C. V.RAMAN GLOBAL UNIVERSITY, BHUBANESWAR

Department of Computer Science and Engineering

3rd Year   Group - **28**

PROJECT REPORT

TOPIC: Comprehensive Banking Operations Management System

Submitted By : **Khusi Agarwal**   Reg no: **2201020495**

Guided By : **Mohammad Sikander**

C.V. RAMAN GLOBAL UNIVERSITY

Mahura, Bhubaneswar Odisha -752054

<u>Declaration</u>

Project Declaration

I, **Khusi Agarwal**, hereby declare my intent to undertake a software development project focused on building a **Bank Management System** using **C++** for core logic and **Firebase** as the database backend.

Project Overview

The objective of this project is to develop a **simple CLI-based** Bank Management System that efficiently handles basic banking operations such as account creation, transactions, balance inquiries, and authentication. The implementation will integrate:

Firebase for secure cloud-based data storage

OpenSSL for encryption and secure transactions

cURL for API communication with Firebase

nlohmann/json for efficient JSON data handling

Scope of Work

The system will include functionalities such as:

User registration and authentication

Deposits, withdrawals, and balance inquiries

Transaction logging and history retrieval

Secure data transmission using encryption

By signing this declaration, I commit to carrying out the project as outlined above and delivering a functional prototype demonstrating efficient banking operations with Firebase integration.

Signed by:
Khusi Agarwal

Certificate from Supervisor

This is to certify that Khusi Agarwal, enrolled in the Bachelor of Technology (B. Tech) program at C. V. Raman Global University, has successfully completed the project titled "Bank Management System using C++, Firebase, and OpenSSL" during the academic year 2024-2025.

The project involved the development of a CLI-based Bank Management System integrating Firebase for secure cloud storage, OpenSSL for encrypted transactions, cURL for API communication, and nlohmann/json for efficient data handling.

Khusi Agarwal’s dedication and effort in designing and implementing a secure and efficient banking system demonstrate commendable technical proficiency.

I hereby endorse that the work presented by Khusi Agarwal is original, authentic, and represents a valuable contribution to the field of banking systems and secure software development. The project highlights the ability to address real-world challenges using innovative solutions based on academic principles and industry-relevant technologies.

Supervisor

Mohammad Sikander
C. V. Raman Global University

ACKNOWLEDGEMENT

The completion of this case study on Bank Management System Using C++ could not have been possible without the participation and assistance of **Mohammad Sikander** for their endless support, kindness, and understanding during the project duration.

I would also like to extend my appreciation to my peers and mentors for their continuous motivation and constructive suggestions, which have helped refine my understanding of the subject.

THANK YOU!

Index

Introduction to C++

The CppBank project represents a console-based banking application that demonstrates practical implementation of C++ programming for financial systems. This application integrates Firebase cloud storage for data persistence while providing core banking functionalities through an organized command-line interface. The project successfully balances simplicity with essential banking features, making it suitable as both an educational tool and a foundation for more advanced banking systems.

Key Features of C++:

Encapsulation and Abstraction: These features help in securing sensitive data by restricting direct access and exposing only necessary functionalities.

Inheritance and Polymorphism: They allow for code reuse and flexibility, reducing redundancy and improving efficiency.

Memory Management: C++ provides control over memory allocation and deallocation using pointers, ensuring efficient resource utilization.

High Performance: The compiled nature of C++ ensures faster execution compared to many other high-level languages.

This banking management system project leverages C++ due to its structured programming approach and efficient data handling capabilities.

Overview of the Banking Management System

The CppBank project is a console-based banking management system designed to handle fundamental banking operations. It allows users to create accounts, deposit and withdraw money, and check their balance. The project utilizes object-oriented programming principles to maintain a modular and scalable structure.

Objectives of the System:

Develop a reliable and secure banking management tool.

Provide an intuitive and simple interface for banking transactions.

Implement structured logic for efficient processing of transactions.

Store user account details securely using Firebase as a cloud database.

System Workflow:

User Registration: New users create an account by providing necessary details such as an account number and password.

Authentication: Users are required to verify their credentials before accessing account details.

Banking Transactions: Users can perform deposits, withdrawals, and balance inquiries.

Data Storage: All transactions and account details are securely stored in Firebase, ensuring real-time data retrieval.

Features of the System

Core Features:

Encapsulation: The BankAccount class is used to store and manage user account details securely.

Comprehensive Banking Operations: Users can create accounts, deposit and withdraw money, and check their account balance.

Structured Menu Navigation: A switch-case-based menu allows users to select and perform operations efficiently.

Cloud Storage Integration: Firebase is utilized for secure data storage and retrieval, ensuring seamless access across devices.

Functional Workflow:

Operation Phase	Key Functions	Technical Details
System Entry	displayIntroAnimation(), main()	Initialization and main menu presentation
Account Creation	createAccount(), sendToFirebase()	Data validation and Firebase document creation
Authentication	authenticate(), getFromFirebase()	Credential verification with retry limitations
Transaction Processing	depositMoney(), withdrawMoney()	Balance manipulation with constraint validation
Account Management	changePassword(), deleteAccount()	Security and lifecycle management

---

Implementation - Source Code

```
#include <iostream>
#include <cstdlib>
#include <curl/curl.h>
#include <nlohmann/json.hpp>
#include <chrono>
#include <thread>
#include <vector>

using namespace std;
using json = nlohmann::json;

const string FIREBASE_URL = "https://bank-9c0da-default-rtdb.firebaseio.com/";
const string JSON_EXT = ".json";
const double MAX_TRANSACTION_AMOUNT = 100000.0; // Maximum
deposit/withdrawal limit
const double MIN_BALANCE = 100.0; // Minimum balance required
const int MAX_LOGIN_ATTEMPTS = 3; // Maximum password attempts

size_t callback(void* contents, size_t size, size_t nmemb, void* userp) {
   ((string*)userp)->append((char*)contents, size * nmemb);
   return size * nmemb;
}

void loadingEffect(const string& message) {
   cout << "\033[1;33m" << message;
   for (int i = 0; i < 3; ++i) {
      cout << ".";
      cout.flush();
      this_thread::sleep_for(chrono::milliseconds(500));
   }
   cout << "\033[0m" << endl;
}

string getFromFirebase(const string& path) {
   loadingEffect("Fetching data");
   CURL* curl = curl_easy_init();
   string response;
   if (curl) {
      string url = FIREBASE_URL + path + JSON_EXT;
      curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
      curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, callback);
      curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);
      CURLcode res = curl_easy_perform(curl);
      if (res != CURLE_OK) {
            cerr << "\033[1;31mError: " << curl_easy_strerror(res) << "\033[0m" <<
endl;
        }
        curl_easy_cleanup(curl);
    }
    return response;
}

void sendToFirebase(const string& path, const string& data, const string& method)
{
    loadingEffect("Updating data");
    CURL* curl = curl_easy_init();
    if (curl) {
        string url = FIREBASE_URL + path + JSON_EXT;
        curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
        curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, method.c_str());
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data.c_str());
        curl_easy_perform(curl);
        curl_easy_cleanup(curl);
    }
}

void displayIntroAnimation() {
    cout << "\033[1;36m";
    cout << "==============================================\n";
    cout << "|                                            |\n";
    cout << "|          WELCOME TO CPP BANK               |\n";
    cout << "|                                            |\n";
    cout << "==============================================\n";
    cout << "\033[0m";
    this_thread::sleep_for(chrono::milliseconds(1000));
    cout << "\033[1;32mLoading your banking experience";
    for (int i = 0; i < 3; ++i) {
        cout << ".";
        cout.flush();
        this_thread::sleep_for(chrono::milliseconds(500));
    }
    cout << "\033[0m" << endl;
}

void createAccount() {
    string accNo, name, password;
    double balance;
    cout << "\033[1;34mEnter Account Number: \033[0m";
    cin >> accNo;
    // Check if account already exists
    string existingData = getFromFirebase("accounts/" + accNo);
    if (!existingData.empty() && existingData != "null") {
        cout << "\033[1;31mAccount already exists!\033[0m\n";
        return;
    }

    cout << "\033[1;34mEnter Name: \033[0m";
    cin >> name;
    cout << "\033[1;34mEnter Password: \033[0m";
    cin >> password;
    do {
        cout << "\033[1;34mEnter Initial Balance (Min: " << MIN_BALANCE << "):
\033[0m";
        cin >> balance;
    } while (balance < MIN_BALANCE);

    string data = "{\"name\": \"" + name + "\", \"password\": \"" + password + "\",
\"balance\": " + to_string(balance) + "}";
    sendToFirebase("accounts/" + accNo, data, "PUT");
    cout << "\033[1;32mAccount Created Successfully!\033[0m\n";
}

bool authenticate(const string& accNo) {
    string storedData = getFromFirebase("accounts/" + accNo);
    if (storedData.empty() || storedData == "null") {
        cout << "\033[1;31mAccount not found!\033[0m\n";
        return false;
    }

    try {
        json accountData = json::parse(storedData);
        string storedPassword = accountData["password"];
        string enteredPassword;

        for (int attempts = 0; attempts < MAX_LOGIN_ATTEMPTS; ++attempts) {
            cout << "\033[1;34mEnter Password: \033[0m";
            cin >> enteredPassword;
            if (enteredPassword == storedPassword) {
                return true;
            }
            cout << "\033[1;31mIncorrect Password! Attempts left: " <<
(MAX_LOGIN_ATTEMPTS - attempts - 1) << "\033[0m\n";
        }
        cout << "\033[1;31mToo many incorrect attempts. Access denied!\033[0m\n";
        return false;
    } catch (json::parse_error& e) {
        cout << "\033[1;31mJSON Parsing Error: \033[0m" << e.what() << endl;
        return false;
    }
}

void changePassword(const string& accNo) {
    string newPassword;
    cout << "\033[1;34mEnter New Password: \033[0m";
    cin >> newPassword;

    string storedData = getFromFirebase("accounts/" + accNo);
    json accountData = json::parse(storedData);
    accountData["password"] = newPassword;

    sendToFirebase("accounts/" + accNo, accountData.dump(), "PUT");
    cout << "\033[1;32mPassword Changed Successfully!\033[0m\n";
}

void viewAccountDetails(const string& accNo) {
    string storedData = getFromFirebase("accounts/" + accNo);
    json accountData = json::parse(storedData);
    cout << "\033[1;34mAccount Number: \033[0m" << accNo << endl;
    cout << "\033[1;34mName: \033[0m" << accountData["name"] << endl;
    cout << "\033[1;34mBalance: \033[0m" << accountData["balance"] << endl;
}

void depositMoney(const string& accNo) {
    double amount;
    cout << "\033[1;34mEnter Amount to Deposit: \033[0m";
    cin >> amount;

    if (amount <= 0 || amount > MAX_TRANSACTION_AMOUNT) {
        cout << "\033[1;31mInvalid Amount!\033[0m\n";
        return;
    }

    string storedData = getFromFirebase("accounts/" + accNo);
    json accountData = json::parse(storedData);
    double currentBalance = accountData["balance"];
    double newBalance = currentBalance + amount;
    accountData["balance"] = newBalance;

    sendToFirebase("accounts/" + accNo, accountData.dump(), "PUT");
    cout << "\033[1;32mAmount Deposited Successfully!\033[0m\n";
}
```

```
void withdrawMoney(const string& accNo) {
   double amount;
   cout << "\033[1;34mEnter Amount to Withdraw: \033[0m";
   cin >> amount;

   if (amount <= 0 || amount > MAX_TRANSACTION_AMOUNT) {
      cout << "\033[1;31mInvalid Amount!\033[0m\n";
      return;
   }

   string storedData = getFromFirebase("accounts/" + accNo);
   json accountData = json::parse(storedData);
   double currentBalance = accountData["balance"];

   if (currentBalance - amount < MIN_BALANCE) {
      cout << "\033[1;31mInsufficient Balance!\033[0m\n";
      return;
   }

   double newBalance = currentBalance - amount;
   accountData["balance"] = newBalance;

   sendToFirebase("accounts/" + accNo, accountData.dump(), "PUT");
   cout << "\033[1;32mAmount Withdrawn Successfully!\033[0m\n";
}

void checkBalance(const string& accNo) {
   string storedData = getFromFirebase("accounts/" + accNo);
   json accountData = json::parse(storedData);
   double currentBalance = accountData["balance"];
   cout << "\033[1;34mCurrent Balance: \033[0m" << currentBalance << "\n";
}

void deleteAccount(const string& accNo) {
   sendToFirebase("accounts/" + accNo, "", "DELETE");
   cout << "\033[1;32mAccount Deleted Successfully!\033[0m\n";
}

void userMenu(const string& accNo) {
   int choice;
   do {
      cout << "\n\033[1;36m===== User Menu =====\033[0m";
      cout << "\n1. View Account Details";
      cout << "\n2. Deposit Money";
      cout << "\n3. Withdraw Money";
```

```
        cout << "\n4. Check Balance";
        cout << "\n5. Change Password";
        cout << "\n6. Delete Account";
        cout << "\n7. Logout";
        cout << "\n\033[1;34mEnter your choice: \033[0m";
        cin >> choice;
        switch (choice) {
            case 1: viewAccountDetails(accNo); break;
            case 2: depositMoney(accNo); break;
            case 3: withdrawMoney(accNo); break;
            case 4: checkBalance(accNo); break;
            case 5: changePassword(accNo); break;
            case 6: deleteAccount(accNo); return; // Logout after deleting account
            case 7: cout << "\033[1;32mLogging out...\033[0m\n"; break;
            default: cout << "\033[1;31mInvalid Choice!\033[0m\n";
        }
    } while (choice != 7);
}

int main() {
    displayIntroAnimation();
    int choice;
    do {
        cout << "\n\033[1;36m===== Cpp Bank Main Menu =====\033[0m";
        cout << "\n1. Create Account";
        cout << "\n2. Login";
        cout << "\n3. Exit";
        cout << "\n\033[1;34mEnter your choice: \033[0m";
        cin >> choice;
        switch (choice) {
            case 1: createAccount(); break;
            case 2: {
                string accNo;
                cout << "\033[1;34mEnter Account Number: \033[0m";
                cin >> accNo;
                if (authenticate(accNo)) {
                    userMenu(accNo);
                }
                break;
            }
            case 3: cout << "\033[1;32mExiting...\033[0m\n"; break;
            default: cout << "\033[1;31mInvalid Choice!\033[0m\n";
        }
    } while (choice != 3);
    return 0;
}
```

Advantages and Disadvantages

Advantages:

The project follows a modular programming approach, ensuring better code organization and maintainability.

The use of object-oriented principles allows for easy future enhancements and modifications.

The system provides a simple yet effective console-based user interface, making it accessible and easy to use.

Firebase integration ensures that user data is stored securely and can be accessed remotely.

Disadvantages:

The system relies on a text-based console interface, which may not be user-friendly for all users.

Passwords are stored in plaintext, posing a security risk that could be mitigated using hashing mechanisms.

The absence of advanced features such as transaction history and fund transfers limits the functionality of the system.

The system is currently designed to run on Linux and may require modifications for compatibility with other operating systems.

Future Enhancements and Conclusion

Potential Improvements:

Develop a graphical user interface (GUI) using frameworks such as Qt or GTK for better user experience.

Implement password hashing techniques such as bcrypt to enhance security.

Add a feature to store and display a transaction history for better record-keeping.

Enable fund transfers between accounts to enhance banking functionality.

Expand platform support to ensure the system runs smoothly on Windows and macOS.

Conclusion:

The banking management system developed in C++ serves as a demonstration of core programming concepts, particularly in the areas of object-oriented programming, structured logic, and database integration. The system effectively allows users to manage their bank accounts through essential operations such as deposits, withdrawals, and balance inquiries. While the current implementation is functional, there is scope for further improvement in terms of security, user interface, and additional banking features. Future updates to the system can help transform it into a more robust and practical banking solution.