**BANKING MANAGEMENT SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

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

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

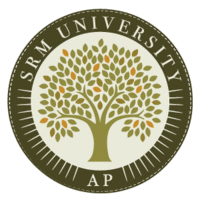
Submitted by

Kommineni Surya – AP22110010933

Jagadeeswar Chirmirala – AP22110010946

Ganesh Botla – AP22110010947

Mokshanag Kuppam – AP22110010980



**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[NOVEMBER, 2023]**

**Acknowledgements**

* **Menu-Driven Interface**: The use of a menu-driven interface makes the program user-friendly and allows users to choose the operation they want to perform easily.
* **Object-Oriented Approach**: The code utilizes a class (BMS) to represent account details, which is a good practice for encapsulating related data and operations.
* **Input Validation**: The code includes some input validation, such as checking for valid account types and ensuring that deposit and loan amounts meet certain criteria.
* **Loan Interest Calculation**: The code calculates the interest amount for a loan based on a fixed interest rate, providing additional functionality for managing loans.
* **Error Handling**: The code includes error messages to guide users in case of invalid input or other issues.
* **Serial Number Usage**: The use of a serial number for each account and mobile number verification enhances security and provides a unique identifier for each account.

4

**Table of Contents**

Certificate..................................................................................................................................

Acknowledgements...............................................................................................................

Table of Contents.....................................................................................................................

Abstract..................................................................................................................................

Abbreviations........................................................................................................................

List of Tables........................................................................................................................

List of Figures....................................................................................................................... List of Equations...................................................................................................................

1. Introduction.....................................................................................................................
   1. Heading 2...............................................................................................................................
      1. Heading 3.......................................................................................................................
2. Methodology....................................................................................................................
   1. Heading 2...............................................................................................................................
      1. Heading 3.......................................................................................................................
3. Discussion........................................................................................................................
4. Concluding Remarks......................................................................................................
5. Future Work.....................................................................................................................

References...............................................................................................................................

**Abstract**

**Account Management**:

Users can create accounts by providing essential details such as name, mobile number, and account type (Savings or Current).

Each account is assigned a unique serial number, and mobile number verification is implemented for added security.

**Transaction Operations:**

Users can deposit funds into their accounts, with specific minimum deposit requirements based on the account type.

Loan requests can be submitted, with automatic calculation of interest based on a predefined interest rate.

Loans can be repaid partially or in full, with excess amounts contributing to the account balance.

**Account Information:**

Users have the option to display and update their account details, including name and mobile number.

**Money Transfer:**

The system supports transferring funds between accounts, ensuring that the sender has sufficient funds for the transfer.

**User-Friendly Interface:**

The program features a menu-driven interface that guides users through available operations, enhancing overall user experience.

**Abbreviations**

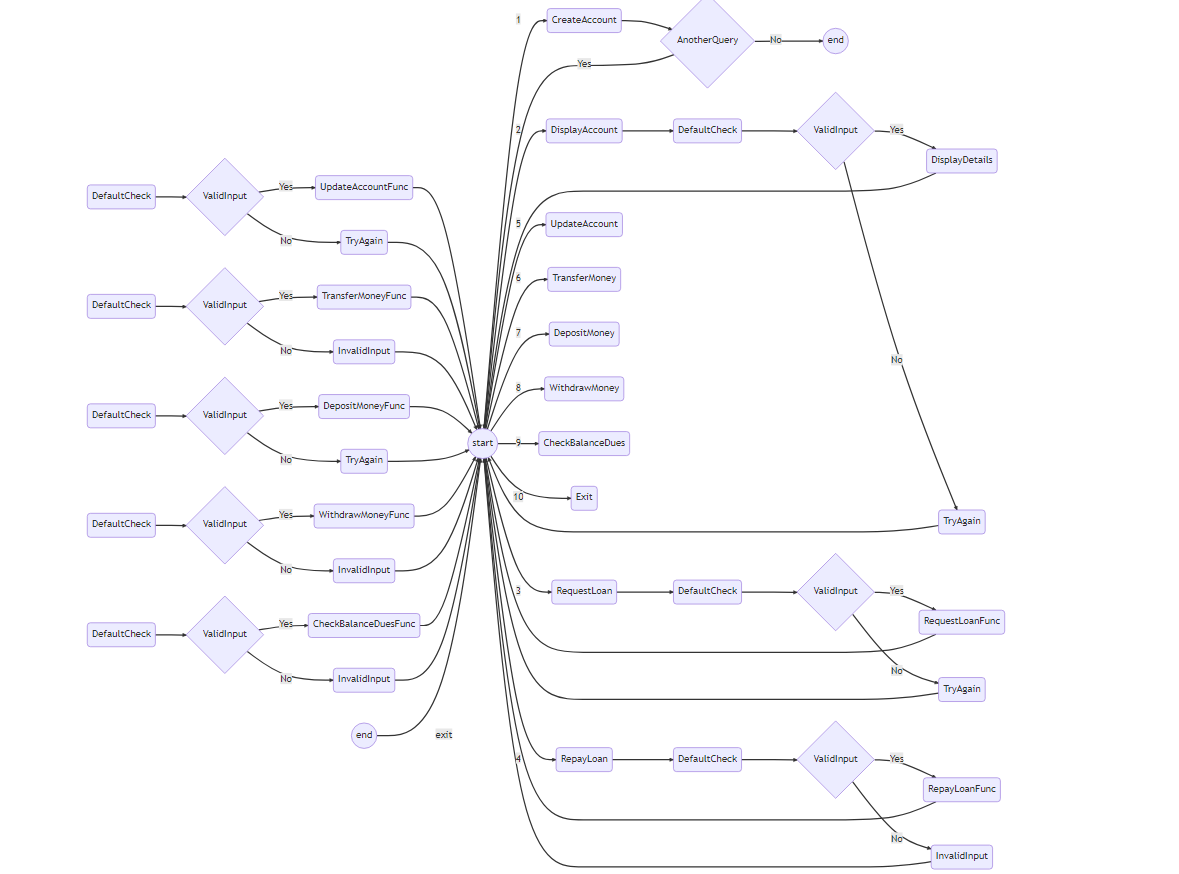
* **BMS:** Banking Management System
* **i**: Index (used as a loop counter)
* **mbno** : Mobile Number
* **accno** : Account Number
* **type**: Account Type (Savings or Current)
* **amount**: Account Balance
* **sno**: Serial Number
* **obj:** Object (referring to instances of the BMS class)
* **ch:** Choice (user input for menu selection)
* **k:** Indicator (used for various checks)
* **ans:** Answer (user input for yes/no questions)
* **r:** Result (return value from functions)
* **temp:** Temporary (used for storing temporary values)
* **dep:** Deposit
* **with**: Withdraw
* **upd:** Update
* **tr:** Transfer
* **disp:** Display
* **reqLoan:** Request Loan
* **repayLoan:** Repay Loan
* **defaultCheck:** Default Check
* **addAcc:** Add Account
* **checkAcc:** Check Account

**Tables**

Table 1 …………………………..…………………………..10

|  |  |  |
| --- | --- | --- |
| **FUNCTION** | **DESCRIPTION** | **USER INTERACTION** |
| ADD ACCOUNT | Add a new account with user-provided details | Account creation  process. |
| DISPLAY | Display account details. | View account  information. |
| REQUEST LOAN | Request a loan with user-specified amount. | Apply for a loan. |
| REPAY LOAN | Repay a portion or the full amount of the loan. | Repay a loan installment. |
| UPDATE ACCOUNT | Update account details such as name and mobile number. | Modify account  Information |
| TRANSFER MONEY | Transfer money between two accounts. | Send money to another account. |
| DEPOSIT | Deposit funds into the account. | Add money to the account. |
| WITHDRAW | Withdraw funds from the account. | Take money out of the account. |
| CHECK | Validate account information for various operations. | Verify account details. |
| DEFAULT CHECK | Perform a default check for serial number and mobile. | Verify account  through defaults |

**Figures**

Figure 1. Text for an example figure……………………………...……………………….6

**List Of Equations**

Equation 1. Text of equation …………………………………………………………...…15

**Interest Calculation for Loan:**

* Interest Amount=Loan Amount × Interest Rate
* Interest Amount=Loan Amount × Interest Rate

**Total Loan Amount after Interest**:

* Total Loan Amount=Loan Amount+ Interest Amount
* Total Loan Amount=Loan Amount +Interest Amount

**Account Balance Update after Deposit:**

* For Savings Account: New Balance=Old Balance +Deposit Amount
* New Balance=Old Balance +Deposit Amount

**For Current Account:**

* New Balance=Old Balance+ New Balance

**Deposit Amount:**

* New Balance = Old Balance + Deposit Amount

**Transfer Money between Accounts:**

* Sender’s New Balance = Sender’s Old Balance − Transfer Amount
* Sender’s New Balance = Sender’s Old Balance − Transfer Amount
* Receiver’s New Balance = Receiver’s Old Balance + Transfer Amount
* Receiver’s New Balance=Receiver’s Old Balance+ Transfer Amount

**Loan Repayment and Account Update:**

**If the repayment amount exceeds the loan amount:**

* Excess Amount=Repayment Amount−Loan Amount
* Excess Amount=Repayment Amount− Loan Amount
* New Loan Amount=0
* New Loan Amount=0 (Loan fully repaid)
* New Account Balance=Excess Amount+ Old Account Balance
* New Account Balance=Excess Amount+ Old Account Balance

**If the repayment amount is less than the loan amount:**

* New Loan Amount=Loan Amount−Repayment Amount
* New Loan Amount=Loan Amount−Repayment Amount

**1. Introduction**

The Simple Banking Management System (BMS) implemented in C++ serves as a foundational platform for managing basic banking operations in the rapidly evolving landscape of digital financial management. This introduction sets the stage for understanding the purpose and structure of the code, highlighting its role in providing a user-friendly and secure interface for individuals seeking efficient financial management solutions.

In recent years, digitalization has reshaped the way individuals interact with financial systems, emphasizing the need for accessible and secure tools. The Simple BMS code addresses this demand by offering a menu-driven interface, allowing users to perform key banking operations seamlessly. From creating accounts to managing loans, the system aims to cater to a diverse user base.

**1.1 Overview of Simple BMS**

The Simple BMS code's primary feature lies in its menu-driven interface, a crucial element in ensuring user accessibility. This subsection delves into the user-friendly design and the range of functionalities available. The menu-driven approach enables users to navigate through various options easily, promoting a hassle-free experience. By providing a straightforward and intuitive interface, the Simple BMS positions itself as an accessible tool for individuals seeking uncomplicated financial management solutions.

**1.1.1 Significance of Object-Oriented Approach**

The adoption of an object-oriented approach in the Simple BMS code is pivotal for maintaining code organization and promoting reusability. This subheading explores how the use of classes encapsulates account details, resulting in a modular and scalable structure. Object-oriented programming principles enhance the code's readability and maintainability, allowing for smoother integration of new features and modifications. The significance of this approach lies in its ability to facilitate future development and adaptability to changing requirements in the dynamic field of banking.

**2. Methodology**

The methodology section provides insights into the structured approach taken in the development of the Simple BMS code. This includes the organization of code, the implementation of key functionalities, and the establishment of a user-friendly interface.

**2.1 Code Organization and Structure**

A cornerstone of the code's methodology is its organization. Through the use of classes and functions, the code achieves a modular structure that enhances maintainability and readability. This subsection explains the rationale behind the chosen organization and its implications for future development. The systematic arrangement of code components ensures clarity and ease of comprehension, laying the groundwork for future enhancements.

**2.1.1 Implementation of Key Functionalities**

The heart of the Simple BMS lies in its functionalities. From account creation to loan management, each operation is meticulously implemented to ensure accuracy and security. This subsection delves into the coding strategies behind key functionalities, shedding light on the decision-making processes involved. By understanding the intricacies of each functionality, developers and users alike gain insights into the robustness of the system.

**3. Discussion**

The discussion section critically examines the Simple BMS code, providing an in-depth analysis of its strengths and areas that may benefit from improvement. Key considerations include input validation, error handling, and the overall user experience. By dissecting these aspects, the discussion aims to provide a comprehensive evaluation of the code's performance, guiding future development and refinement efforts.

The Simple BMS code exhibits strengths in its user-friendly design and the implementation of core banking functionalities. The menu-driven interface enhances usability, allowing users to navigate through various options intuitively. Moreover, the use of an object-oriented approach contributes to the code's modularity and readability, providing a solid foundation for future expansion and modification.

However, certain areas warrant attention for improvement. The code's input validation mechanisms, while present, could be further enhanced to handle a wider range of scenarios and edge cases. Error handling, especially in cases of invalid user input, requires additional refinement to ensure a seamless and error-free user experience. These considerations serve as valuable insights for developers seeking to optimize the Simple BMS code.

**4. Concluding Remarks**

In conclusion, the Simple BMS code emerges as a valuable tool for basic banking operations, demonstrating its potential for streamlined financial management. The user-friendly design, coupled with a secure and structured coding approach, positions it as a promising solution in the digital banking landscape. The code's modular organization and adherence to object-oriented principles contribute to its adaptability and ease of maintenance.

The Simple BMS has the capacity to evolve further, incorporating refinements based on the discussion's insights. Addressing areas for improvement, such as input validation and error handling, will enhance the overall reliability and robustness of the system. As digital financial management continues to evolve, the Simple BMS stands poised for further development and adaptation to meet the dynamic needs of users.

**5. Future Work**

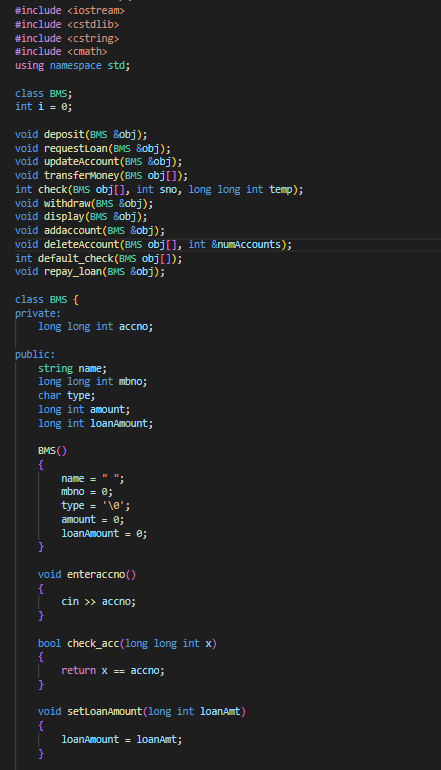
The future work section outlines potential areas for enhancement and development within the Simple BMS code. This roadmap serves as a guide for advancing the code to meet evolving banking system requirements.

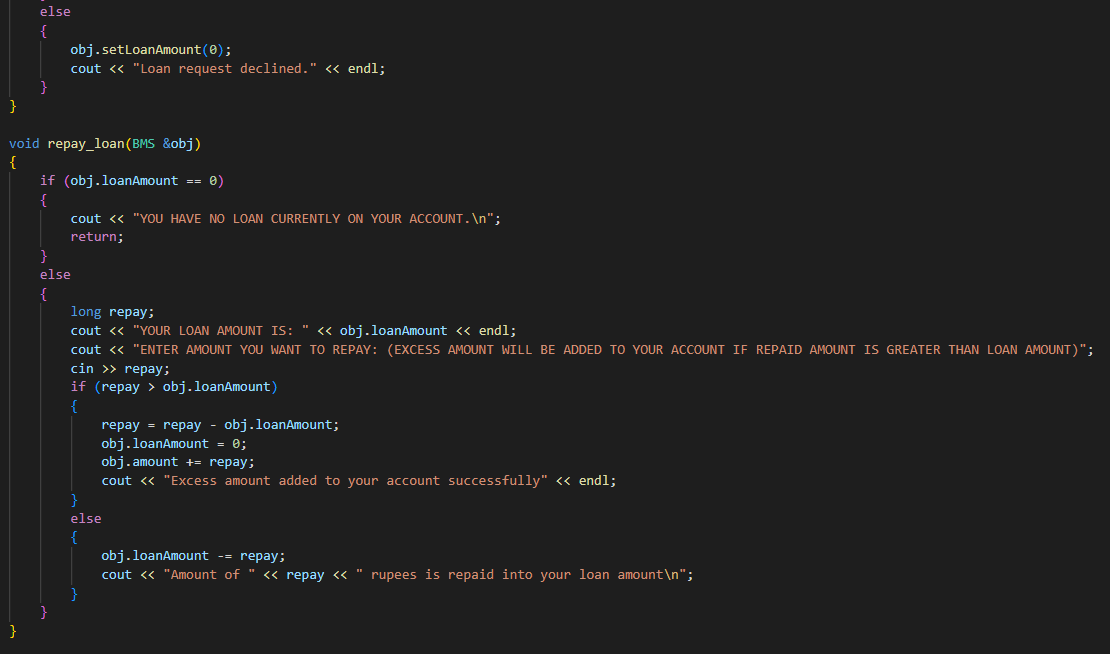
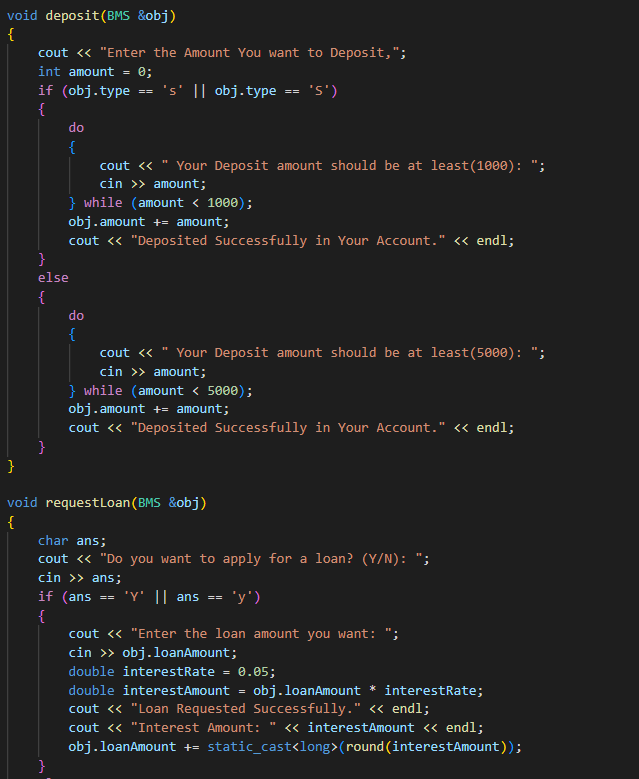
Looking ahead, a key area of focus is code optimization. Refactoring the code to enhance modularity and readability further can improve its maintainability and ease of future development. The introduction of data persistence mechanisms, such as integration with a database or file storage system, is crucial for retaining user data between program executions. This step enhances the user experience by providing continuity in financial records.

Security considerations also warrant attention in future development. Integrating advanced security measures, including password protection and encryption, enhances the overall security posture of the Simple BMS. This is particularly important in a banking system where the safeguarding of user data is paramount.

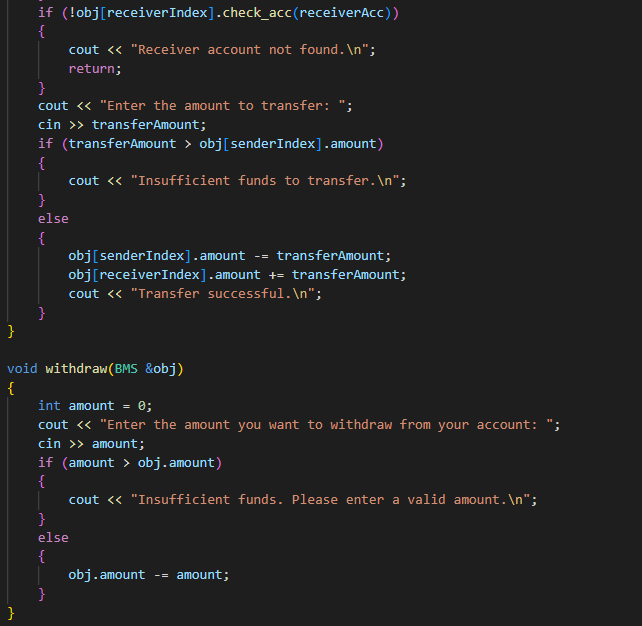
User authentication mechanisms represent another avenue for future work. Implementing robust authentication processes ensures secure access to account-related operations, adding an extra layer of protection to user accounts. Authentication measures could include multi-factor authentication or biometric verification, aligning the Simple BMS with contemporary security standards.

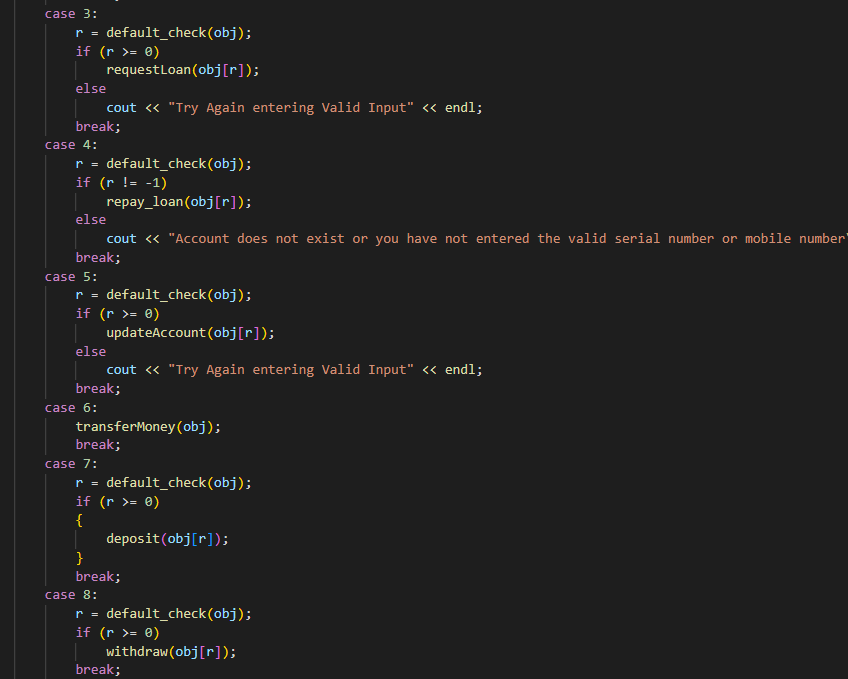
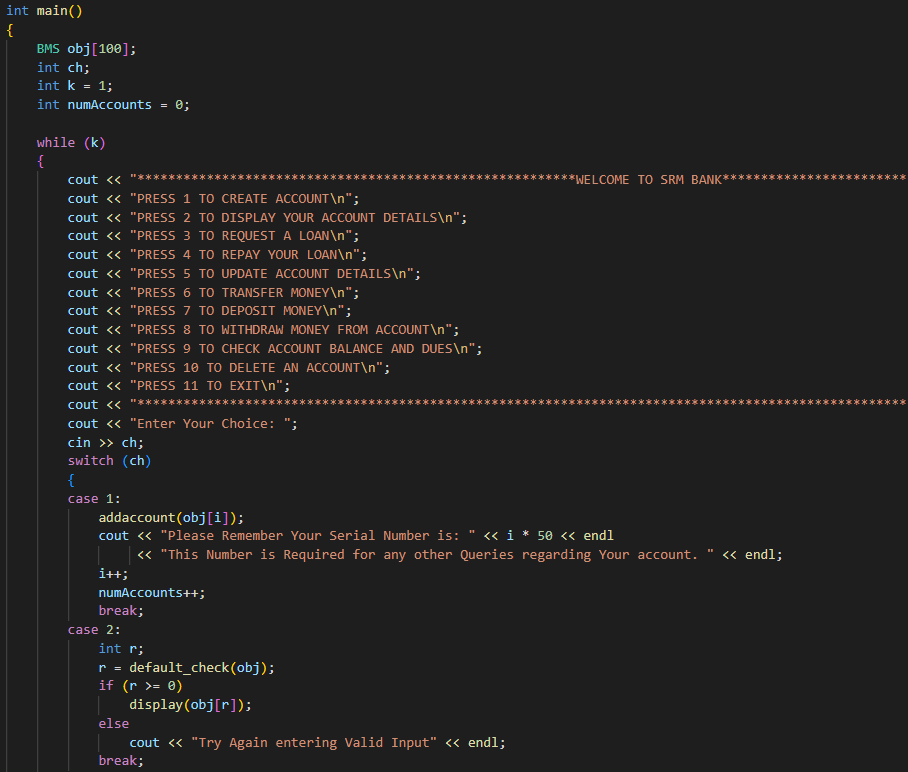
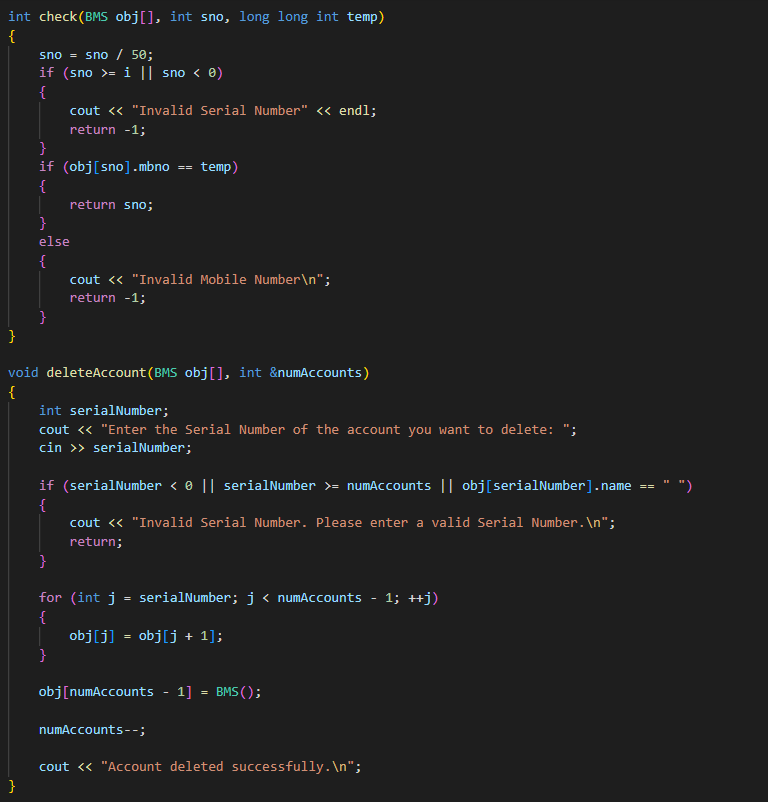
In conclusion, the future work section provides a roadmap for developers seeking to enhance the Simple BMS code. By addressing areas such as code optimization, data persistence, security measures, and user authentication, the code can evolve into a more sophisticated and secure banking management system, meeting the dynamic needs of users in an ever-changing digital landscape.

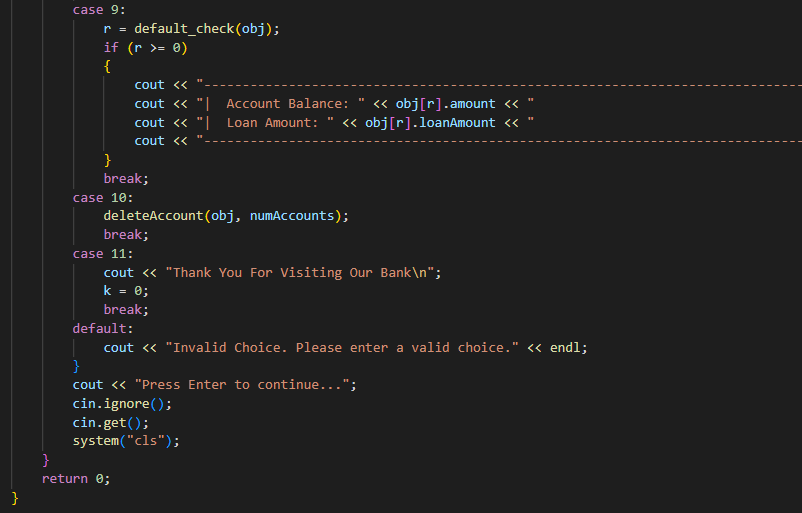






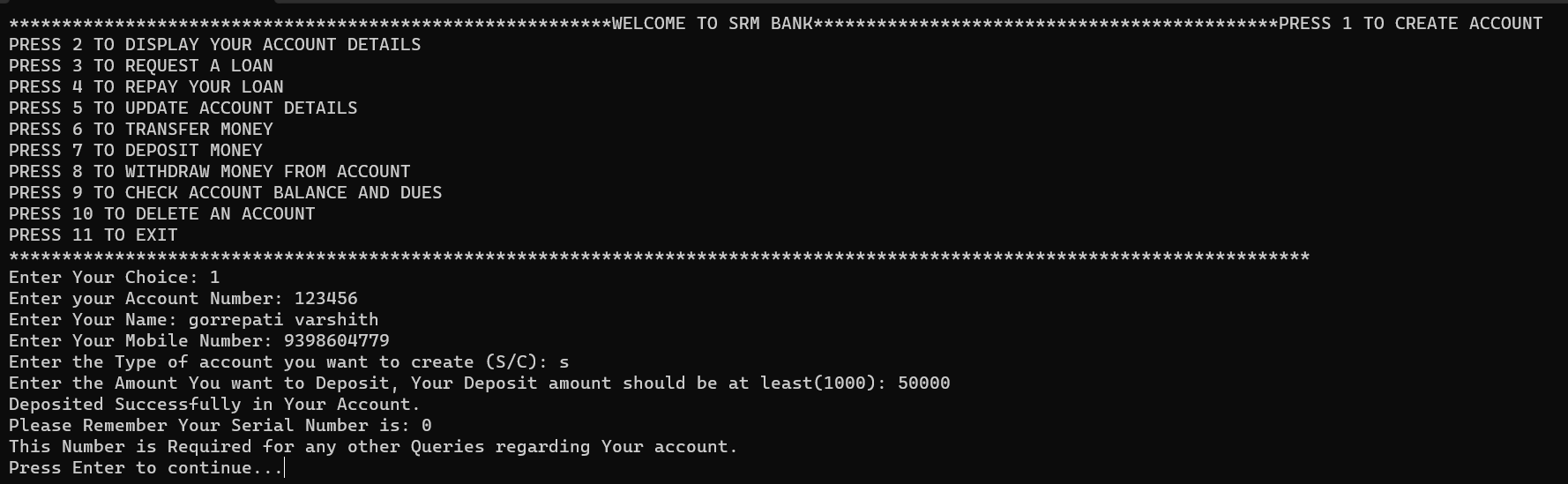


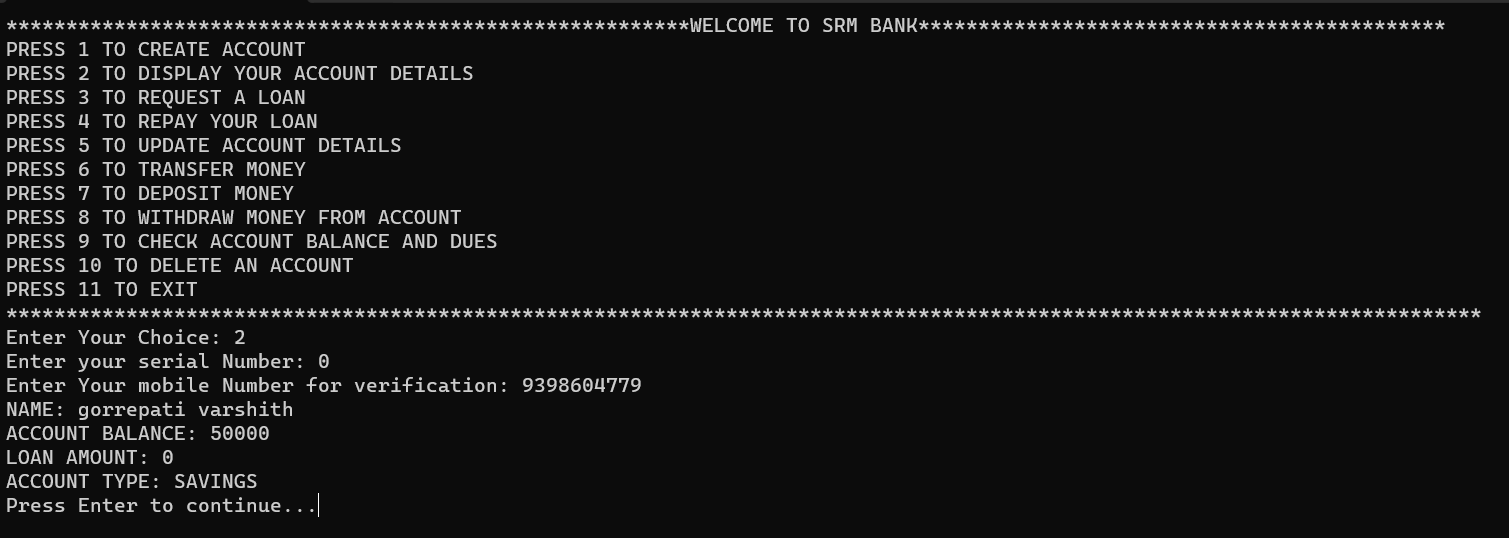
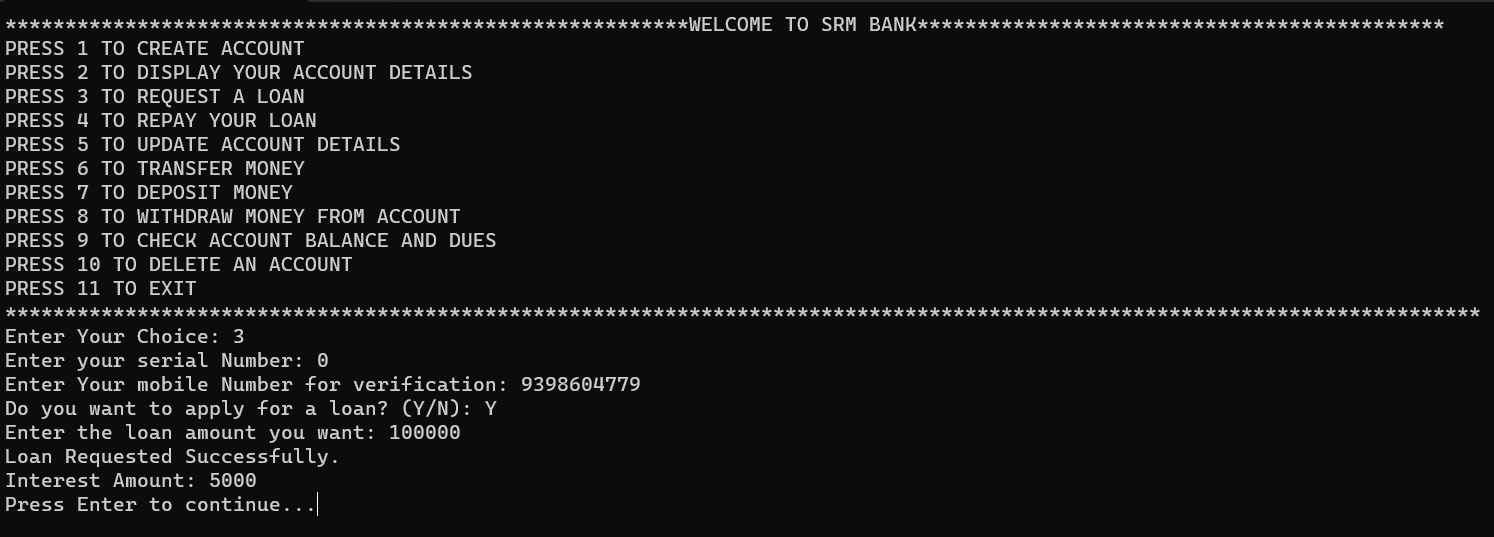
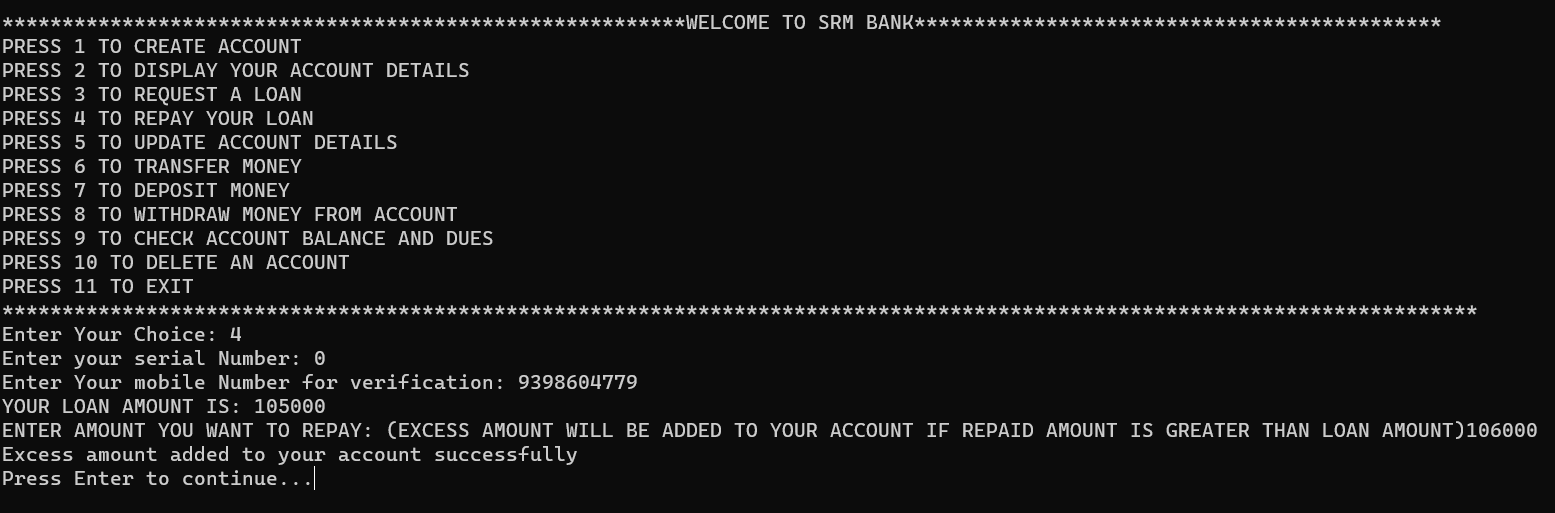




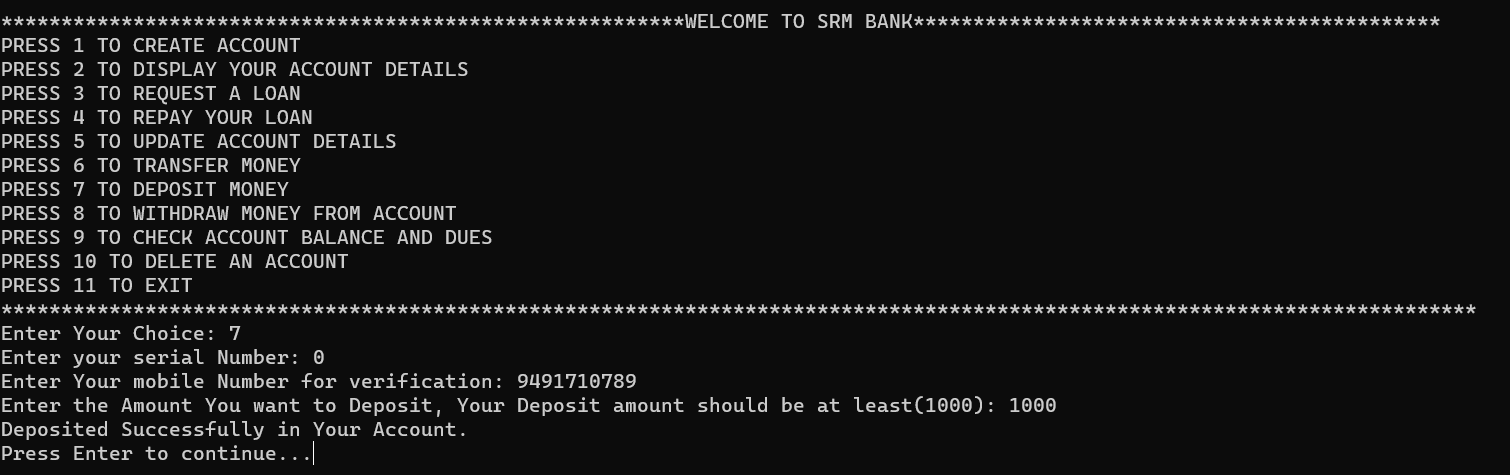
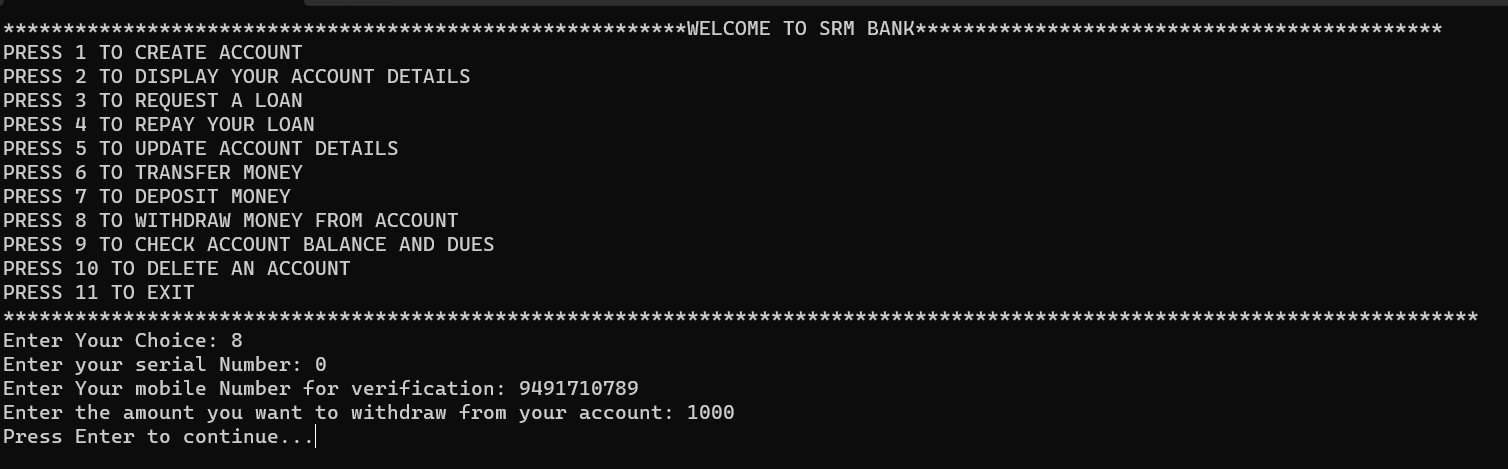
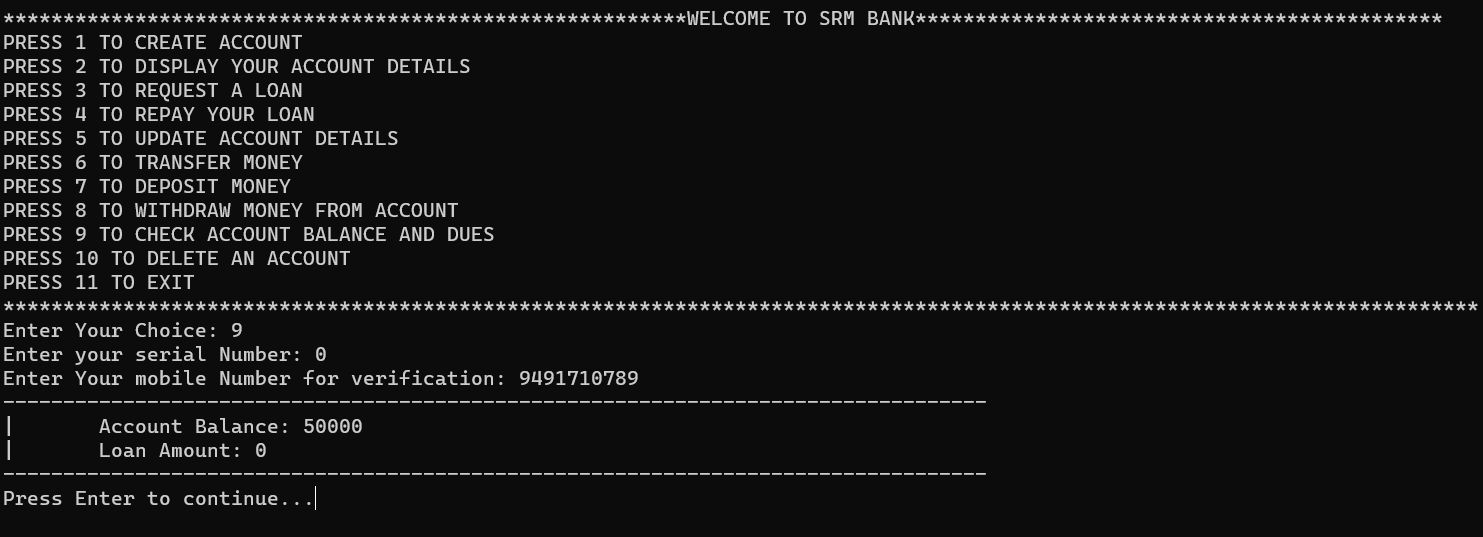
**OUTPUTS**

* **CASE 1:**



* **CASE 2:**
* **CASE 3:**
* **CASE 4:**
* **CASE 5:**

# CASE 6:

* **CASE 7:**
* **CASE 8:**
* **CASE 9:**

# CASE 10:

# CASE 11: