoans(LoanNode\* head, Customer customers[], int n) {

printf("\n--- Active Loans ---\n");

LoanNode\* curr = head;

while (curr != NULL) {

// Find customer name

char\* cname = "Unknown";

for (int i = 0; i < n; i++) {

if (customers[i].id == curr->customerId) {

cname = customers[i].name;

break;

}

}

printf("Customer: %s | Loan Amount: %.2f | EMI: %.2f\n", cname, curr->loanAmount, curr->emi);

curr = curr->next;

}

}

int main() {

// Sample customer data

Customer customers[MAX\_CUSTOMERS] = {

{1, "Alice", 35000, 0},

{2, "Bob", 25000, 0},

{3, "Charlie", 40000, 0},

{4, "Diana", 18000, 0},

{5, "Eve", 50000, 0}

};

// Check eligibility for each customer

printf("--- Customer Eligibility ---\n");

for (int i = 0; i < MAX\_CUSTOMERS; i++) {

customers[i].eligible = checkEligibility(customers[i].income);

printf("Customer: %s | Income: %.2f | Eligible: %s\n",

customers[i].name,

customers[i].income,

customers[i].eligible ? "Yes" : "No");

}

// Linked list for active loans

LoanNode\* loanList = NULL;

// Example: Add loans for eligible customers

for (int i = 0; i < MAX\_CUSTOMERS; i++) {

if (customers[i].eligible) {

float loanAmount = 100000; // Example loan amount

float emi = loanAmount / 12; // Simple EMI calculation

loanList = addLoan(loanList, customers[i].id, loanAmount, emi);

}

}

// Display all active loans

displayLoans(loanList, customers, MAX\_CUSTOMERS);

// Free memory

LoanNode\* temp;

while (loanList != NULL) {

temp = loanList;

loanList = loanList->next;

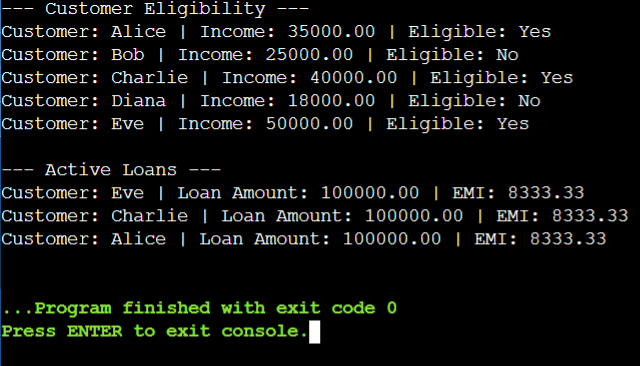
free(temp);

}

return 0;

}

Output:



Advantages:

* Fast and automated loan processing.
* Reduced human error and paperwork.
* User-friendly interface.
* Secure data handling.
* Efficient tracking of repayments.
* Saves time and operational cost.
* Easy to scale or integrate with other systems.
* Enhances customer satisfaction.

Conclusion:

This C program checks loan eligibility based on income, assigns loans to eligible customers using a linked list, and displays active loans. It demonstrates structure usage, dynamic memory, and basic loan processing logic.

References:

* GeeksforGeeks: https://www.geeksforgeeks.org
* MDN Web Docs: https://developer.mozilla.org
* W3Schools: https://www.w3schools.com
* Python/Django Docs: https://docs.djangoproject.com
* MySQL Documentation: https://dev.mysql.com/doc
* Stack Overflow: https://stackoverflow.com
* GitHub open-source loan system examples

THANKYOU