A First Year Project Report on

Simple Banking System

Submitted in Partial Fulfillment of the Requirements for the Degree of **BCA** under Pokhara University

Submitted by:

Prabesh Shakya, 242049

Rajin Maharjan, 242026

Rohit Khaling Rai, 242030

Date: 30/07/2025

Department of Bachelors of Computer Application

NEPAL COLLEGE OF

INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

ACKNOWLEDGEMENT

We are sincerely thankful to all those who have guided and supported us throughout this project. We would like to express our heartfelt gratitude to our college, **Nepal College of Information Technology**, and the Department of Computer Application for giving us this valuable opportunity to apply our knowledge and skills in a real-world project. We are also deeply appreciative of our project supervisor for their constant encouragement, clear guidance, and kind assistance during the entire preparation and development process.

Prabesh Shakya

Rajin Maharjan

Rohit Rai

ABSTRACT

This project presents a Simple Banking System built as a terminal application using the C programming language. It facilitates fundamental banking tasks, enabling users to open accounts, check transaction history, make deposits and withdrawals and handle customer data effectively. To ensure the system remains lightweight and efficient, it employs a file-based storage mechanism, making it ideal for environments with limited resources or no internet connectivity. The application includes a user authentication mechanism combined with permission controls to differentiate access rights between roles like manager and customers. Overall, this approach delivers a compact, reliable, and easy-to-use banking management tool focused on core functionalities without the need for complex libraries or online access.

Keywords: *Simple Banking System*, *File-based storage*, *Terminal-based*, *User authentication*, *etc.*

Contents

List of Abbreviations

SBS: Simple Banking System

Prod: Procedural

Prog: Programming

Trans: Transactions

Dep: Deposit

With: Withdraw

List of Figures

List of charts

Chapter 1 Introduction

This project aims to tackle the difficulties people face when handling basic banking operations manually by developing a straightforward, command-line banking system using the C language. It includes essential features like creating accounts, viewing balances, and processing deposits and withdrawals. While not designed for real-world commercial use, the project serves as an educational tool to showcase how structured programming techniques can be applied to solve practical problems through clear logic and modular coding [4].

Students working on this project gain hands-on experience with fundamental C programming elements such as variables, functions, data structures, and file management for storing information persistently [5]. The process highlights best practices in code organization, modular development, and debugging, contributing to higher software reliability. By constructing the system from scratch, learners enhance their analytical thinking and develop a deeper insight into how separate pieces of code come together to form a cohesive application [6].

1.1. Background:

In recent years, the demand for efficient and accessible banking solutions has grown, especially in regions where advanced digital infrastructure is limited. Traditional banking methods involving manual record-keeping or paper-based transactions are prone to errors, delays, and difficulties in maintaining accurate financial data. While modern banking systems often rely on internet-connected platforms and complex software, these solutions may not be feasible in areas with inconsistent network availability or limited technological resources [7].

To address these challenges, lightweight and offline banking applications that operate through terminal or command-line interfaces have gained importance. Such systems do not require continuous internet access or high-end hardware, making them ideal for small-scale or resource-constrained environments. Terminal-based banking solutions focus on core banking functionalities such as account management, deposits,

withdrawals, and transaction history, providing a simple yet effective way to manage financial data securely and efficiently [8].

Our project implements a Simple Banking System using C programming, designed with two key modules: the customer module and the manager module. The customer module enables users to create accounts, deposit and withdraw funds, and view their transaction history. The manager module supports administrative operations, including adding or removing users, viewing transaction histories for specific accounts, and approving high-value transactions exceeding 50,000 units, ensuring secure oversight of large transfers. This modular approach balances user autonomy with managerial control, creating a versatile and practical banking solution suitable for low-resource environments [9].

By employing a file-based backend for data storage, the system maintains portability and ease of use on basic hardware without reliance on internet connectivity. The terminal interface emphasizes simplicity and efficiency, allowing users to navigate essential banking functions without extensive training or sophisticated software requirements [10].

1.2. Problem Definition:

Many small banks and financial institutions rely on outdated or manual methods to manage transactions and accounts, which often results in inefficiencies and errors. Limited internet access in underserved areas makes it difficult to adopt modern digital banking solutions. According to the International Finance Corporation, lightweight and offline-capable systems are essential in such environments to ensure secure and timely services [11]. These constraints pose significant challenges for banks seeking to provide efficient financial management.

This project aims to develop a terminal-based Simple Banking System that supports essential banking functions like account creation, deposits, withdrawals, and transaction history tracking. It also includes a manager module to control user access and approve significant transactions, offering a practical and accessible solution for banks operating in low-resource settings.

1.3. Objectives

The main aim of this project is to design and develop a simple banking system using the C programming language that replicates essential banking features. This terminal-based application allows users to perform key functions such as account creation, deposits, withdrawals, balance checks, and account removal. The system utilizes fundamental C programming concepts like loops, conditional statements, functions, and structures to ensure a modular and maintainable design [12]. To achieve data persistence, file handling techniques are integrated, enabling the program to save and retrieve user information across multiple sessions [13]. Additionally, the system incorporates basic input validation to minimize invalid operations and enhance overall reliability. This project offers a practical platform for students to apply theoretical concepts to real-life scenarios, improving their logical thinking and debugging skills through hands-on development [14].

1.4. Scopes and Limitation

The proposed system, developed in the C programming language, is designed to handle basic banking tasks such as account creation, balance checking, deposits, withdrawals, and viewing transaction history through a simple command-line interface. Its primary focus is on simplicity, efficiency, and offline operation, making it suitable for small-scale or educational purposes. However, the system does not include advanced banking features like online transactions, loan processing, or credit scoring. Security measures are kept minimal, with only basic user authentication, and it lacks a graphical interface or multi-user support. These limitations are intentional to keep the project lightweight and focused on demonstrating core ideas.

Chapter 2 System Analysis and Design

2.1. Feasibility Study:

2.1.1. Technical Feasibility:

The project is technically feasible as the banking system can be implemented using C, covering file handling, structures, and basic logic.

2.1.2. Economic Feasibility:

The project is cost-effective since it only requires a computer with a C compiler, which is freely available.

2.1.3. Operational Feasibility:

The system will have a simple text-based interface, making it easy for students and evaluators to operate and test.

2.1.4. Schedule Feasibility:

With proper planning and regular progress, the project can be successfully developed and submitted within a semester.

2.1.5. Legal Feasibility:

As a non-commercial, academic project using dummy data, it complies with all legal and ethical guidelines.

2.2. Algorithm of the System:

Step 1: Start

Step 2: Display the Login Menu

Step 3: Ask user to enter Login credentials

Step 4: Validate Login credentials

If false, go to step 2.

If true, go to step 5.

Step 5: Check the role of the user (Manager / Customer)

Step 6: Display Menu based on role of the user.

Step 7: Execute the function user selected.

Step 8: Ask the user if they want to log out

If Yes, go to Step 9.

If No, go to step 6.

Step 9: End

2.3. Flowchart:

2.3.1. Customer Module

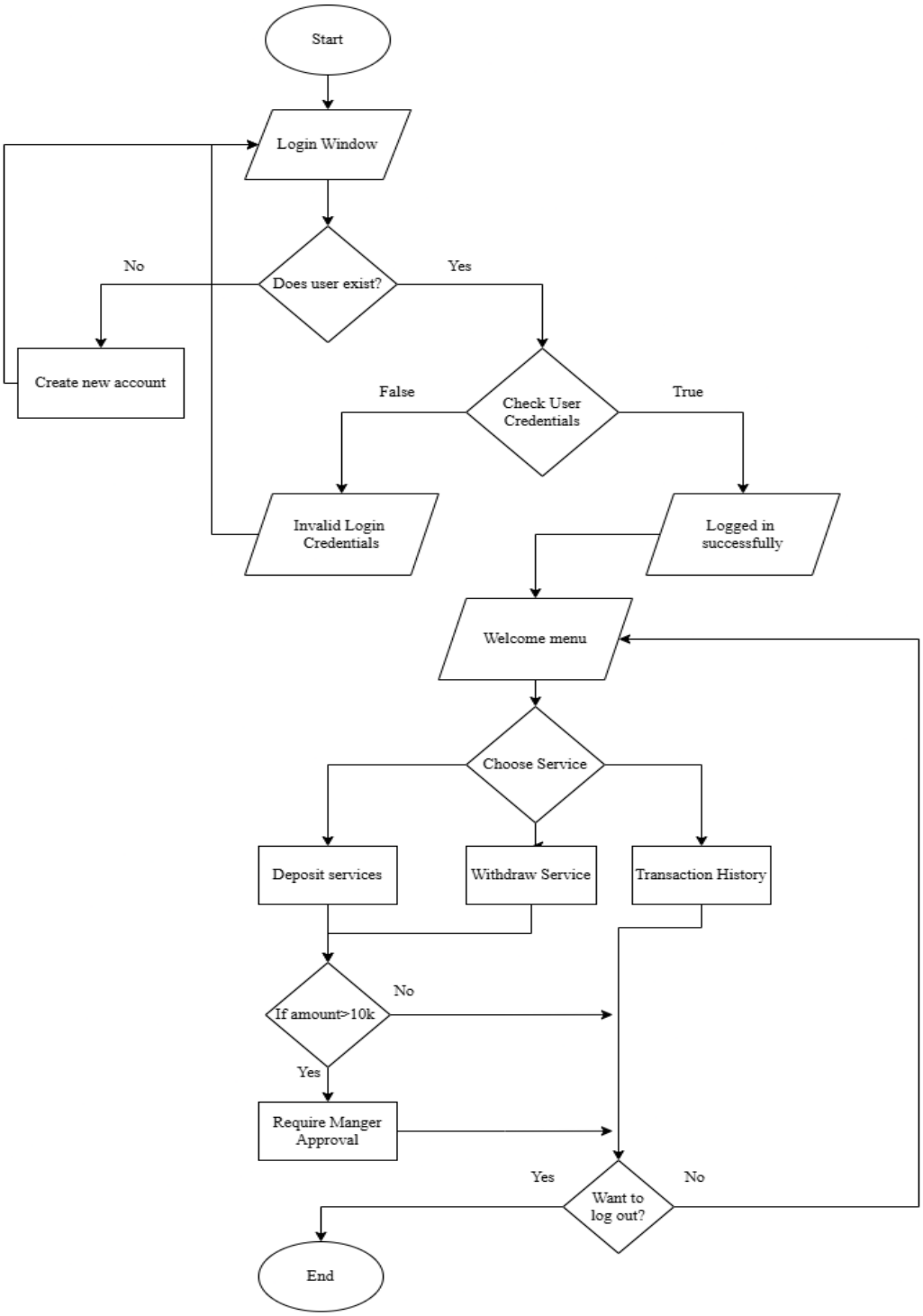

Figure 1 Customer Modul

2.3.2. Manager Module

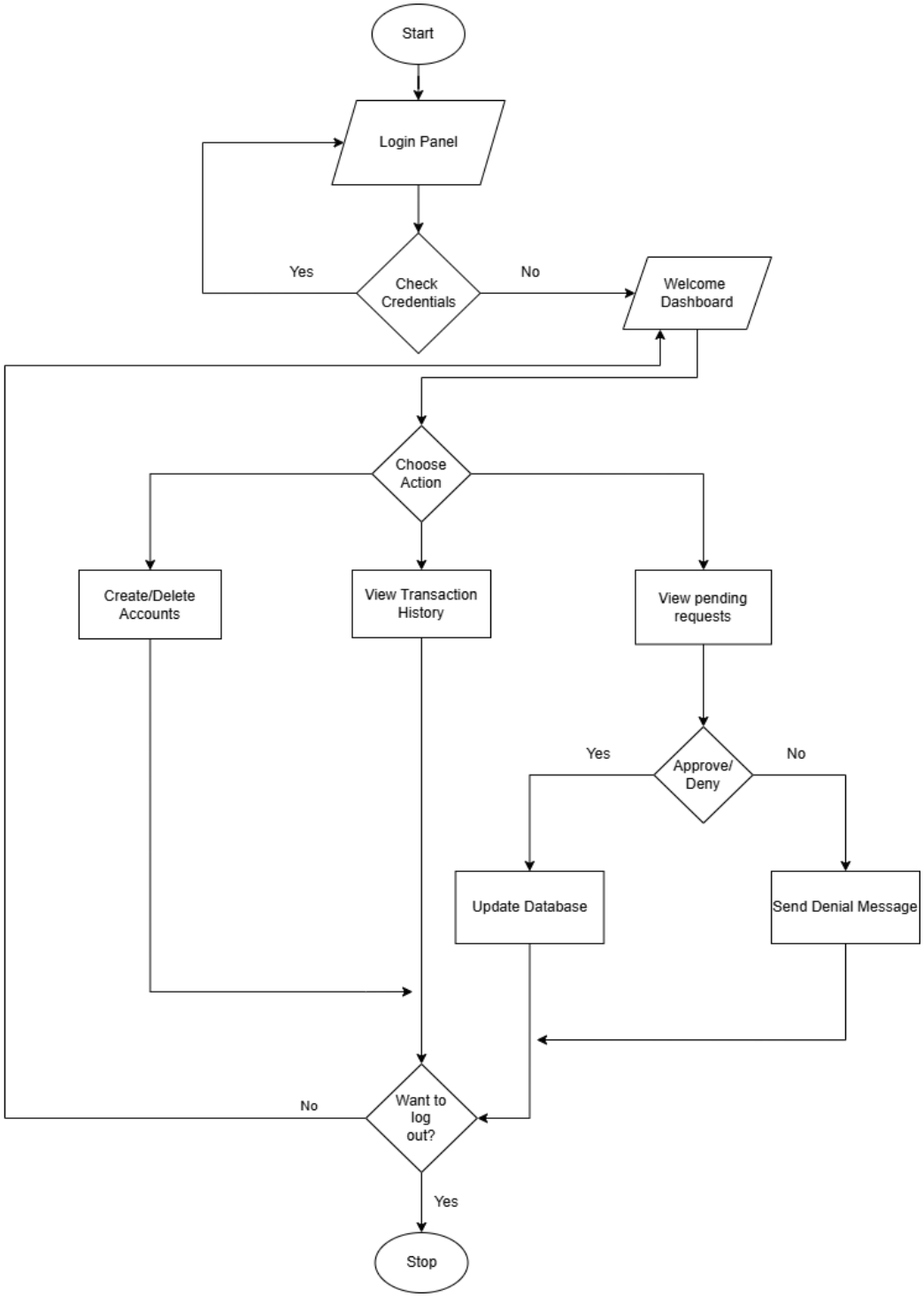

Figure 2 Manager Module

2.4. Context Diagram:

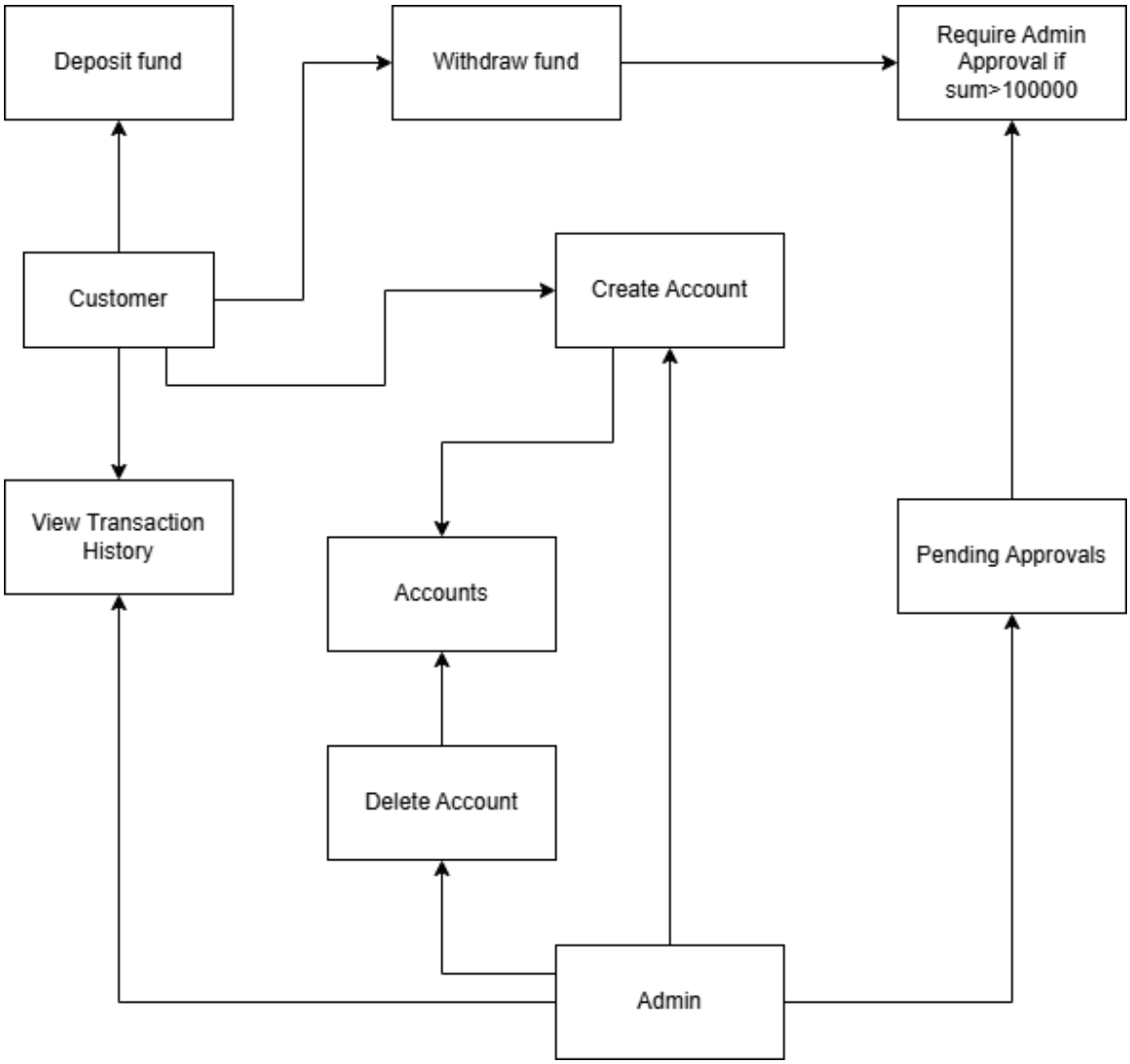

Figure 3 Context Diagram of SBS

Chapter 3 Development and Testing

3.1. Development Model: Waterfall Model

For the development of SBS, the Waterfall Model of SDLC was followed. This model consists of following phases:

1. **Requirement Analysis & Feasibility Study**

   Here, all the requirements for the development of the software is studied. Also, different types of feasibility study is conducted to ensure that the project is full proof for future.

2. **System Design**

   Here, the initial layout of the system is designed, which lays as the foundation for our software. It includes Algorithm, Flowchart, Context Diagram, etc.

3. **System Development**

   In this phase, the system undergoes the process of developing. C language is used to develop the modules required for the system.

4. **System Testing**

   In this phase, all the features of the system is tested and testers check extreme limit of the system for any kind of bugs or error.

5. **Implementation and Maintenance**

   Here, the system is presented to the users for the use. The feedbacks of the user is noted and the system will be further improved.

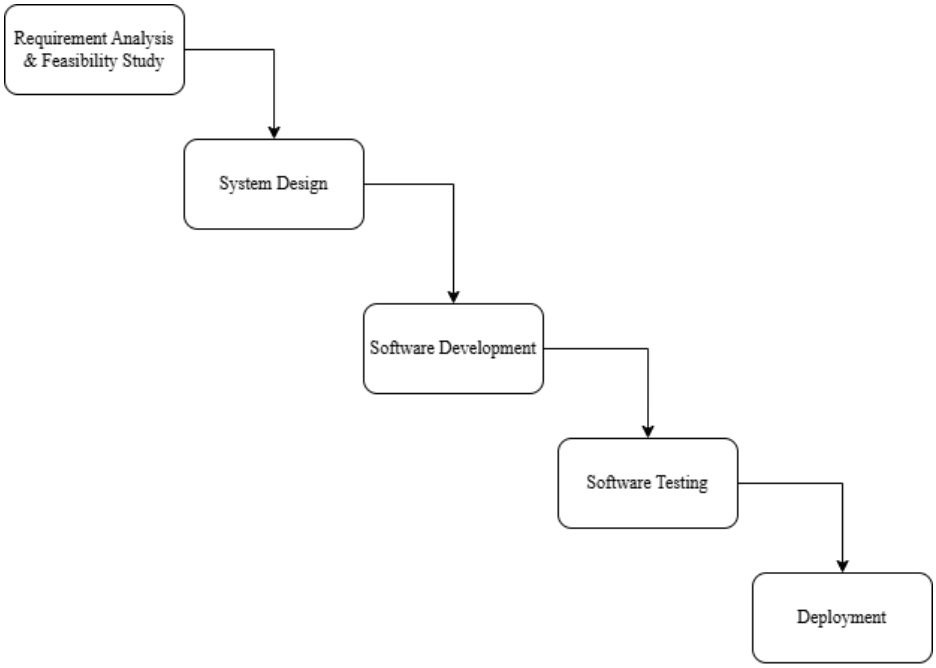

Figure 4 Waterfall Model

3.2. Development Phase

3.2.1. Tools Used:

- **Language**: C
- **Complier:** GCC
- **Editor/IDE:** VS Code, Embarcadero
- **Storage:** File Handling (.txt / .dat)
- **Platform:** Windows

3.2.2. Core Modules

- **Customer Module:**

  Create account → Deposit/ Withdraw amounts / Request transaction history → Logout
- **Manager Module:**

  Login → Create/Delete user accounts / Approve transactions / View transaction history → Logout.

3.3. Testing Phase

- **Functional Testing:**

  Test each function/module (account creation, deposit, etc.) independently to ensure they meet the required behavior.
- **Boundary Testing:**

  Check the system's behavior when minimum and maximum values (e.g., zero balance, maximum withdrawal) are used.
- **Error Handling Verification:**

  Intentionally provide invalid input (e.g., wrong account number, negative deposit) to ensure the system responds appropriately.

S.n	Unit	Test	Expected Result	Test Output
1.	Admin Module	To check if manager logins with the credentials	Logged in successfully	Sucessful
2.	Customer Module	To check whether customer can login and access service	Able to deposit withdraw money	Sucessful
3.	Validation testing	Check whether the dep/with amt is greater than 50k or not	If amt is greater than 50k, requires manager perm	Sucessful
4.	System Testing	Whether the system handles all trans or not	Customer can easily complete all trans	Sucessful

Table 1Test Case for Different Modules

Chapter 4 Literature Review

4.1. Analysis of Traditional Banking Practices

Conventional banking operations, especially in small financial institutions, often rely on manual record-keeping methods such as registers or spreadsheets. These approaches are prone to data inconsistencies, calculation errors, and delays in processing transactions. The lack of automation increases administrative workload and makes it difficult to track deposits, withdrawals, or account balances efficiently. Such systems also face limitations in providing quick access to historical transaction data, which can affect customer service and operational accuracy

4.2. Overview of Proposed banking system

The proposed terminal-based banking system aims to replace outdated, error-prone methods with a structured and automated solution. By using the C programming language, the system introduces key features like account creation, balance inquiries, deposits, withdrawals, and transaction history tracking, all managed through a lightweight file-based backend.

Chapter 5 Task Breakdown

Team Member	Task Assigned
Prabesh Shakya	Coding (Manager Module), Documentation
Rajin Maharjan	Coding (Customer Module), Documentation
Rohit Rai	Testing, Documentation

Table 2 Work Distribution

Chapter 6 Result and Conclusion

6.1 Final Result

The Simple Banking System delivers an efficient and user-friendly solution for managing essential banking operations. Built with C and file handling, it ensures smooth functionality for both customers and managers while maintaining reliable data storage.

- Main Menu Navigation:

➢ A command-line interface that allows seamless movement between the Customer and Manager modules.

- Customer Module:

➢ Customers can create accounts, check balances, deposit or withdraw funds, and review their transaction history.

- Manager Module:

➢ Managers can add or remove users, oversee account activities, and approve transactions exceeding a defined limit.

- File Handling:

➢ All account and transaction details are maintained through C file handling, ensuring data remains consistent across sessions.

- Transaction Management:

➢ Managers can monitor all user transactions while customers can easily track their own financial activity.

6.2 Conclusion

The Simple Banking System successfully demonstrates how fundamental banking operations can be efficiently managed using the C programming language. By focusing on essential features such as account creation, balance inquiry, deposits, withdrawals, and transaction history, the project delivers a lightweight yet functional solution for small-scale financial management. The use of file handling ensures data persistence, allowing users and managers to access records across multiple sessions without relying on external databases or internet connectivity. While the system is designed for simplicity, its modular structure and command-line interface make it easily adaptable for future enhancements. Overall, this project serves as both a practical tool and an educational exercise, reinforcing core programming concepts and problem-solving techniques.

Chapter 7 References

[1] MIT OpenCourseWare. (n.d.). *Practical Programming in C*. Retrieved from https://ocw.mit.edu/courses/6-087-practical-programming-in-c-january-iap-2010/

[2] Linux Documentation Project. (n.d.). *C Programming/File IO*. Retrieved from

https://tldp.org/LDP/lpg/node115.html

[3] Carnegie Mellon University. (n.d.). *Error Handling in C*. Retrieved from https://www.cs.cmu.edu/~guna/15-123S11/Lectures/ErrorHandlingInC.pdf

[4] University of Washington. (n.d.). *Modular Programming and Design*. Retrieved from
https://courses.cs.washington.edu/courses/cse374/13au/lectures/ModularProgramming.pdf

[5] IBM Developer. (n.d.). *Mastering C File I/O – Best Practices*. Retrieved from https://developer.ibm.com/articles/au-fileio/

[6] Stanford University. (n.d.). *CS Education Library – Debugging in C*. Retrieved from https://web.stanford.edu/class/cs106b/materials/cs106b-debugging-handout.pdf

[7] World Bank. Financial Inclusion Overview. 2021. https://www.worldbank.org/en/topic/financialinclusion

[8] MIT OpenCourseWare. Software Construction (6.005) Lecture Videos. 2019. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-005-software-construction-fall-2016/lecture-videos/

[9] Stanford CS Education Library. CS106A Programming Methodology. 2020. https://cs.stanford.edu/people/eroberts/courses/cs106a/

[10] Carnegie Mellon University. Introduction to Programming in Java. 2018. https://www.cs.cmu.edu/~213/

[11] International Finance Corporation., 2019. https://www.ifc.org/wps/wcm/connect/topics_ext_content/ifc_external_corporate_site/financial+institutions/resources/publications/em+compass+note+52+financial+inclusion+and+digital+finance.

[12] University of Toronto. (n.d.). *Structured Programming in C*. Retrieved from https://www.cs.toronto.edu/~penny/csc180/11s/notes/structured_programming.pdf

[13] Delft University of Technology. (n.d.). *C Programming: File Handling Basics*. Retrieved from

https://cse.tu-delft.nl/education/courses/ti1100/

[14] Harvard University CS50. (n.d.). *CS50 Notes – Debugging and C Programming Practices*. Retrieved from https://cs50.harvard.edu/x/2023/notes/4/

Chapter 8 ANNEX

8.1 Main Menu

```
==================================================
        WELCOME TO BANKING MANAGEMENT SYSTEM
==================================================

               Developed by:
                 - Prabesh Shakya
                 - Rajin Maharjan
                 - Rohit Rai

==================================================


1. Login (Manager/Customer)
2. Create Account
3. Exit
Choose an option:
```

8.2 Customer Login

```
1. Login (Manager/Customer)
2. Create Account
3. Exit
Choose an option: 1
Enter Account ID: 1001
Enter Password: 1234
Login successful! Welcome, Rajin.

===== Customer Menu =====
Current Balance: 0
1. Deposit Money
2. Withdraw Money
3. View Transaction History
4. Logout
Choose an option:
```

8.3 Manager Login

```
1. Login (Manager/Customer)
2. Create Account
3. Exit
Choose an option: 1
Enter Account ID: 1000
Enter Password: admin123
Manager login successful!

===== Manager Menu =====
1. Add User
2. Delete User
3. View Transactions of Specific User
4. Approve Pending Transactions
5. Logout
Choose an option:
```