

Banking System Simulation

By - Uday Mayank Dhodi

● Introduction to the problem

The problem addressed by this code is the creation of a basic bank management system using C programming. This System aims to facilitate various banking operations such as creating new user accounts, viewing details, editing, deleting, searching and conducting transactions like balance inquiries, cash deposit, and withdrawals.

Additionally the program provides an “Administrator” mode for system management, allowing administrative users to access and control various functionalities.

Note: If you log in as an administrator, the username and password are both ***‘admin’***. However, if you attempt to log in as a regular user, the username and password should be ***‘user’***.

● How the problem is solved

The problem of creating basic bank management system is solved through a systematic approach:

- 1) User Interface
- 2) Administrator and User Mode
- 3) Data Handling with Structures
- 4) Menu - Driven Function
- 5) Data Verification
- 6) Positioning the Cursor
- 7) File Handling
- 8) Exit and About

In summary, the code addresses the problem by providing an organized and interactive Interface for managing customer accounts in a bank. It uses a combination of user Authentication, file handling, structured data storage, and modular functions to create a Basic bank management system.

- **Detailed Methodology with working codes**

1. User Interface

- a. The program started by presenting users with a user-friendly menu-driven interface.
- b. Users can choose between three main options: Administrator login, User login, or New registration.

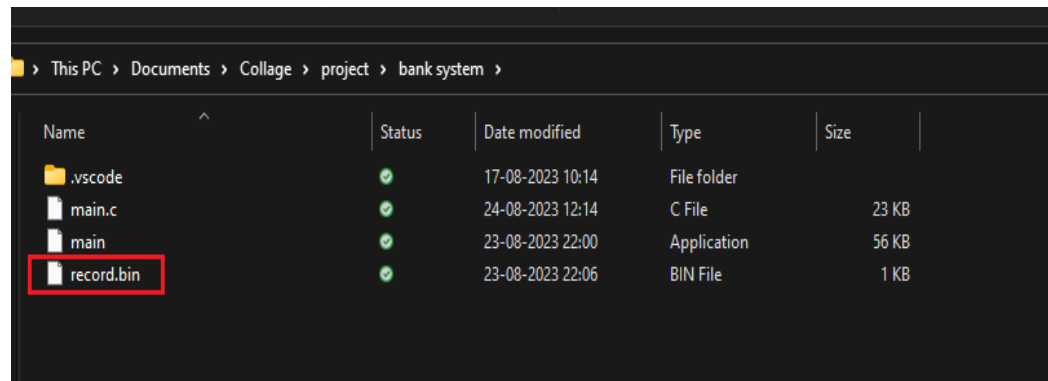
2. Administrator and User Mode

- a. Administrator Login
 - i. If the user selects the administrator login, they are prompted to enter a username and password (hardcoded as ***“admin”*** and ***“admin”*** in this code).
 - ii. Upon successful login, the user gains access to administrative features.
 - iii. Administrative tasks include viewing customer accounts and transactions.
- b. User Login
 - i. If the user selects User login, they are prompted to enter a username and password (hardcoded as ***“user”*** and ***“user”*** in this code).
 - ii. Upon successful login, users can perform account-related operations.
- c. New Registration
 - i. If the user selects a new registration, they can create a new customer account.
 - ii. The code collects essential customer information, such as name, account number, phone, address, email, citizenship number (optional), and initial deposit amount.
 - iii. A unique User ID is generated for each new account.

- iv. The account details are stored in a binary file named ***“record.bin”***.

3. Data Handling with Structures

- a. Customer account information is structured using a ‘record’ structure that includes fields for various details such as name, account number, phone number, address, email citizenship number, balance, and a User ID.
- b. The structure is used to read and write customer data to and from the ***“record.bin”*** binary file.



Name	Status	Date modified	Type	Size
.vscode	✓	17-08-2023 10:14	File folder	
main.c	✓	24-08-2023 12:14	C File	23 KB
main	✓	23-08-2023 22:00	Application	56 KB
record.bin	✓	23-08-2023 22:06	BIN File	1 KB

4. Menu - Driven Function

- a. The code organizes functionalities into modular functions, such as ***‘view()’, ‘add()’, ‘edit()’, ‘del()’, ‘search()’, and ‘transaction()’***.
- b. These functions handle tasks such as viewing account details, adding new accounts, editing account information, deleting accounts, searching for accounts, and processing transactions.

5. Data Verification

- a. The code verifies users credentials(username and password) using the ***‘verify()’*** function before granting access to specific functionalities.

6. Positioning the Cursor

- a. The '**gotoxy()**' function is used to position the cursor on the console window, enhancing the user interface by displaying information at specific locations.

```
void gotoxy(int a, int b) {  
    coord.X = a;  
    coord.Y = b;  
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);  
}
```

7. File Handling

- a. Data is stored and retrieved from the "**record.bin**" binary file using file handling functions like '**fread()**' and '**fwrite()**'.
- b. Data integrity and security are important aspects of the code's file handling operations.

8. Exit and About

- a. Users can exit the program gracefully through the "**Exit**" option, which displays a thank-you message and close the application.
- b. An "**About**" option provides brief information about the project and its creator.

● Outputs and Screenshots of Working Project

I have submitted a screenshot folder containing visual representations of the output generated by my project. These screenshots provide a comprehensive overview of the functioning and user interface of the bank management system implemented in your project.

These visuals will aid in better understanding the practical aspects and user experience of my project's execution.

- **Hardware Requirements**

The code is designed to run on a standard computer. It has no specific hardware requirements beyond what is typically found in a PC or laptop.

- **Software Requirements**

To run the code, I installed a C compiler on the computer. Popular options include GCC (GNU Compiler Collection) for Windows or Linux, or Microsoft Visual Code for Windows. Additionally , the code uses standard C libraries and functions, so no additional software is required.