COE125/C2 – Software Engineering

Group Members:

Antolin, Arryll Mori N.

Jawod, Danielle Nicole O.

Reyes, Jean Marc D.

**ADJ BANKING SOFTWARE**

1. **Introduction**

ADJ Banking Software maintains client’s bank accounts and providing basic banking services such as deposit, withdraw, and change PIN. The group tried to recreate a real-life banking management services by providing a Python application of such which consist of two main parts: The administrative module and the ATM module. The administrative module is used and accessed by the System Administrators or Bank Tellers. It has the following functionalities: To open, close accounts and view account information. In opening accounts, the new account will be added to the database system. In closing accounts, an existing account will be deleted in the database. In view account information, an account number is required to be entered to view the particular account. On the other hand, the ATM module side can be used by the client in order to do deposit, withdraw, balance inquiry, or change pin transaction. First, the user must be authenticated by putting his/her card number (ATM card insertion in real-life ATM) together with the PIN number. All except balance inquiry will update the database after execution. In summary, the administrative module has open close account functionalities while ATM module has deposit, withdraw, change pin and check balance transaction.

1. **Objectives**

* To be able to implement a banking management software using Python programming by providing an application that can offer basic bank transactions.
* To be able to utilize SQLite as the database management system of the software.

1. **Functions**

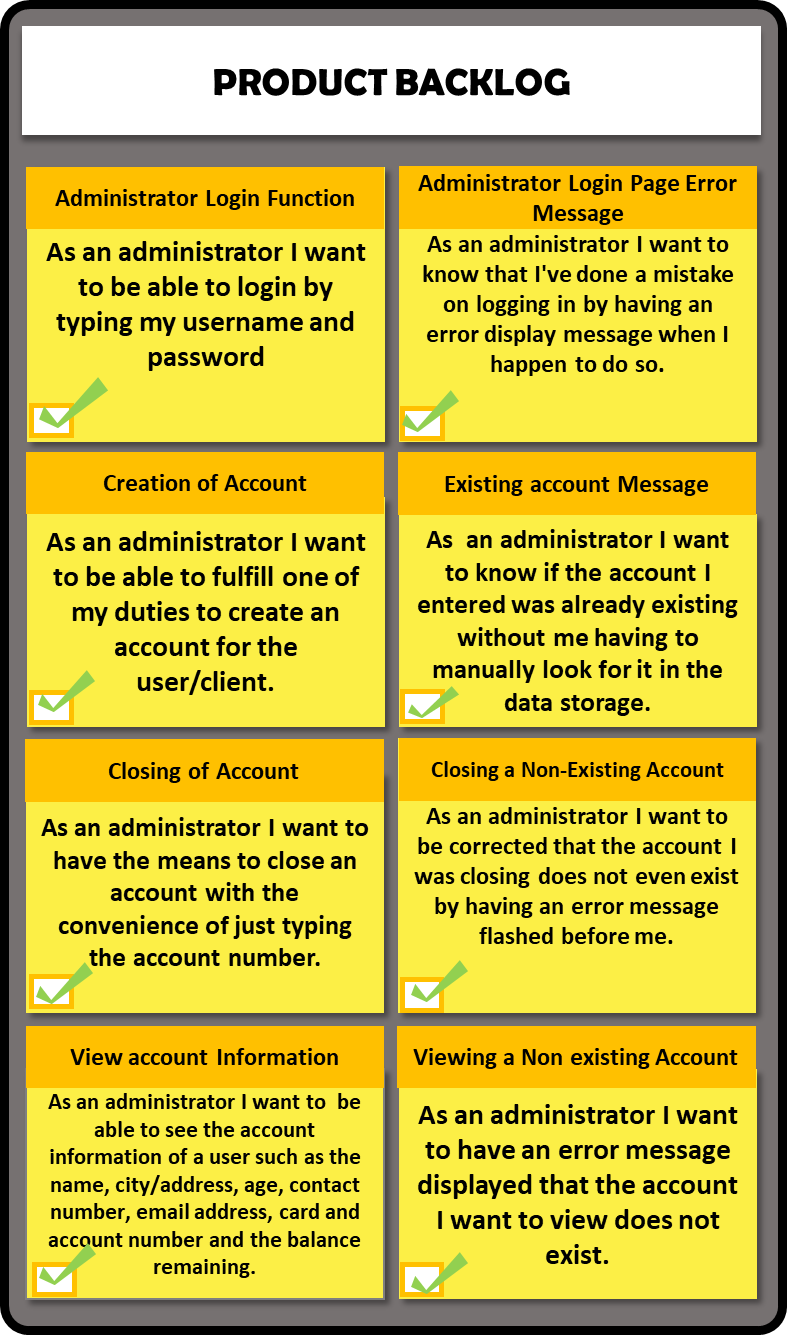
**Administrator**

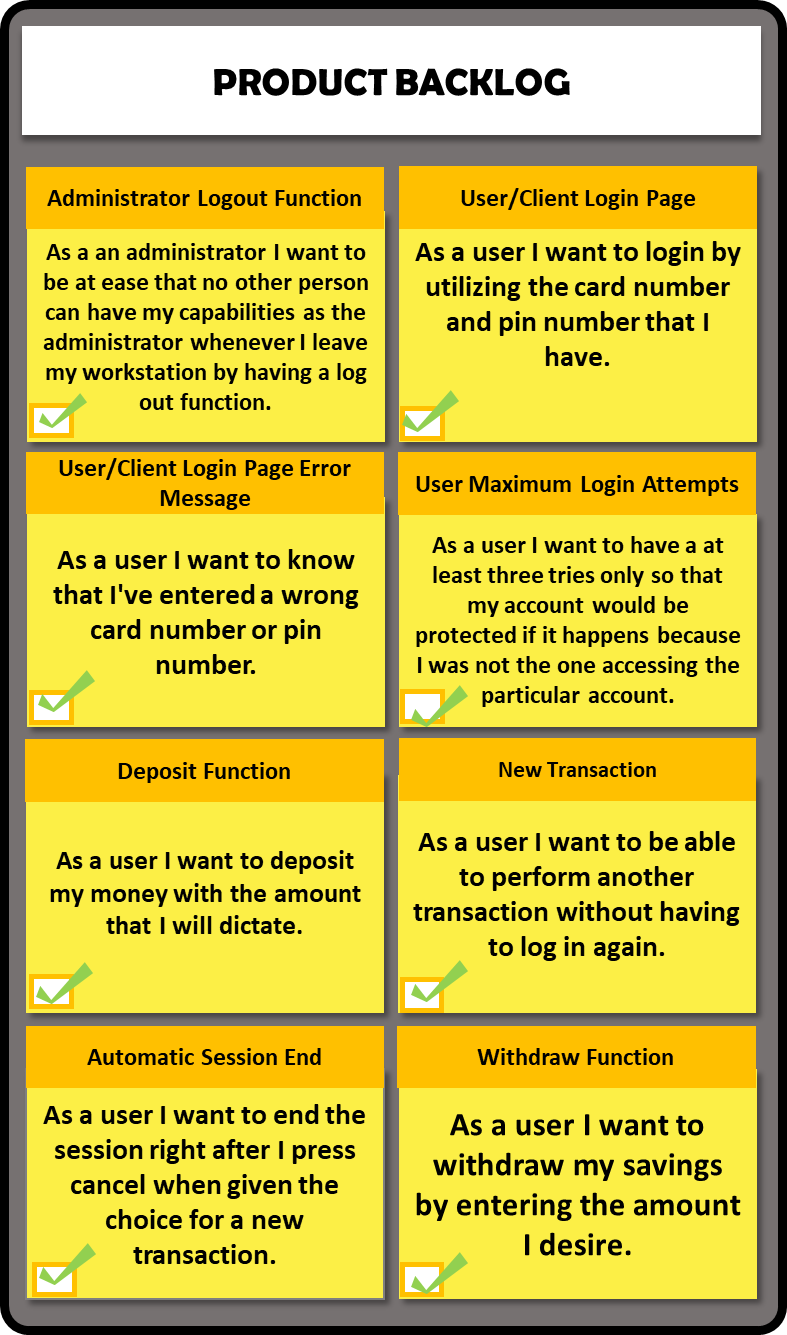
* Open Account - the new account will be added to the database system.
* Close Account - an existing account will be deleted in the database.
* View Account - an account number is required to be entered to view the particular account.

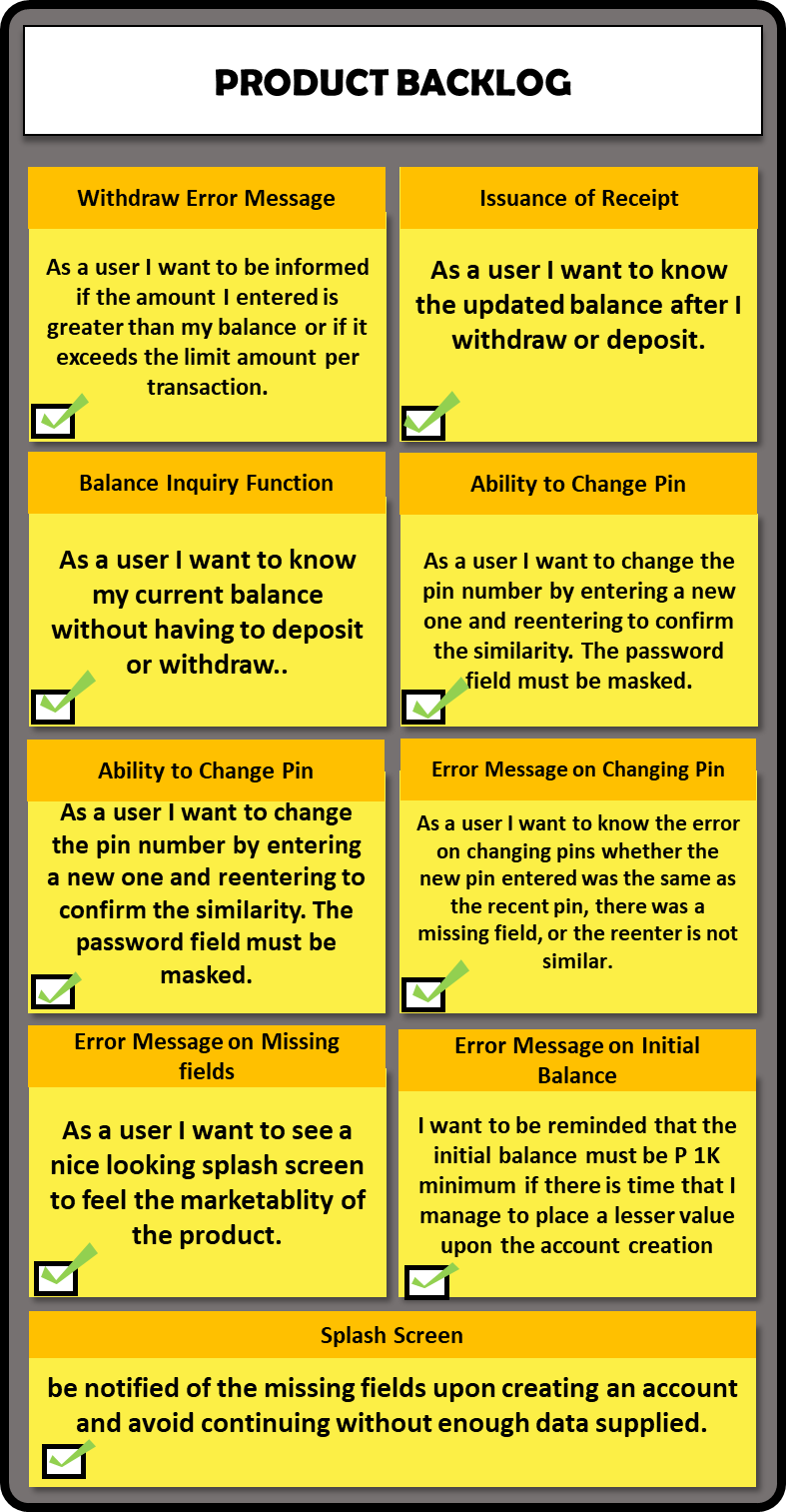
**Client (ATM MODULE)**

* Deposit – prompts the user to enter the desired amount to be deposited.
* Withdraw - prompts the user to enter the desired amount to be withdrawn.
* Balance inquiry – Displays the current balance of the user as determined in the SQLite database.
* Change Pin transaction – Requires the user to enter a new pin number. Reentering the new pin number is also required to make sure that the entered pin is the correct desired pin.

1. **Product Backlog**

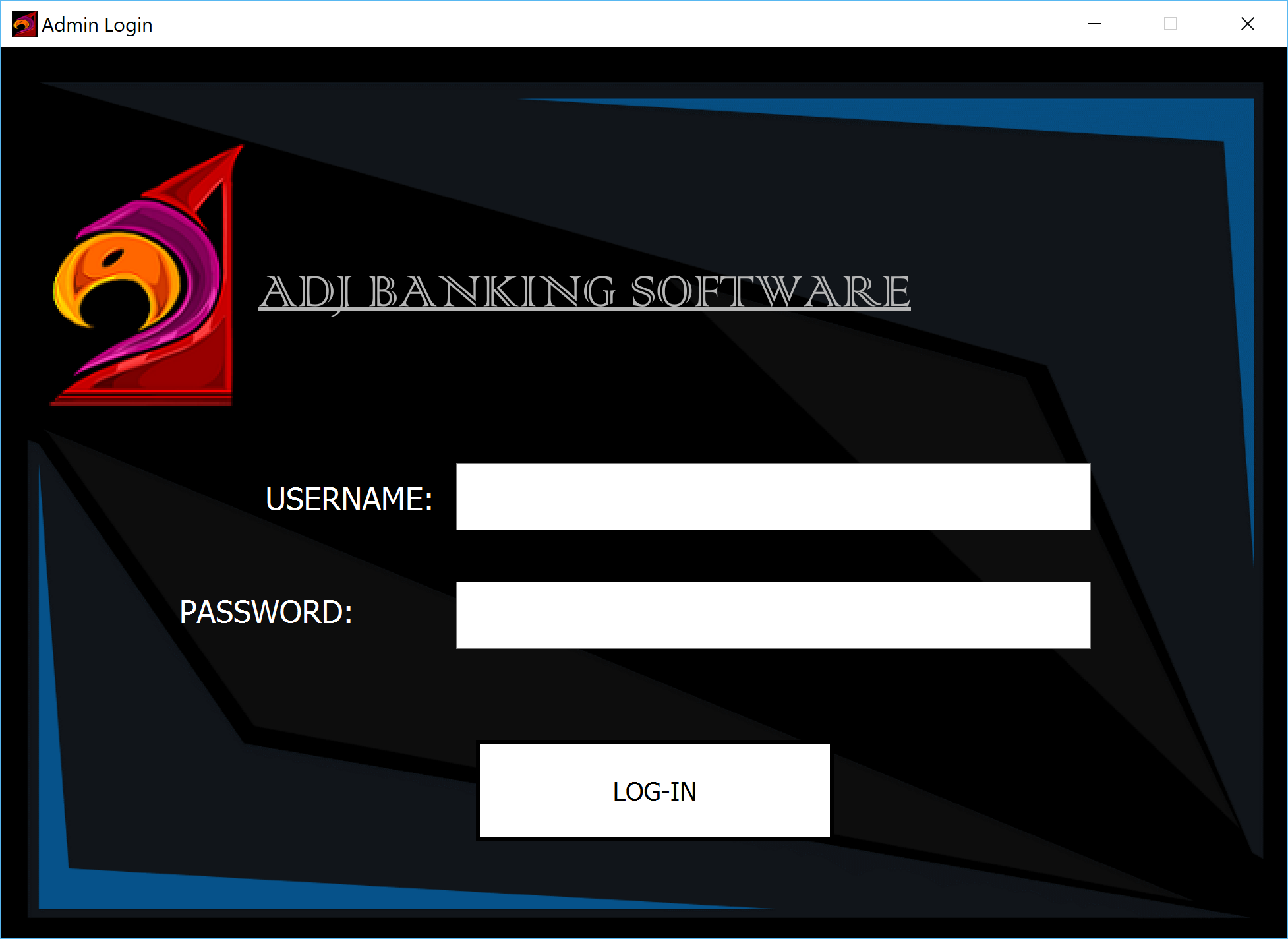
****

****

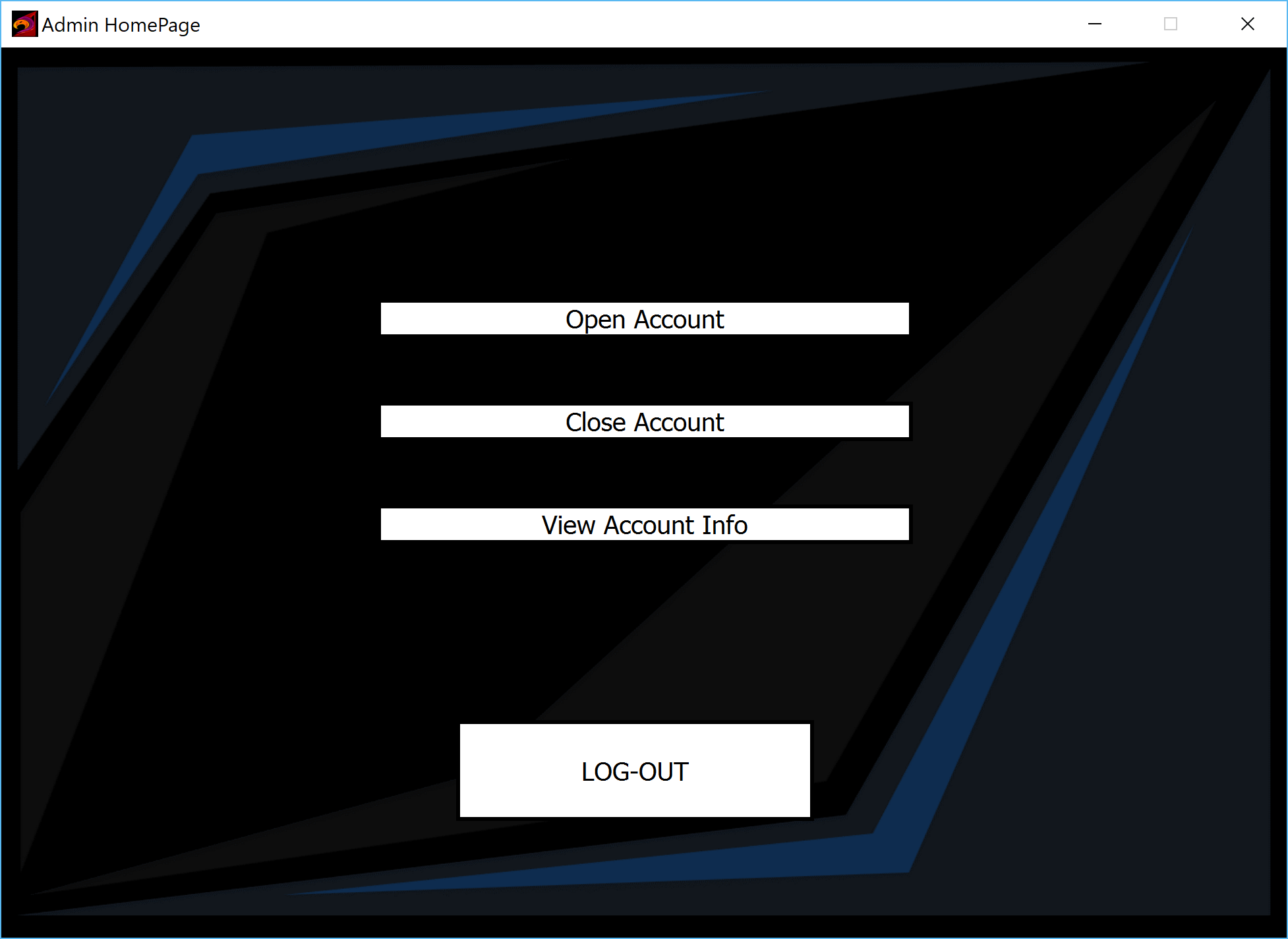
****

1. **Graphical User Interface**

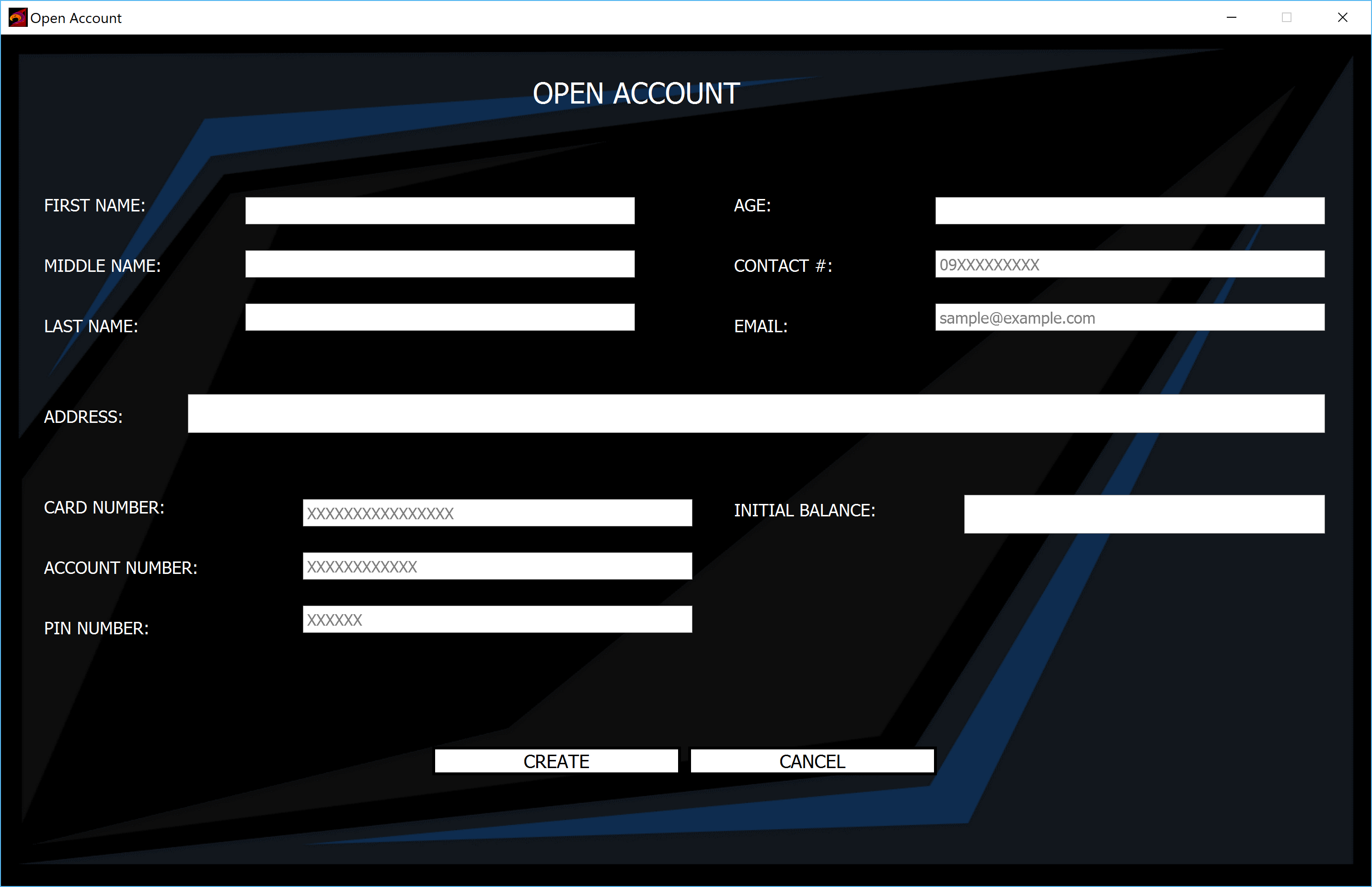
**Admin UI**



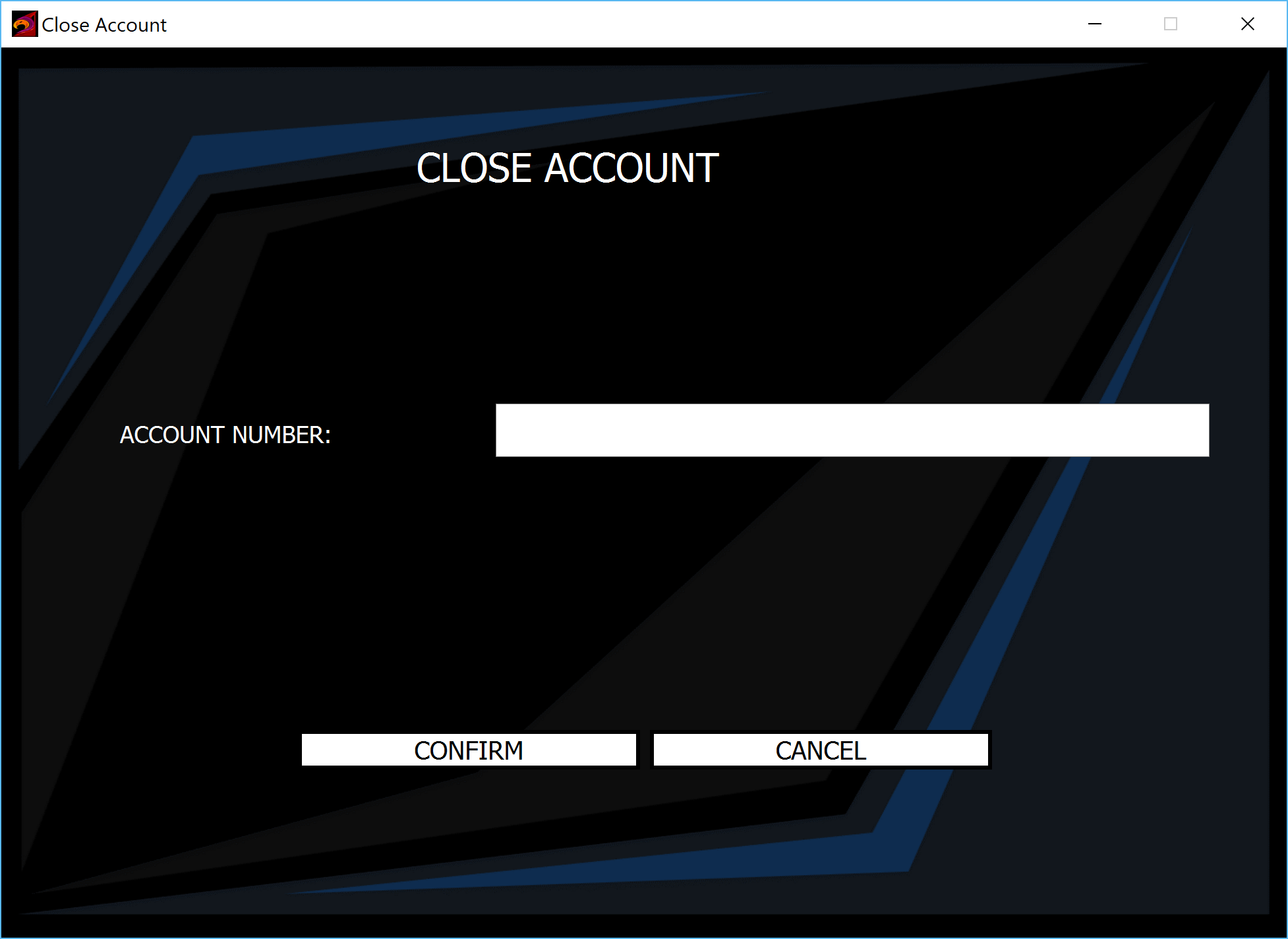
*Login Page (Administrator)*



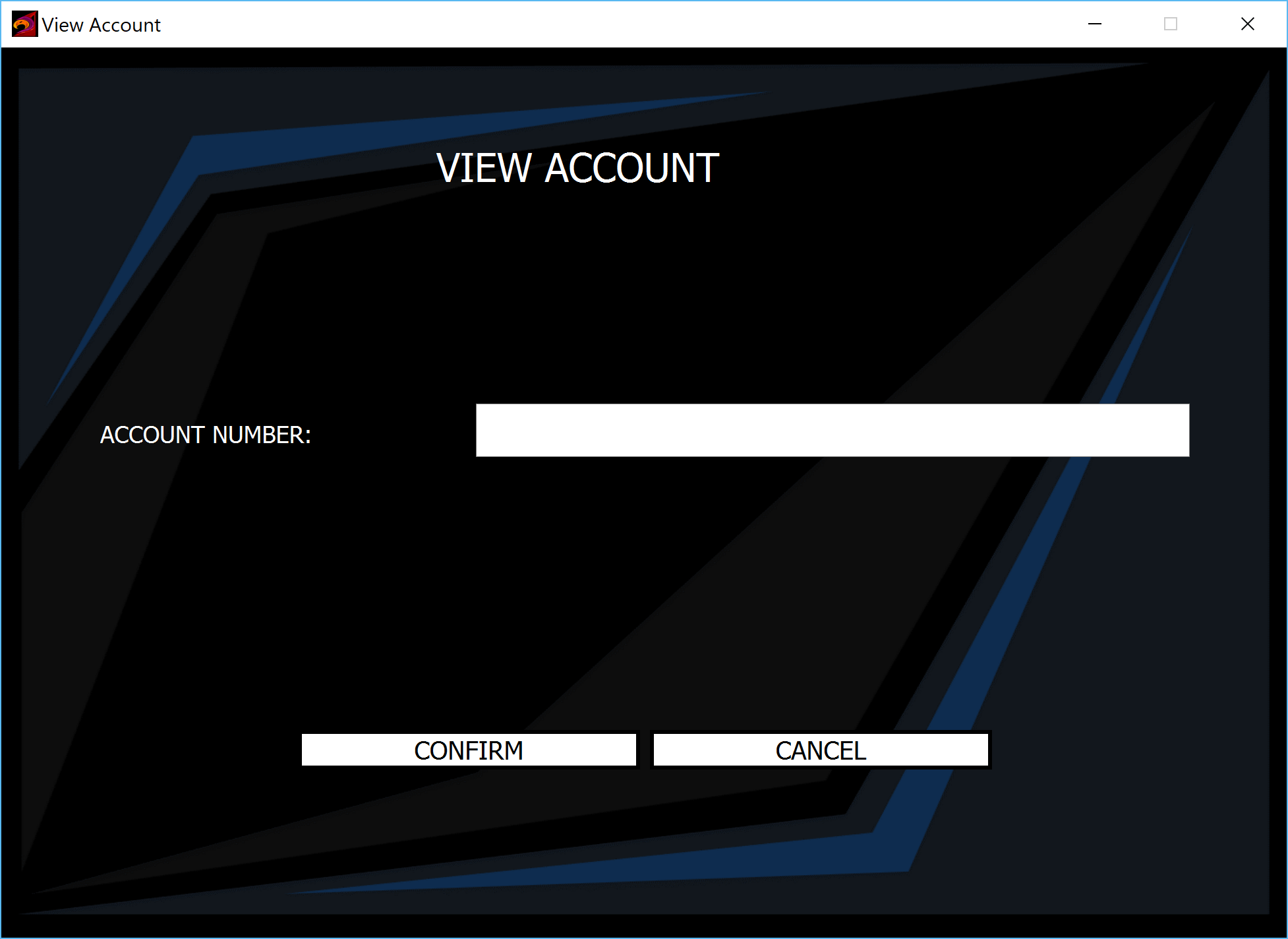
*Homepage (Administrator)*



*Open Account (Administrator)*

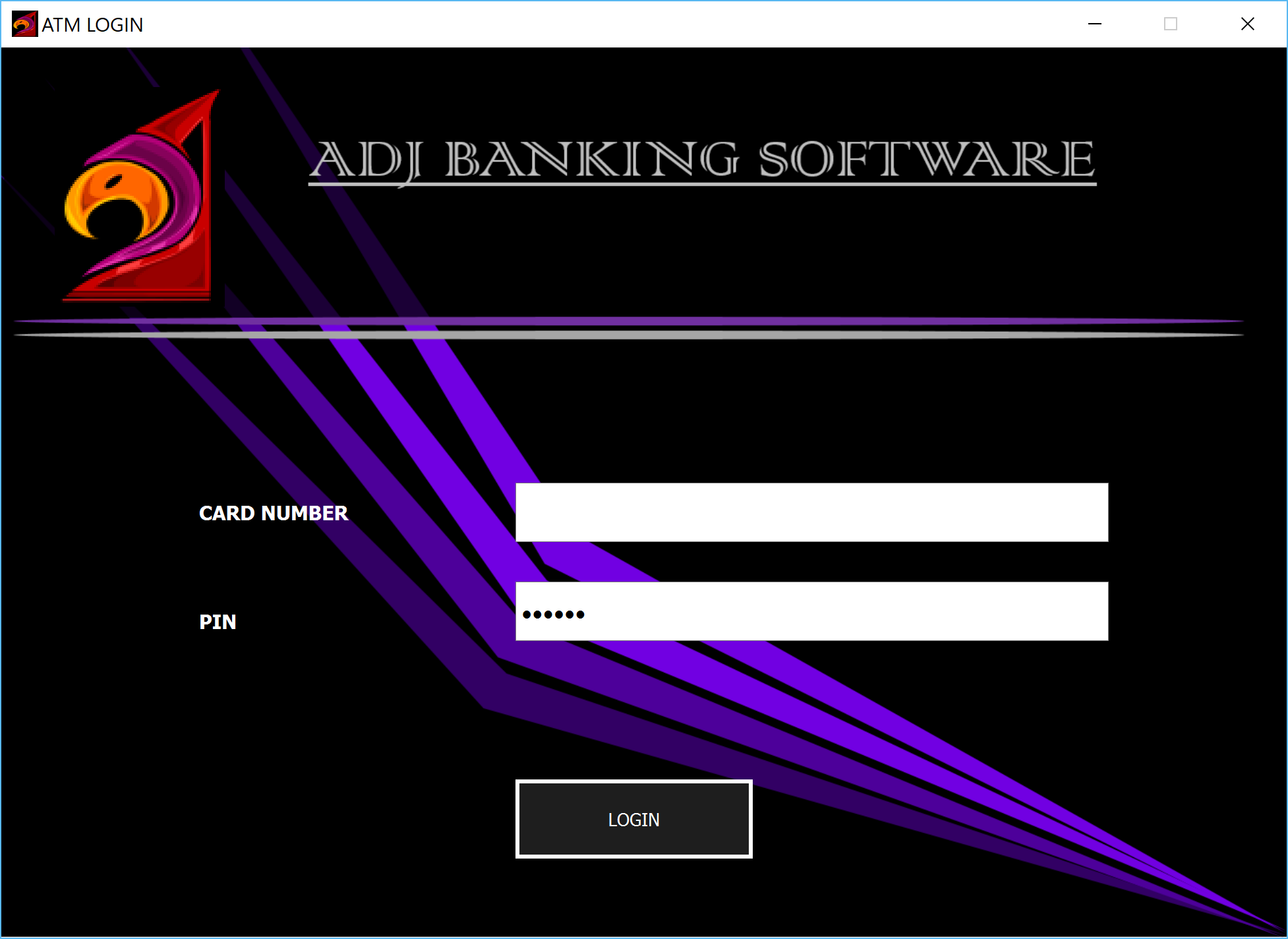


*Close Account (Administrator)*

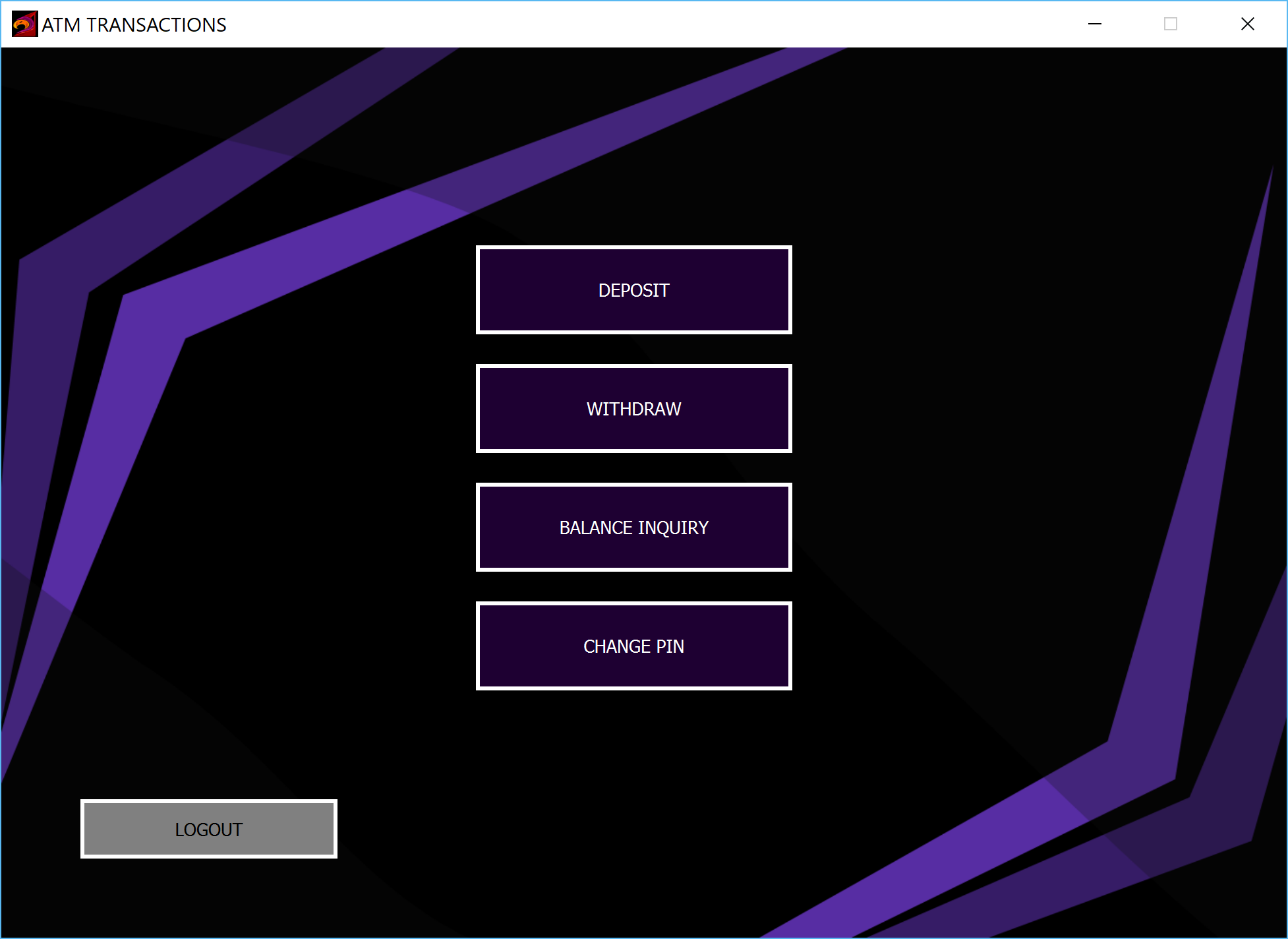


*View Account (Administrator)*

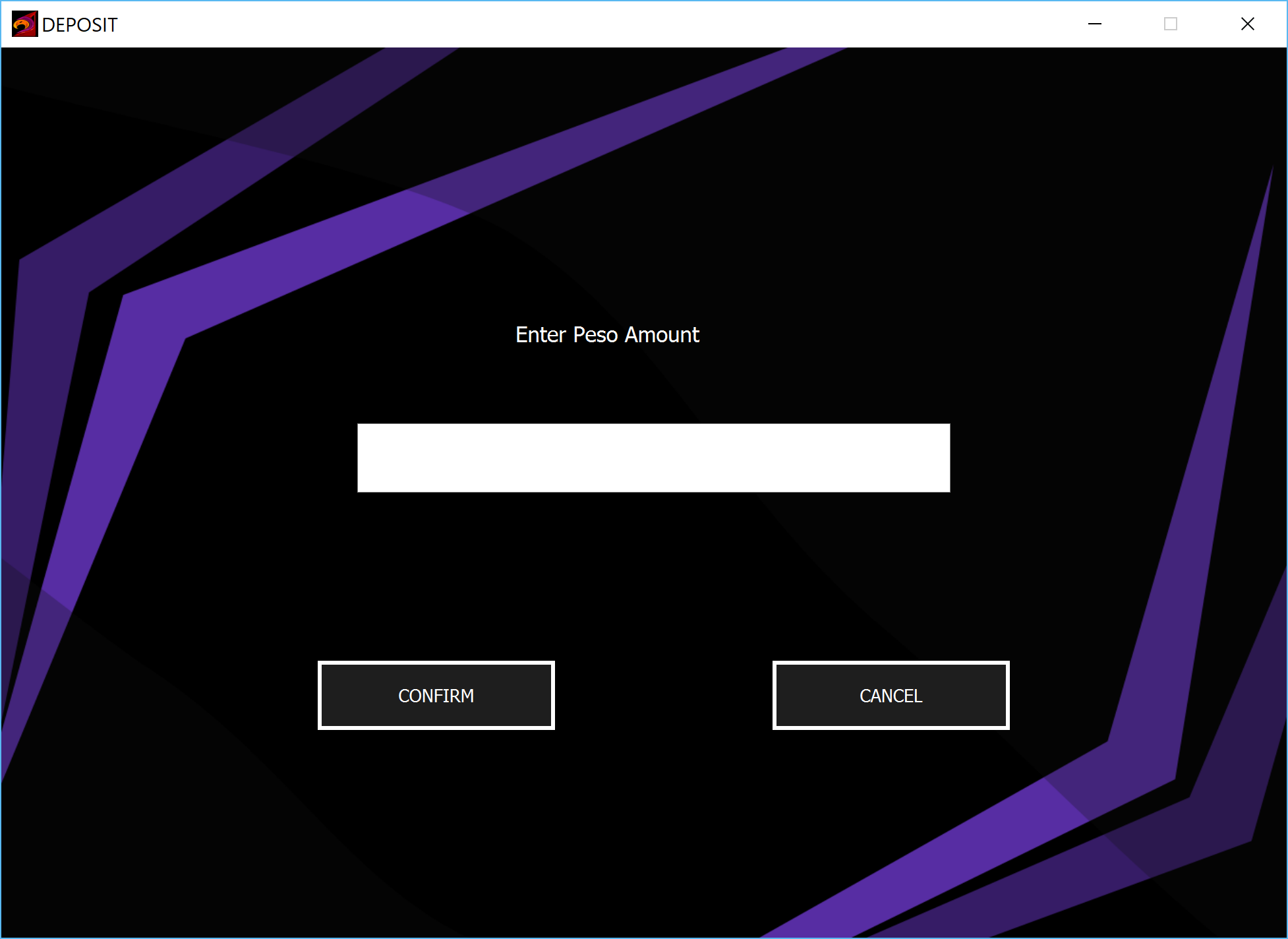
**Client UI**



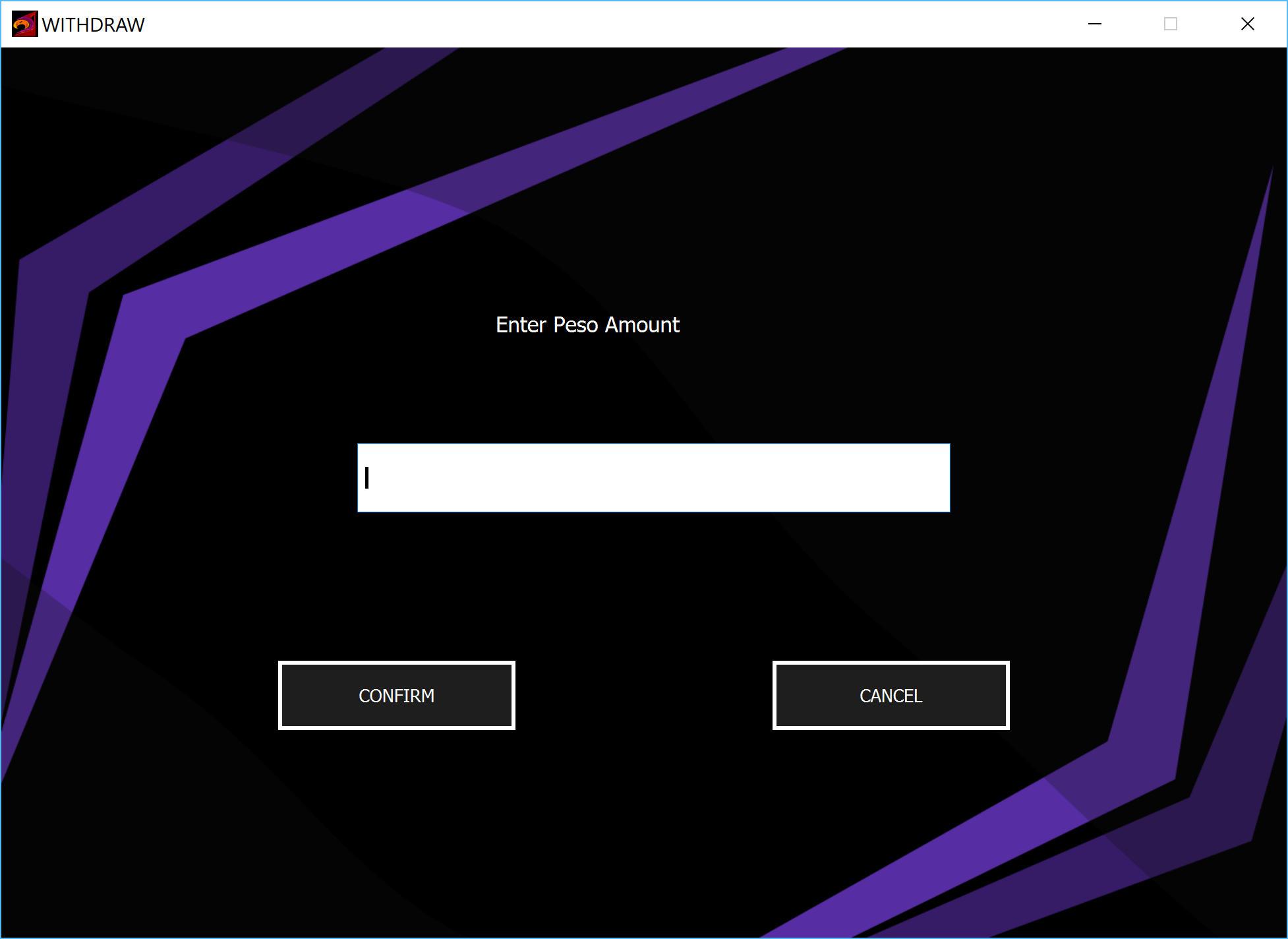
*Login Page (Client)*



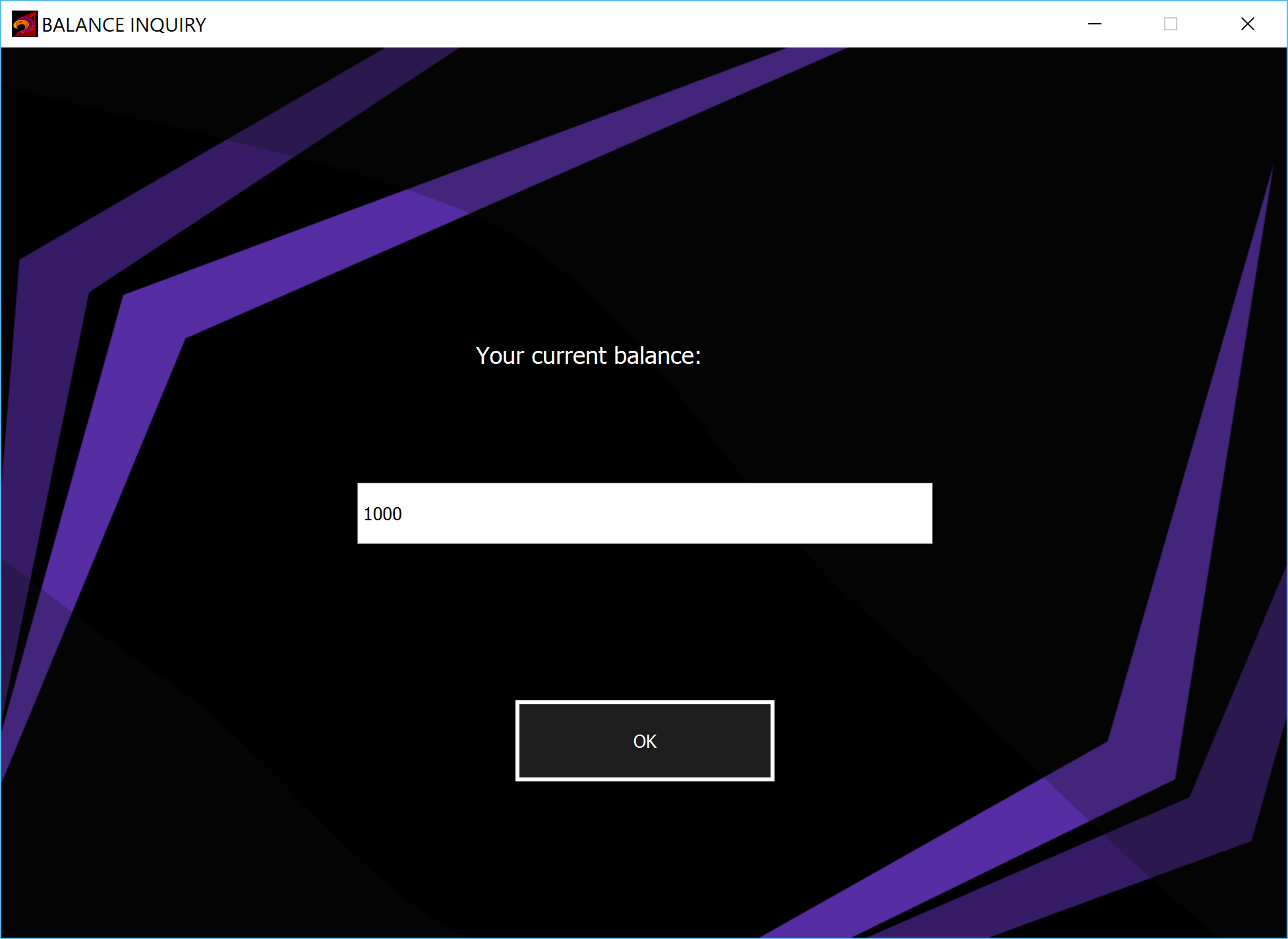
*Transactions (Client)*



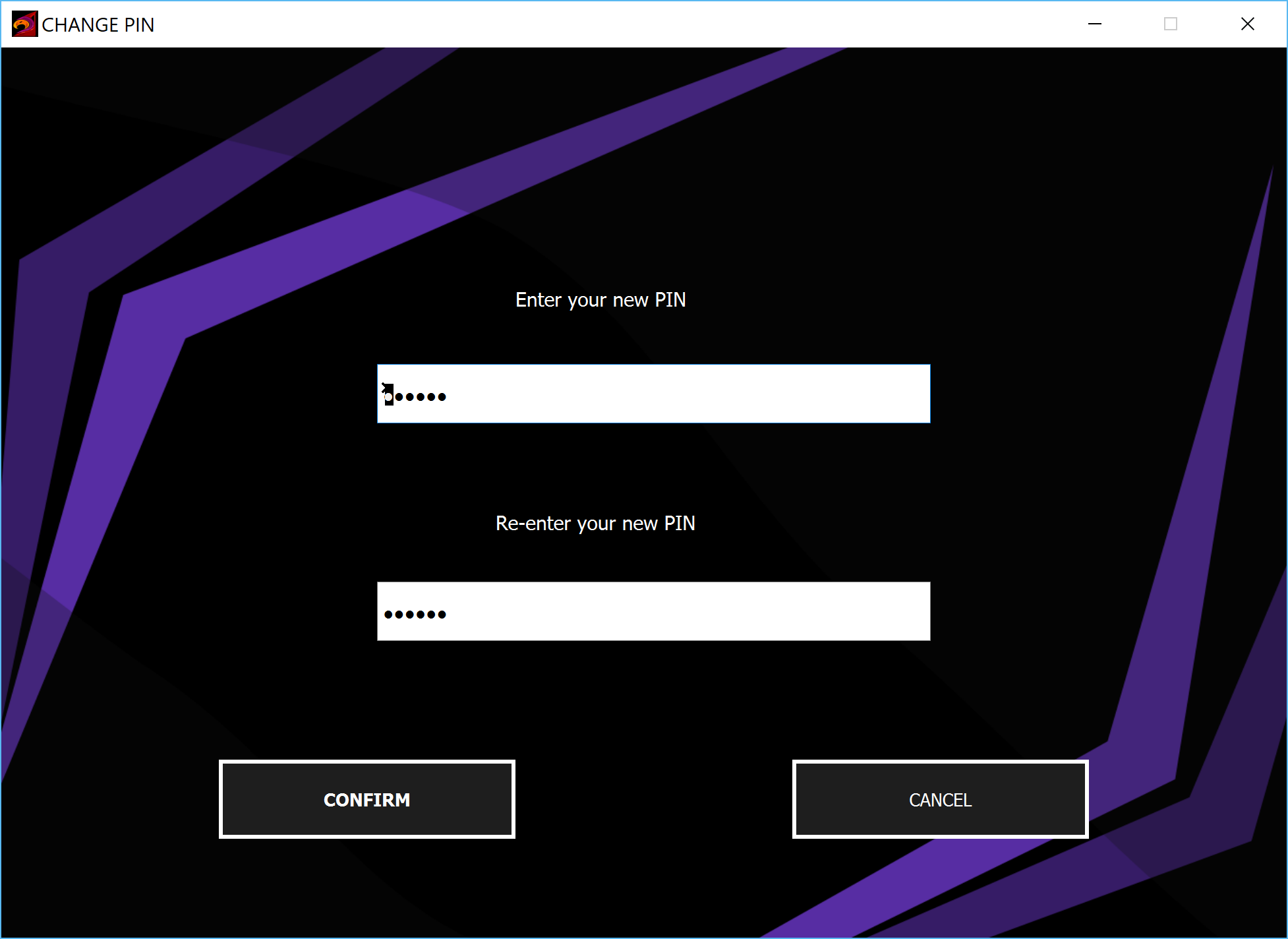
*Deposit (Client)*



*Withdraw (Client)*

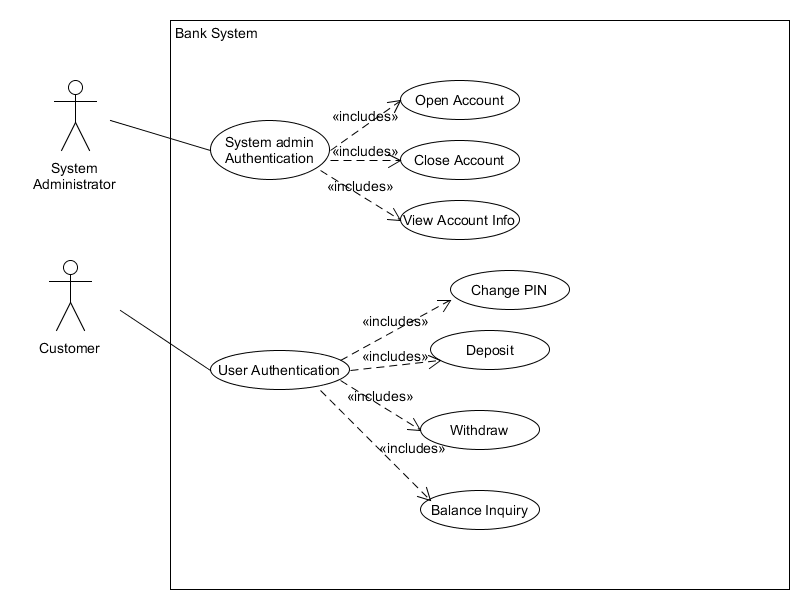


*Balance Inquiry (Client)*



*Change PIN (Client)*

1. **Use Case Diagram**

****

The Use Case diagram above is the representation of the overall banking services of the software. It consists of two actors (system administrator, customer). Each actor own sets of functionalities represented by the ovals (use case). The system administrator is prompt for authentication before being able to open, close and view account info as well as the customer before giving the choices of transaction such as deposit, withdraw, balance inquiry, and change PIN.

1. **Use Case Narrative**

|  |  |
| --- | --- |
| **Use Case ID:** | UC1 |
| **Use Case Name:** | System Admin Authentication |
| **Actors:** | System Administrator |
| **Description:** | This use case describes the event of a System Administrator authenticating the username and password to login and access such system admin functionalities such as opening, closing, and viewing an account. |
| **Precondition:** | The person authenticating must be an administrator |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The System Administrator provides a valid username and password. | 1. The system responds by verifying that all required information has been provided. 2. The system verifies the system admin’s username and password. 3. Once authentication is confirmed, the system displays the homepage for system admin functionalities. |
| **Alternative Flow of Events:** | 1. The system admin has not provided all the information necessary for authentication. System displays incorrect username and/or password and prompts to enter again. 2. The system admin has provided incorrect information. System displays incorrect username and/or password and prompts to enter again. 3. System will not display the homepage when the System admin failed to login. |
| **Postcondition:** | The System Administrator has access for opening, closing, and viewing of account. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC2 |
| **Use Case Name:** | Open Account |
| **Actors:** | System Administrator |
| **Description:** | This use case describes the event of a System Administrator opening an account for the client. |
| **Precondition:** | The System Administrator has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The System Administrator chooses the Open Account option.   3. The System Administrator provides necessary client account information. | 1. The system responds by verifying the chosen option and displays the Open Account session. 2. The system verifies the provided client account information. 3. Once creation of account is successful, the system displays the homepage. |
| **Alternative Flow of Events:** | 1. The system admin has chosen a different option. The system therefore displays a different session. 2. The system admin provided incorrect/invalid information. The system displays error message and prompts to enter again. 3. Account creation is not successful. |
| **Postcondition:** | The System Administrator has created an account. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC3 |
| **Use Case Name:** | Close Account |
| **Actors:** | System Administrator |
| **Description:** | This use case describes the event of a System Administrator closing an account for the client. |
| **Precondition:** | The System Administrator has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The System Administrator chooses the Close Account option.   3. The System Administrator provides the necessary account# of the account to be deleted. | 1. The system responds by verifying the chosen option and displays the Close Account session. 2. The system verifies the provided account#. 3. Once deletion of account is successful, the system displays the homepage. |
| **Alternative Flow of Events:** | 1. The system admin has chosen a different option. The system therefore displays a different session. 2. The system admin provided incorrect/invalid account#. The system displays error message and prompts to enter again. 3. Account deletion is not successful. |
| **Postcondition:** | The System Administrator has deleted an account. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC4 |
| **Use Case Name:** | View Account Info |
| **Actors:** | System Administrator |
| **Description:** | This use case describes the event of a System Administrator viewing an account of the client. |
| **Precondition:** | The System Administrator has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The System Administrator chooses the View Account Info option.   3. The System Administrator provides the necessary account# of the account to be viewed. | 1. The system responds by verifying the chosen option and displays the View Account Info session. 2. The system verifies the provided account#. 3. Once viewing of account info is successful, the system displays the homepage. |
| **Alternative Flow of Events:** | 1. The system admin has chosen a different option. The system therefore displays a different session. 2. The system admin provided incorrect/invalid account#. The system displays error message and prompts to enter again. 3. Account info viewing is not successful. |
| **Postcondition:** | The System Administrator has viewed an account info. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC5 |
| **Use Case Name:** | User Authentication |
| **Actors:** | Customer |
| **Description:** | This use case describes the event of a Customer authenticating the card number and PIN to login and access such ATM transactions such as deposit, withdraw, balance inquiry and change PIN. |
| **Precondition:** | The customer authenticating must have a valid bank account. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The Customer provides a valid card# and PIN. | 1. The system responds by verifying that all required information has been provided. 2. The system verifies the Customer’s card# and PIN. 3. Once authentication is confirmed, the system displays the Transactions page for ATM functionalities. |
| **Alternative Flow of Events:** | 1. The customer has not provided all the information necessary for authentication. System displays incorrect card# and/or PIN and prompts to enter again. 2. The customer has provided incorrect information. System displays incorrect card# and/or PIN and prompts to enter again. 3. System will not display the Transactions page when the customer failed to login. |
| **Postcondition:** | The Customer has access for deposit, withdraw, balance inquiry and change PIN. |

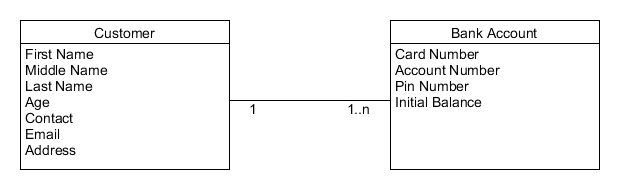
|  |  |
| --- | --- |
| **Use Case ID:** | UC6 |
| **Use Case Name:** | Deposit |
| **Actors:** | Customer |
| **Description:** | This use case describes the event of a Customer performing a Deposit Transaction. |
| **Precondition:** | The Customer has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The Customer chooses the Deposit Transaction. 2. The Customer provides the necessary amount to deposit. | 1. The system responds by verifying the chosen option and displays the Deposit Transaction session. 2. The system verifies the Customer’s deposited amount. 3. Once Deposit Transaction is successful, the system displays the receipt and prompts for another transaction. |
| **Alternative Flow of Events:** | 1. The customer has chosen a different option. The system therefore displays a different session. |
| **Postcondition:** | The Customer has performed a Deposit Transaction. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC7 |
| **Use Case Name:** | Withdraw |
| **Actors:** | Customer |
| **Description:** | This use case describes the event of a Customer performing a Withdraw Transaction. |
| **Precondition:** | The Customer has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The Customer chooses the Withdraw Transaction. 2. The Customer provides the necessary amount to withdraw. | 1. The system responds by verifying the chosen option and displays the Withdraw Transaction session. 2. The system verifies the Customer’s withdrawn amount. 3. Once Withdraw Transaction is successful, the system displays the receipt and prompts for another transaction. |
| **Alternative Flow of Events:** | 1. The customer has chosen a different option. The system therefore displays a different session. 2. The customer provided incorrect/invalid amount. The system displays error message and prompts to enter again. 3. Withdraw Transaction is not successful. |
| **Postcondition:** | The Customer has performed a Withdraw Transaction. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC8 |
| **Use Case Name:** | Balance Inquiry |
| **Actors:** | Customer |
| **Description:** | This use case describes the event of a Customer performing a Balance Inquiry Transaction. |
| **Precondition:** | The Customer has successfully logged in. |
| **Flow of Events:** | |
| **Actor Action:** | **System Response:** |
| 1. The Customer chooses the balance Inquiry Transaction. | 1. The system responds by verifying the chosen option and displays the Balance Inquiry Transaction session. 2. The system displays receipt showing current balance. 3. Once Balance Inquiry Transaction is successful, the system prompts for another transaction. |
| **Alternative Flow of Events:** | 1. The customer has chosen a different option. The system therefore displays a different session. |
| **Postcondition:** | The Customer has performed a Balance Inquiry Transaction. |
| **Use Case ID:** | UC9 |
| **Use Case Name:** | Change PIN |
| **Actors:** | Customer |
| **Description:** | This use case describes the event of a Customer performing a Change PIN session. |
| **Precondition:** | The Customer has successfully logged in. |
| **Flow of Events:** |
| **Actor Action:** | **System Response:** |
| 1. The Customer chooses the Change PIN session. 2. The Customer provides New PIN and re-entered New PIN. | 1. The system responds by verifying the chosen option and displays the Change PIN session. 2. The system verifies the Customer’s New PIN and re-entered New PIN. 3. Once Change PIN session is successful, the system prompts for another transaction. |
| **Alternative Flow of Events:** | 1. The customer has chosen a different option. The system therefore displays a different session. 2. The customer provided incorrect/invalid New PIN. The system displays error message and prompts to enter again. 3. Change PIN session is not successful. |
| **Postcondition:** | The Customer has performed a Change PIN session. |

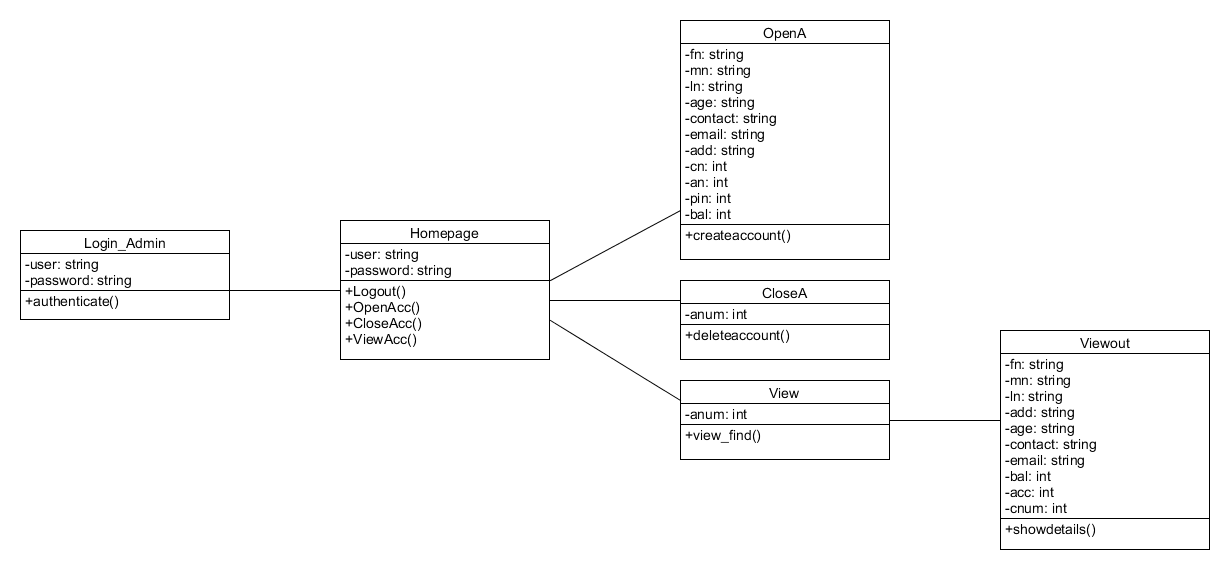
1. **Class Diagram**

**Conceptual Class Diagram**

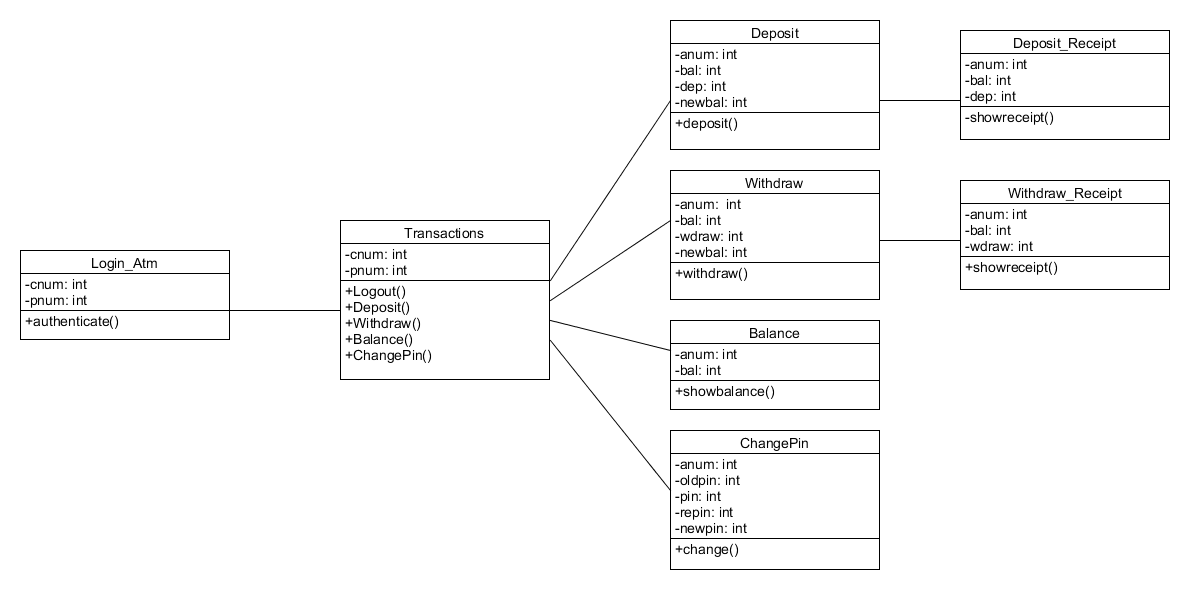
****

The diagram above is in a business level type of representation which focuses on the perspective and language of the subject matter. The conceptual class diagram also has a multiplicity notation between the association of the classes. It indicates the number of instances the Bank account class is linked to one instance of the Customer class. This tells us that one customer will have one or more bank accounts, but each bank account belongs to just one customer.

**Detailed Class Diagram**

****

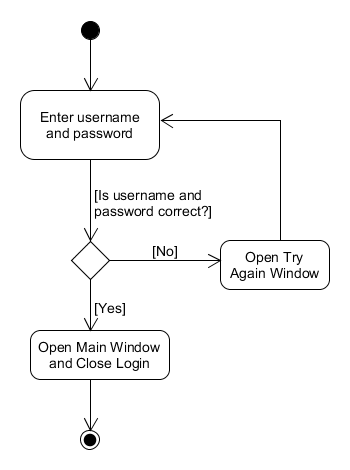
*(Admin)*

****

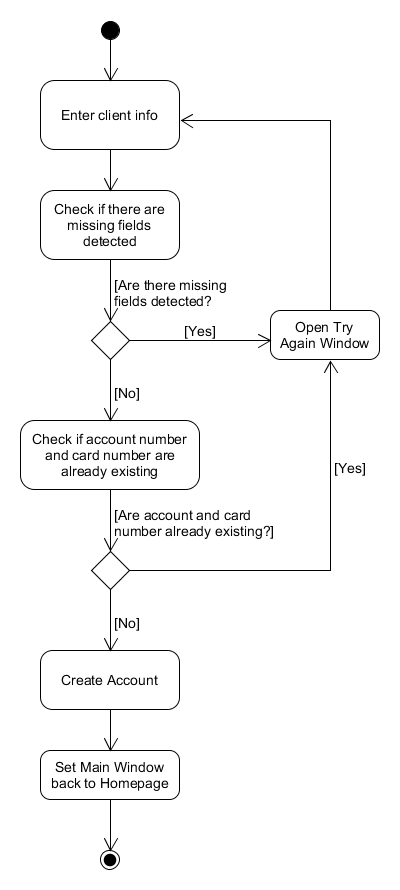
*(Client)*

Detailed as the name suggest is comprised of additional contents in the class diagram that shows all attributes relevant for the problem at hand, the data types of these attributes, and methods. It is the code implementation itself which corresponds to the classes used from the source code. The administrator side has 6 major classes (Login\_Admin, Homepage, OpenA, CloseA, ViewA, and Viewout). While the client side has 8 major classes (Login\_Atm, Transactions, Deposit, Deposit\_Receipt, Withdraw, Withdraw\_Receipt, Balance, and ChangePin). Such classes can be accessed only after authentication from the Login class. Other minor classes that were used for error and prompt messages were not shown for the detailed class diagram.

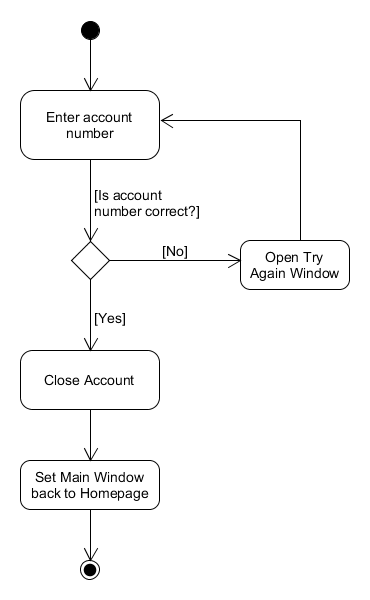
1. **Activity Diagram**



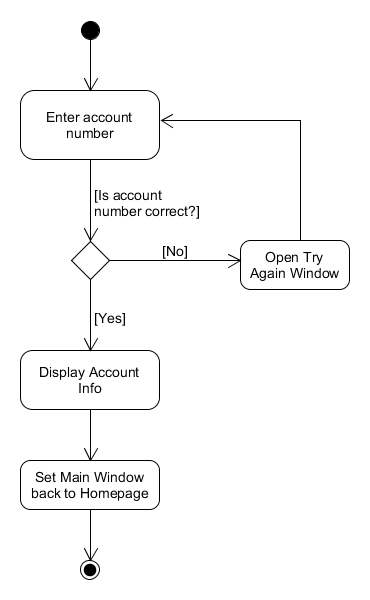
*(Admin Login)*

**

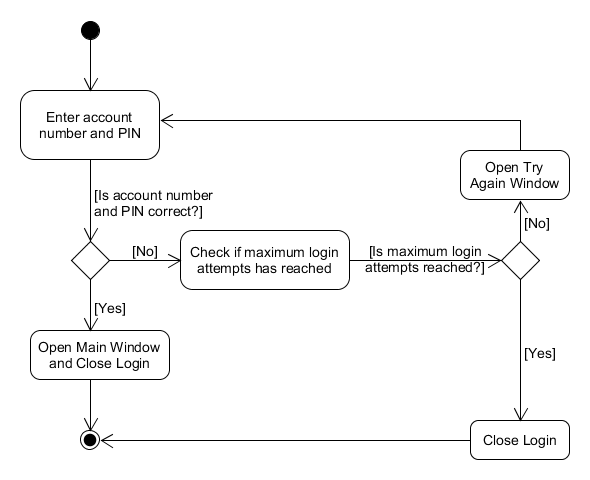
*(Admin Open Account)*

**

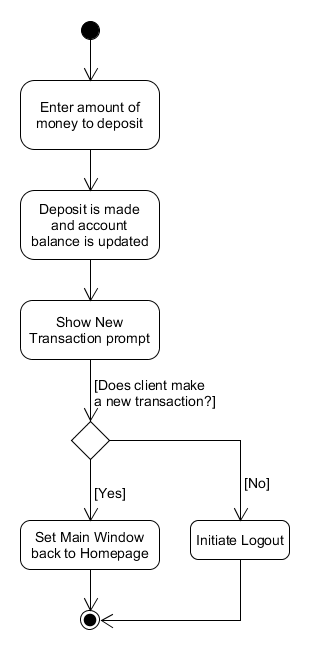
*(Admin Close Account)*

**

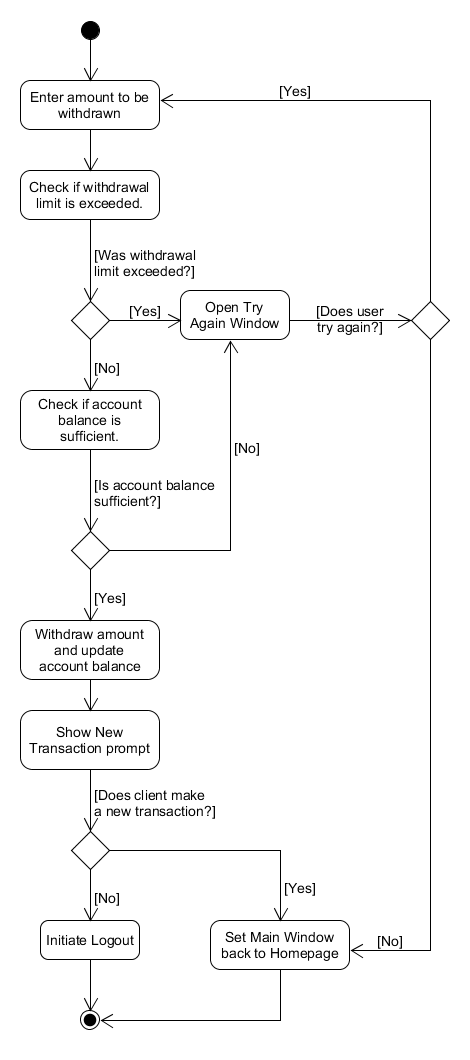
*(Admin View Account Info)*

**

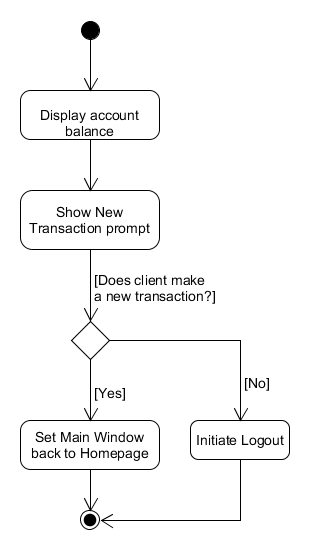
*(Client Login)*

**

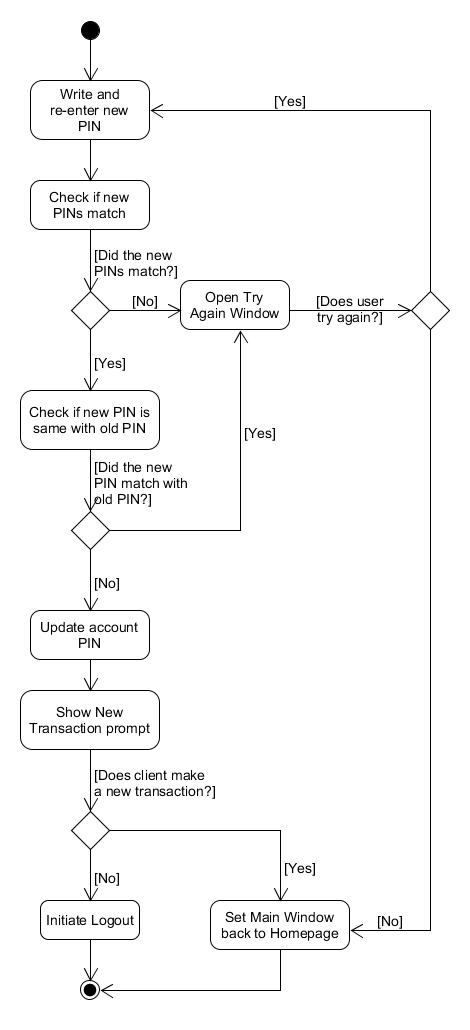
*(Client Deposit)*

**

*(Client Withdraw)*

**

*(Client Balance Inquiry)*

**

*(Client Change PIN)*

1. **Test Case Diagram**

ADMIN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#1 – Admin Login | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_A1 | Authenticate System Administrator | System is on and displaying a login page | Input system admin username and password | Username=”admin”  Password=”admin” | System displays system administrator homepage |
| TC\_A2 | System rejects system admin username and/or password | System is on and displaying a login page | Input invalid system admin username and/or password | Username=”admin123”  Password=”admin123” | System displays an error screen and is ready to start a new session |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#2 – Creating an Account | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_B1 | System allows system admin to choose Open account option | The user must be logged-in as System Administrator | Choose Open account option | Choose “Open Account” | System displays Create account option session |
| TC\_B2 | Create non-existing account with valid initial balance of 1000PhP and above | Create account session is displayed | 1. Type-in user details  2. Type-in account details of non-existing account and valid initial balance | 1.   * First name=”Chris” * Middle name=”Robert” * Last name=”Evans” * Age=”37” * Contact #=”09154848488” * Email=[crevans@gmail.com](mailto:crevans@gmail.com) * Address=”Boston”   2.   * Card Number=”9999999999999999” * Account Number=”999999999999” * PIN Number=”999999” * Initial Balance=”10000” | System displays User account successfully created and goes back to homepage |
| TC\_B3 | Create existing account with valid initial balance of 1000PhP and above | Create account session is displayed | 1. Type-in user details  2. Type-in account details of existing account and valid initial balance | 1.   * First Name=”Chris” * Middle Name=”Robert” * Last Name=”Evans” * Age=”37” * Contact #=”09154848488” * Email=[crevans@gmail.com](mailto:crevans@gmail.com) * Address=”Boston”   2.   * Card Number=”1111111111111111” * Account Number=”111111111111” * PIN Number=”111111” * Initial Balance=”10000” | System displays Account Number and/or Card Number already existed and prompts to enter again |
| TC\_B4 | Create non-existing account with an invalid initial balance less than 1000PhP | Create account session is displayed | 1. Type-in user details  2. Type-in account details of non-existing account and invalid initial balance | 1.   * First name=”Chris” * Middle name=”Robert” * Last name=”Evans” * Age=”37” * Contact #=”09154848488” * Email=[crevans@gmail.com](mailto:crevans@gmail.com) * Address=”Boston”   2.   * Card Number=”9999999999999999” * Account Number=”999999999999” * PIN Number=”999999” * Initial Balance=”500” | System displays that initial balance must be 1000 or above and prompts to enter again |
| TC\_B5 | Create an account with missing field/s | Create account session is displayed | Type-in missing fields on user and account details | First Name=” “  Middle Name=” “  Last Name=” “  Age=” “  Contact #=” “  Email=” “  Address=” “  Card Number=” “  Account Number=” “  PIN Number=” “  Initial Balance=” “ | System displays there is/are missing field/s detected and prompts to enter again |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#3 – Closing an Account | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_C1 | System allows system admin to choose Close account option | The user must be logged-in as System Administrator | Choose Close account option | Choose “Close Account” | System displays Close account option session |
| TC\_C2 | Close existing account | Close account session is displayed | Enter valid account# | Account Number=”111111111111” | System displays User account successfully closed and goes back to homepage |
| TC\_C3 | Close non-existing account | Close account session is displayed | Enter invalid account# | Account Number=”444444444444” | System displays account number doesn’t exist and prompts to enter again |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#4 – View Account Info | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_D1 | System allows system admin to choose View Account Info | The user must be logged-in as System Administrator | Choose View Account Info option | Choose “View Account Info” | System displays View Account Info session |
| TC\_D2 | View existing user account info | View Account Info session is displayed | Enter account# | Account Number=”111111111111” | System displays User account info successfully and goes back to homepage |
| TC\_D3 | View non-existing user account info | View Account Info session is displayed | Enter invalid account# | Account Number=”444444444444” | System displays account number doesn’t exist and prompts to enter again |

CLIENT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#5 – ATM Transaction (Login) | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_E1 | System authenticates the user | System is on and displaying a login page | Insert card (user card#) and PIN | Card Number=”1111111111111111”  PIN=”111111” | System displays a menu of transaction types |
| TC\_E2 | System rejects invalid user card (card#) and/or PIN | System is on and displaying a login page | Insert invalid card (user card#) and/or PIN | Card Number=”4444444444444444”  PIN=”444444” | System displays invalid Card Number and/or PIN and prompts to enter again |
| TC\_E3 | Customer exceeds max number of invalid PIN tries (3 entries) and system aborts session | System is on and displaying a login page | Enter 3rd try ofinvalid card (user card#) and/or PIN | Card Number=”4444444444444444”  PIN=”444444” | System displays maximum number of login attempts has exceeded and session is terminated |

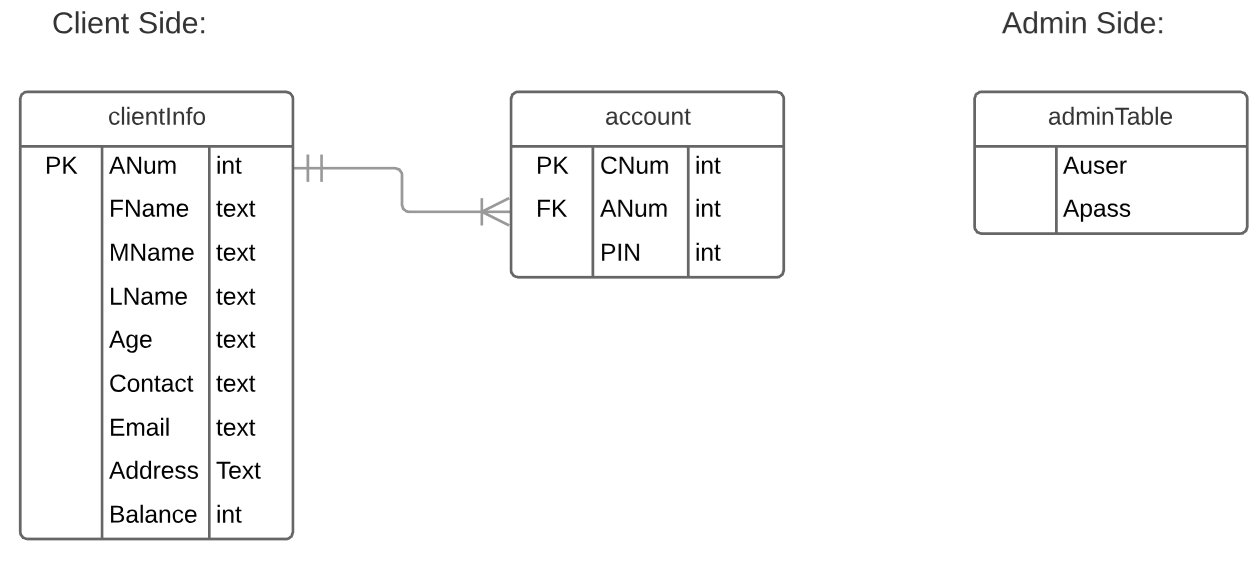
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#6 – ATM Transaction (Deposit) | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_F1 | System allows customer to perform a Deposit transaction | Menu of transaction types is being displayed | Choose Deposit transaction | Choose “DEPOSIT” | System displays the Deposit transaction session |
| TC\_F2 | System allows customer to enter a peso amount to deposit | System is displaying a request for the customer to type a peso amount | Enter a peso amount | Amount=”5000” | System prints receipt showing deposited amount and updated balance |
| TC\_F3 | System automatically sets missing field on amount to 0PhP | System is displaying a request for the customer to type a peso amount | Enter missing field | Amount=” “ | System prints receipt showing deposited amount of 0PhP and updated balance |
| TC\_F4 | System allows customer to cancel Deposit Transaction | System is displaying a request for the customer to type a peso amount | Choose cancel option | Choose “CANCEL” | System displays a menu of transaction types |
| TC\_F5 | System allows multiple transactions in one session | System is asking whether customer wants another transaction | Choose yes | Choose “YES” | System displays a menu of transaction types |
| TC\_F6 | Session ends when customer chooses not to do another transaction | System is asking whether customer wants another transaction | Choose no | Choose “NO” | System terminates and is ready to start a new session |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#7 – ATM Transaction (Withdrawal) | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_G1 | System allows customer to perform a Withdrawal transaction | Menu of transaction types is being displayed | Choose Withdrawal transaction | Choose “WITHDRAW” | System displays the Withdrawal transaction session |
| TC\_G2 | System verifies that customer's balance is sufficient to fulfill the request | System is displaying a request for the customer to type a peso amount | Choose an amount that is not greater than the account balance | Amount=”500” | System prints receipt showing withdrawn amount and updated balance |
| TC\_G3 | System verifies that customer's balance is insufficient to fulfill the request | System is displaying a request for the customer to type a peso amount | Choose an amount that is greater than the account balance | Amount=”2000” | System displays Insufficient Balance error and prompts to enter again |
| TC\_G4 | System verifies that customer's withdrawal amount is greater than per transaction limit amount (10000Php) | System is displaying a request for the customer to type a peso amount | Choose an amount that is greater than per transaction limit amount | Amount=”11000” | System displays withdrawal limit per transaction has exceeded and prompts to enter again |
| TC\_G5 | System automatically sets missing field on amount to 0PhP | System is displaying a request for the customer to type a peso amount | Enter missing field | Amount=” “ | System prints receipt showing withdrawn amount of 0PhP and updated balance |
| TC\_G6 | System allows customer to cancel Withdrawal Transaction | System is displaying a request for the customer to type a peso amount | Choose cancel option | Choose “CANCEL” | System displays a menu of transaction types |
| TC\_G7 | System allows multiple transactions in one session | System is asking whether customer wants another transaction | Choose yes | Choose “YES” | System displays a menu of transaction types |
| TC\_G8 | Session ends when customer chooses not to do another transaction | System is asking whether customer wants another transaction | Choose no | Choose “NO” | System terminates and is ready to start a new session |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#8 – ATM Transaction (Balance Inquiry) | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_H1 | System allows customer to perform a Balance Inquiry transaction | Menu of transaction types is being displayed | Choose Balance Inquiry transaction | Choose “BALANCE INQUIRY” | System prints receipt showing balance |
| TC\_H2 | System allows multiple transactions in one session | System is asking whether customer wants another transaction | Choose yes | Choose “YES” | System displays a menu of transaction types |
| TC\_H3 | Session ends when customer chooses not to do another transaction | System is asking whether customer wants another transaction | Choose no | Choose “NO” | System terminates and is ready to start a new session |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC#9 – ATM Transaction (Change PIN) | | | | | |
| TC\_ID# | Test Objective | Precondition | Steps | Test Data | Expected Result |
| TC\_I1 | System allows customer to perform Change PIN | Menu of transaction types is being displayed | Choose Change PIN | Choose “CHANGE PIN” | System displays the Change PIN session |
| TC\_I2 | System verifies that customer's new PIN is valid | System is displaying a request for the customer to enter new PIN | Enter valid new PIN | New PIN=”222222”  Re-enter New PIN=”222222” | System prompts the customer that current PIN is successfully changed |
| TC\_I3 | System verifies that customer's new PIN is not the same with re-entered PIN | System is displaying a request for the customer to enter new PIN | Enter valid new PIN different with re-entered PIN | New PIN=”222222”  Re-enter New PIN=”333333” | System displays New PIN doesn’t match Re-entered PIN and prompts to enter again |
| TC\_I4 | System verifies that customer's new PIN is the same as old PIN | System is displaying a request for the customer to enter new PIN | Enter invalid or old PIN | New PIN=”111111”  Re-enter New PIN=”111111” | System displays Invalid PIN where New PIN is the same as old PIN and prompts to enter again |
| TC\_I5 | System verifies that there is/are missing field/s | System is displaying a request for the customer to enter new PIN | Enter missing fields | New PIN=” “  Re-enter New PIN=” “ | System displays missing field detected and prompts to enter again |
| TC\_I6 | System allows customer to cancel Change PIN session | System is displaying a request for the customer to enter new PIN | Choose cancel option | Choose “CANCEL” | System displays a menu of transaction types |
| TC\_I7 | System allows multiple transactions in one session | System is asking whether customer wants another transaction | Choose yes | Choose “YES” | System displays a menu of transaction types |
| TC\_I8 | Session ends when customer chooses not to do another transaction | System is asking whether customer wants another transaction | Choose no | Choose “NO” | System terminates and is ready to start a new session |

1. **ERD**



1. **Source Codes**
2. **Sprint 1**

* main.py
* res.py
* admin\_log.py
* admin\_hp.py
* admin\_open.py
* admin\_close.py
* admin\_view.py
* admin\_viewout.py
* admin\_prompt.py
* admin\_prompt2.py
* admin\_error.py
* admin\_error2.py

1. **Sprint 2**

**Admin**

* SPLASH\_screen.py
* admin\_close.py
* admin\_error.py
* admin\_error2.py
* admin\_error3.py
* admin\_error4.py
* admin\_hp.py
* admin\_log.py
* admin\_open.py
* admin\_prompt.py
* admin\_prompt2.py
* admin\_view.py
* admin\_viewout.py
* data\_handler.py
* main.py
* reso.py
* test\_main.py

**Client**

* Atm\_Balance.py
* Atm\_ChangePin.py
* Atm\_ChangePin\_Blank.py
* Atm\_ChangePin\_Reenter.py
* Atm\_ChangePin\_Success.py
* Atm\_Deposit.py
* Atm\_Deposit\_Receipt.py
* Atm\_Login.py
* Atm\_Login\_Error.py
* Atm\_Login\_Error\_Max.py
* Atm\_Main.py
* Atm\_NewTransaction.py
* Atm\_Transactions.py
* Atm\_Withdraw.py
* Atm\_Withdraw\_Error\_Exceed.py
* Atm\_Withdraw\_Error\_Insuff.py
* Atm\_Withdraw\_Receipt.py
* SPLASH\_screen.py
* client\_data\_handles.py
* res1.py
* reso.py

1. **Sprint 3**

**BLL**

**Admin**

* Admin\_Close.py
* Admin\_Error.py
* Admin\_Error2.py
* Admin\_Error3.py
* Admin\_Error4.py
* Admin\_Error5.py
* Admin\_HP.py
* Admin\_Log.py
* Admin\_Main.py
* Admin\_Open.py
* Admin\_Prompt.py
* Admin\_Prompt2.py
* Admin\_View.py
* Admin\_Viewout.py

**Client**

* atm\_balance.py
* atm\_changepin.py
* atm\_changepin\_blank.py
* atm\_changepin\_error.py
* atm\_changepin\_reenter.py
* atm\_changepin\_success.py
* atm\_deposit.py
* atm\_deposit\_receipt.py
* atm\_login.py
* atm\_login\_error.py
* atm\_login\_error\_max.py
* atm\_newtransaction.py
* atm\_transactions.py
* atm\_withdraw.py
* atm\_withdraw\_error\_exceed.py
* atm\_withdraw\_error\_insuff.py
* atm\_withdraw\_receipt.py
* Client\_Main.py

**DAL**

* Data\_handler.py

**UI**

* admin\_close.py
* admin\_error.py
* admin\_error2.py
* admin\_error3.py
* admin\_error4.py
* admin\_error5.py
* admin\_hp.py
* admin\_log.py
* admin\_open.py
* admin\_prompt.py
* admin\_prompt2.py
* admin\_view.py
* admin\_viewout.py
* Atm\_Balance.py
* Atm\_ChangePin.py
* Atm\_ChangePin\_Blank.py
* Atm\_ChangePin\_Error.py
* Atm\_ChangePin\_Reenter.py
* Atm\_ChangePin\_Success.py
* Atm\_Deposit.py
* Atm\_Deposit\_Receipt.py
* Atm\_Login.py
* Atm\_Login\_Error.py
* Atm\_Login\_Error\_Max.py
* Atm\_NewTransaction.py
* Atm\_Transactions.py
* Atm\_Withdraw.py
* Atm\_Withdraw\_Error\_Exceed.py
* Atm\_Withdraw\_Error\_Insuff.py
* Atm\_Withdraw\_Receipt.py
* res.py
* res1.py
* reso.py
* resource.qrc
* SPLASH\_screen.py

**Testing**

* unit\_test.py

1. **Burndown Chart**
2. **Errors Encountered**

|  |  |
| --- | --- |
| **Date** | **Errors** |
| 9/21/18 | No errors encountered yet, the task was recently distributed. |
| 9/22/18 | Two account types (Checking and Savings) were initially mentioned but was corrected in the day 2 of sprint 1 to be savings account only.  The mockups lack display for error messages.  The team noticed that it is unnecessary to add the pin number in order for the admin to authenticate before being able to close an account. |
| 9/23/18 | Some of the documents were only uploaded in the github and not on the file exchange. |
| 9/24/18 | Accessing data from the csv file took several trial and errors but was accomplished later on. |
| 9/25/18 | The program was not running on the cmd when the created database was incorporated with the program. |
| 9/26/18 | There is no button that signifies the confirm action when the user is asked to try again which leaves the user to rely on the close button. The burn down chart must be set in such a way that the last connecting line in the graph falls into zero. |
| 9/27/18 | There is a problem with the program flow linking on the deposit transaction after a successful login. |
| 9/28/18 | Allowing the ESC button to be an alternative for the closing of error displayed message is unsuccessful. |
| 9/29/18 | Few trial and errors were taken in order to correct the way on how the deposit and withdraw function should work such as adding and subtracting respectively and updating it into the csv file.  Changing the background using the designer platform chosen takes a bit of work and needed caution, otherwise, the background design would not appear in the window. |
| 9/30/18 | There was an error noticed on fetching the current pin of the client as well as overwriting it in the csv file.  Due to the lack of information on the internet with the designer that is currently being utilized by the team regarding the splash screen makes the member unable to decide whether to include the mentioned feature. |
| 10/1/18 | Errors encountered were linking all client functions with one another. Some functions were not able to result with the desired function linked them.  An error with regards to the deposit and withdraw functions were noticed after a running the program. It crashes when there is no amount placed by the user on the text field. |
| 10/2/18 | There were times when implementing the program unit testing that there is an incorrect usage of assert methods to check for and report failures.  There was a piece of code related on the splash screen which ends the whole program. |
| 10/3/18 | some parts in the product backlogs' priority levels were incorrect |
| 10/4/18 | One of the problems encountered was analysing and identifying on how to include the client part with the existing detailed class diagram which only involved the admin part. A bit of a hard time deciding on how to show the relationship and connect these two sides, the client and admin side, into one detailed class diagram. |
| 10/7/18 | Errors encountered regarding the separation of classes into different python files were quite a lot because it was a domino effect of revising each classes to link them with one another since from the first setup we had. Some had missing file imports and unknown modules or attributes along the way. |
| 10/8/18 | The update on the product backlog gave a confusion since the updated priority given for each tasks doesn’t make sense to the sprint assigned to it. The sprint number was removed from the table. |

1. **Conclusion**

The team was able to create the ADJ banking software using python programming language as required by the product owner Engr. Dionis Padilla within the time duration (3 sprints). The graph on the Burndown chart indicates the team’s performance on the last 3 sprints. It can be observed that the time was maximized and took only few efforts on sprint 3. The group were able to implement the functionalities specified by Engr. Padilla for the administrator (open, close, and view account) and client (Deposit, Withdraw, Balance Inquiry, Change Pin). SQLite was used for the data storage purposes instead of opting for a simple csv file.

Concerns about the presence of the unit testing on the feedback done on sprint 1 were addressed. The team agreed as well on other suggestions such as enabling function keys, placing limit on the initial balance and placing a maximum number of attempts. The team believed that creating original designs for the logo, splash screen and background for the graphical user interfaces would make the appearance look more marketable.

1. **Recommendation**

The group recommends users to provide feedbacks in order to improve the usability of the ADJ Banking Software and to fix bugs that may arise upon utilization.