TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

Bank Management System

Submitted By:

Bidhan Shrestha THA081BEI009

Kabindra Panta THA081BEI012

Nirajan Chaudhary THA081BEI022

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2025

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report
On
Bank Management System

Submitted By:

Bidhan Shrestha THA081BEI009

Kabindra Panta THA081BEI012

Nirajan Chaudhary THA081BEI022

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics and Communication Engineering.

Under the Supervision of

Prajwal Pakka

March, 2025

1. DECLARATION

We hereby declare that the report of the project entitled **"Bank Management System"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics Communication and Information Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Bidhan Shrestha (Class Roll No: [081/BEI/009]) _____

Kabindra Panta (Class Roll No: [081/BEI/012]) _____

Nirajan Chaudhary (Class Roll No: [081/BEI/022]) _____

Date: March, 2025

2. CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**Bank Management System**" submitted by **Bidhan Shrestha, Kabindra Panta** and **Nirajan Chaudhary** in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

.
Project Supervisor
Saroj Shakya
Department of Electronics and Computer Engineering, Thapathali Campus

.
External Examiner
Prof. Dr. Dinesh Kumar Sharma
Department of Electronics and Computer Engineering, Pulchowk Campus

.
Project Coordinator
Dinesh Baniya Kshatri
Department of Electronics and Computer Engineering, Thapathali Campus

.
Janardan Bhatta
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2025

3. COPYRIGHT

4. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the Institute of Engineering, Tribhuvan University, for incorporating the major project into the Bachelor's course in Electronics, Communication, and Information Engineering. We are also thankful to our supervisor, **Prajwal Pakka**, as well as the Department of Electronics and Computer Engineering, Thapathali Campus, for their valuable guidance, support, and resources that contributed to the successful completion of this project.

Bidhan Shrestha (Class Roll No.: 081/BEI/009)

Kabindra Panta (Class Roll No.: 081/BEI/012)

Nirajan Chaudhary (Class Roll No.: 081/BEI/022)

5. ABSTRACT

This project focuses on the development of a Bank Management System using the C programming language. The system is designed to automate and simplify essential banking operations, such as user registration, account creation, fund deposits, withdrawals, and money transfers. By leveraging C's efficiency in handling structured data and file operations, the system ensures secure and reliable management of user accounts and transactions. Key features include a user-friendly interface for account management, real-time transaction tracking, and the ability to view account details and transaction history. The project addresses the inefficiencies of manual or outdated banking systems by providing a streamlined, automated solution for small-scale financial institutions. Through this system, users can perform basic banking tasks with ease, while the underlying code demonstrates the practical application of C programming concepts such as file handling, data structures, and modular programming.

The primary goal of this project is to create a functional and efficient banking system that can be used as a foundation for more advanced applications. By focusing on core functionalities, the system ensures simplicity and ease of use, making it suitable for educational purposes or small-scale banking operations. The project also highlights the importance of secure data handling and efficient transaction processing, which are critical in modern banking systems. This project serves as a stepping stone for understanding the practical implementation of programming concepts in real-world applications, paving the way for future enhancements such as graphical user interfaces (GUIs) and advanced security features.

Keywords: Bank Management System, C programming, account management, transaction tracking, file handling, data structures, user-friendly interface.

6. Table of Contents

1. DECLARATION ..... **i**

2. CERTIFICATE OF APPROVAL ..... **ii**

3. COPYRIGHT ..... **iii**

4. ACKNOWLEDGEMENT ..... **iv**

5. ABSTRACT ..... **v**

6. Table of Contents ..... **vi**

7. List of Tables ..... **ix**

8. List of Figures ..... **x**

9. List of Abbreviations ..... **xi**

1. Introduction **1**

1.1 Background 1

1.2 Motivation 1

1.3 Problem Definition 2

1.4 Project Objectives 2

1.5 Project Scope and Applications 2

1.6 Report Organization 3

2. LITERATURE REVIEW **4**

3. REQUIREMENT ANALYSIS **6**

3.1 Hardware Requirements 6

3.2 Software Requirements 6

3.3 Feasibility Study 7

3.3.1 Technical Feasibility 7

3.3.2 Economic Feasibility 7

3.3.3 Operational Feasibility 7

3.4 Challenges 7

4. SYSTEM ARCHITECTURE AND METHODOLOGY **9**

4.1 System Architecture 9
4.1.1 User structure (struct User): 9
4.1.2 File Storage: 9
4.1.3 Function modules: 9
4.1.4 Main Control Flow: 9
4.2. Methodology 10
4.2.1. Data Flow 10
4.2.2. Workflow 10
4.2.3 Error Handling and Validation 11
4.2.4 File Management 11
4.2.5 System Flowchart 12
5. IMPLEMENTATION DETAILS 13
5.1 Software Components 13
5.1.1 Core Functions: 13
5.1.2 File Management: 14
5.1.3 Interfacing and Protocols 14
5.2 System Workflow 14
5.2.1 User Interaction: 14
5.2.2 Transaction Handling 15
5.2.3 Setting Management: 15
5.2.4 System Integration: 16
5.3 Testing 16
6. RESULTS AND ANALYSIS 17
6.1 Introduction to Results 17
6.2 Some menu driven interfaces 17
6.3 Sample inputs and outputs for each functions 19
6.4 Error Analysis and limitations 25

6.5 Comparison of Theory vs. Practical Results 25

7. FUTURE ENHANCEMENT 26

8. CONCLUSION 27

9. APPENDICES 28

Appendix A: User Manual 28

9.1 Introduction 28

9.2 Features 28

9.3 System Requirements 28

9.4 Stepwise procedure to run the program 28

References: 31

7. List of Tables

Table 4. 1 Structured format of uesrdetail.txt file 11
Table 4. 2 Structured format of transaction_log.txt file 11
Table 6. 1 Key differences between theoretical expectations and practical results 25
Table 9. 1 Issues and solutions during program execution..... 30

8. List of Figures

Figure 4. 1 System workflow 12
Figure 5. 1 userdetail.txt File 14
Figure 5. 2 transaction_log.txt File 14
Figure 6. 1 Home page menu interface 17
Figure 6. 2 Customer menu interface 18
Figure 6. 3 User Setting menu interface 19
Figure 6. 4 Account creation page 19
Figure 6. 5 Login page 20
Figure 6. 6 userdetails.txt file after account creation 20
Figure 6. 7 Deposit money interface 20
Figure 6. 8 userdetails.txt file after money deposit 20
Figure 6. 9 Withdraw money interface 21
Figure 6. 10 userdetails.txt file after money withdraw 21
Figure 6. 11 Transfer money interface 21
Figure 6. 12 userdetails.txt file after money transfer 21
Figure 6. 13 Account statements details in transaction_log.txt file 22
Figure 6. 14 Personal detail interface 22
Figure 6. 15 Change password interface 23
Figure 6. 16 userdetails.txt file after password change 23
Figure 6. 17 Change Email interface 23
Figure 6. 18 userdetails.txt file after email change 23
Figure 6. 19 Change phone no. interface 24
Figure 6. 20 userdetails.txt file after phone no. change 24
Figure 6. 21 Bank welcome animation 24
Figure 6. 22 Bank exit animation 25

9. List of Abbreviations

DBMS	Database Management System
GUI	Graphical User interface
BMS	Bank Management System
IDE	Integrated Development Environment
OTP	One-Time Password
AI	Artificial Intelligence
CRM	Customer Relationship Management
PCB	Printed Circuit Board

1. Introduction

A bank is a financial institution that accepts deposits from the public and creates credit. Banks play a crucial role in the economy by providing various financial services to individuals, businesses, and governments.

A Bank Management System (BMS) is a software application that automates banking operations and helps manage customer accounts, transactions, and other banking services efficiently.

1.1 Background

Banking systems form the operational backbone of financial institutions, enabling efficient account management, transaction processing, and regulatory compliance. Over decades, these systems evolved from manual ledgers to sophisticated digital platforms that handle millions of transactions daily. Digital banking progressed from basic mainframe computers in the 1950s to today's AI-powered, mobile-first solutions with features like real-time payments and personalized analytics. Computerized systems are now essential for banks because they provide the necessary scale, speed, accuracy, and security required in modern financial operations. They enable 24/7 service delivery, minimize human error, reduce operational costs through automation, and create opportunities for innovative financial products and services. Without these digital systems, banks could not meet customer expectations, competitive pressures, or regulatory requirements in today's interconnected financial landscape.

1.2 Motivation

Selecting a "Bank Management System" as a C Programming minor project offers an opportunity to apply fundamental concepts like structures, pointers, file handling, and memory management to a recognizable real-world scenario while also implementing practical programming skills such as input validation, and error handling. The project's clearly defined scope—encompassing account creation, transaction processing, and balance inquiries—provides sufficient complexity to be challenging while remaining achievable within academic timeframes. Furthermore, the banking domain is

universally understood and directly relevant to professional software development environments.

1.3 Problem Definition

Traditional banking systems face numerous operational challenges that hinder efficiency and customer satisfaction. These include time-consuming manual paperwork creating service bottlenecks, human errors in calculations and data entry compromising financial record accuracy, and significant security vulnerabilities due to inadequate safeguards for sensitive information. Physical documents risk damage or theft, while the absence of real-time processing delays account updates and limits access to current information. Additional problems include restricted service availability due to banking hours, compliance difficulties without proper tracking capabilities, inefficient staff allocation to routine tasks, and limited scalability for growing customer bases. Our Bank Management System addresses these issues through automation of critical processes, real-time transaction capabilities, and comprehensive audit trails—collectively improving efficiency, accuracy and customer experience while reducing operational costs.

1.4 Project Objectives

The main objectives of our project are listed below**:**

- To demonstrate the use of programming concepts, database management, and software design principles.
- To design and implement a functional banking system that mimics real-world banking operations.

1.5 Project Scope and Applications

The Bank Management System is designed for small-scale financial institutions, catering to two primary user roles: customers and bank staff. Customers can create and manage accounts, perform transactions (deposits, withdrawals), check balances, and view transaction history, while bank staff oversee account approvals, monitor transactions and generate statements. The system offers essential features like account management, transaction processing, and reporting, making it ideal for local credit

unions, community banks, microfinance institutions, and cooperative societies. It is cost-effective and scalable though it currently lacks advanced features like loan processing. The system aims to streamline banking operations, enhance efficiency, and improve customer satisfaction for small financial institutions.

1.6 Report Organization

The summary below offers a concise overview of each chapter, enabling readers to grasp the document's structure and content. Chapter 1 introduces the project, covering its background, motivation, problem statement, objectives, scope, applications, and an outline of the report's organization. Chapter 2 examines existing research, technologies, and systems in banking management, highlighting gaps and how this project addresses them. Chapter 3 delves into the project's requirements, including hardware and software specifications, along with a feasibility study to evaluate its technical, economic, and operational viability. Chapter 4 outlines the system's design and methodology, featuring high-level architecture, flowcharts, and algorithms. Chapter 5 details the implementation process, including the hardware and software components utilized. Chapter 6 presents the project's results, analyzing system performance, user feedback, and the system's effectiveness in achieving its goals. Chapter 7 explores potential future enhancements to improve the system's functionality and scalability. Chapter 8 concludes the report by summarizing the project's achievements, challenges, and overall impact, emphasizing its value for small-scale financial institutions. Chapter 9 contains supplementary materials like circuit diagrams, PCB layouts, module specifications, and relevant commands or code snippets. Chapter 10 lists all references, research papers, and materials cited in the report.

2. LITERATURE REVIEW

Bank management systems have evolved significantly, transforming from traditional manual operations to automated digital platforms. These systems play a crucial role in ensuring efficient management of banking operations, enhancing security, and improving customer experience. Various studies have analyzed different aspects of bank management systems, including their architecture, functionalities, and limitations.

One of the primary functions of a Bank Management System is to automate daily banking operations, such as account management, transaction processing, and customer record maintenance. Research highlights that modern banking platforms incorporate secure financial management tools, providing a seamless experience for both customers and banking staff. These systems aim to reduce human errors and improve transaction speed through digital automation [1].

The architecture of banking systems varies based on the requirements of financial institutions. A study on bank management systems emphasizes the use of database management systems (DBMS), encryption algorithms to handle large-scale transactions efficiently. The integration of secure authentication mechanisms, such as OTP (One-Time Password) and biometric verification, has enhanced security in modern banking applications [2]. Another research paper suggests that blockchain technology and AI-driven fraud detection algorithms can further improve banking security and prevent cyber threats [3].

The importance of banking management systems extends beyond transaction handling. Studies indicate that these systems facilitate financial planning, risk management, and regulatory compliance for banking institutions. They also provide customer relationship management (CRM) features, enabling banks to offer personalized services based on customer data analysis [4]. Additionally, online banking portals and mobile applications have expanded banking accessibility, allowing users to perform transactions remotely without visiting physical branches.

Despite these advancements, existing banking systems face certain limitations. One major drawback is cybersecurity threats, including phishing attacks, data breaches, and hacking attempts. Research suggests that weak encryption protocols and outdated security measures in some banking systems make them vulnerable to financial fraud

[3]. Another issue is system downtime and maintenance challenges, which can affect banking operations and customer satisfaction [2]. Moreover, the high cost of implementing and maintaining advanced banking management systems limits their adoption in small-scale financial institutions [4].

Criticism of current banking systems revolves around their complexity and user experience. Some studies argue that banking applications lack user-friendly interfaces, making it difficult for non-tech-savvy individuals to navigate [5]. Others point out that excessive dependency on internet connectivity in digital banking platforms makes them inaccessible in remote areas with poor network coverage. Addressing these issues requires continuous innovation and enhancement of banking technologies to improve reliability and accessibility.

The challenges identified in existing banking systems motivate the development of improved solutions. The proposed Bank Management System in this project aims to enhance security through robust encryption, introduce a simplified user interface, and optimize database performance. By addressing key drawbacks of current banking platforms, this project intends to provide an efficient, secure, and user-friendly banking management system tailored for modern financial needs [2].

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

Laptop/Computer:

- Minimum Specifications: At least 3 GB RAM.
- Purpose: Used to run development tools like VS Code, Dev C++, and Turbo C. The laptop is the primary hardware for writing, compiling, and debugging code.
- Justification: The program is lightweight and does not require high-end hardware, making it accessible to most users.

3.2 Software Requirements

- Integrated Development Environments (IDEs):
  - Visual Studio Code (VS Code):
    - Extensions Used: Code-runner, MinGW C Compiler, C/C++ extensions.
    - Purpose: Used for writing, compiling, and debugging code. VS Code is lightweight and highly customizable, making it ideal for development.
  - Dev C++:
    - Purpose: An alternative IDE for writing and running C/C++ programs. It is simpler and easier to use for beginners.
  - Turbo C:
    - Purpose: A legacy IDE for C/C++ programming. It is included for compatibility with older systems or educational purposes.
- MinGW (C Compiler)
    - Purpose: Used to compile C/C++ code into executable programs. It is compatible with VS Code and provides efficient compilation.

3.3 Feasibility Study

3.3.1 Technical Feasibility

The project is technically feasible as all required hardware and software components are readily available and well-documented. The system requirements are minimal, with only 3 GB of RAM needed, ensuring that the program can run smoothly on most modern computers. Although there are some challenges with certain functions like usleep() and clrscr(), these have been addressed with platform-specific alternatives such as Sleep() from windows.h and system("cls") for Windows or system("clear") for Linux. These solutions ensure compatibility across different platforms without significant issues.

3.3.2 Economic Feasibility

The project is highly economically feasible due to its minimal operating costs. Most users already own a laptop with the required specifications, such as 3 GB of RAM, which eliminates the need for additional hardware investments. All the necessary software tools, including Visual Studio Code, Dev C++, Turbo C, and MinGW, are free and open-source, removing licensing costs. The use of free tools and minimal hardware requirements ensures that the project remains cost-effective.

3.3.3 Operational Feasibility

The program is designed with a user-friendly, menu-driven interface that makes it easy to use, even for beginners with no programming knowledge. By running on the terminal, it ensures compatibility across different systems, including Windows, Linux, and macOS. Furthermore, maintenance is straightforward due to the lightweight nature of the program and the use of standard tools, which makes ongoing support and updates easy to manage.

3.4 Challenges

- Compatibility Issues: Functions like usleep() are not natively supported on Windows. This issue is mitigated by using platform-specific alternatives such as Sleep() from windows.h.

- Outdated Functions: Functions like conio.h and clrscr() are outdated and not supported in modern IDEs. This is resolved by using system("cls") for Windows and system("clear") for Linux.

- Hardware Limitations: While the system requirements are minimal, performance may be limited on older hardware or when dealing with larger projects.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 System Architecture

The system is designed using the procedural programming language (POP's). It consists of the following components.

4.1.1 User structure (struct User):

It acts as the central data structure for managing information. It consists of the user details such as accountNumber, username, password, phone, balance. dateofBirth, address and email.

4.1.2 File Storage:

- userdetatils.txt: Stores all user details in structured format
- transaction_log.txt: Logs all transitions with time history.

4.1.3 Function modules:

- **createAccount():** Create a new user account.
- **login():** Verify a user identity.
- **depositMoney():** Adds money in account.
- **withdrawMoney():** Take back money.
- **accountStatement():** Display transaction history.
- **isValidUsername(), isValidPassword(), isValidDateFormat(), isValidPhoneNumber(), isValidEmail():** Validate user inputs.
- **continueKey():** Waits for the user to press any key.
- **Bufferflush():** Clear input buffer.
- **header():** Display the program header.
- **boot():** Display the boot screen.

4.1.4 Main Control Flow:

The main() function acts as the central controller, managing the flow between the system dashboard, account creation, login, and banking operations.

4.2. Methodology

The system follows a procedural programming methodology, where the program is divided into functions, and each function performs a specific function. Below is the step-by-step methodology:

4.2.1. Data Flow

1. User Input:
   User communicates with the system through a text-based menu. Inputs are validated using helper methods (**isValidUsername(), isValidPassword(),** etc.).
2. Data Storage:
   User details are stored in userdetail.txt. Transactions are stored in transaction_log.txt.
3. Data Processing:
   Functions like **depositMoney()** and **withdrawMoney()** update the user's balance and log transactions. Functions like **changePassword()** and **changeEmail()** modify user data in the file.
4. Output:
   The system displays results (e.g., account balance, transaction history) to the user.

4.2.2. Workflow

1. Boot Screen:
   The **boot()** function displays a welcome screen with a loading animation.
2. System Dashboard:
   The user is presented with options to: Create a new account. Log in to an existing account. Exit the program.
3. Account Creation:
   The c**reateAccount()** function: Prompts the user to enter personal details. Validates inputs using helper functions. Generates a unique account number. Saves the new account details to userdetail.txt.
4. Login:
   The **login()** function: Authenticates the user by matching the account number, username, and password. If successful, the user is directed to the banking menu.

5. Banking Menu:

   The user can: Deposit money **(depositMoney()),** Withdraw money (**withdrawMoney()),** View account statements (**accountStatement()),** Access settings (**changePassword(), changeEmail(), changePhoneNumber()),** Log out or exit the program.
6. Transaction Logging:

   All deposits and withdrawals are logged in transaction_log.txt with a timestamp, account number, transaction type, amount, and balance.
7. Settings Management:

   The user can update their password, email, or phone number. Changes are saved to userdetail.txt.

4.2.3 Error Handling and Validation

1. Error Handling:

   Invalid inputs are not allowed and the user prompted to re-type the data. File operations (e.g., file openings) are checked for any mistake.
2. Input Validation:
   i. Username: Must be between 5 and 30 characters and contain only letters
   ii. Password: Must be at least 8 characters long and have at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character.
   iii. Date of Birth: Must be in the YYYY-MM-DD format.
   iv. Phone Number: Must be 10 digits and start with 97 or 98.
   v. Email: Must contain '@' and '.'.

4.2.4 File Management

1. **userdetail.txt :** Stores user details in structured format

Account Number	Username	Phone	Password	Balance	Date of Birth	Address	Email

Table 4. 1 Structured format of uesrdetail.txt file

2. **transaction_log.txt:** Logs transactions in the format

timestamp	accountNumber	amount	transactionType	balance

Table 4. 2 Structured format of transaction_log.txt file

4.2.5 System Flowchart

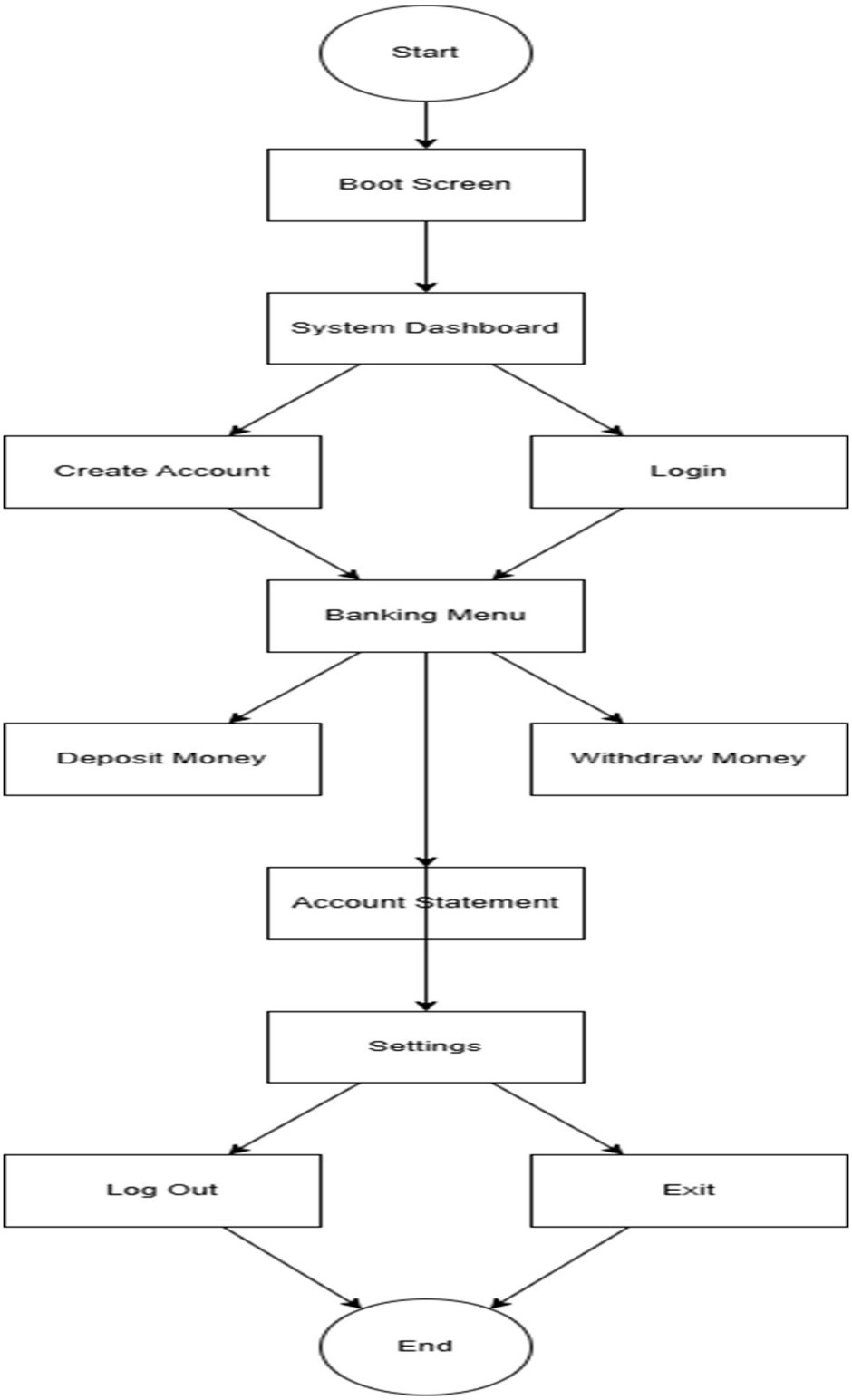


Figure 4. 1 System workflow

5. IMPLEMENTATION DETAILS

The Bank Management System is a C project that has been implemented by using modular design and file handling to manage user accounts and transactions. The description of the implementation of the system is given below:

5.1 Software Components

The system is divided into a number of functions, each handling to a unique functionality. The functions interact with each other and together form a complete bank management system.

5.1.1 Core Functions:

- Account management functions:
  - **createAccount(), login()**
    - → Handles user account creation and authorization.
    - → Reads/writes user details to userdetail.txt
- Banking operation Functions:
  - **depositMoney(), withdrawMoney(), accountStatement()**
    - → Managing financial transactions and displays account statements.
    - → update user current state in the userdetail.txt and logs transactions in the transaction_log.txt.
- Settings management functions:
  - **changePassword(), changeEmail(), changePhoneNumber()**
    - → Allows user to update their profile.
    - → change the user details in userdetail.txt
- Input Validation Functions:
  - **isValidUsername(), isValidPassword(), isValidDateFormat(), isValidPhoneNumber(), isValidEmail()**
    - → Verify the user inputs meet the requirements or criteria.
    - → Used by other functions to verify the user input.
- Other Functions:
  - **ContinueKey(), Bufferflush(), header(), boot(),** Input handling, screen clearing, display headers.

5.1.2 File Management:

- userdetail.txt:
  - ❖ store user details in structured form
  - ❖ each line contain the one user

```
12345679 john_doe 9876543210 Password@123 1000.00 1990-01-01 123_Fake_Street john.doe@example.com
```

Figure 5. 1 userdetail.txt File

- transaction_log.txt:
  - ❖ logs all transaction details with date and time.
  - ❖ each line represents the transaction

```
2024-03-14 10:15:30 12345679 500.00 DEPOSIT 1500.00
2024-03-14 10::20:45 12345679 200.00 WITHDRAW 1300.00
```

Figure 5. 2 transaction_log.txt File

5.1.3 Interfacing and Protocols

- Input/Output: User interacts with the system via a text-based interface. Inputs are validated with the Validation Module before processing.
- File Handling: Files are accessed with the standard C file I/O functions (**fopen (), fscanf (), fprintf (), fclose ()).** Data is read and written in structured form to ensure consistency.
- Error Handling: File operations are checked for error (e.g., **fopen()** returns NULL if the file cannot be opened). The invalid inputs are rejected, and the user will be prompted to re-enter data.

5.2 System Workflow

5.2.1 User Interaction:

1. Boot Screen:

   The boot function displays the welcome screen with loading screen.

2. System Dashboard:

   The user can have the option to choose the following options:

A. Create Account
B. Login
C. Exit

3. Account creation:
   The create **Account ()** function collects user details and saves them in userdetail.txt.

4. Login:
   The **login ()** function authenticates the user by matching the account number, username, and password.

5. Banking Menu:
   After successful login, the user can:
   A. Deposit money.
   B. Withdraw money.
   C. View account statements.
   D. Update settings (password, email, phone number).
   E. Log out or exit the program.

5.2.2 Transaction Handling

1. Deposit money:
   The **depositMoney()** function adds amount and user balance and transaction.

2. Withdraw money:
   The **withdrawMoney()** function draws out the balance and logs the transaction.

3. Account statement:
   The **accountStatement()** function displays the all transactions for the logged in user.

5.2.3 Setting Management:

1. Change Password: The **changPassword()** function updates the users password in the userdetail.txt.

2. Change Email: The **changPassword()** function updates the users email in the userdetail.txt.

3. Change Phone Number: The c**hangPhoneNumber()** function updates the users phone number in the userdetail.txt.

5.2.4 System Integration:

All the functions are interconnected through the function call and file handling.

5.3 Testing

1. Input validation: Test all the validation functions with various inputs to ensure that they work as expected.
2. File Handling: Verify that the userdetail.txt and the transaction_log.txt are correctly ready for work or not.
3. Transaction Logging: Ensure that all the transactions are logged in the transaction_log.txt or not.

6. RESULTS AND ANALYSIS

6.1 Introduction to Results

The Results and Analysis section presents the outcomes of the banking system project. It includes numerical and graphical representations of the system's performance, showcasing key functionalities such as account creation, transactions, balance inquiries, and error handling. The results are analyzed to compare theoretical expectations with practical implementations, highlighting any deviations and their possible causes.

Additionally, this section includes error analysis, performance evaluation, and a discussion on potential improvements to enhance system efficiency. The findings are supported by tables, graphs, and charts where applicable.

6.2 Some menu driven interfaces

6. Home page : This page consists of three functionalities
    1. Creating account for new user or customer
    2. Login page for those who are already been a user
    3. Exit the bank (End the program)

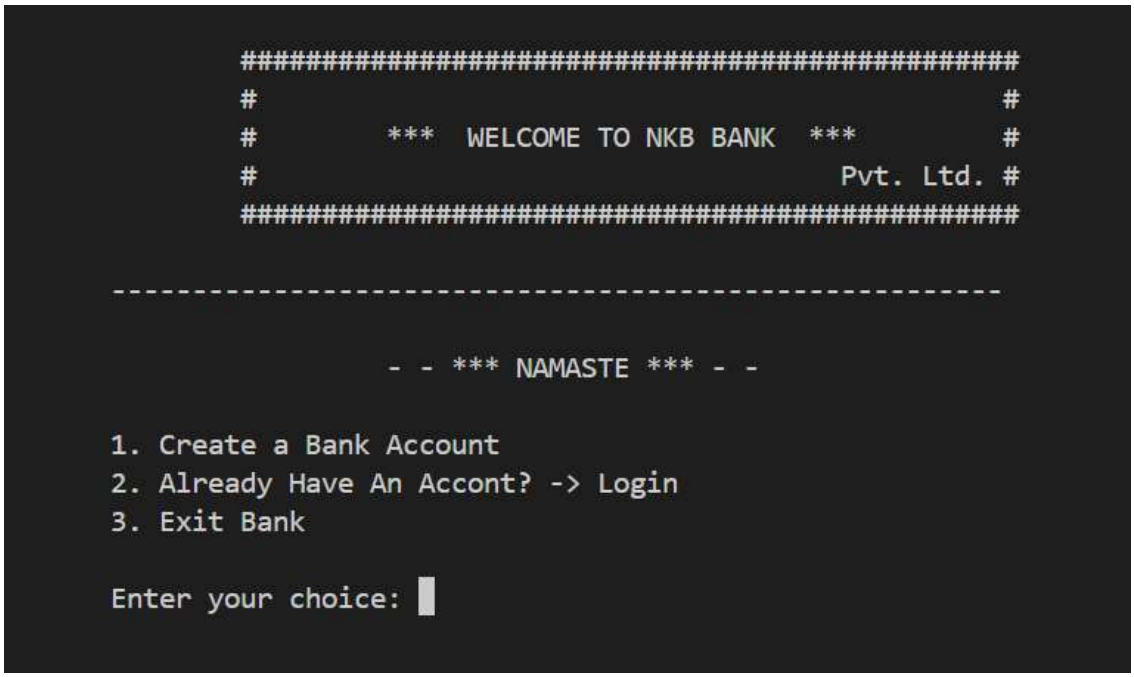


Figure 6. 1 Home page menu interface

- Customer menu sessions: It consists of following functionalities
    1. Deposit : To deposit money in the bank
    2. Withdraw: To withdraw amount from the bank
    3. Transfer: To transfer money to the other user via acc no.
    4. Transactions: For showing the transaction details
    5. Setting: Leads to the other menu driven interface
    6. Log out: Returns back to the home page interface
    7. Exit : Exit from the bank

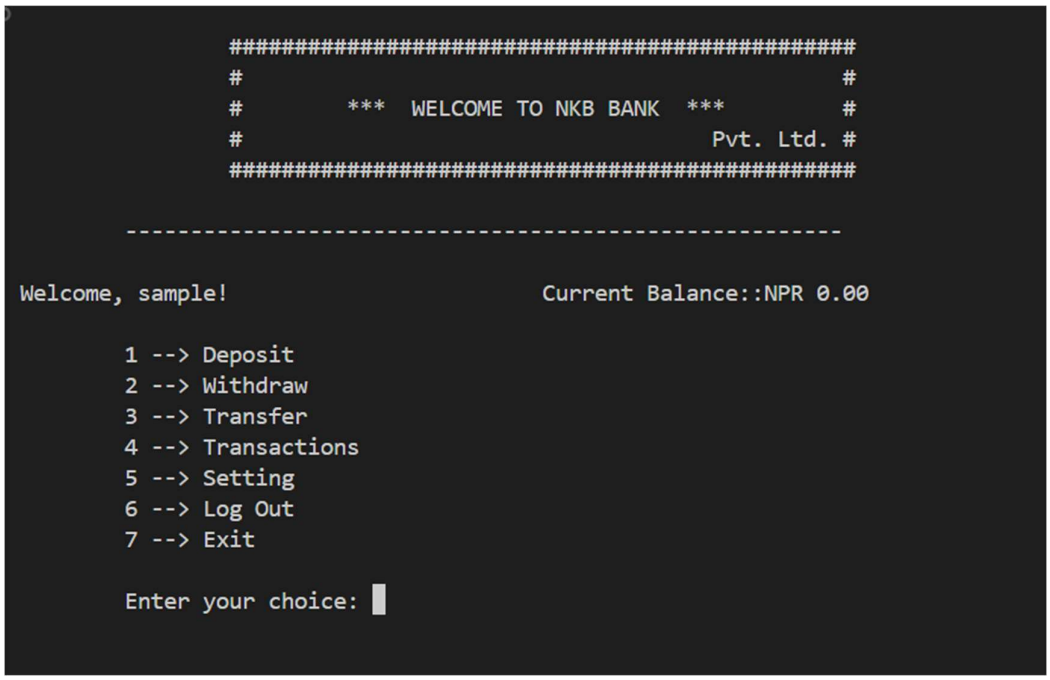


Figure 6. 2 Customer menu interface

- Setting Menu : Consists of following features
    1. Personal information: Displays the overall details of the user
    2. Change password: Facilitate user to change their password
    3. Change Email: Facilitate user to change their email
    4. Change Phone no.: Facilitate user to change their Mobile number
    5. Back : Returns the program execution to the customer menu interface

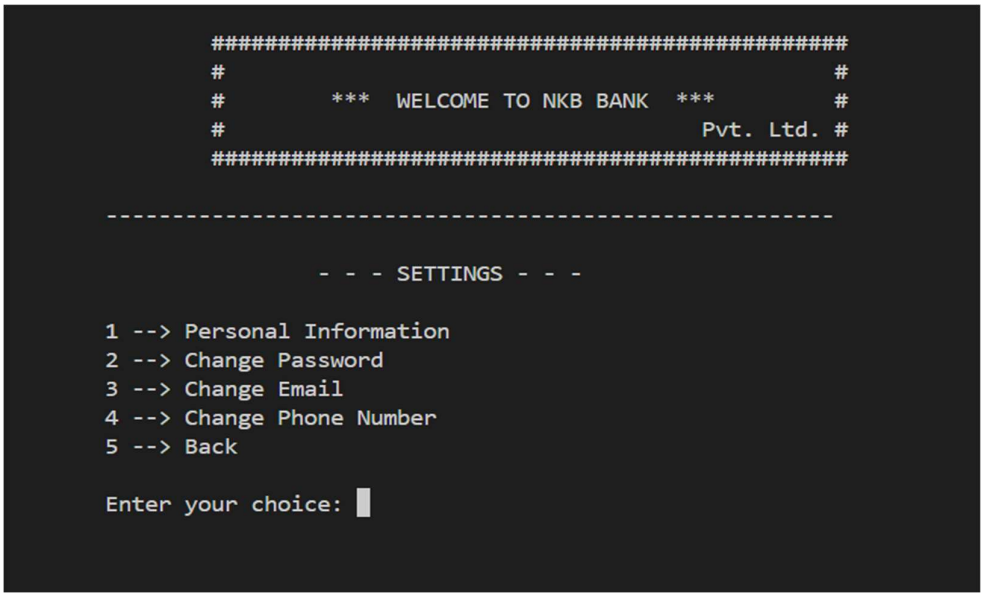


Figure 6. 3 User Setting menu interface

6.3 Sample inputs and outputs for each functions

1. Account creation and login
    A. User interface

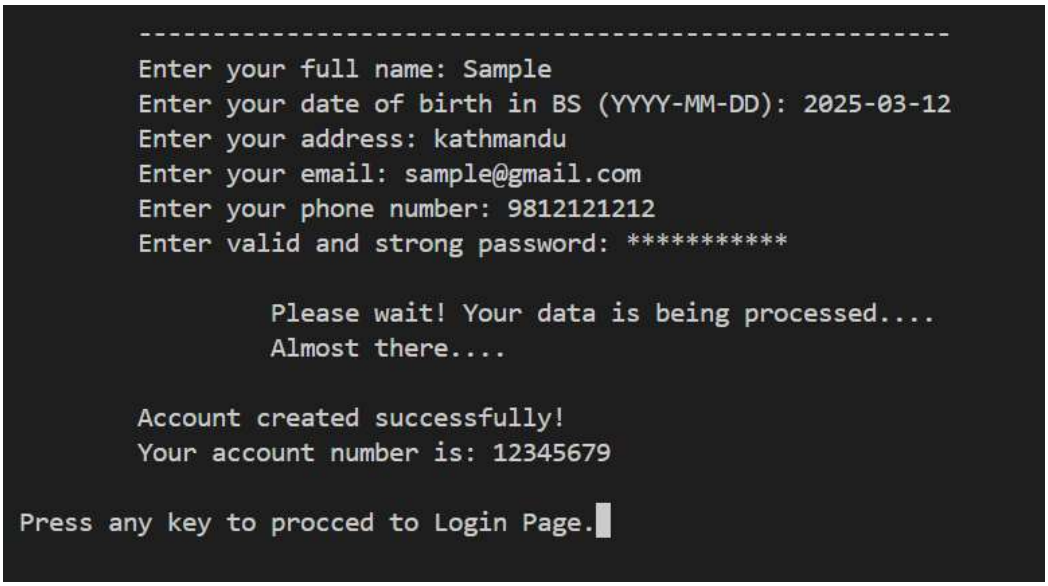


Figure 6. 4 Account creation page

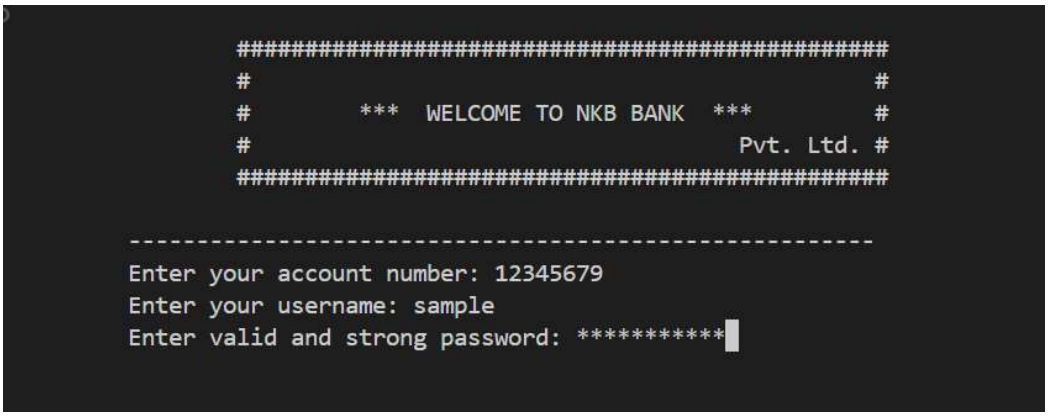


Figure 6. 5 Login page

B. Output (in user.txt file)

Figure 6. 6 userdetails.txt file after account creation

2. Deposit

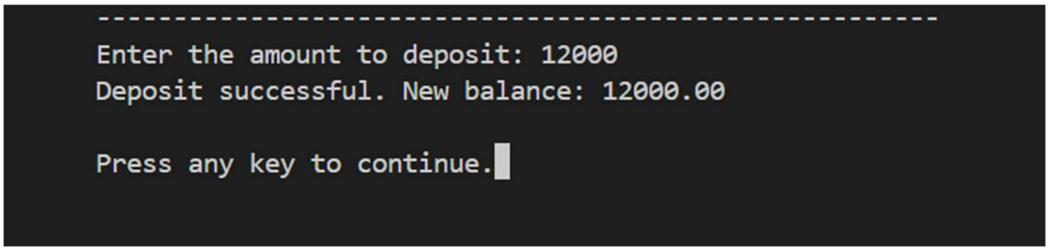


Figure 6. 7 Deposit money interface

Figure 6. 8 userdetails.txt file after money deposit

3. Withdraw

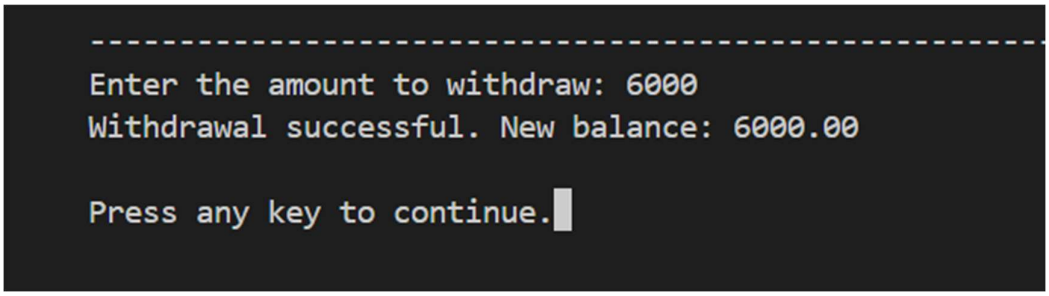
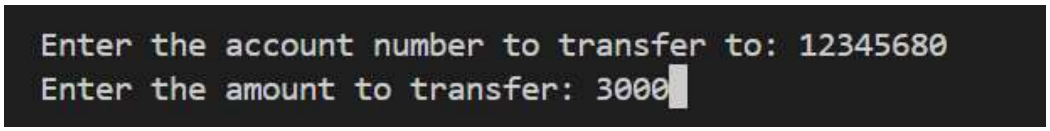


Figure 6. 9 Withdraw money interface

Figure 6. 10 userdetails.txt file after money withdraw

3. Transfer



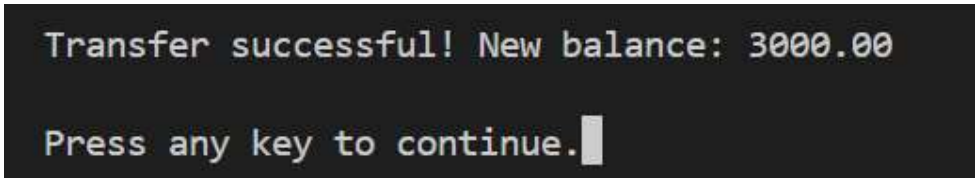


Figure 6. 11 Transfer money interface

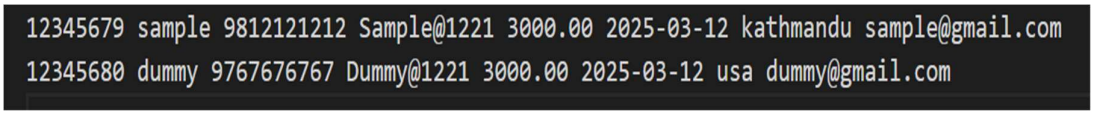


Figure 6. 12 userdetails.txt file after money transfer

Note: Before transferring the amount it is necessary to have existing account no. of receiver

4. Account statement (Transaction history)

- Output from transaction_log.txt file

```
2025-03-14 12:53:28 12345679 12000.00 DEPOSIT 12000.00
2025-03-14 12:53:34 12345679 6000.00 WITHDRAW 6000.00
2025-03-14 12:54:52 12345679 3000.00 TRANSFER 3000.00
2025-03-14 12:54:52 12345680 3000.00 RECEIVE 3000.00
```

Figure 6. 13 Account statements details in transaction_log.txt file

5. Personal details

```
                ** Personal details ***
Account Number: 12345679
Name: sample
Phone no.: 9812121212
Password: Sample@1221
Email: sample@gmail.com
Balance: NPR 3000.00
DOB: 2025-03-12 BS
Address: kathmandu
Email: sample@gmail.com


Press any key to continue.
```

Figure 6. 14 Personal detail interface

6. Change password

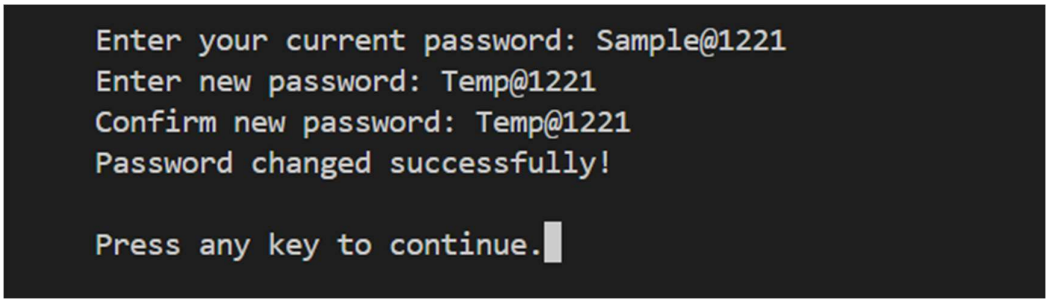


Figure 6. 15 Change password interface

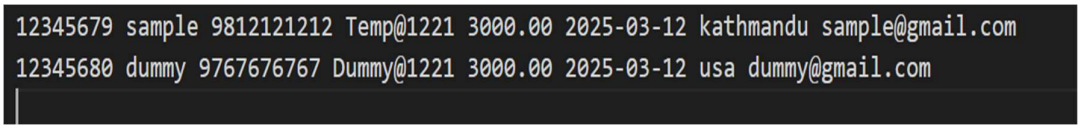


Figure 6. 16 userdetails.txt file after password change

7. Change Email

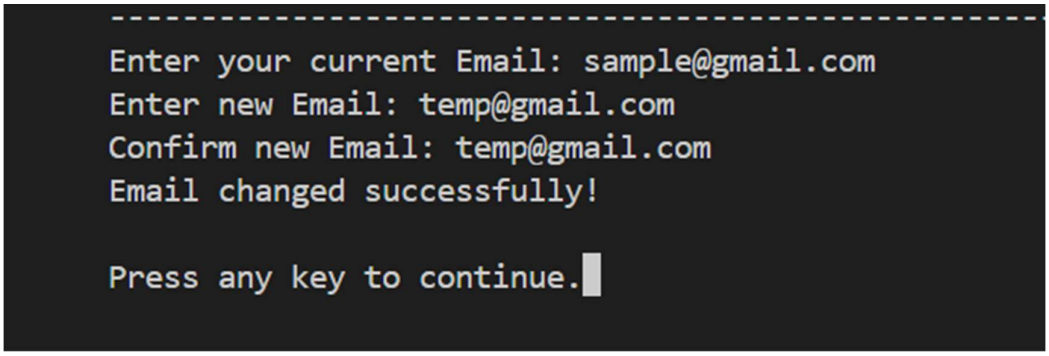


Figure 6. 17 Change Email interface

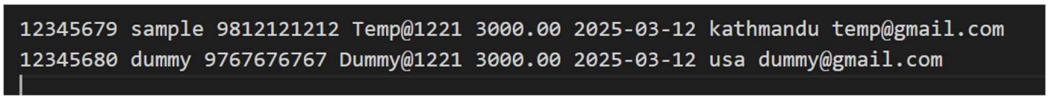


Figure 6. 18 userdetails.txt file after email change

8. Change Phone no.

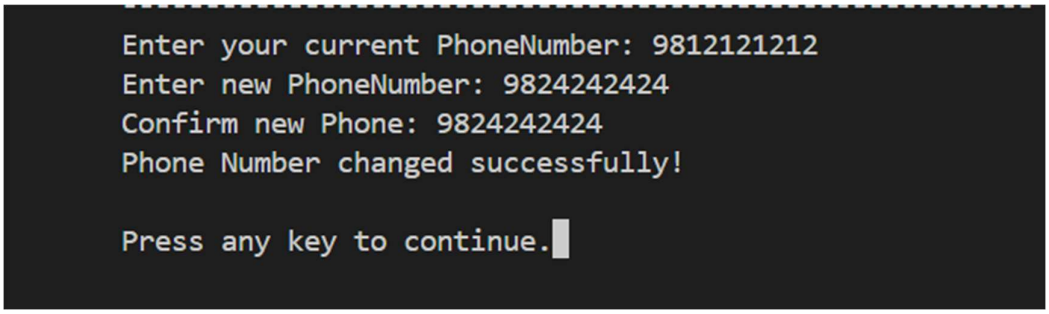


Figure 6. 19 Change phone no. interface

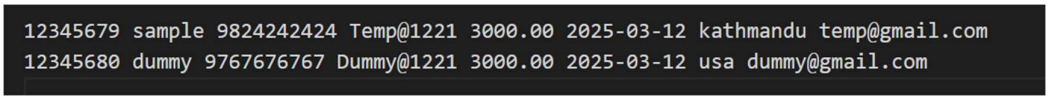


Figure 6. 20 userdetails.txt file after phone no. change

9. Boot()

- Runs at the very beginning of the program for a sec, showing the name of the bank

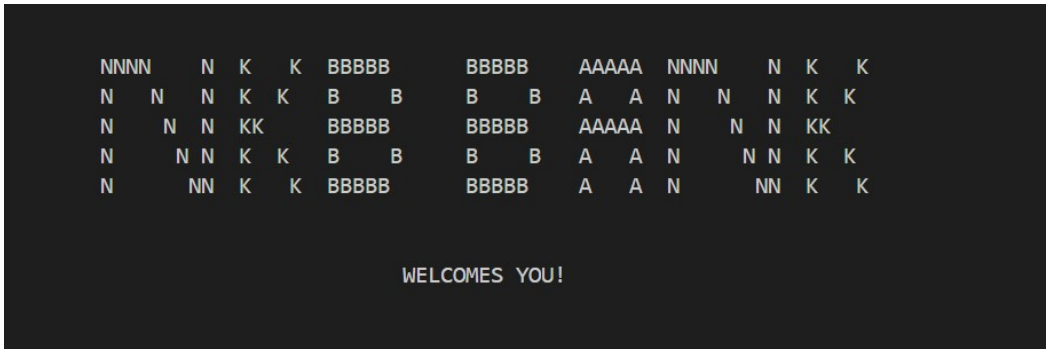


Figure 6. 21 Bank welcome animation

10. Exit ()

- Shows bank details and follow up pages at the end of the program

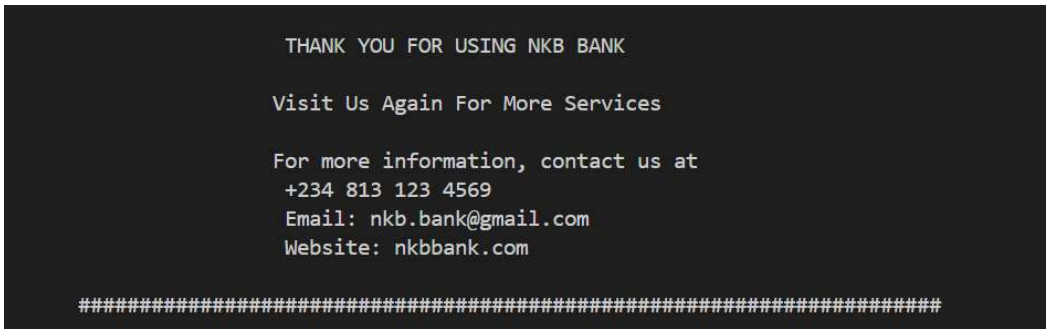


Figure 6. 22 Bank exit animation

6.4 Error Analysis and limitations

The overall program runs very well because of simple programming design and user friendly interfaces. While exploring more and more some minor issues can be faced out like during execution of each function, we couldn't step back .There is no approach of getting back during the execution of each function until the user completes the current approach. Some problems might include lack of dynamic interface, security issues etc.

6.5 Comparison of Theory vs. Practical Results

The practical implementation closely aligns with theoretical expectations in terms of functionality. However, security enhancements, such as encryption and stricter access control, are necessary to meet industry standards. Future improvements could focus on implementing hashed passwords, user authentication, and secure database integration instead of relying on text files.

Aspects	**Theory (Expected)**	**Practical (Observed)**
Data Security	Encrypted storage for security	Plaintext storage, potential risk
Transaction Handling	Smooth execution without failures	Works well but requires more validation
Error Handling	Robust handling for all cases	No errors, but logging can improve
Unauthorized Access	Restricted via authentication	Limited security, needs enhancements

Table 6. 1 Key differences between theoretical expectations and practical results

7. FUTURE ENHANCEMENT

1. Add features like interest calculation, loan management and transaction pin..
2. For added security, hashing algorithms should be employed in storing and validating credentials
3. Currently, all transactions are recorded in a single file. As the customer base grows, this approach may slow down the performance. So, we may have transactions of individual customers recorded in respective separate files.
4. The current system allows only one account per customer. For this system to be implemented in banks, it should be allowing a customer to have a variety of accounts like saving accounts, current accounts, business accounts etc.

8. CONCLUSION

The development of our Banking System in C has provided a comprehensive understanding of financial transaction management, file handling, and system security. Through the implementation of core banking functionalities such as user account management, deposits, withdrawals, and balance inquiries, we successfully created a functional and efficient system.

A comparison between theoretical expectations and practical implementation highlighted key insights. While the system performed as expected, certain areas, such as data security, error handling, and scalability, require improvements. Storing user details in a plaintext file (userdetails.txt) poses security risks, which can be mitigated through encryption and authentication mechanisms. Additionally, enhancing validation techniques would improve the reliability and robustness of transactions.

Despite these challenges, the project successfully demonstrated how a basic banking system can be implemented using C, reinforcing our understanding of file handling, data management, and security considerations. Future improvements can focus on database integration, GUI development, multi-user access, and enhanced security features to make the system more practical for real-world applications.

This project has been a valuable learning experience, bridging the gap between theoretical knowledge and practical application in banking system development, programming logic, and system security.

9. APPENDICES

Appendix A: User Manual

9.1 Introduction

This Banking System is a console-based application developed in C. It allows users to create accounts, deposit, withdraw funds, transfer money, and many more. The system ensures data security by using a password-protected login for each user.

9.2 Features

- User Registration
- Secure Login System
- Deposit and Withdrawal
- Money Transfer
- Account Management
- Transaction History

9.3 System Requirements

- Operating System: Windows/Linux
- Compiler: GCC or any C compiler
- Console-based interface

9.4 Stepwise procedure to run the program

1. Launching the Program

- Open a terminal or command prompt.
- Navigate to the directory that contains the .c file typing → **cd path/to/file**
- Compile the .c file → **gcc Main-UI.c -o main_ui**
- Run the compiled program → **./main_ui**

2. User Registration

- Select the option to create a new account.
- Enter the required details:
    - Name

- Date of Birth
- Address
- Email
- Phone no.
- Valid password

3. Logging In

- Enter your account number and username
- Provide the correct password to access your account.

4. Depositing Money

- Select the deposit option.
- Enter the amount you want to deposit.
- The new balance will be updated accordingly.

5. Withdrawing Money

- Choose the withdrawal option.
- Enter the amount to withdraw.
- Ensure you have sufficient balance.
- The system updates the balance after withdrawal.

6. Transferring Money

- Select the money transfer option.
- Enter the recipient's account number.
- Specify the amount to be transferred.
- If funds are available, the amount will be deducted and transferred.

7. Updating Account Details

- Choose the setting option.
- Modify personal details (e.g., address, contact number).
- Confirm changes before exiting.

8. Viewing Transaction History

- Select the transaction history option.
- The system will display recent transactions for your account.

9. Logging Out

- Select the logout option to head to the home page.

9.5 Security Measures

- Password encryption for account security.
- Restricted access to unauthorized users.
- Secure transaction processing to prevent fraud.

9.6 Troubleshooting

Issues	Solutions
File not found	Ensure **userdetails.txt and transaction_log.txt** exist or let the program create it.
Program crashes	Check for missing inputs and **recompile** the program.
Unexpected result	Ensure **correct inputs** for different data types during account creation.
Takes longer time to compile	Open the new terminal, **compile and run** again.

Table 9. 1 Issues and solutions during program execution

References:

[1] D. Bisht, S. Kumari, A. Katheria, and G. Yadav, "*Bank Management System*," International Research Journal of Modern Engineering and Technology Studies (IRJMETS), 11 Nov 2024. [Online]. Available: http://www.irjmets.com/uploadedfiles/paper//issue_11_november_2024/64514/final/fin_irjmets1732689383.pdf. [Accessed: 10-Mar-2025].

[2] A. Kumar, T. Sharma, S. Kumar, and P. Kumar, "To study the Bank Management System," *Journal of Emerging Technologies in Engineering and Research (JETIR)*, vol. 10, no. 5, May 2023. [Online]. Available: https://www.jetir.org/papers/JETIR2305489.pdf. [Accessed: 10-Mar-2025].

[3] Kamal Acharya, "*Online Banking Management System*," *ResearchGate*, 2024. [Online] Available: https://www.researchgate.net/publication/380214416_Online_banking_management_system.[Accessed: 10-Mar-2025].

[4] "Bank Management System," *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)".* [Online] Available: https://ijercse.com/viewabstract.php?id=14871&issue=Issue8&volume=Volume8 .[Accessed: 10-Mar-2025].

[5] "*The Design and Implementation of a Bank Management System*," *Academia.edu*, 2024. [Online]. Available: https://www.academia.edu/97047712/The_Design_and_Implementation_of_Bank_Management_System. [Accessed: 10-Mar-2025].