**User Bank System Management C++**

The project is a Bank Management System implemented in C++. It allows users to perform various banking operations such as creating new accounts, depositing and withdrawing cash, searching for accounts, viewing all accounts, closing accounts, editing account details, and sorting accounts. The system also includes user authentication through login and registration functionality.

|  |  |
| --- | --- |
| **ABOUT PROJECT** | **PROJECT DETAILS** |
| **Project Name :** | User Bank System Managament |
| **Project Platform :** | C++ |
| **Programming Language Used:** | C++ Programming Language |
| **Developer Name :** | Heng Chhorpong |
| **IDE Tool (Recommended):** | Visual Studio Code |
| **Project Type :** | CLI |
| **Data Storage (File or Database):** | Stores data in .DAT file |
| **Created Date:** | 29/06/2024 |
| **Modified Date:** |  |

**Project Details and Technology :**

The project is a Bank Management System implemented in C++. It allows users to perform various banking operations such as creating new accounts, depositing and withdrawing cash, searching for accounts, viewing all accounts, closing accounts, editing account details, and sorting accounts. The system also includes user authentication through login and registration functionality.

The main components of the project are:

1. **Account Class**: Represents a bank account and provides methods for creating, displaying, renaming, depositing, withdrawing, and reporting account details.
2. **AccountUser Functions**: A set of functions that handle file operations for reading, writing, and modifying account records in a binary file.
3. **Application Class**: Manages the user interface and handles user interactions with the Bank Management System.
4. **DataInputValidation Class**: Validates user input for account numbers, balances, and names.
5. **Login and Register Classes**: Handle user authentication and registration processes.
6. **BadInputException Class**: A custom exception class that is thrown when invalid user input is detected.

**Technologies Used:**

1. **Programming Language**: The project is written in C++, a compiled, general-purpose programming language.
2. **File I/O**: The project uses file input/output operations to store and retrieve account data from a binary file named "account.dat".
3. **Exception Handling**: The project implements exception handling using the BadInputException class to handle invalid user input.
4. **Standard C++ Libraries**: The project utilizes various standard C++ libraries, including:

* <iostream>: For input/output operations.
* <fstream>: For file input/output operations.
* <cctype>: For character handling functions (e.g., toupper()).
* <iomanip>: For input/output manipulators (e.g., setw(), setfill()).
* <cstdlib>: For random number generation (rand(), srand()).
* <ctime>: For time-related functions (time()).
* <stdexcept>: For standard exception classes.
* <string>: For string manipulation.

1. **User Interface**: The project provides a command-line user interface (CLI) for user interactions, displaying menus, and prompting for user input.
2. **Random Number Generation**: The project uses the rand() and srand() functions from the <cstdlib> library to generate random 9-digit account numbers.
3. **Data Structures**: The project utilizes arrays (char[]) and strings (string) to store and manipulate data.

Overall, the Bank Management System project is a console-based application developed in C++, utilizing standard C++ libraries and file I/O operations to manage bank accounts and user authentication.

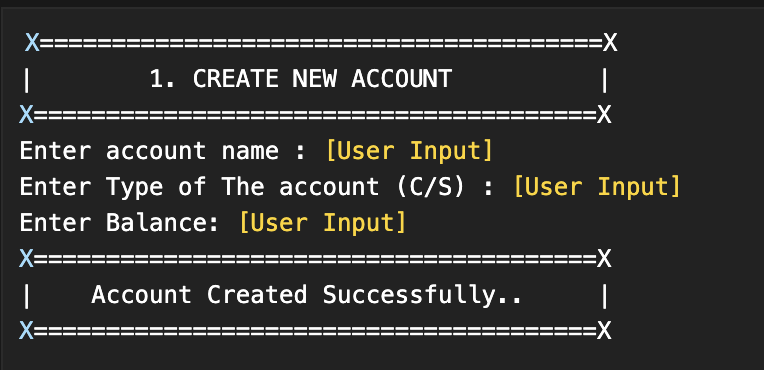
**UML Class Diagram**



**Major Functionalities User Bank System Management**

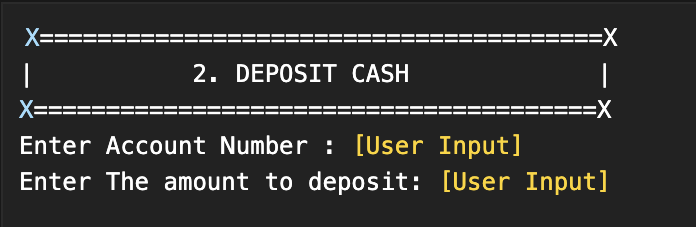
1. **Create New Account**:

* **Proposal**: The create\_account() function in the Account class is responsible for creating a new account. It prompts the user to enter the account name, account type (current or savings), and the initial deposit amount. The function then generates a random 9-digit account number and displays a success message upon account creation.
* **Display :**



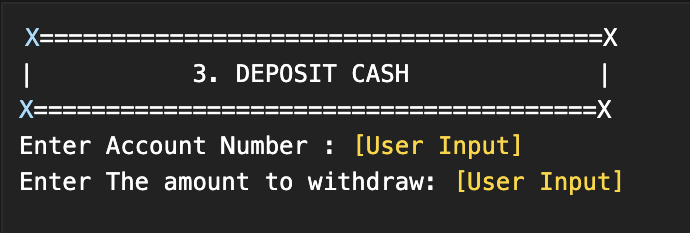
1. **Deposit Cash**:

* **Proposal**: The deposit\_withdraw() function in the AccountUserFunctions namespace handles the deposit operation. It prompts the user to enter the account number and the amount to be deposited. The function then updates the account balance accordingly.
* **Display** :



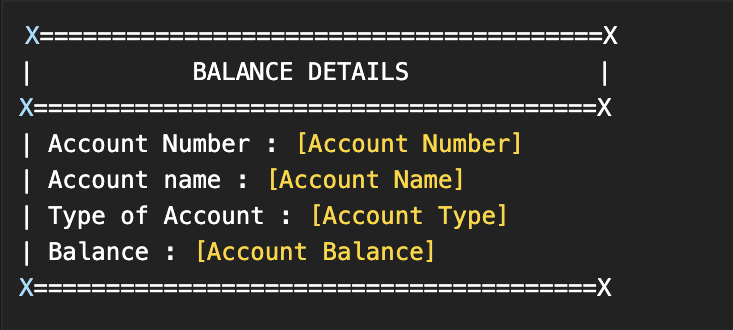
1. **Withdraw Cash**:

* **Proposal**: The deposit\_withdraw() function in the AccountUserFunctions namespace also handles the withdrawal operation. It prompts the user to enter the account number and the amount to be withdrawn. The function checks if the account has sufficient balance and updates the account balance accordingly.
* **Display**:



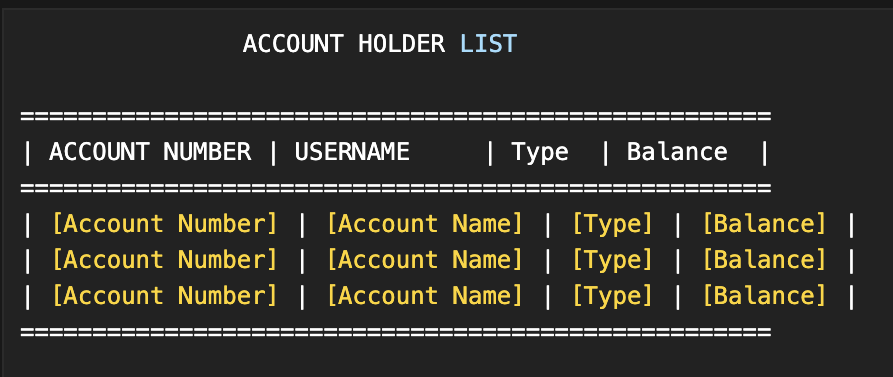
1. **Search Account**:

* **Proposal**: The display\_sp() function in the AccountUserFunctions namespace searches for an account based on the provided account number. It reads the account records from the binary file and displays the account details if found.
* **Display**:



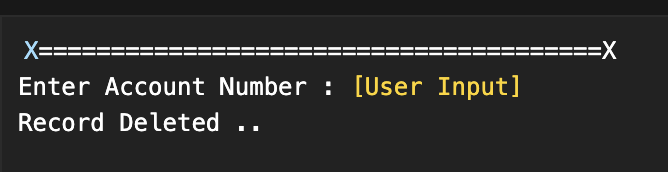
1. **View All Accounts**:

* **Proposal**: The display\_all() function in the AccountUserFunctions namespace reads all account records from the binary file and displays them in a tabular format, showing the account number, account holder's name, account type, and balance.
* **Display**:



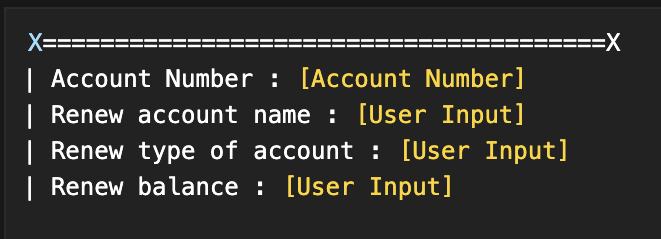
1. **Close Account**:

* **Proposal**: The delete\_account() function in the AccountUserFunctions namespace deletes an account record from the binary file based on the provided account number. It creates a temporary file, copies all records except the one to be deleted, and then replaces the original file with the temporary file.
* **Display**:



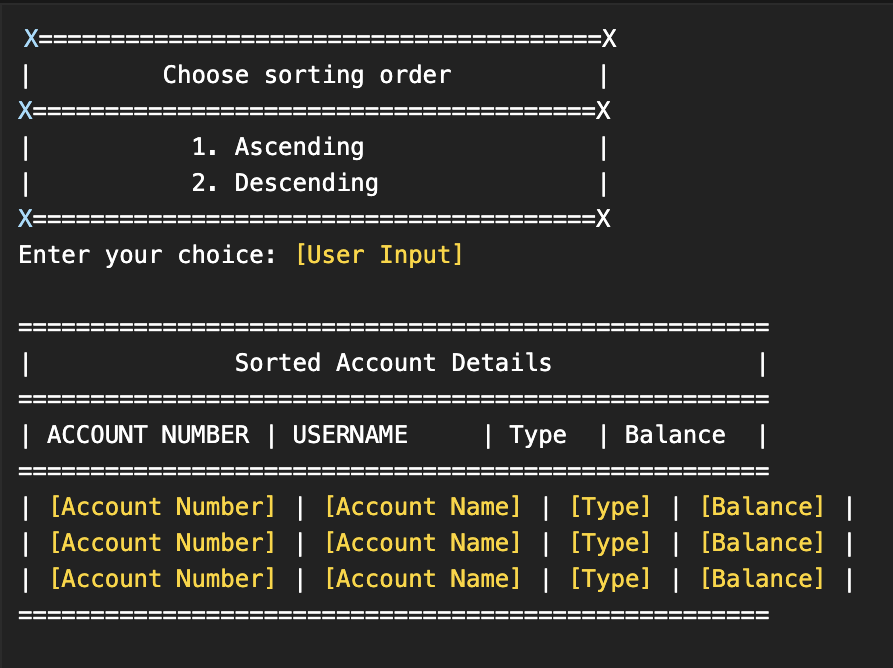
1. **Edit Account Details**:

* **Proposal**: The rename\_account() function in the AccountUserFunctions namespace allows users to edit the account details, such as the account holder's name, account type, and balance, by providing the account number. It reads the account record, prompts the user for new details, and updates the record in the binary file.
* **Display**:



1. **Sort Accounts**:

* **Proposal**: The sort\_account\_numbers() function in the AccountUserFunctions namespace sorts the account records based on the account number in ascending or descending order, as chosen by the user. It reads all account records, sorts them using the Bubble Sort algorithm, and displays the sorted list.
* **Display**:



1. **Log Out**:

* **Proposal**: The LOG\_OUT case in the bankManagement() function of the Application class simply displays a message indicating that the user is being taken back to the login and registration menu.
* **Display**:

