**MINI BANK LOAN SYSTEM**

**Project report submitted in partial fulfillment of the Requirements for the Award of the**

**Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

SD.SHABNAM - 24KB1A05KL

T. GEETHA - 24KB1A05KZ

T.KHUSHITHA - 24KB1A05LJ

U.SREE VIDYA - 24KB1A05LX

**Under the Guidance of**

**SMT. Shruthi**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**N.B.K.R Institute of Science and Technology**

**Affiliated by JNTU Anathapur**  
Vidyanagar, Kota, Tirupati Dist-524413, Andhra Pradesh

India

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



###### **CERTIFICATE**

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

SD.SHABNAM - 24KB1A05KL

T. GEETHA - 24KB1A05KZ

T.KHUSHITHA - 24KB1A05LJ

U.SREE VIDYA - 24KB1A05LX

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the N.B.K.R Institute of Science and Technology is a record of bonafied work carried out under my guidance and supervision.

|  |  |
| --- | --- |
| **SMT.SRUTHI**  **Assistant Professor**  **Designation** | **Dr A Raja Sekhar Reddy**  **M.Tech, Ph.D**  **Head of the Department** |

**DECLARATION**

I hereby declare that the dissertation entitled **MINI BANK LOAN SYSTEM** submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: SD..SHABNAM (24KB1A05KL) Date : T. GEETHA (24KB1A05KZ)

T.KHUSHITHA (24KB1A05LJ)

U.SREE VIDYA (24KB1A05LX)

**ACKNOWLEDGEMENT**

I sincerely thank my faculty guide, Ms. B. Shruthi, for her guidance and support throughout my project, "Mini Bank Loan system". I also thank Dr. A. Raja Sekhar Reddy, Head of Department, for providing the necessary resources and encouragement.

Gratitude to my friends and teammates for their cooperation and motivation, and to my family for their constant support.

This project enhanced my knowledge of C Programming, especially in Linked Lists, Structures, and Dynamic Memory Management, while giving me valuable insight into real-world application development.

Thank you all once again!

### **ABSTRACT OF THE PROJECT**

### **PROJECT TITLE: MINI BANK LOAN SYSTEM**

The Mini Bank Loan System is a simple yet effective C-based application designed to simulate basic banking operations related to customer loan management. It demonstrates core programming principles such as modularity, memory management, and data structure utilization, making it ideal for educational and training purposes.

This project enables users to manage a list of customers eligible for loans using static arrays and to maintain a dynamic record of active loans using linked lists. A menu-driven interface allows users to add and view eligible customers, as well as record and display loan details including loan amount, EMI, and tenure.

The system offers practical exposure to:

• Implementing and manipulating data structures.

• Designing user-interactive programs in C.

• Structuring code using modular functions.

• Handling real-time input/output scenarios.

Despite its simplicity, the Mini Bank Loan System provides a foundation for understanding how larger financial systems operate and can be further extended with file handling, search features, and graphical interfaces.

### **TABLE OF CONTENTS**

**Chapter 1 INTRODUCTION------------------------------------------------------------------------**

**1.1.1 Problem Statements--------------------------------------------------------------------**

**1.1.2 Scope---------------------------------------------------------------------------------------------**

**1.1.3 Objectives-----------------------------------------------------------------------------------**

**Chapter 2 LITERALLY SURVEY/EXISTING SYSTEM-----------------------------------**

**Chapter 3 SOFTWARE REQUIREMENT ANALYSIS-------------------------------------**

**3.1.1 Functional Requirements------------------------------------------------------------**

**3.1.2 Non-Functional Requirements--------------------------------------------------------------**

**Chapter 4 SOFTWARE DESIGN------------------------------------------------------------------------**

**4.1.1 Data Structures Used-----------------------------------------------------------------------------**

**4.1.2 Module Design-------------------------------------------------------------------------------------**

**4.1.3 Flow Chart------------------------------------------------------------------------------------------**

**Chapter 5 CODING---------------------------------------------------------------------------------------**

**5.1.1 Sample Code Snippet----------------------------------------------------------------------------**

**5.1.2 Source Code----------------------------------------------------------------------------------------**

**Chapter 6 TESTING-------------------------------------------------------------------------------------**

**6.1.1 Testing Approach----------------------------------------------------------------------------------**

**6.1.2 Black Box Testing---------------------------------------------------------------------------------**

**6.1.3 White Box Testing---------------------------------------------------------------------------------**

**Chapter 7 OUTPUT SCREEN/RESULTS--------------------------------------------------------------------**

**7.1.1 Sample Output: Eligible Customers----------------------------------------------------**

**7.1.2 Sample Output: Active Loans----------------------------------------------------------------**

**7.1.3 Output Screenshots---------------------------------------------------------------------------------**

**Chapter 8 CONCLUSION AND FURTHER WORK--------------------------------------------------**

**8.1.1 Conclusion--------------------------------------------------------------------------------------------**

**8.1.2 Future Enhancements-------------------------------------------------------------------------------**

**REFERENCE--------------------------------------------------------------------------------------------------------**

### **CHAPTER 1**

**1. INTRODUCTION**

Banking systems today are complex and heavily database-driven. However, for educational purposes, simulating a smaller version using simple C code can serve as a strong learning experience. This Mini Bank Loan System simulates eligibility tracking and loan record management using arrays and linked lists, offering a hands-on approach to learning core concepts such as memory management, pointers, and data structure integration.

This project models two key aspects:

1. Managing loan eligibility (using arrays for fixed-size customer storage).

2. Handling loan issuance and record-keeping (using linked lists for dynamic, scalable data management).

**1 1.1 Problem Statement**

Existing banking software is too advanced for beginners to grasp. There is a need for an academic project that teaches core principles of software design and data structures through practical examples. The aim is to simulate essential operations like checking eligibility and processing loan records with a simple user interface.

**1.1.2 Scope**

The scope of this project includes:

- Storing a fixed list of customers eligible for loans.

- Adding, displaying, and managing active loan records.

- Implementing CLI-based interactive functionality.

- Creating modular and scalable code for future upgrades.

**1.1.3 Objectives**

- Utilize arrays for fixed-length data (eligible customers).

- Employ linked lists for dynamic data (active loans).

- Build modular, readable C code demonstrating good programming practices.

- Create an educational tool to illustrate the practical use of data structures in real-world contexts.

**CHAPTER 2**

**2. LITERATURE SURVEY / EXISTING SYSTEM**

In most real-world banking systems, data is handled using relational databases with a robust backend and GUI front-end. These systems are complex and involve multiple modules like customer profiles, transaction histories, credit scores, and regulatory compliance checks. For beginners in programming, such systems are too large to comprehend.

This project narrows the scope to simulate just the loan eligibility and management module, using data structures taught in early programming courses. While traditional systems use SQL databases, here arrays and linked lists are used to offer similar storage and retrieval logic on a smaller scale.

By using C language, students gain experience in memory allocation, pointer manipulation, and algorithmic flow, which helps build a solid foundation for advanced programming and software development.

**CHAPTER 3**

**3. SOFTWARE REQUIREMENT ANALYSIS**

**3.1.1 Functional Requirements**

- Add eligible customer to an array (static storage).

- Add a new loan record with customer name, amount, EMI, and tenure.

- Display all eligible customers.

- Display all active loans.

- Simple menu-driven command-line interface for interaction.

**3.1.2 Non-functional Requirements**

- Platform Independent: Runs on any machine with a C compiler.

- Performance: Instant processing due to in-memory structures.

- Usability: Simple menu makes it easy to navigate.

- Scalability: Linked list allows adding any number of loans without predefined limits.

**CHAPTER 4**

**4. SOFTWARE DESIGN**

The software is structured using two main components: the eligibility module using arrays and the loan management module using linked lists. The system follows a modular approach, where each functionality is implemented as a separate function. This separation aids in debugging, code reusability, and logical clarity.

**4.1.1 Data Structures Used**

- Array: Used to store names of eligible customers. Limited in size but easy to use and access.

- Linked List: Used to dynamically store active loan records. Offers flexibility to grow at runtime and ideal for frequent inserts/deletes.

**4.1.2 Module Design**

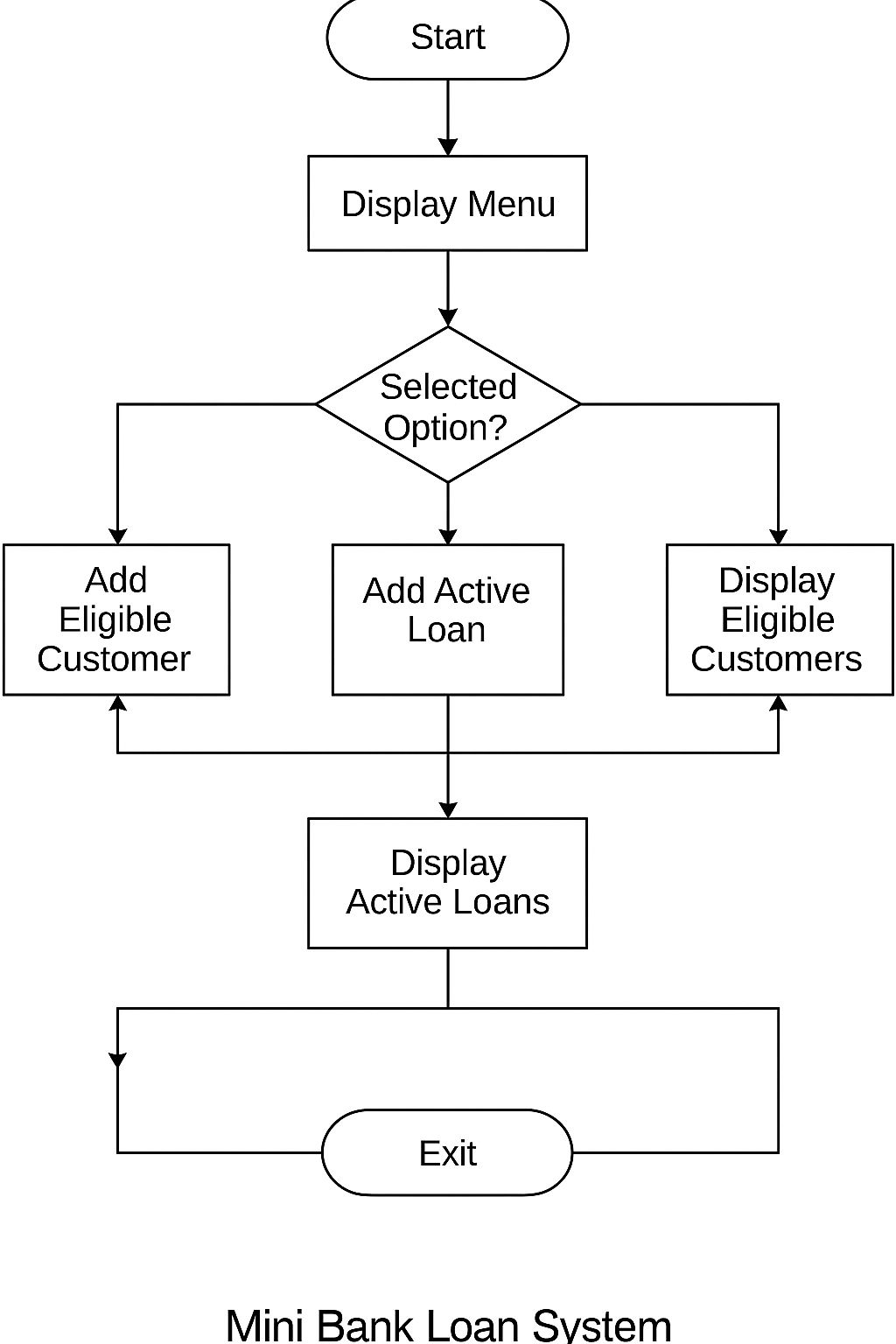
- Main Menu: Offers options to add customers, loans, and view data.

- Eligibility Module: Adds and stores fixed number of eligible customers.

- Loan Module: Dynamically adds loan entries using a linked list structure.

- Display Functions: Traverse and display data from both data structures.

**4.1.3 Flowchart**

****

### **CHAPTER 5**

### **5. CODING**

### The code is written in the C programming language. It consists of several modular functions for adding eligible customers, managing loan entries, and displaying the stored data.

### Key elements of the code include:

### - Structures for defining customer and loan data.

### - Functions for each operation: adding/displaying customers and loans.

### - Use of dynamic memory allocation for the linked list.

### **5.1.1 Sample Code Snippet**

### // Loan structure

### struct Loan {

### char name[50];

### float amount;

### float emi;

### int tenure;

### struct Loan\* next;

### };

### // Add loan function

### void addLoan(char name[], float amount, float emi, int tenure) {

### struct Loan\* newLoan = (struct Loan\*)malloc(sizeof(struct Loan));

### strcpy(newLoan->name, name);

### newLoan->amount = amount;

### newLoan->emi = emi;

### newLoan->tenure = tenure;

### newLoan->next = NULL;

### // Logic to insert at end of linked list

### }

### **5.1.2 SOURCE CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CUSTOMERS 100

#define NAME\_LENGTH 50

// Array for Eligible Customers

char eligibleCustomers[MAX\_CUSTOMERS][NAME\_LENGTH];

int totalEligible = 0;

void addEligibleCustomer(const char\* name) {

if (totalEligible < MAX\_CUSTOMERS) {

strcpy(eligibleCustomers[totalEligible], name);

totalEligible++;

} else {

printf("Eligible customer list is full!\n");

}

}

void displayEligibleCustomers() {

printf("\nList of Customers Eligible for Loans:\n");

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

printf("%d. %s\n", i + 1, eligibleCustomers[i]);

}

}

// Linked List for Active Loans

typedef struct LoanNode {

char customerName[NAME\_LENGTH];

float loanAmount;

float emiAmount;

int tenureMonths;

struct LoanNode\* next;

} LoanNode;

LoanNode\* loanHead = NULL;

LoanNode\* createLoanNode(const char\* name, float loan, float emi, int tenure) {

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

strcpy(newNode->customerName, name);

newNode->loanAmount = loan;

newNode->emiAmount = emi;

newNode->tenureMonths = tenure;

newNode->next = NULL;

return newNode;

}

void addActiveLoan(const char\* name, float loan, float emi, int tenure) {

LoanNode\* newLoan = createLoanNode(name, loan, emi, tenure);

newLoan->next = loanHead;

loanHead = newLoan;

}

void displayActiveLoans() {

printf("\nActive Loans with EMI Details:\n");

LoanNode\* current = loanHead;

while (current != NULL) {

printf("Customer: %s | Loan: %.2f | EMI: %.2f | Tenure: %d months\n",

current->customerName, current->loanAmount,

current->emiAmount, current->tenureMonths);

current = current->next;

}

}

// Main Function with User Input

int main() {

int choice;

char name[NAME\_LENGTH];

float loan, emi;

int tenure;

while (1) {

printf("\n--- Mini Bank Loan System ---\n");

printf("1. Add Eligible Customer\n");

printf("2. View Eligible Customers\n");

printf("3. Add Active Loan\n");

printf("4. View Active Loans\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar();

switch (choice) {

case 1:

printf("Enter customer name: ");

fgets(name, NAME\_LENGTH, stdin);

name[strcspn(name, "\n")] = '\0';

addEligibleCustomer(name);

break;

case 2:

displayEligibleCustomers();

break;

case 3:

printf("Enter customer name: ");

fgets(name, NAME\_LENGTH, stdin);

name[strcspn(name, "\n")] = '\0';

printf("Enter loan amount: ");

scanf("%f", &loan);

printf("Enter EMI amount: ");

scanf("%f", &emi);

printf("Enter tenure in months: ");

scanf("%d", &tenure);

getchar();

addActiveLoan(name, loan, emi, tenure);

break;

case 4:

displayActiveLoans();

break;

case 5:

printf("Exiting...\n");

**CHAPTER 6**

**6. TESTING**

**6.1.1 Testing Approach**

### We used both black-box and white-box testing methods to validate functionality and ensure memory safety in operations involving dynamic allocation.

### **6.1.2 Black Box Testing**

### This approach tests the system without knowledge of internal code. The test cases are based on input/output behavior.

### | Test Case | Input | Expected Output |

### |-----------|------------------------------|------------------------|

### | TC1 | Add eligible customer | Customer added |

### | TC2 | Add loan for existing user | Loan added successfully |

### | TC3 | Display customers | List of names printed |

### | TC4 | Display loans | Loan details printed |

### **6.1.3 White Box Testing**

### We performed line-by-line debugging and pointer validation to ensure memory safety, proper initialization of nodes, and correct pointer traversal during list insertion and display.

### **CHAPTER 7**

### **7. OUTPUT SCREENS / RESULTS**

### The program compiles successfully and executes without runtime errors. Below are the sample results and output screenshots from test execution.

### **7.1.2 Sample Output: Eligible Customers**

### 1. Shabnam

### 2. Geetha

### 3. Vidya

### 4. Khushitha

### **7.1.2 Sample Output: Active Loans**

### Loan Records:

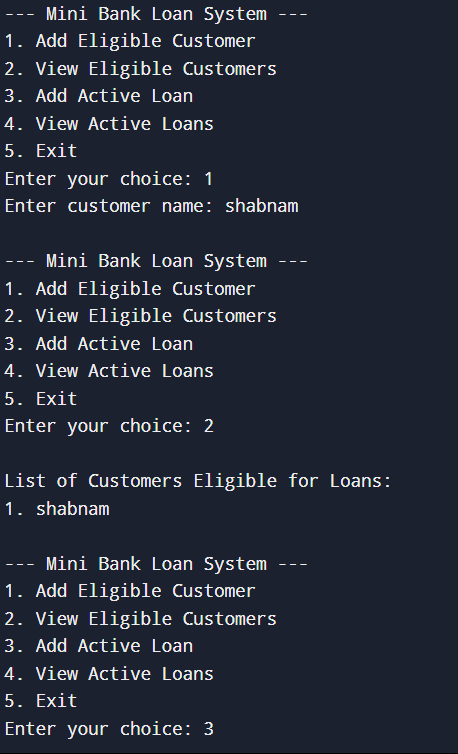
### Customer: Geetha

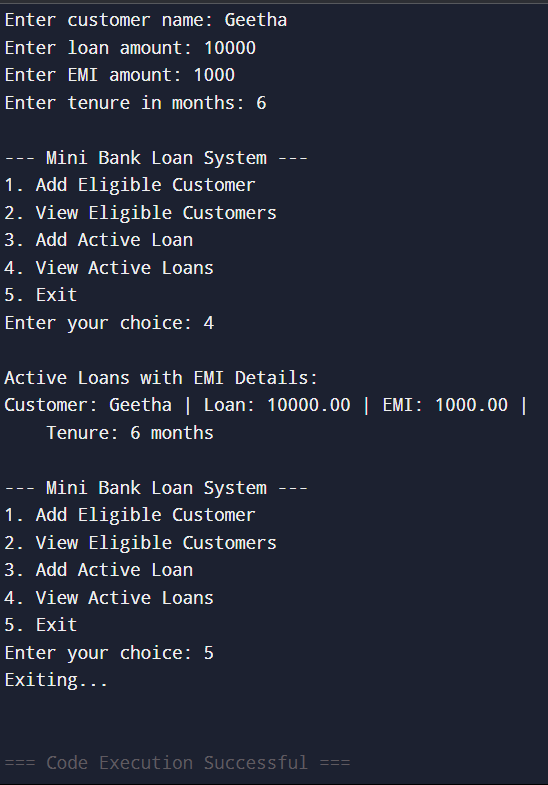
Loan Amount: 10000

EMI: 1000

Tenure: 6 months

### **7.1.3 Output Screenshot**





**CHAPTER 8**

**8.CONCLUSION AND FURTHER WORK**

**8.1.1 Conclusion**

The Mini Bank Loan System successfully demonstrates how fundamental data structures such as arrays and linked lists can be used to simulate banking functionalities. It provides an educational platform for students to explore logic implementation, memory management, and modular programming in C.

**8.1.2 Future Enhancements**

- Introduce file handling to store and retrieve data permanently.

- Expand the program to support loan approval criteria.

- Implement search and delete functionalities for loan records.

- Add a graphical or menu-driven interface for better usability.

**REFERENCES**

1. Brian W. Kernighan, Dennis M. Ritchie. "The C Programming Language", 2nd Edition, Prentice Hall.

2. Tanenbaum, A. S. "Data Structures Using C", Pearson Education.

3. https://www.geeksforgeeks.org/data-structures/ (for linked list implementation references)

4. C Programming Tutorials and Examples - https://www.programiz.com/c-programming.