[BANK MANAGEMENT SYSTEM]

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

GROUP-4

Under the Guidance of

(MRS. POONAM YADAV)

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[12,2023]

Certificate

Date: 3-Dec-23

This is to certify that the work present in this Project entitled "**BANK MANAGEMENT SYSTEM**" has been carried out by
Sreemayi Billa (AP22110010549)
Sriya Balanagu(AP22110010539)
Sowjanya Thota (AP22110010543)
Indu Meghana Kalluri (AP22110010540)
Vineetha Pemmasani (AP22110010580)

under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

Prof. / Dr. Poonam Pandey

Designation,

Affiliation.

Acknowledgements

I extend my sincere appreciation to the individuals and resources that played a pivotal role in the realization of this C++ bank management system project.

First and foremost, heartfelt thanks to the dedicated contributors who invested their time and expertise, shaping this project into its final form. Their commitment and collaboration were instrumental in its success.

The inspiration for this project stems from the innovative ideas from Mrs. Poonam Yadav, serving as a guiding light throughout the development process.

The journey of learning and implementing C++ was made smoother by referring to Educational Resources provided by our faculty in charge, which provided valuable insights and knowledge. The authors of these resources deserve recognition for their role in expanding our understanding.

Special recognition goes to Mrs. Poonam Yadav ma'am] for her unwavering support and guidance. Her mentorship not only facilitated technical aspects but also enriched the project with strategic insights, pushing its boundaries by monitoring frequently and giving advice to improve the learning experience.

Lastly, but equally important, I express my gratitude to friends and family for their continuous encouragement. Their unwavering support fueled the project's momentum and made this accomplishment a shared success.

In conclusion, I extend my thanks to everyone involved, directly or indirectly, in making this project a reality.

Table of Contents

Certificate i

Acknowledgements iii

Table of Contents iv

Abstract v

Abbreviations vii

List of Figures viii

1. Introduction ix

2. Methodology x

   2.1 Requirements

   2.2 Design

   2.3 Implementation

   2.4 Verification

   2.5 Maintenance

3. Discussion xii

   3.1. DFD

      3.1.1 .Level 0 DFD

      3.1.2. Level 1 DFD

      3.1.3. Level 2 DFD

   3.2 Flowchart

4. Concluding Remarks xvii

5. Future Work xviii

References xix

Abstract

In today's dynamic and competitive financial landscape, banks are constantly seeking innovative solutions to streamline operations, enhance customer satisfaction, and maintain a competitive edge. The Bank Management System (BMS) in C++ emerges as a powerful tool addressing these critical needs, offering a comprehensive and secure platform for managing the diverse facets of banking operations.

Core Functionalities

The BMS seamlessly integrates a suite of modules designed to cater to the intricate requirements of modern banking:

- Customer Account Management: This module empowers bank personnel to efficiently manage customer accounts, encompassing account creation, updation, and deletion.
- Transaction Processing: The system facilitates seamless transaction processing, handling deposits, withdrawals, transfers, and other financial transactions with accuracy and precision.
- User Authentication and Authorization: Employing robust authentication and authorization protocols, the BMS safeguards sensitive financial data by ensuring only authorized personnel can access critical information and perform designated tasks.
- Account Inquiries: Customers can conveniently access their account details, transaction history, and statements through a user-friendly interface.
- Employee Management: The BMS streamlines employee management by enabling the creation, updation, and deletion of employee records, along with assigning roles and permissions.
- Data Security: Data security remains paramount, with the BMS implementing advanced encryption techniques and access control mechanisms to protect sensitive financial information.

Technical Implementation

Leveraging the power and versatility of the C++ programming language, the BMS delivers a robust and scalable solution, capable of handling the ever-growing demands of banking institutions. The object-oriented programming paradigm ensures modularity, maintainability, and extensibility, while adhering to industry-standard best practices.

Impact and Significance

The BMS revolutionizes banking operations by:

- Enhancing Operational Efficiency: Automating routine tasks and streamlining processes significantly reduces operational costs and improves overall efficiency.
- Elevating Customer Experience: A user-friendly interface and convenient access to account information foster customer satisfaction and loyalty.
- Strengthening Data Security: Rigorous security measures safeguard sensitive financial data, mitigating the risk of cyber-attacks and data breaches.
- Promoting Scalability: The BMS's modular architecture and C++ foundation enable seamless adaptation to evolving business needs and increasing transaction volumes.

Conclusion

The Bank Management System in C++ stands as a testament to the transformative power of technology in the banking industry. By streamlining operations, enhancing security, and elevating customer experience, the BMS empowers banks to thrive in the competitive landscape and forge a path towards a secure and prosperous future.

Abbreviations

BMS	Bank Management System
OOP	Object Oriented Programming
DFD	Data Flow Diagram

List of Figures

Figure 1. Level 0 Data Flow Diagram……...…….12

Figure 2. Level 1 Data Flow Diagram……...…….13

Figure 3. Level 2 Data Flow Diagram……...…….14

Figure 4. Flowchart……...……...…….16

1. Introduction

The Bank Management System (BMS) is a comprehensive software solution designed to enhance the efficiency of banking operations. It includes modules for customer account management, transaction processing, and user authentication.

The BMS empowers banks by effectively managing customer accounts, facilitating account creation, modification, and deletion. It ensures precise transaction processing for deposits, withdrawals, and transfers. Additionally, users gain convenient access to account details and transaction history, promoting transparency.

This system enables users to create, deposit, withdraw, and check balances. Customer details are recorded, allowing users to view all account holders, close accounts, and modify account information. The project incorporates various features to constitute a robust banking system.

To automate and streamline banking operations, the C++-based Bank Management System facilitates customer account handling, transactions, and essential banking functions. It ensures modularity, extensibility, and easy maintenance through object-oriented programming (OOP).

The system's features encompass user management, account handling, transaction processing, balance inquiries, account updates, and closures. Its OOP design provides a secure and user-friendly interface for customers and bank employees, facilitating diverse operations like account creation, management, deposits, withdrawals, fund transfers, and report generation.

Beyond managing customer accounts, the system tracks different account types (savings, checking, fixed deposits), records transactions, and generates reports and statements. Overall, the Bank Management System serves as a comprehensive tool for efficient and secure banking processes.

2. Methodology

The methodology consists of 5 phases:

- Requirement phase
- Design phase
- Implementation phase
- Verification phase
- Maintenance phase

Let's look into the different phases in detail.

2.1 Requirements:

In this phase, we gather and document the requirements for the banking management system. These could include features like user account creation, transaction processing, security, and performance requirements.

In order to create a C++ banking management system, the project needs to include basic features such strong user authentication that allows users to access the system securely. With a unique identification for every account, account management functions such as creation, closure, and changes are essential. Basic financial operations including fund transfers, withdrawals, and deposits should be supported by the system. In order to provide smooth responses to unforeseen circumstances, the system should also have error management.

2.2 Design:

During the design phase, we create a detailed blueprint of your banking system. This includes designing the structure of the program and dividing it into different
modules for easier understanding.

Account Management:

- Allow the creation of new customer accounts.
- Enable customers to have multiple bank accounts.

Transaction Handling:

- Record transactions for deposits and withdrawals.
- Ensure that transactions are atomic and maintain data consistency.

Customer Interaction:

- Provide functionality for customers to view their account details.

2.3 Implementation:

In this phase we will write the actual code for displaying the main menu and using a switch case structure for taking the input from the user.
There are 8 modules in the implementation part:

1. **New account**: it takes input from the user that is required for account creation and creates a new account.

2. **Deposit**: It takes the amount to be deposited into the account as input from the user and updates the account balance.

3. **Withdrawal:** it takes the amount that must be withdrawn as input from the user and updates the account balance accordingly .

4. **Balance:** it takes the input from the user and displays the account balance.

5. **All bank holder list**: it displays all the names of the account holders of the bank

6. **Close account**: it deletes the bank account of the user

7. **Update account**: it takes the necessary input from the user and updates the information in the bank account.

8. **Exit**: it takes the control flow out of the program.

2.4 Verification:

Verification includes testing to ensure that the system functions correctly.We wrote test cases to validate different aspects of your banking system, such as account creation, money transfers, and security measures.Perform debugging, and testing to identify and fix any issues or defects.

2.5 Maintenance:

After the system is deployed, the maintenance phase begins. It involves ongoing updates, and enhancements to keep the banking management system running smoothly.Address any issues, bugs, that arise. Add new features or improvements as the requirements change.

3.Discussion

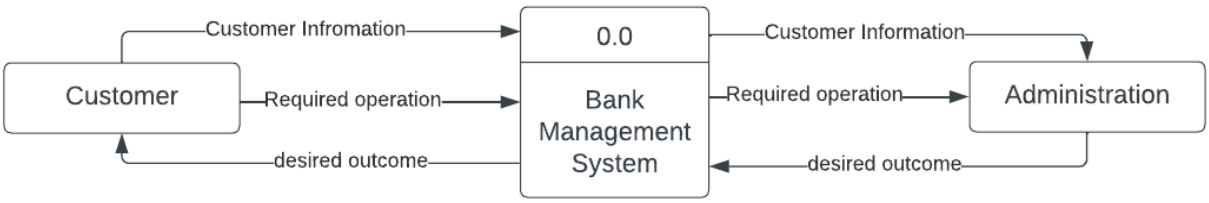
Now let's look deeper into our project.

3.1. DFD

DFD stands for Data Flow Diagram. It is a graphical representation of the flow of data within an information system. It illustrates how data is input, processed, stored, and output in a system. The data flow diagram for any project is very essential to identify the flow of data and the necessary operations to be done on the data.
It provides an overview of how the system processes the data, which data is stored and what are the expected outputs or results.
DFDs can be divided into different levels, which provide varying degrees of detail about the system. For our project we have taken 3 levels of dfd's:

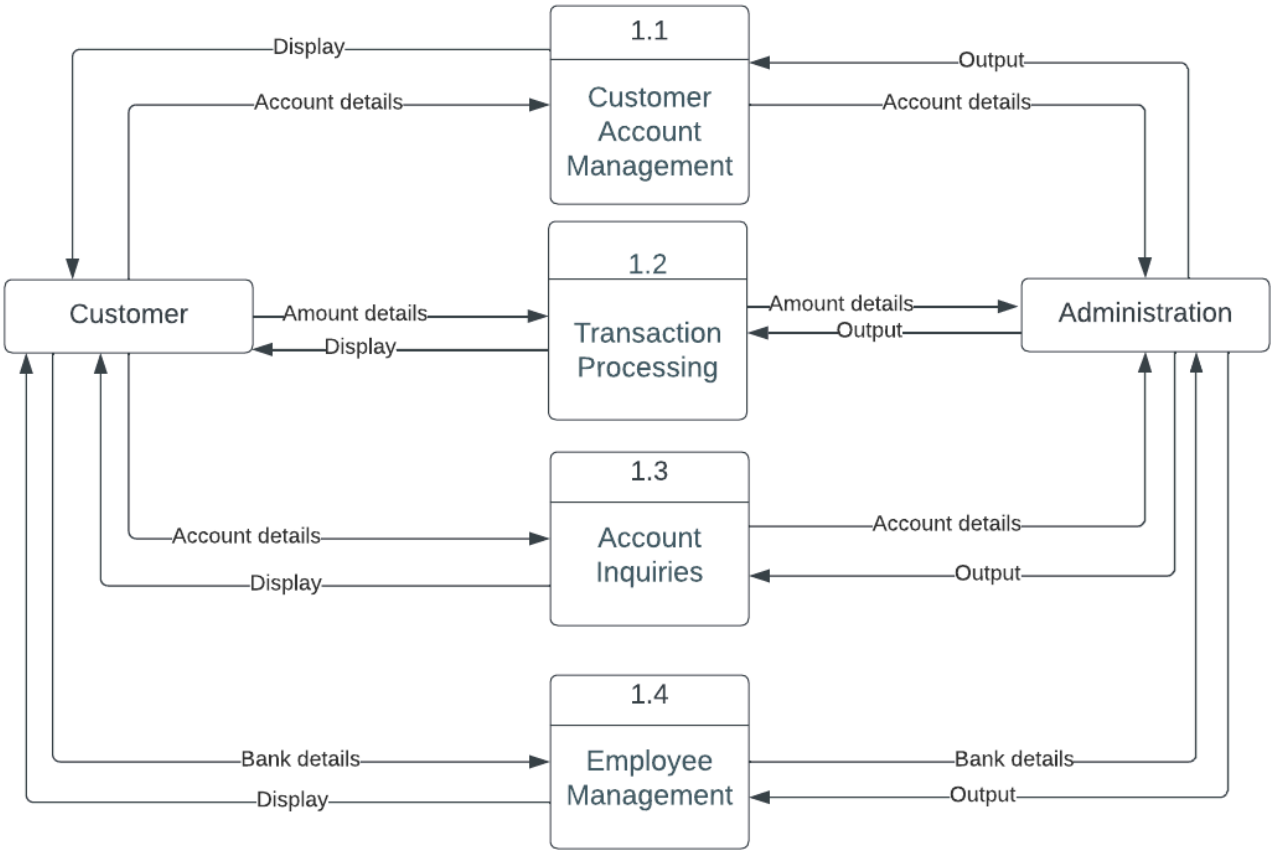
3.1.1. 0 Level DFD for Bank Management System

It is the highest level of abstraction and provides an overview of the entire system.
Level 0 shows the main entities interacting with the Bank Management System, which are the Admin and the Customer.
The user feeds data into the system and then receives the output from it.



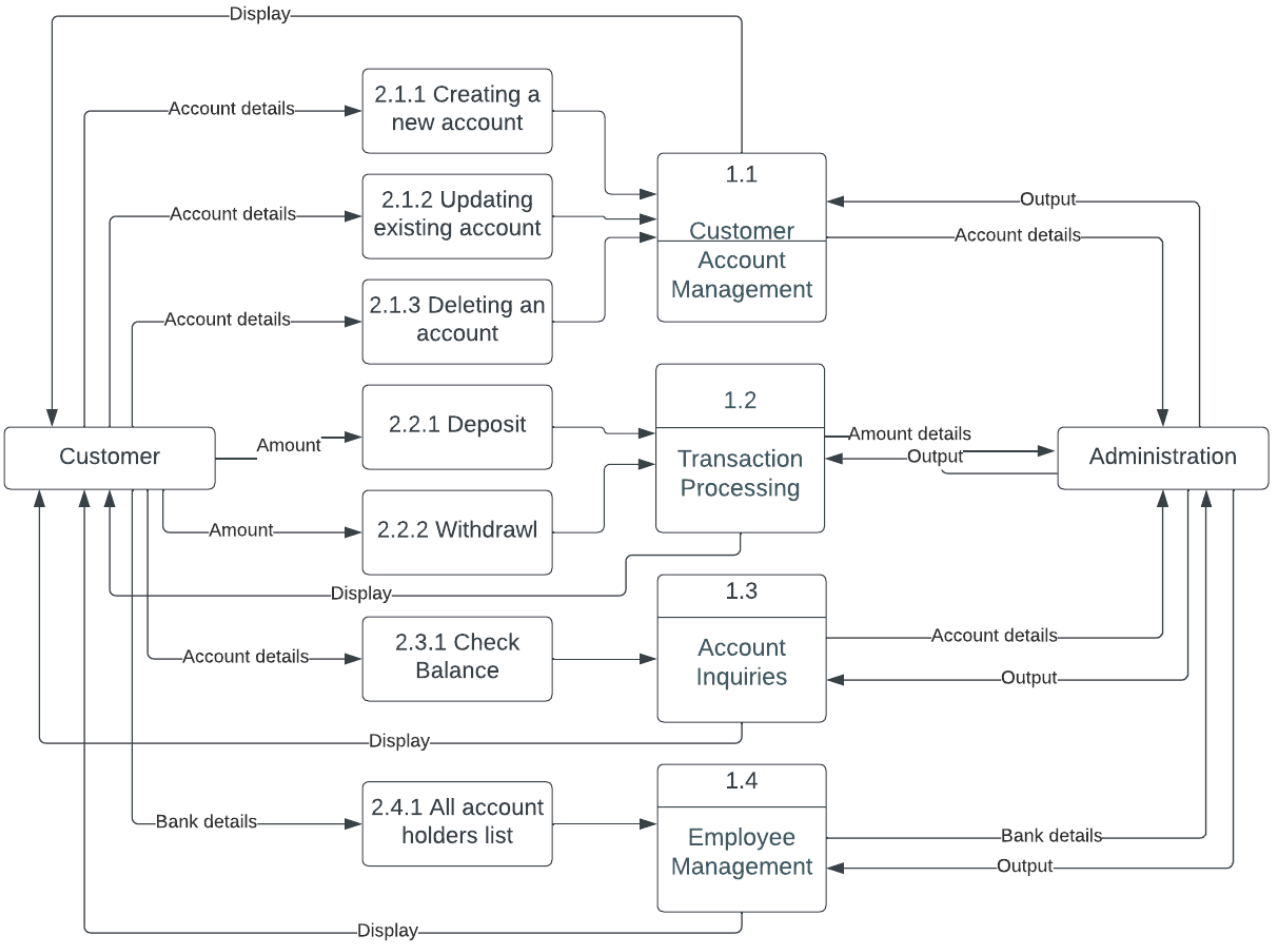
3.1.2. Level 1 DFD for Bank Management System

The Level 1 DFD delves deeper into the system, breaking down the processes identified in the Level 0 DFD into sub-processes.This is to clarify the paths (flow) of data and its transformation from input to output.



3.1.3. Level 2 DFD for Bank Management System

At this stage, a more intricate perspective of the system is presented, achieved by dividing the sub-processes that were identified in the level 1 DFD into even smaller sub-processes. On the level 2 DFD, every sub-process is illustrated as an individual process. The diagram also includes the data flows and data stores related to each sub-process.


Display
Account details
2.1.1 Creating a new account
Account details
2.1.2 Updating existing account
Account details
2.1.3 Deleting an account
1.1
Customer Account Management
Output
Account details
Customer
Amount
2.2.1 Deposit
Amount
2.2.2 Withdrawl
1.2
Transaction Processing
Amount details
Output
Administration
Display
Account details
2.3.1 Check Balance
1.3
Account Inquiries
Account details
Output
Display
Bank details
2.4.1 All account holders list
1.4
Employee Management
Bank details
Output
Display

3.2. FLOWCHART

A flowchart serves as a visual representation of the systematic flow of processes within a Bank Management System project. It encapsulates the logic and sequence of activities, illustrating how different components interact to achieve specific functionalities. In the context of this project, the flowchart visually articulates the journey from user authentication to account management, transaction processing, and various customer interactions. By providing a bird's-eye view of the project's structural design, the flowchart becomes a valuable tool for comprehending the intricate flow of operations, facilitating effective communication, and aiding in the seamless implementation of the Bank Management System.

The various symbols used and their significance is given below:

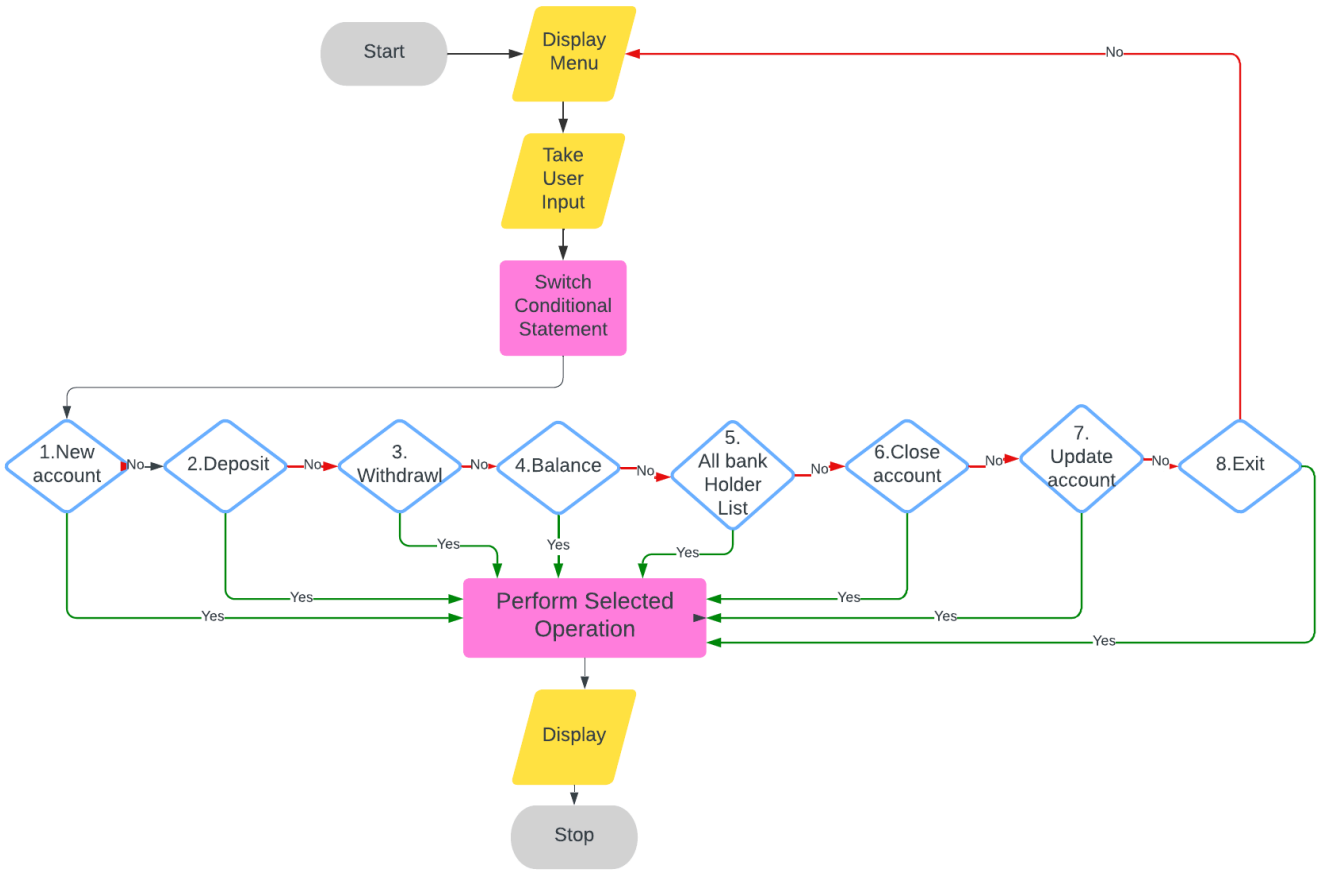
Terminal Symbol:
Oval shape is used to represent the start and end of the flowchart

Process Symbol:
Rectangles are used to represent processes or actions.
In the context of a switch case, rectangles may contain the actions or tasks associated with each case.

Decision Symbol:
A decision symbol is used when the flow of the program depends on a condition. In the case of a switch statement, it represents the evaluation of the variable or expression.

Case Symbols:
Each case inside the switch is represented by a rectangle containing the specific value or condition associated with that case.A default case, which is optional, can be represented by a rectangle labeled "default."

Arrow Line:
Arrows indicate the flow of control from one step to the next.



3.3. FUNCTIONALITIES USED

The C++ code for the Bank Management System encompasses a set of essential functions to manage customer accounts effectively.

- The `create_account()` function facilitates the creation of a new customer account, collecting necessary details such as account holder name, account number, type (savings or current), and initial balance.
- The `show_account()` function allows users to view account details, providing transparency in their financial transactions.
- The `modify()` function permits users to update account information, ensuring the system remains current and accurate.
- Deposit and withdrawal functionalities are handled by the `dep(int)` and `draw(int)` functions, respectively, adjusting the account balance accordingly.
- The `report()` function generates a report summarizing account details.
- Additionally, various auxiliary functions like `write_account()`, `display_sp(int)`, `modify_account(int)`, `delete_account(int)`, `display_all()`, and `deposit_withdraw(int, int)` complement these primary account management functions.
- The `intro()` function serves as an introductory display for the Bank Management System. Together, these functions form a cohesive system for efficient and user-friendly banking operations.

4. Concluding Remarks

In the culmination of this Bank Management System project, we have achieved a significant milestone in developing a comprehensive and efficient C++ codebase for managing customer accounts within a banking framework. The implementation successfully addresses the fundamental requirements of user authentication, account creation, modification, and the execution of secure financial transactions. The object-oriented design principles employed in structuring the system contribute to its modularity, making it adaptable and extensible for future enhancements.

The dynamic array for storing accounts proves pivotal in handling varying account loads, ensuring scalability and optimal memory management. The system's functionalities, including deposit, withdrawal, and account reporting, adhere to industry standards and user expectations. The code's clarity and organization, influenced by foundational programming principles, promote maintainability and ease of understanding for future developers.

5. Future Work

As we reflect on the achievements of the current implementation, the roadmap for future improvements is equally exciting. Strengthening security measures through the incorporation of advanced authentication methods and encryption protocols is imperative to fortify the system against evolving cybersecurity threats. Introducing real-time notifications and alerts can enhance user engagement and financial awareness, fostering a more transparent and responsive banking experience.

Exploration of cloud-based solutions for data storage and backup can contribute to the system's robustness, ensuring data resilience and accessibility. The integration of advanced data analytics techniques may provide valuable insights into customer behaviors, facilitating personalized services and aiding in fraud detection. Continuous testing, debugging, and adherence to evolving industry standards will remain integral to maintaining the system's integrity.

In essence, the future trajectory involves a commitment to innovation, security, and adaptability, positioning the Bank Management System as a dynamic and reliable solution amidst the ever-changing landscape of banking technologies and customer expectations.

References

1. Stroustrup, B. (2014). "Programming: Principles and Practice Using C++." Addison-Wesley.

2. Josuttis, N. M. (2012). "The C++ Standard Library: A Tutorial and Reference." Addison-Wesley.

3. Eckel, B. (2000). "Thinking in C++." Prentice Hall.

4. Balagurusamy, E. (2008). "Object-Oriented Programming with C++." McGraw-Hill Education.

5. Meyers, S. (2014). "Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14." O'Reilly Media.

These references encompass foundational C++ programming principles, object-oriented design, and specific insights into the C++ standard library. They have been instrumental in shaping the coding practices, design decisions, and overall approach to implementing the Bank Management System project.