Operator Use Cases:

Use Case ID: O001

Use Case Name: Deposit cash

Relevant Requirements: 3.1.3.5

Primary Actor:

- Operator Module

Pre-condition:

- The operator is authenticated.

- Operator is authorized to operate on that account.

Post-condition:

- The cash amount is deposited to the account in which the operator is authorized.

Basic Flow:

1. The operator log in to the account.

2. The operator requests to deposit some balance amount.

3. The client module uses the message module to send the request.

4. The server module receives the request.

5. The server module invoke correct account's methods to change its balance.

6. The interaction is logged.

Alternate Flows:

1. The operator tries to deposit a non-supported input, non-positive numeric inputs, or decided undesirable amounts.

2. Client module intercept input.

3. Client module deny action.

Exceptions:

- The server may fail to receive or fulfil the request due to reasons such as data corruption or connection issues.

Related Use Cases:

- Log in, Withdraw cash

---

Use Case ID: O002

Use Case Name: Withdraw cash

Relevant Requirements: 3.1.2.4, 3.1.3.1, 3.1.3.6

Primary Actor:

- Operator Module

Pre-condition:

- The operator is authenticated.

- Operator is authorized to operate on that account.

Post-condition:

- The retrieved cash must be subtracted from the account balance.

Basic Flow:

1. The operator log in to the account.

2. The operator requests to withdraw some balance amount.

3. The client module uses the message module to send the request.

4. The server module receives the request.

5. The server module invoke correct account's methods to withdraw.

6. The interaction is logged.

Alternate Flows:

1. The operator tries to deposit a non-supported input, non-positive numeric inputs, or decided undesirable amounts.

2. Client module intercept input.

3. Client module deny action.

Exceptions:

- The server may fail to receive or fulfil the request due to reasons such as data corruption or connection issues.

- If balance is lower than withdraw request, server will deny request.

Related Use Cases:

- Deposit cash

---

Use Case ID: O003

Use Case Name: Transferring request

Relevant Requirements: 3.3.9, 3.1.2.2

- Operator can request transfer fund between accounts.

Primary Actor:

- Operator Module

Pre-condition:

- Authenticated Operator access on Account is currently authorized.

Post-condition:

- The transfer is finalized, balance on both accounts reflect the change.

Basic Flow:

1. The operator initiates a transfer money request.

2. The operator is prompted to enter the account they wish to transfer money from.

3. The operator enters the desired amount to transfer.

4. Client call message to send request to Server using that data.

5. The Server receives request.

6. The Server call Account's transfer method.

7. Server fulfill request to Client.

8. Client invoke correct GUI showing result.

Alternate Flows:

1. Account's transfer method deny due to lack balance or minimum balance unfulfilled.

2. Server deny request.

3. Client trigger correct response.

4. Events logged.

Exceptions:

- The operator has met the account transfer limit.

- Account lack balance

- Account locked

- Minimum amount not met.

Related Use Cases:

Account transfer

---

Client and Server section:

Use Case ID: CS001

Use Case Name: Retrieve message from server

Relevant Requirements: 3.1.4.1, 3.1.4.2, 3.1.

Primary Actor:

- Message Module

Pre-condition:

- The information requested is available in the bank account.

Post-condition:

- The information asked for will be made available to the client.

Basic Flow:

1. The client makes a request using the message module.

2. The server processes the request and sends a response to the client.

3. The client receives the response and uses it as needed.

Alternate Flows:

Exceptions:

- The server is down.

- The server doesn't exist.

Related Use Cases:

- Retrieve cash, deposit cash, M001

---

Use Case ID: CS002

Use Case Name: Send message to server

Relevant Requirements: 3.1.4.1, 3.1.4.2, 3.1.4.3, 3.1.4.5.

Primary Actor:

- Message Module

Pre-condition:

- The message to send has been initialized with proper information.

Post-condition:

- The message is sent to the server.

Basic Flow:

1. The operator makes a request (e.g., deposit cash).

2. The request is packed and sent via the message module.

3. The request is received by the server module and unpacked.

Alternate Flows:

- None

Exceptions:

- This module is likely entirely private with the sole purpose of moving back and forth between the server and the client.

Related Use Cases:

- Retrieve cash, deposit cash, M002

---

Use Case ID: CS001

Use Case Name: Log in

Relevant Requirements: 3.1.2.1, 3.1.2.2, 3.1.2.3

Primary Actor:

- Client Module

Pre-condition:

- The operator is not authenticated.

Post-condition:

- The operator is authenticated.

Basic Flow:

1. Client show operator a log in interface.

2. The operator enters their authenticator.

3. Client invoke message to send request to Server using authenticator.

4. Server pass authenticator to Operator modules to authenticate.

5. Once success, a read-only version of the authenticated Operator is return to Client, include a reference table of authorized accounts.

6. The operator is authenticated.

7. Show user GUI

Alternate Flows:

1. The authentication failed.

2. Server denies request.

3. Client invoke correct GUI response.

4. Events are logged

1. entered authenticator belongs to superuser

2-6 as basic Flow

7. show superuser GUI

Exceptions:

- If operator doesn't exist, response same as authentication failed.

Related Use Cases:

- Account authorization

---

Use Case ID: CS002

Use Case Name: Account authorization

Relevant Requirements: 3.1.2.4.6

Primary Actor:

- Operator module

Pre-condition:

- The operator has authenticated.

- Client has received the table of reference accounts

Post-condition:

- The operator is authenticated.

Basic Flow:

1. Operator select interface to access account.

2. Operator enter desired account ID.

3. Client check with account authorization reference table.

4. Events are logged.

Alternate Flows:

1. Entered account ID did not exist in table.

2. Client deny request.

3. Client invoke GUI response.

4. Events are logged.

Exceptions:

- Table empty, invoke correct client response.

Related Use Cases:

- Log in

---

---

Messaging Module Use Cases:

Use Case ID: M001

Use Case Name: Sending Message

Relevant Requirements: 3.1.4.2, 3.1.4.3, 3.1.4.4, 3.1.4.5, 3.1.4.5.1, 3.1.4.5.4

Primary Actor: Sender

Pre-condition:

Sender is called and provided data, current address, receiving address are provided (3.1.4.3)

Post-condition:

Package is sent

Basic Flow:

1. Sender is called and fork a Sender child

2. Child call Processor to encapsulate a package

3. Processor return package ready to be sent

4. Child read receiving address on package and make a connection

5. Once connection is made, package is sent

6. Kill Child

Alternate Flows:

Exceptions:

1. If Sender is evoked with lacking arguments, throw error according to UI

2. The server is not connected to the internet.

3. The client is not connected to the internet.

Related Use Cases: Receiving Message, Encapsulate data

---

Use Case ID: M002

Use Case Name: Receiving Message

Relevant Requirements: 3.1.4.2, 3.1.4.3, 3.1.4.4, 3.1.4.6, 3.1.4.6.1, 3.1.4.6.3

Primary Actor: Receiver

Pre-condition:

Receiver sit on designated network and listen for connections

Post-condition:

package received and decapsulated

Basic Flow:

1. Receiver fork a child upon handling an incoming package on listened port

2. Assign the child to handle that request

3. Child receive package from connection

4. Child call Processor to decapsulate the package

5. Child returns encapsulated data

6. Kill Child.

Alternate Flows:

Exceptions:

1. The server is not connected to the internet.

2. The client is not connected to the internet.

3. The server returns an unrecognized response.

Related Use Cases: Sending Message, Decapsulate data

Use Case ID: M003

Use Case Name: Encapsulate data

Relevant Requirements:

Primary Actor: Processor

Pre-condition: Processor get called with data

Post-condition: Processor returns a package

Basic Flow:

1. Processor get called with data

2. Processor fork a child to process data

3. Child create a package

4. Kill Child, return package

Alternate Flows:

Exceptions:

Related Use Cases: Sending Message

Use Case ID: M004

Use Case Name: Decapsulate data

Relevant Requirements: 3.1.4.4, 3.1.4.5.1.

Primary Actor: Processor

Pre-condition: Processor get called with a package

Post-condition: Processor returns encapsulated data

Basic Flow:

1. Processor get called with package

2. Processor fork a child to process package

3. Child parse and tokenize package

4. Kill Child, return data

Alternate Flows:

Exceptions:

Related Use Cases: M002

---

Account use cases:

Use Case ID: A001

Use Case Name: Account transfer

Relevant Requirements:

- received a request of transferring

Primary Actor:

- Account module

Pre-condition:

- Account is accessing by authorized operator.

- Server invoke account transfer method.

Post-condition:

- Account transferred, balance update.

Basic Flow:

1. Checking current balance against transfer amount.

2. Checking transfer constraint.

3. Account initialized correct destination accounts.

4. Update balance from both accounts to reflect change.

5. Update new balance to files.

Alternate Flows:

Exceptions:

- Account don’t exist, invoke correct client response.

- Transfer constraint invoked, abort

Related Use Cases:

- Transferring request

---

Use Case ID: A002

Use Case Name: Creating an account

Relevant Requirements:

- A user wants to create an account to be able to store

their money in either a checking or savings.

Primary Actor:

- Client Module

- Server Module

Pre-condition:

- The user is in position where it is beneficial to create

an account.

Post-condition:

- The user has now made an account and has gained access to

further pursue a bank account.

Basic Flow:

1. The user initiates an account creation request.

2. The user provides necessary personal information.

3. The user's identification begins to be verified.

4. The superuser verifies the user's information.

5. The user now has an account.

Alternate Flows:

1. The user enters any type of wrong information during

the process of wanting to create an account. So, the user is

prompted to make changes to the incorrect information.

2. The user's request to create an account was rejected.

Exceptions:

1. The user provides invalid personal information.

2. The user's ID and password are already taken so the user

must think of a different ID and password.

3. The ID and or password don't meet bank's security requirements.

4. The user doesn't meet the bank's requirements in order to

create an account. (Ex: Age).

Related Use Cases:

Logging in, get authorized, send message to server, retrieve message

from server

---

Use Case ID: A003

Use Case Name: Deleting an account

Relevant Requirements:

- A user's account has reached a point in which it is not needed anymore.

Primary Actor:

- Client Module

- Server Module

Pre-condition:

- The user's funds have been removed from the account and

has been verified for deletion of account.

Post-condition:

- The user's account has been deleted.

Basic Flow:

1. Initialize deleting an account.

2. User is authenticated to proceed with the deleting

process.

3. The account's information is under review and the user

is notified of deleting an account.

4. Funds are removed from the account, if any.

5. The account has been deleted.

Alternate Flows:

1. The user makes the decision of not wanting to delete

their account.

2. The account can't be deleted if the user has any type

of outstanding charges.

Exceptions:

1. The user fails to authenticate credentials.

2. There are outstanding charges to the user's account.

3. The server or client run into technical issues

throughout the deletion process.

4. The user's account has already been deleted.

5. The user's account can't be deleted if the account's

current status is locked, or restricted.

Related Use Cases:

- Log in

---

Use Case ID: A004

Use Case Name: Add a user to an account

Relevant Requirements: 3.1.3.15, 3.1.3.16.

Primary Actor: Account Module

Pre-condition: An account exists and a user exists.

Post-condition: The account and user are connected and the user has access to the account.

Basic Flow:

1. The primary holder of the account logs-in from the Teller's interface.

2. The user provides the user ID of the user that is to be added.

3. The to be added user logs in through the teller interface.

Alternate Flows:

none

Exceptions:

1. The user does not have the required access to the account.

2. The user is already added to the account.

Related Use Cases:

Use Case ID: A005

Use Case Name: Remove a user from an account.

Relevant Requirements: 3.1.3.15, 3.1.3.16.

Primary Actor: Client Module.

Pre-condition: An additional user is a part of an account.

Post-condition: The user is removed from the account.

Basic Flow:

1. The primary holder of the account logs-in from the Teller's interface.

2. The user provides the user ID of the user that is to be deleted.

3. The user's access to the account is erased.

Alternate Flows:

Exceptions:

1. The user does not have the required access to the account.

2. The user is already removed from the account.

Related Use Cases:

---

Use Case ID: A006

Use Case Name: Accounts Module, Client Modules,

Relevant Requirements: 3.1.2.4.3, 3.1.2.3.5.

Primary Actor: Operator

Pre-condition:

1. The customer must be logged into the system with valid credentials.

2. The customer must have at least one bank account registered.

3. The customer initiates a request to view their transaction history.

Post-condition:

The transaction history is presented to the customer.

Basic Flow:

1. The customer navigates to the "Account Summary" or "Transactions" section.

2. The system displays a default view of the most recent transactions for the selected account.

Alternate Flows:

No Transactions: If the account has no transactions for the selected date range, the system will display a “No transactions found” message.

Filter Error: If the filters entered by the customer are invalid (e.g., future date), the system prompts the operator to correct the input.

Exceptions:

System downtime or unavailability might prevent transaction data from being retrieved.

If there’s a lag in the backend systems, some recent transactions may not appear immediately.

Related Use Cases:

---