Abdelmalek Essaadi University
National School of Applied Sciences Al-Hoceima

Project Report

Development of a bank account management application
(Advanced Data Structure and Algorithmics)
ID1

DIRECTED BY:
MOHAMED-SABER EL GUELTA
SOUKAINA EL HADIFI

SUPERVISED BY: **Dr. Abdelkhalek Bahri**

Academic year: 2023-2024

Contents

Problematic

The development of a C language-based banking account management application, designed to seamlessly operate across both Unix and Windows operating systems, presents intricate and specific challenges. The problem statement for this project can be broken down into several key aspects:

1. Portability and Compatibility:

- How can optimal portability of the application between Unix and Windows environments be ensured, given the fundamental differences in terms of file systems, system calls, and libraries?

2. System Resource Management:

- How can system resource management, particularly memory, be optimized to ensure balanced performance across operating systems with different management models?

3. Unified User Interface:

- How can a unified user interface be designed to provide a consistent user experience, regardless of the underlying operating system, while leveraging the specificities of each platform?

4. Security and Authentication:

- How can the security of banking operations be ensured, taking into account authentication mechanisms specific to each operating system?

5. Maintenance and Updates:

- How can the maintenance and update process of the application on both platforms be streamlined, minimizing manual interventions and ensuring feature consistency?

About

This project revolves around the creation of a robust banking account management application using the C programming language. The application is strategically developed in two versions, tailored to seamlessly operate on both Unix and Windows systems. The culmination of this effort aims to provide financial institutions with a versatile and efficient tool for managing banking accounts.

Project Overview:

The primary objective of this project is to design and implement a comprehensive banking application with C language, ensuring compatibility with both Unix and Windows environments. The dual-version approach is adopted to address the distinct needs of users across these two prevalent operating systems, fostering widespread accessibility and usability.

Project Objectives:

- Develop a C language-based banking application for Unix systems.
- Develop a C language-based banking application for Windows systems.
- Ensure consistency and reliability across both versions.
- Implement core functionalities for secure and efficient account management.
- Provide a comprehensive user guide for seamless application usage.

Target Audience:

This application caters to banking institutions and financial organizations seeking a cross-platform solution for streamlined account management. The dual compatibility ensures that users operating on both Unix and Windows platforms can benefit from the application's features.

Methodology:

The project adheres to an agile development methodology, emphasizing iterative development, continuous feedback, and adaptability. Regular sprints are scheduled to ensure alignment with project objectives.

Challenges:

- Addressing platform-specific nuances in system calls and libraries
- Ensuring seamless compatibility across different file systems

Timeline:

- Phase 1: Project Planning and Requirement Analysis (Weeks 1-2)
- Phase 2: Unix Version Development (Weeks 3-4)
- Phase 3: Windows Version Development (Weeks 3-4)
- Phase 4: Integration and Testing (Weeks 5-6)
- Phase 5: Documentation, User Guide Preparation, and Finalization (Week 7)

The development of this C language-based banking account management application represents a significant advancement in providing financial institutions with a versatile and user-friendly solution. The inclusion of both Unix and Windows versions enhances the application's adaptability and accessibility.

Technological Stack

- ***Operating Systems:***

- ✓ **Unix:**

  -The application is optimized for Unix-based operating systems, ensuring compatibility with Unix-specific features and leveraging the stability of Unix environments.

- ✓ **Windows:**

  - A version tailored for Windows operating systems is developed to provide a seamless and native user experience on Windows platforms.

- ***Collaboration & Version Control***:

- ✓ **Git** :

  Git is a distributed version control system, meaning each developer has a local copy of the entire repository. This allows for independent work on local branches, providing flexibility and enabling collaboration without the need for a constant network connection.

- ✓ **GitHub** :
  GitHub is a web-based platform that serves as a hosting service for Git repositories. It provides a collaborative environment and facilitates code review processes.

- *IDE:*

✓ **Visual Studio :**

Visual Studio is a comprehensive integrated development environment (IDE) developed by Microsoft. It is widely used for software development across various platforms and programming languages. Visual Studio provides a suite of tools, features, and services that streamline the entire software development lifecycle, from coding and debugging to testing and deployment.

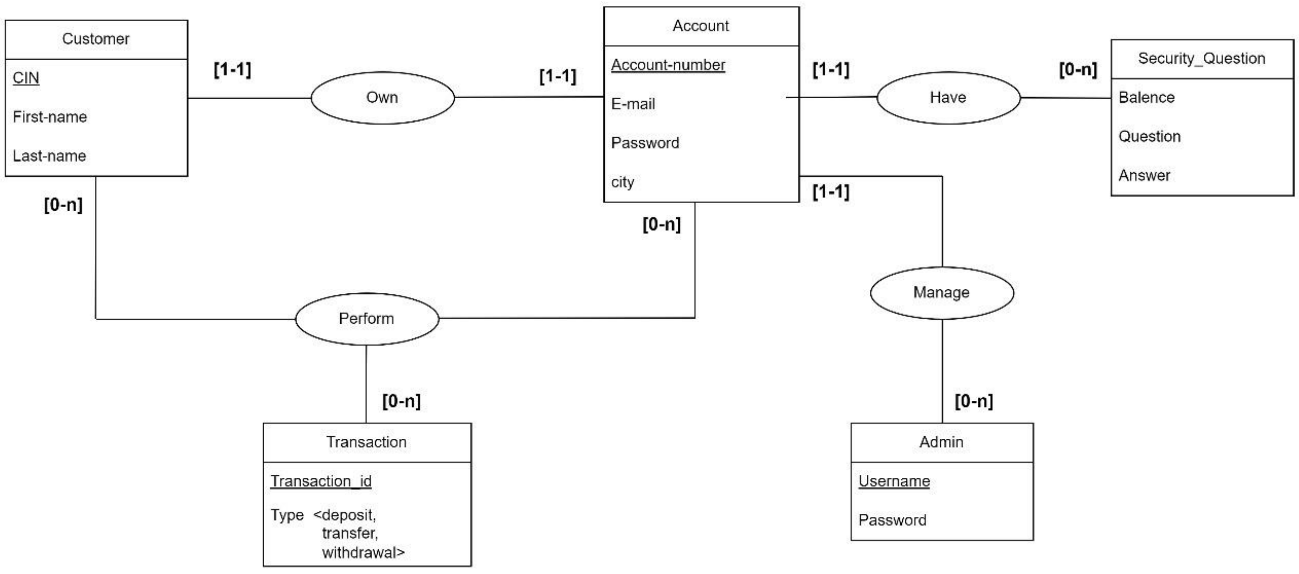
- ***Programming Language:*** C

- The application is developed using the C programming language, chosen for its efficiency, low-level capabilities, and cross-platform support.

This carefully selected technological stack aims to ensure efficient development, cross-platform compatibility, and collaborative version control throughout the lifecycle of the banking account management application. The combination of C, Git/GitHub, and Visual Studio reflects a strategic approach to address the specific requirements of both Unix and Windows environments.

Modeling

✓ E - A Model :



✓ Relational Model :

- **CUSTOMERS** (**CIN :** varchar ,city: integer , First-name: varchar, Last-name : varchar, Password :integer ,E-mail: varchar)
- **ACCOUNTS** (**Account-number :** integer, E-mail: varchar, Password :integer, city: integer , **# Username**)
- **ADMINS** (**Username** : varchar , Password : varchar)
- **SECURITY-QUESTIONS** (Balance: integer , Question : integer , Answer : varchar)
- **PERFORMANCES** (**#CIN , #Account-number, #Transaction_id**)
- **TRANSACTIONS** (**Transaction_id** : integer , Type : integer)

Key Features

The following key features will be thoroughly discussed and implemented in the subsequent sections of the project

- ✓ ***USER :***
  - Account request
  - Authentication
  - Status Request
  - Password recovery

- ✓ ***Client :***
  - Balance Inquiries
  - Secure Fund Transfers
  - Withdrawal
  - Make Deposits

- ✓ ***Admin :***
  - Check Account Requests
  - Display Accounts
  - Update Accounts
  - Disable/Enable Account

The implementation process will delve into the specific functionalities and procedures required to ensure the seamless operation of each feature for the respective user roles."

C libraries

- **< Stdio.h >**

  The stdio.h library is a standard library of the C language that contains functions for the manipulation of inputs and outputs. For example, this library allows to read and write data on the console, on files, or on strings of characters.

- **< Stdlib.h >**

  The stdlib.h is a header file in the C language that defines four variable types, several macros, and various functions for performing general functions. Some of the functions include dynamic memory management, random number generation, communication with the environment, integer arithmetic, searching, sorting, and converting.

- **< String.h >**

  The string.h is a header file in the C language that defines several functions for manipulating strings (arrays of characters). Some of the functions include copying, concatenating, comparing, searching, and manipulating strings.

- **< Ctype.h >**

  In the C programming language, ctype.h is a header file that stands for "character type." It is part of the standard C library and provides functions for testing and manipulating character data. The functions declared in ctype.h are primarily used to classify characters (e.g., whether a character is alphabetic, numeric, etc.) and to convert between uppercase and lowercase.

- **< Windows.h >**

  windows.h is a header file that provides declarations for a wide range of functions, data types, and constants needed for Windows programming. This header file is a fundamental part of the Windows API (Application Programming Interface), which allows developers to create Windows applications.

- **< Conio.h >**

  The conio.h header file is not a standard part of the C or C++ language. It is specific to certain compilers, notably those that target DOS and Windows environments. The name "conio" is derived from "console input/output," and the header provides a set of functions for simple console-based input and output operations.

- **< Unistd.h >**

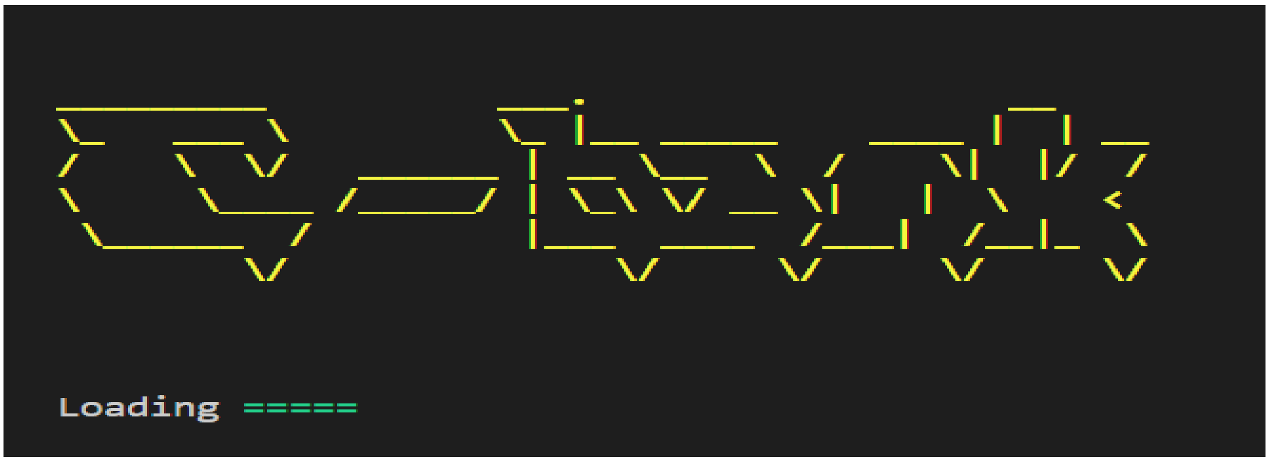
  The unistd.h header file is primarily associated with Unix and Unix-like operating systems, and it provides declarations for various system calls and other constants. It is not part of the C standard, but it is widely used in Unix-based systems.

- **< Termios.h >**

  The termios.h header file is part of the POSIX standard and is commonly used in Unix-like operating systems. It provides declarations for terminal I/O (Input/Output) operations, allowing programs to interact with the terminal in a more sophisticated manner. The name "termios" stands for "terminal input/output structure.

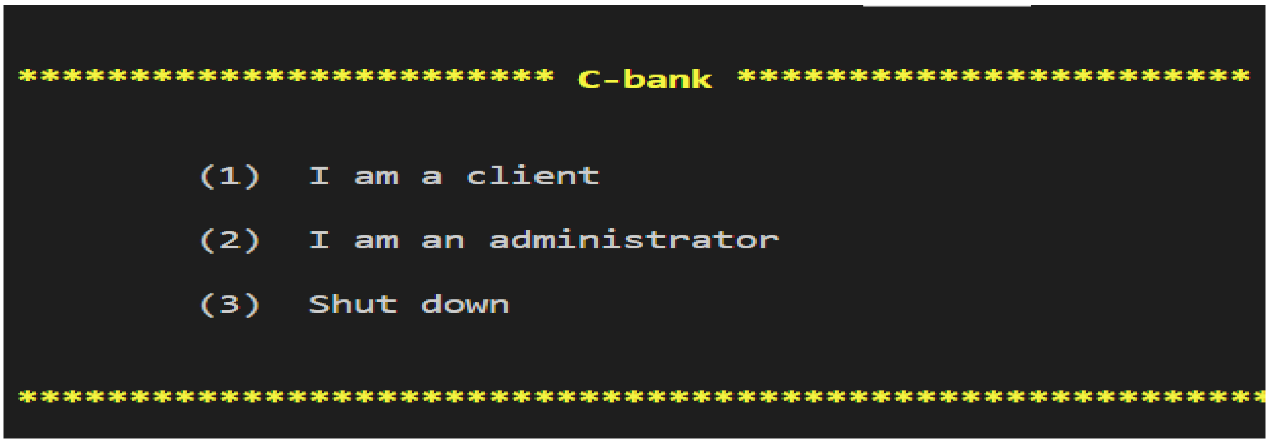
Implementation

(Users are prompted to specify their role as either a client or an administrator. This initial choice determines the set of functionalities they can access within the application, tailoring the experience based on their designated role.)



```
Loading =====
```

Figure 1

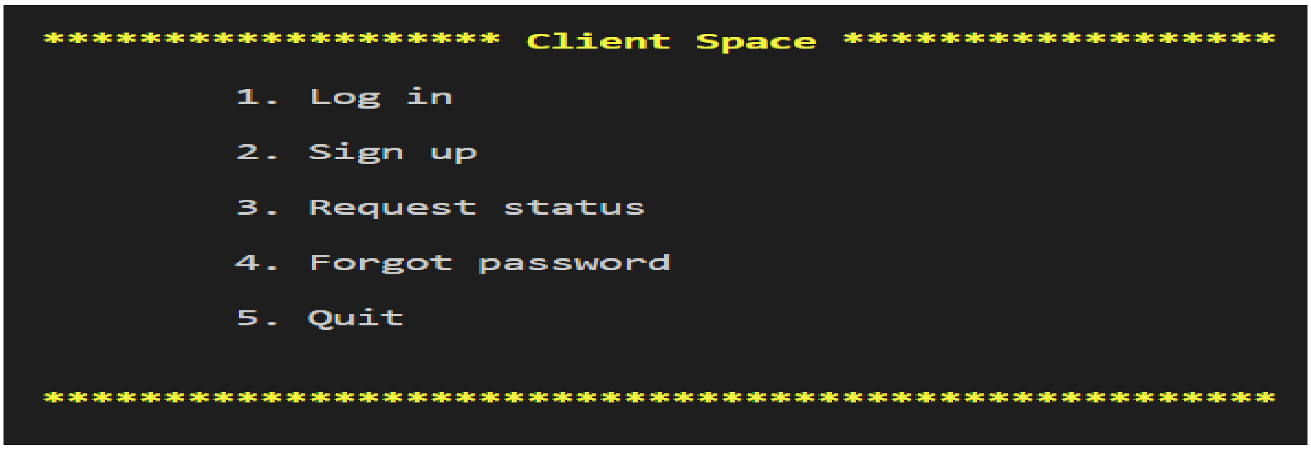


```
************************ C-bank ***********************

        (1)  I am a client

        (2)  I am an administrator

        (3)  Shut down

*******************************************************
```

Figure 2

I. User Space

1-key functionalities :



```
******************** Client Space *******************
        1. Log in
        2. Sign up
        3. Request status
        4. Forgot password
        5. Quit

*****************************************************
```

Figure 3

2- Control flow

- **Sign up (Request account) :**

  the function captures user information, performs checks to avoid duplicates, sets up security measures, allows the user to confirm or cancel the operation, and provides feedback throughout the process.

- **Log in**

  This process is designed to verify the identity of a user attempting to log in by checking their account number, verifying the password, and ensuring the account is active. The user receives notifications and feedback at each step of the authentication process.

- **Request status :**

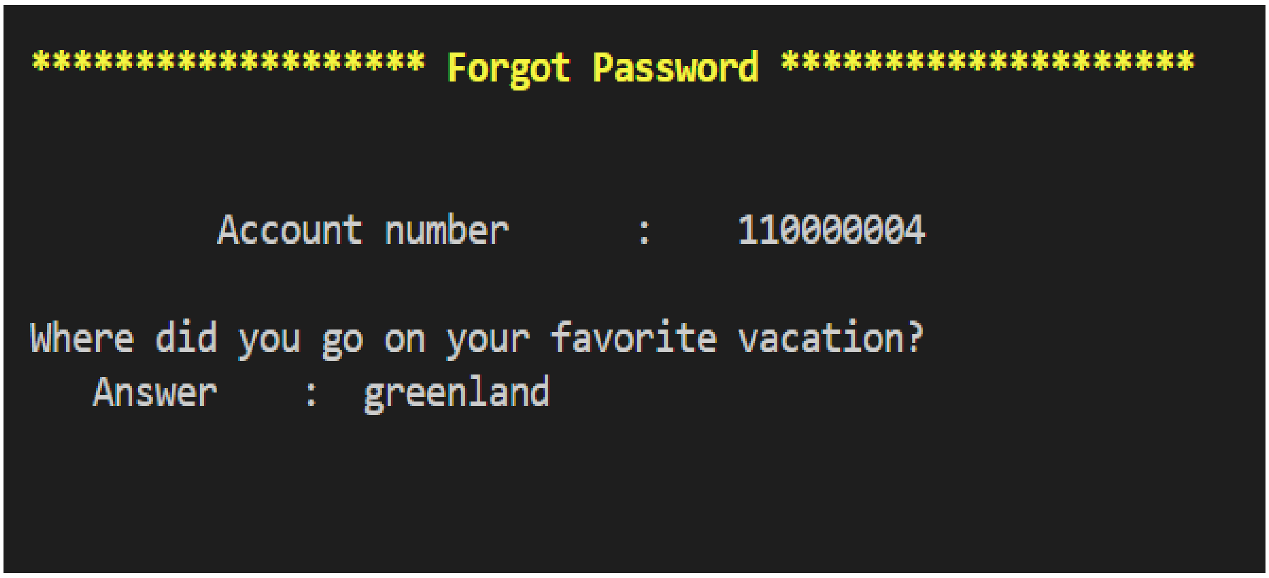
  This process involves user authentication, checking account status, and providing relevant notifications based on the user's input and account information.

- **Forgot password :**

  This process involves user interaction, security verification through a security question, and password reset functionality with corresponding notifications.

3-Demo

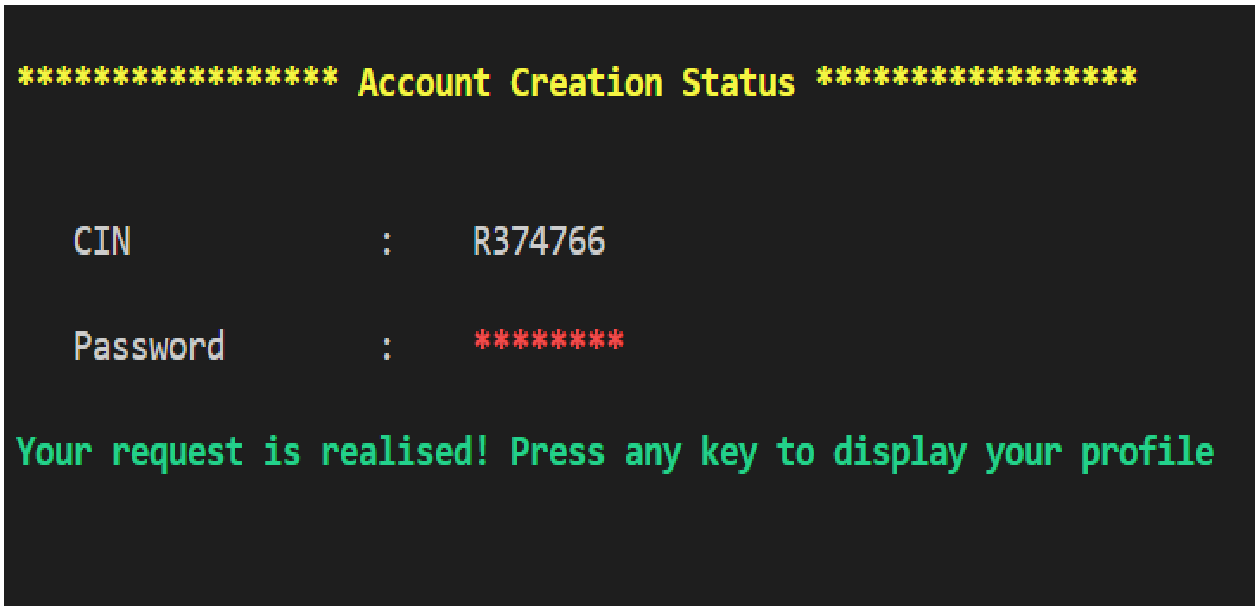
➢ Forgot password



```
******************* Forgot Password ********************


         Account number      :    110000004

Where did you go on your favorite vacation?
   Answer    :  greenland
```

Figure 4

➢ Request Status

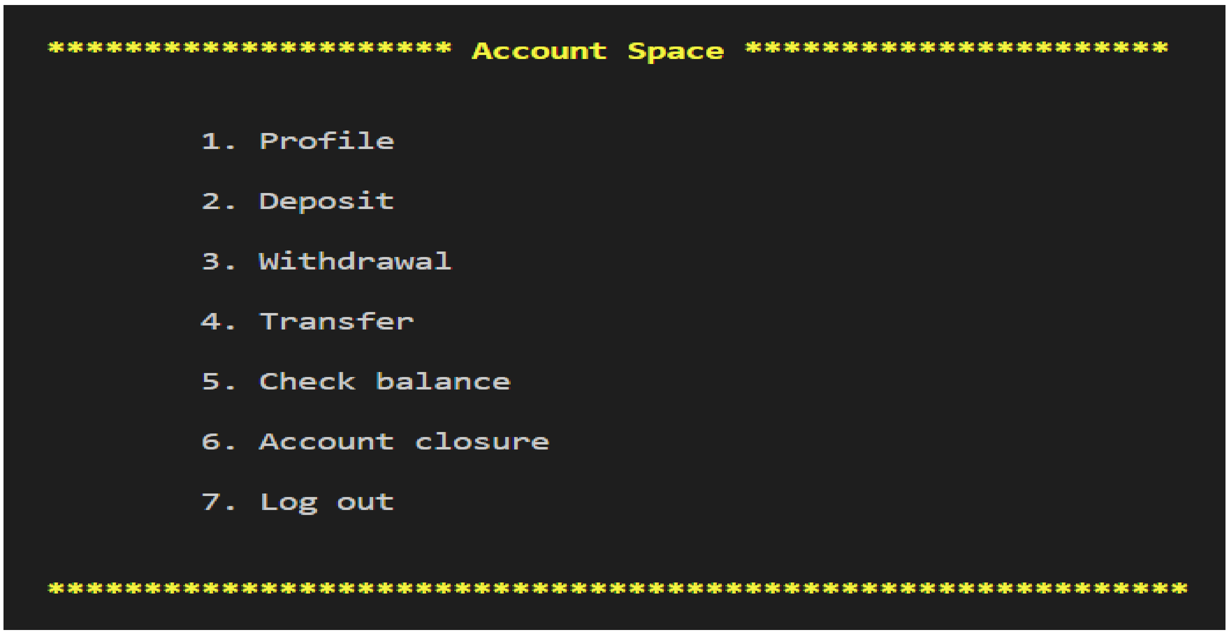


```
**************** Account Creation Status ****************


   CIN              :    R374766

   Password         :    ********

Your request is realised! Press any key to display your profile
```

Figure 5

II. Client Space :

1-key Functionalities :



```
********************** Account Space ***********************

        1. Profile

        2. Deposit

        3. Withdrawal

        4. Transfer

        5. Check balance

        6. Account closure

        7. Log out

************************************************************
```

Figure 6

2- Control flow

- **Transfer :**

  the function handles user input for the destination account and transfer amount, validates the inputs, checks for the existence of the destination account, and executes the transfer if the conditions are met. It provides status feedback on the success or failure of the transfer operation.

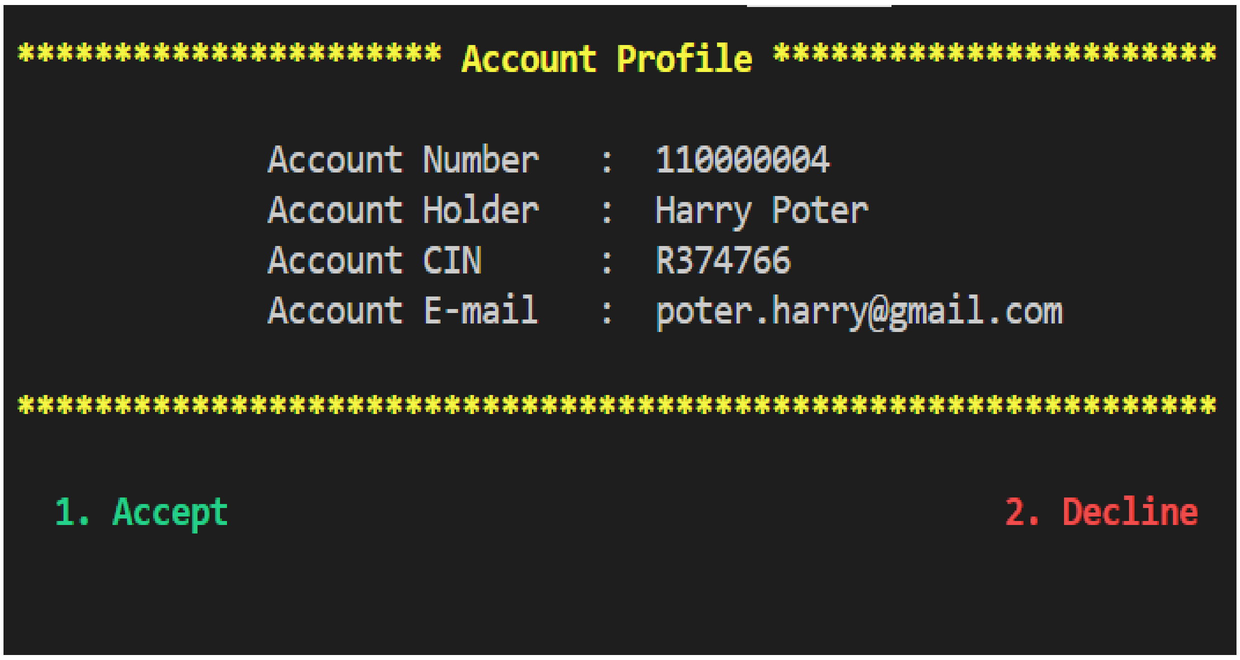
- **Deposit:**

  the function prompts the user for a deposit amount, validates the input, updates the client's balance, performs necessary file updates, and provides status feedback on the success or failure of the deposit operation.

- **Withdrawal:**

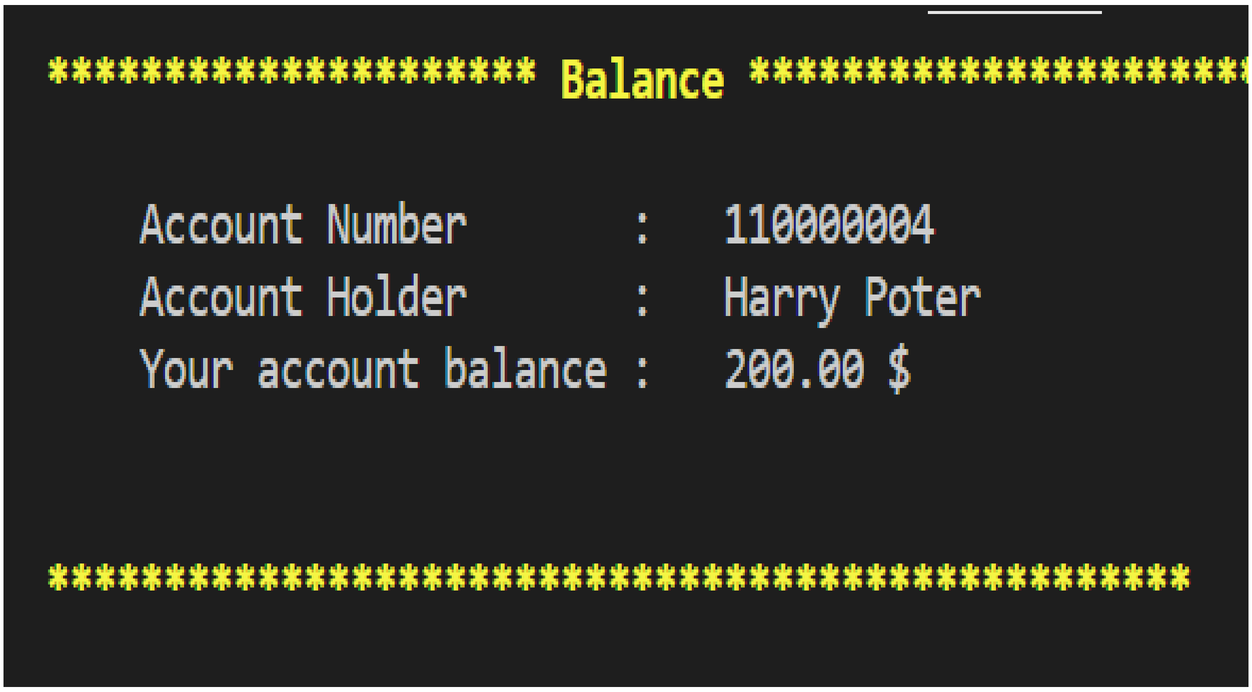
  the function handles user input for the withdrawal amount, validates the input, checks for sufficient funds, and executes the withdrawal if the conditions are met. It provides status feedback on the success or failure of the withdrawal operation.

3-Demo



```
********************** Account Profile ***********************

             Account Number    :  110000004
             Account Holder    :  Harry Poter
             Account CIN       :  R374766
             Account E-mail    :  poter.harry@gmail.com

**************************************************************

  1. Accept                                          2. Decline
```

Figure 7

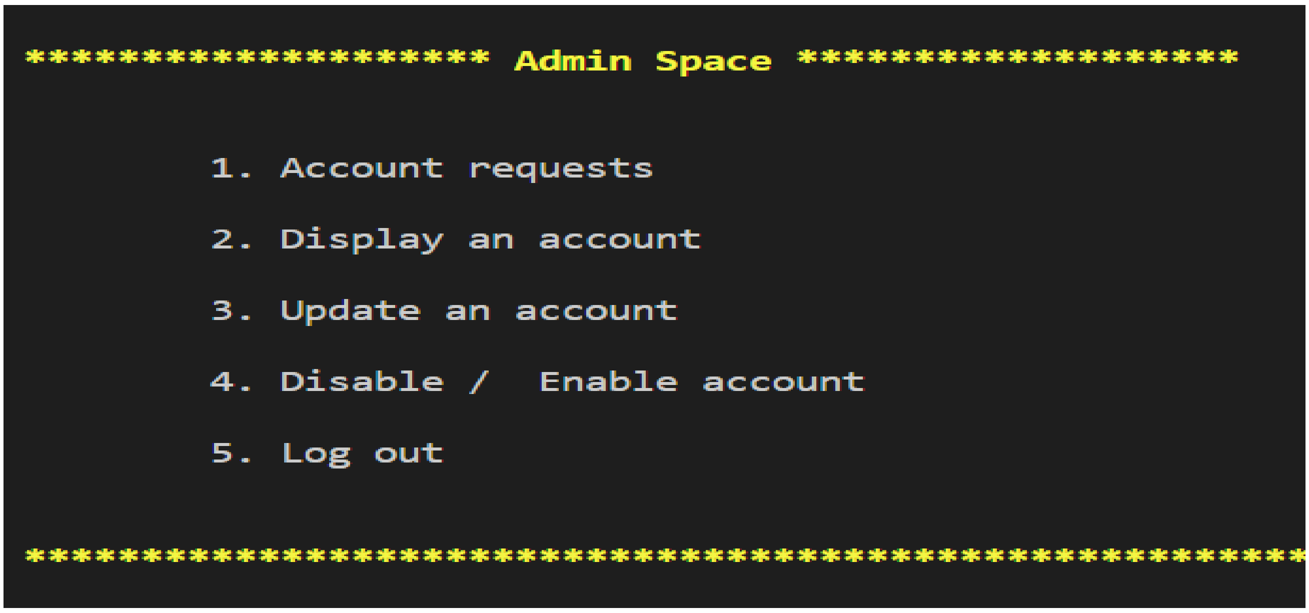


```
********************** Balance ***********************

    Account Number       :   110000004
    Account Holder       :   Harry Poter
    Your account balance :   200.00 $


*****************************************************
```

Figure 8

III. Admin Space :

1-key Functionalities :



```
******************** Admin Space ********************


        1. Account requests

        2. Display an account

        3. Update an account

        4. Disable /  Enable account

        5. Log out


*******************************************************
```

Figure 9

2- Control flow

➢ **Check requests**

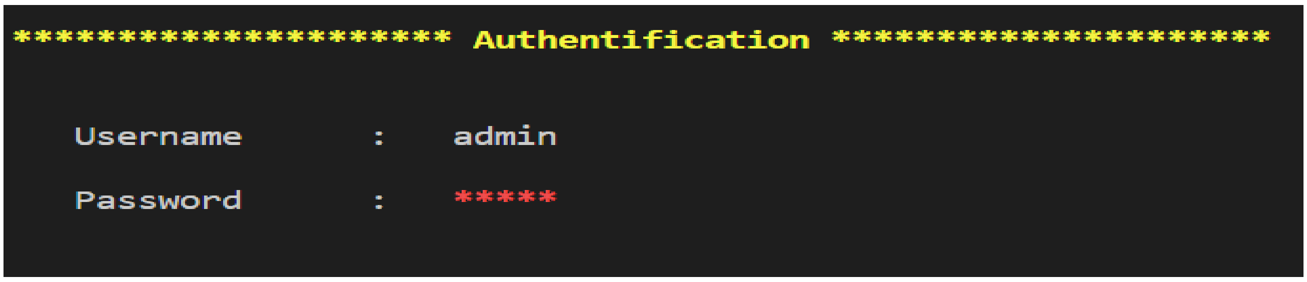
The administrator is given the option to accept or decline the account creation request. If accepted, the client's information is written to the main accounts file; if declined, the request is simply marked as declined. The function provides feedback messages to the administrator throughout the process, including notifications for existing clients.

➢ **Disable or enable account**

The function allows the administrator to disable or enable a user account based on the entered account number. It prompts the administrator to enter the account number, displays the profile information of the corresponding client, and then presents options to either enable or disable the account. After the administrator makes a choice, the function performs the corresponding action, such as activating or closing the account, and provides feedback messages accordingly.

3-Demo

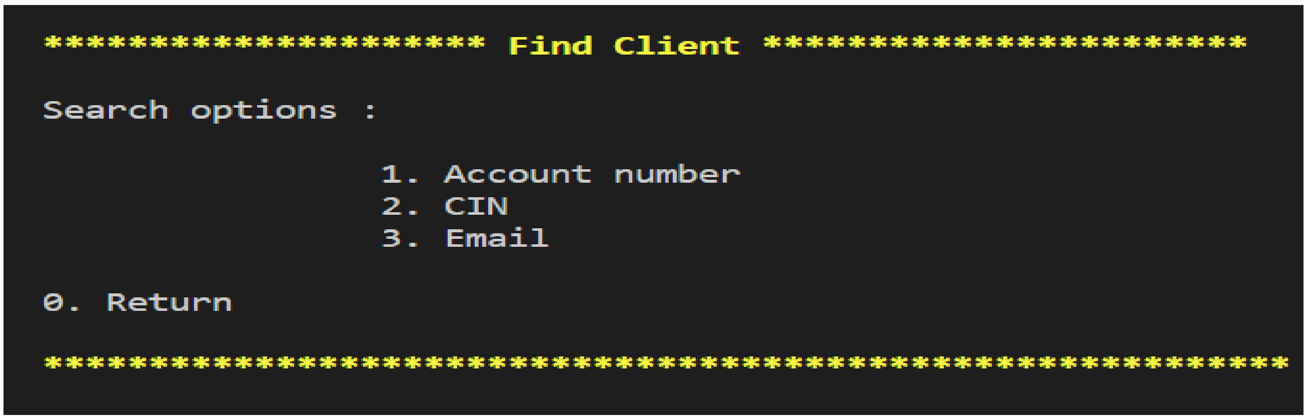
- **Admin authentification**



```
********************** Authentification **********************


   Username        :    admin

   Password        :    *****
```

Figure 10

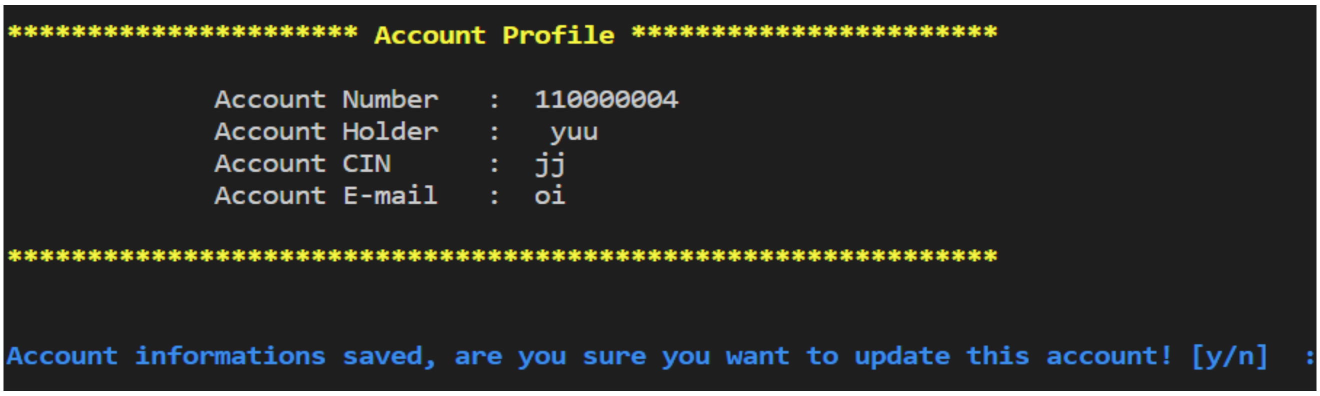
- **Find client option in order to display it**



```
 ********************** Find Client ************************

 Search options :

                 1. Account number
                 2. CIN
                 3. Email

 0. Return

 ***************************************************************
```

Figure 11

- **Update an account**



```
****************** Account Profile *******************

          Account Number   :  110000004
          Account Holder   :   yuu
          Account CIN      :  jj
          Account E-mail   :  oi

***************************************************


Account informations saved, are you sure you want to update this account! [y/n]  :
```

Figure 12

C-bank strengths

1. Enhanced Security:

- C-Bank prioritizes the security of user accounts by implementing robust password encryption techniques, ensuring the confidentiality of sensitive information.

2. Visual Appeal and User Experience:

- The application incorporates visually appealing elements such as processing animations, color-coded messages, and sleep functions, enhancing the overall user experience and making interactions more engaging.

3. Effective Handling of Unexpected Input:

- C-Bank is designed to handle unexpected user inputs gracefully. The system provides clear error messages, prompts for valid input, and ensures a smooth user experience even in scenarios with unexpected responses.

4. User Interaction and Confirmation:

- The application prioritizes user interaction by incorporating functionalities like user confirmation prompts for critical operations. Users are prompted to confirm actions, reducing the likelihood of accidental operations and providing a more user-friendly experience.

5. Intuitive Design:

- C-Bank features an intuitive design that guides users through various functionalities, ensuring ease of use and reducing the learning curve for both novice and experienced users.

End