INFO 620: Information Systems Analysis and Design Project Milestone #3

*Rishabh Pandey (ID: **rp946**)*
*David Nana Dwomoh Sarpong (ID: **dd993**)*
*Mohamed Ashiq Basheer Ahamed (ID: **mb4498**)*
*Anushka Awasthi (ID: **aa4756**)*

*06/04/2024*

*Project Category: Analysis & Design*

## Requirements

### 1.1. Functional Requirements

The system will be password-protected. MEXS will be a multi-user system. MEXS needs to perform the following functions:

- Secure login and logout for authorized money exchange employees at the airport.
- Store exchange rates and archive historical rates.
- Calculate current exchange rates from one currency to another.
- Apply commission percentage to exchanges.
- Store transaction details including:
    - Transaction ID
    - Transaction date
    - Customer name, passport details, government issued ID/National Id
    - Amount exchanged/ Currency amount received from the customer
    - Exchange rate applied
    - Commission amount
    - Total amount charged to customer
    - Currency issued to customer
    - Currency exchange issuing country
- Allow voiding of transactions by authorized users (currency exchange agent).
- Generate reports:
    - Generate daily reports summarizing exchanges by country
    - Generate monthly reports summarizing exchanges by country
- Allow traveler/customer to select type of medium of exchange (i.e. cash or credit card card, check)

### 1.2. Data Requirements

- For Currencies: code, name, symbol
- Exchange rates: currency from, currency to, rate, effective date/time
- Transactions: transaction ID, timestamp, employee ID, currency from, amount from, currency to, amount to, rate, commission amount, total charged, customer name, passport number, issuing country, voided yes/no
- Archive of historical exchange rates
- User credentials: user ID, name, secure password, role/access level
- Government issued Id for KYC purpose

### 1.3. Business Rules and Logic

- The Current exchange rate must be used at time of transaction.
- Commission percentage can be dynamically set and changed by authorized personnel.
- Customer information must be stored with each transaction per financial regulations.
- Historical exchange rate data is archived.

### 1.4. Non-Functional Requirements such as usability, security, performance, reliability, etc.

- Secure access and user authentication
- Role-based access control (RBAC) for different user levels
- Audit logging of transactions
- Regular backups of transactional data
- Archived historic data should be accessible but separate from current data
- Responsive system performance for real-time transactions
- Intuitive and easy to use interface
- Ability to handle multiple concurrent users

### 1.5. Other Important Assumptions

- Currency inventory is managed outside of this system
- Exchange rates are imported from an external source
- All currency provided by the customer to the clerk is valid

## 2. Use Case Model

### 2.1. Stories Requirements Document

- **User Story 1**: Exchange Currency As a traveller from the United States visiting Switzerland, I want to exchange 1,000 USD for Swiss Francs (CHF) at the airport's MEXS counter so that I have local currency for my expenses during my stay.

- **User Story 2:** Update Exchange Rates as an MEXS administrator, I want to update the exchange rates in the system whenever there is a change in the currency market to ensure that the MEXS is using the most recent rates for currency exchanges.

- **User Story 3:** Generate Monthly/Daily Statistics Report as an MEXS manager, I want to generate monthly/daily statistics report that provides an overview of the currency exchanges performed at the airport's MEXS counter for the given period.

- **User Story 4:** Process Transaction by Cash/Card with OFAC Check As a traveller from the United Kingdom visiting the United States, I want to exchange 500 GBP for US Dollars (USD) at the airport's MEXS counter and have the option to pay for the transaction using either cash or a credit card, while ensuring compliance with OFAC regulations.

- **User Story 5:** Store, Retrieve, and Filter Transactions as an MEXS clerk, I want to be able to store, retrieve, and filter transaction data so that I can efficiently manage customer inquiries and assist with reporting.

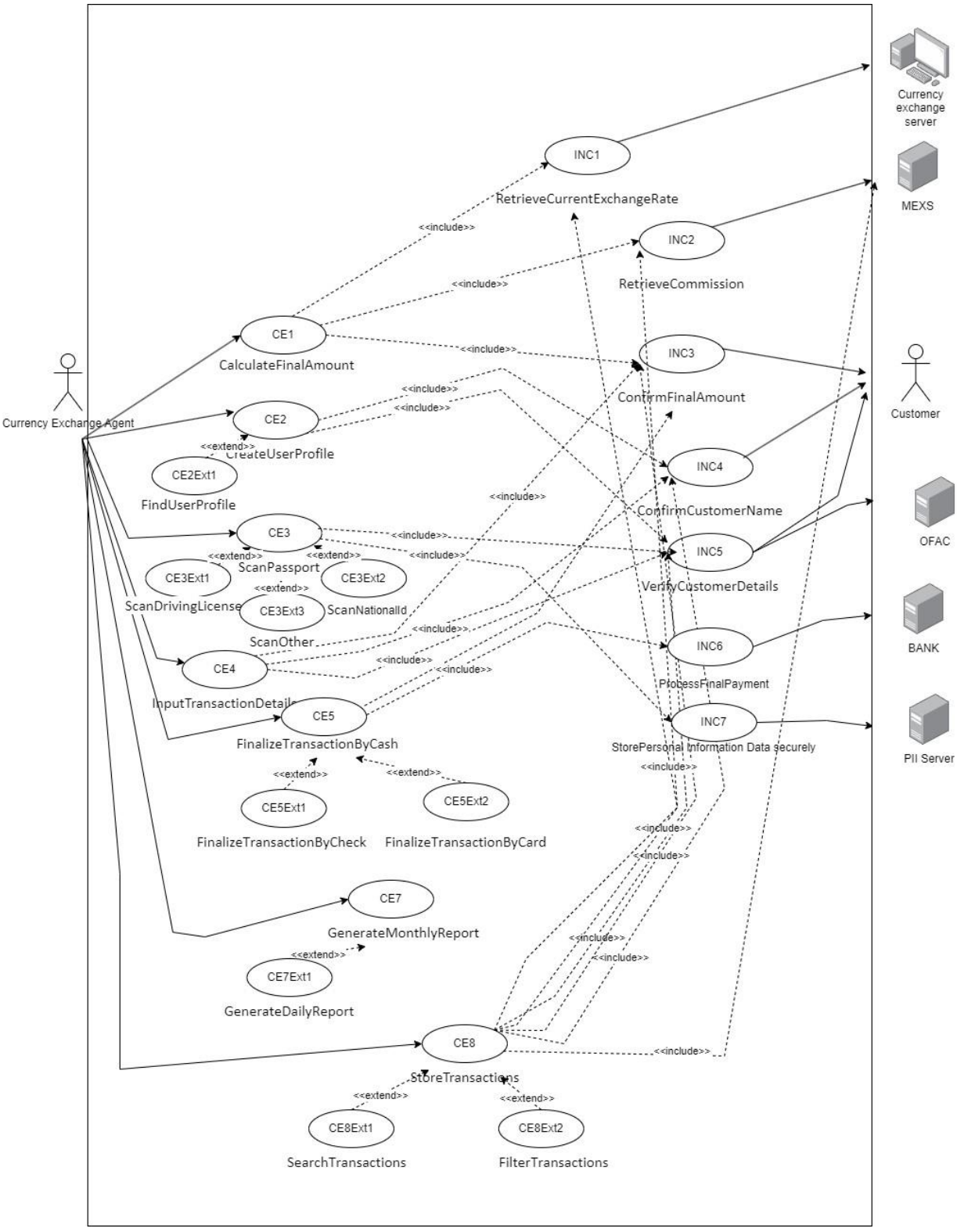
**Example Scenario:**

James Wilson, a traveler from Canada, approaches the MEXS counter at Zurich Airport with 2,000 CAD, requesting to exchange it for USD. The clerk initiates the transaction by entering James' details into the system, which applies the current exchange rate of 0.78 USD per 1 CAD. The system performs an OFAC compliance check, which clears James for the transaction. The clerk processes the exchange, deducting a commission fee, and James receives 1,560 USD. The system records the details of the transaction, including the amount exchanged and the commission deducted. Throughout the day, the MEXS administrator updates the exchange rates based on market fluctuations, ensuring accuracy for all transactions. At the end of the day, the MEXS manager generates a daily statistics report, detailing the total number of transactions, amounts exchanged, and revenue from fees, providing a comprehensive overview of the day's performance and aiding in operational planning.

**2.2. Actors and Their Goals (for each actor, write 1-2 sentence definition)**

- As a traveler, I want to be able to exchange my currency for the local currency of as quickly as possible, so that I can successfully handle the expenses for my trip.
- As an exchange agent, I want to be able to exchange any valid legal tender that a customer hands to me at the Airport terminal in the customer’s currency of choice, gather the required information from the customer for example Passport details, name, address, etc., and store the transaction details in the system.

## 2.3. Use case diagram (entire project)



**2.4. Use case descriptions for the most important primary use cases and their included use cases.**

| | | |
|---|---|---|
| ***USE CASE #*** | CE1 | |
| ***USE CASE Name*** | Calculate Final Amount | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | Determine the final amount to be exchanged for a customer accurately. | |
| ***Overview and scope*** | This use case involves calculating the total amount to be exchanged for a customer based on the entered currency and amount, applying commission rates, and considering any additional fees. | |
| ***Level*** | Base | |
| ***Preconditions*** | The employee is logged into the system.<br>Currency exchange rates are up to date. | |
| ***Postconditions in words (write in passive and past tense)*** | The final amount is calculated and displayed to the employee. | |
| **Trigger** | Employee initiates a currency exchange transaction. | |
| **Included Use Cases** | 1)Retrieve Current Exchange Rate.<br>2) Apply Commission Rate<br>3) Scan Driver License | |
| **Extending Use Cases** | 1) Apply Additional Fees<br>2)Handle Invalid Input | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1.- Employee selects the currency to exchange and enters the amount | Step 2. System retrieves the current exchange rate for the selected currency (INCLUDE Retrieve Current Exchange Rate). |
| | Step 3. System calculates the equivalent amount in the target currency. | Step 4. System applies commission rates to the exchange amount (INCLUDE Apply Commission Rate). |
| | Step 5. System adds any additional fees, if applicable (EXTEND Apply Additional Fees). | **Step 6. INCLUDE I1- System displays the final amount to the employee.** |
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful*** variations of the *normal* execution path, including any extension points using EXTEND use_*name*) | Step 1. Employee selects the currency to exchange and enters the amount. | Step 2. System retrieves the current exchange rate for the selected currency (INCLUDE Retrieve Current Exchange Rate). |
| | Step 3. System calculates the equivalent amount in the target currency. | Step 4. System applies commission rates to the exchange amount (INCLUDE Apply Commission Rate). |

| | | |
|---|---|---|
| | Step 5. System detects an additional fee is applicable. | **Step 6. INCLUDE I1- System adds the additional fee to the final amount.** |
| | **Step 7. System displays the final amount to the employee.** | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1. Employee selects the currency to exchange and enters an invalid amount. | Step 2. System prompts the employee to correct the input (INCLUDE Handle Invalid Input). |
| | Step 3. Employee corrects the input. | Step 4. System continues with the calculation process. |
| **Priority in scheduling** | High | |
| **Frequency** | Occasional | |
| **Business rules and data logic** | The final amount should be calculated accurately based on the current exchange rate, commission rates, and any applicable fees. | |
| **Other non-functional requirements** | The calculation process should be efficient and responsive to ensure a smooth user experience. | |
| **Superordinates** | Currency Exchange System | |
| **Developer** | Anushka Awasthi | |
| **Creation date and last modified date** | 05/04/2024 | |
| | | |
| ***USE CASE #*** | CE1 | |
| ***USE CASE Name*** | Confirm Final Amount | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | To confirm the final amount to be paid to a customer for currency exchange. | |
| ***Overview and scope*** | The system confirms the final amount based on the exchanged currency amount and the current exchange rate. | |
| ***Level*** | <<Included>> | |
| ***Preconditions*** | Currency exchange process is initiated by the cashier. | |
| ***Postconditions in words (write in passive and past tense)*** | The final amount to be paid by the customer is calculated and displayed. | |
| **Trigger** | This event is triggered when the cashier selects the "Confirm final amount" option during the currency exchange process. | |
| **Included Use Cases** | NONE | |
| **Extending Use Cases** | NONE | |
| | **Actor Action** | **System Action** |

| | | |
|---|---|---|
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | 1)Cashier enters the amount of currency to be exchanged. | 2)System retrieves the current exchange rate from the database. |
| | 3)The system calculates the total amount based on the entered currency amount and exchange rate. | 4)System displays the final amount to the cashier. |
| | 5) Cashier selects the "Confirm Final Amount" option. | |
| **Priority in scheduling** | High | |
| **Frequency** | Multiple times per currency exchange transaction | |
| **Business rules and data logic** | 1-The final amount includes the exchanged currency amount.<br>2-The exchange rate may vary based on market conditions. | |
| **Other non-functional requirements** | 1-The system must provide real-time calculation for efficiency.<br>2-The user interface should be user-friendly for cashiers. | |
| **Superordinates** | Currency Exchange System | |
| **Developer** | Anushka Awasthi | |
| **Creation date and last modified date** | 05/04/2024 | |
| Other Comments | | |
| ***USE CASE #*** | CE3Ext1: Scan Driver License | |
| ***USE CASE Name*** | Scan Driver License | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | To capture and verify customer identification through scanning the driver's license. | |
| ***Overview and scope*** | This use case involves scanning the customer's driver's license to capture personal identification details required for the currency exchange transaction. | |
| ***Level*** | Base | |
| ***Preconditions*** | The employee is logged into the system.<br>The system is connected to a functioning scanner. | |
| ***Postconditions in words (write in passive and past tense)*** | The driver's license details are captured and stored in the system.<br>The system verifies the validity of the driver's license. | |
| **Trigger** | Employee initiates the scan of the customer's driver's license. | |
| **Included Use Cases** | None | |
| **Extending Use Cases** | Handle Invalid Input<br>Verify Identification Details | |
| | **Actor Action** | **System Action** |

| | | |
|---|---|---|
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | Step 1.- Employee selects the option to scan the driver's license | Step 2. Employee scans the driver's license using the connected scanner. |
| | Step 3. System captures the details from the scanned driver's license. | Step 4. System verifies the validity of the captured details. |
| | Step 5. System stores the driver's license details in the customer's profile. | |
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful***variations of the *normal* execution path, including any extension points using EXTEND use_*name*) | Step 1. Employee selects the option to scan the driver's license | Step 2. Employee scans the driver's license using the connected scanne |
| | Step 3. System captures the details from the scanned driver's license. | Step 4. System verifies the validity of the captured details. |
| | Step 5. System detects minor issues but still verifies the driver's license as valid. | **Step 6. System stores the driver's license details in the customer's profile.** |
| | | |
| ***UNSUCCESSFUL SCENARIOS***(*erroneous*situations*)* | Step 1. Employee selects the option to scan the driver's license | Step 2. Employee scans the driver's license using the connected scanner. |
| | Step 3. System fails to capture the details from the scanned driver's license.. | Step 4. System prompts the employee to rescan the driver's license (EXTEND Handle Invalid Input). |
| **Priority in scheduling** | High | |
| **Frequency** | Occasional | |
| **Business rules and data logic** | The system should accurately capture and verify the driver's license details.<br>The driver's license information must be stored securely and comply with relevant data protection regulations. | |
| **Other non-functional requirements** | The scanning and verification process should be efficient to avoid delays.<br>The user interface should provide clear instructions and feedback to the employee during the scanning process. | |
| **Superordinates** | Currency Exchange System | |
| **Developer** | Anushka Awasthi | |
| **Creation date and last modified date** | 05/04/2024 | |

| | | |
|---|---|---|
| ***USE CASE #*** | CE2 | |
| ***USE CASE Name*** | Create User Profile | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | Register a new user in the system | |
| ***Overview and scope*** | This use case allows a currency exchange agent to register a new user in the system by entering their details and confirming their name and passport information. | |
| ***Level*** | Base Case | |
| ***Preconditions*** | 1. The agent is logged into the system.<br>2. The agent has the necessary permissions to register new users. | |
| ***Postconditions in words (write passive and past tense)*** | 1. A new user is registered in the system and a Person object newPerson was created.<br>2. Attributes newPerson.name was updated.<br>3. A Customer object newCustomer was created and associated with newPerson.<br>4. An OFAC check was performed on the customer's details, and the result was recorded.<br>5. Data tables were updated.<br>6. newPerson was destroyed.<br>7. newCustomer was destroyed. | |
| **Trigger** | The agent selects the "Create User Profile" option in the MEXS | |
| **Included Use Cases** | ConfirmCustomername (INC4), VerifyCustomerDetails (INC5) | |
| **Extending Use Cases** | FindUserProfile | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference “included use cases” in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1: The agent inititaites a new user profile creation | Step 2: The system displays a form to enter user details. |
| | | Step 3: The system validates the entered details. |
| | | Step 4. INCLUDE 4: **Confirm Customer name** |
| | | Step 5. INCLUDE 5: **Verify customer details** |
| | | Step 6. Perform an ofac check on the customer’s details |
| | Step 7: The agent confirms the user registration. | Step 8: The system creates a new user profile and displays a success message. |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations) | Step 1: Currency Exchange Agent initiates the user profile creation process. | Step 2: System presents a form to enter the customer's personal information. |

| | | |
|---|---|---|
| | Step 3. Currency Exchange Agent enters customer details. | Step 4. OFAC check returns a positive match indicating profile already exists. |
| **Priority in scheduling** | High | |
| **Frequency** | Once per new user profile creation | |
| **Business rules and data logic** | 1. All required fields must be filled.<br>2. Passport number/other customer identification document details must be valid and unique.<br>3. User profile creation cannot proceed if the OFAC check returns a positive match indicating user profile already exists | |
| **Other non-functional requirements** | | |
| **Superordinates** | | |
| **Developer** | Rishabh Pandey (rp946) | |
| **Creation date and last modified date** | Creation: 05/04/2024<br>Updated : 05/18/2024 | |
| Other Comments | | |

| | | |
|---|---|---|
| ***USE CASE #*** | CE2Ext1 | |
| ***USE CASE Name*** | FindUserProfile | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | To find an existing user profile in the currency exchange system. | |
| ***Overview and scope*** | The Currency Exchange Agent searches for a user profile by entering the customer's name or associated identification document details. The system retrieves the matching user profile. | |
| ***Level*** | <<Extended>> | |
| ***Preconditions*** | User profiles exist in the system. | |
| ***Postconditions in words (write in passive and past tense)*** | A Person object matchingPerson was retrieved.<br>A Customer object matchingCustomer associated with matchingPerson was retrieved.<br>A Profile object matchingProfile associated with matchingCustomer was retrieved.<br>An Identification object matchingIdentification associated with matchingProfile was retrieved.<br>matchingPerson was destroyed.<br>matchingCustomer was destroyed.<br>matchingProfile was destroyed.<br>matchingIdentification was destroyed. | |
| **Trigger** | This event is triggered when a Currency Exchange Agent initiates a user profile search. | |
| **Included Use Cases** | VerifyCustomerDetails (INC5) | |
| **Extending Use Cases** | | |
| | **Actor Action** | **System Action** |

| | | |
|---|---|---|
| ***MAIN SUCCESSFUL SCENARIO <u>for this Use Case</u>* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | Step 1. Currency Exchange Agent initiates a user profile search. | Step 2. System presents a search form. |
| | Step 3. Currency Exchange Agent enters the customer's name or passport number.. | Step 4. System searches for a matching Person object based on the provided name/Identification document number via an api call to the OFAC associated database. |
| | | Step 5. System retrieves the matching Person object, associated Customer object, Profile object, and Identification object. |
| | | Step 6. System displays the retrieved user profile information. |
| | Step 7. INCLUDE5 The CEA verifies the retrieved user details with the customer at the desk and proceeds to the exchange process. | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1. Currency Exchange Agent initiates a user profile search. | Step 2. System presents a search form |
| | Step 3. Currency Exchange Agent enters the customer's name or passport number.. | Step 4. System searches for a matching Person object based on the provided name/Identification document number via an api call to the OFAC associated database. |
| | | Step 5. System finds no matching Person object. |
| | | Step 6. System displays a message indicating that no matching user profile was found. |
| **Priority in scheduling** | Medium | |
| **Frequency** | Once per run | |
| **Business rules and data logic** | Search criteria can include customer name/passport number/NationalIdNumber/DrivingLicenseNumber /OtherGovtIssuedIdentificationDocuemnts | |
| **Other non-functional requirements** | Search functionality should be fast and efficient. | |
| **Superordinates** | | |
| **Developer** | Rishabh Pandey (rp946) | |
| **Creation date and last modified date** | 05/18/2024 | |
| Other Comments | | |

| | | |
|---|---|---|
| ***USE CASE #*** | INC4 | |
| ***USE CASE Name*** | Confirm Customer Name | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | Confirm the customer's name during registration. | |
| ***Overview and scope*** | This use case is included in the "CreateUser profile" use case to confirm the customer's name entered by the agent during the registration process. | |
| ***Level*** | Include | |
| ***Preconditions*** | The agent has entered the customer's name in the registration form. | |
| ***Postconditions in words (write in passive and past tense)*** | The customer's name is confirmed and validated. Attributes newPerson.name was updated with the confirmed name. | |
| **Trigger** | This event is triggered when the "Create User Profile" use case invokes this use case. | |
| **Included Use Cases** | | |
| **Extending Use Cases** | | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | | Step 1: The system prompts the agent to confirm the customer's name. |
| | Step 2: The agent confirms the customer's name with the customer at the desk. | Step 3: The system validates the name format and checks for any discrepancies. |
| | | Step 4: The system confirms the name is valid and returns control to the "Create User Profile" use case. |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations) | **Conditions** | **Actions** |
| | | 1. The system detects an invalid name format or discrepancy. |
| | | 2. The system prompts the agent to re-enter or correct the name. |
| | 4. Agent re-enters or corrects the name. | 5. Return to Step 3 in the main successful scenario. |
| | | |
| **Priority in scheduling** | High | |
| **Frequency** | Every time a new user is registered | |
| **Business rules and data logic** | 1. Name must be in a valid format<br>2. Name must not contain any special characters | |
| **Other non-functional requirements** | | |

| | |
|---|---|
| **Superordinates** | Register User (CE2) |
| **Developer** | Rishabh Pandey (rp946) |
| **Creation date and last modified date** | 05/04/2024 |
| Other Comments | |

| | | |
|---|---|---|
| ***USE CASE #*** | INC5 | |
| ***USE CASE Name*** | Verify Customer Details | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | Confirm the customer's passport details during User Profile creation | |
| ***Overview and scope*** | This use case is included in the "Create User Profile" and "Find User Profile" use cases to verify the customer's identification details, such as passport number or other identification document details, entered by the agent during the registration process and perform an OFAC check | |
| ***Level*** | Include | |
| ***Preconditions*** | The agent has entered the customer's passport details in the registration form. | |
| ***ostconditions in words (write passive and past tense)*** | The customer's identification details were verified and validated.<br>An Identification object newIdentification was created.<br>Attributes newIdentification.idNumber, newIdentification.issueDate, newIdentification.expiryDate were updated with the verified details.<br>newIdentification was associated with newProfile.<br>An OFAC check was performed on the customer's identification details, and the result was recorded. | |
| **Trigger** | The "Register User" use case invokes this use case | |
| **Included Use Cases** | | |
| **Extending Use Cases** | | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference “included use cases” in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | | Step 1. The system prompts the agent to confirm the customer's identification document details |
| | Step 2. The agent confirms the customer’s identification document details. | Step 3. System validates the identification details format and checks for |

| | | |
|---|---|---|
| | | uniqueness in the User Profile database. |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | **Conditions** | **Actions** |
| | | The system detects an invalid passport number format or discrepancy |
| | | The system prompts the agent to re-enter or correct the passport details |
| **Priority in scheduling** | High | |
| **Frequency** | Every time a new user is registered | |
| **Business rules and data logic** | Customer Identification document number must be in a valid format<br>Customer Identification document must be unique for each customer | |
| **Other non-functional requirements** | | |
| **Superordinates** | Register User (CE2) | |
| **Developer** | Rishabh Pandey (rp946) | |
| **Creation date and last modified date** | Creation :05/04/2024<br>Update1 : 05/18/2024 | |
| Other Comments | | |
| **USE CASE #** | CE3 | |
| **USE CASE Name** | ScanPassport | |
| **ACTOR** | CurrencyExchangeAgent | |
| **Goal (1 phrase)** | To capture and store information from other traveler's passport. | |
| **Overview and scope** | This use case involves the scanning and processing of a traveler's passport. | |
| **Level** | Base | |
| **Preconditions** | The Currency Exchange Agent (CEA) is logged into the system.<br>The scanning hardware is operational and connected to the system.<br>The customer has provided their passport. | |
| **Postconditions in words (write in passive and past tense)** | | |
| **Trigger** | This event is triggered when the traveler initiates a transaction with the Currency Exchange Agent. | |
| **Included Use Cases** | INC5 | |
| **Extending Use Cases** | | |
| **MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence** | **Actor Action** | **Actor Action** |
| | Step 1. The Currency Exchange Agent (CEA) selects the option to | Step 2. The system activates the scanner and |

| | | |
|---|---|---|
| | scan an "Other" type of identification document. | captures the image of the "Other" identification document. |
| | Step 3. The CEA initiates the scan process by pressing the scan button. | Step 4. The system processes the scanned image to extract relevant details such as name, address, and document type. |
| | | Step 5. The system displays the extracted information on the screen for the CEA to review. |
| | | Step 6. **INCLUDE 5** Verify Customer Details |
| | | Step 7. The system stores the scanned image and the extracted information in the customer's profile. |
| **Priority in scheduling** | High | |
| **Frequency** | Once per transaction | |
| **Business rules and data logic** | The system must support scanning of various alternative identification documents such as utility bills, bank statements, employee ID cards, and other accepted forms of identification in addition to Passport, Driver's Liscence, and National ID. | |
| **Other non-functional requirements** | | |
| **Superordinates** | | |
| **Developer** | David Nana Dwomoh Sarpong (dd993) | |
| **Creation date and last modified date** | 05/17/2024 | |
| Other Comments | | |

| | |
|---|---|
| **USE CASE #** | CE3Ext2 |
| **USE CASE Name** | ScanNationalId |
| **ACTOR** | CurrencyExchangeAgent |
| **Goal (1 phrase)** | To capture and store information from the traveler's Nationally issued ID. |
| **Overview and scope** | This use case involves the scanning and processing of a traveler's nationally issued ID. |
| **Level** | <<Extend>> |

| | | |
|---|---|---|
| **Preconditions** | The Currency Exchange Agent (CEA) is logged into the system.<br>The scanning hardware is operational and connected to the system.<br>The customer has provided their passport. | |
| **Postconditions in words (write in passive and past tense)** | | |
| **Trigger** | This event is triggered when the traveler initiates a transaction with the Currency Exchange Agent. | |
| **Included Use Cases** | INC5 | |
| **Extending Use Cases** | | |
| **MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence** | **Actor Action** | **Actor Action** |
| | Step 1. The Currency Exchange Agent (CEA) selects the option to scan an "Other" type of identification document. | Step 2. The system activates the scanner and captures the image of the "National" identification document. |
| | Step 3. The CEA initiates the scan process by pressing the scan button. | Step 4. The system processes the scanned image to extract relevant details such as name, address, and document type. |
| | | Step 5. The system displays the extracted information on the screen for the CEA to review. |
| | | Step 6. **INCLUDE 5** Verify Customer Details |
| | | Step 7. The system stores the scanned image and the extracted information in the customer's profile. |
| **Priority in scheduling** | High | |
| **Frequency** | Once per transaction | |
| **Business rules and data logic** | The system must support scanning of various alternative identification documents such as utility bills, bank statements, employee ID cards, and other accepted forms of identification in addition to Passport, Driver’s Liscence, and National ID. | |
| **Other non-functional requirements** | | |
| **Superordinates** | | |
| **Developer** | Rishabh Pandey (rp946) | |

| | |
|---|---|
| **Creation date and last modified date** | 05/20/2024 |
| Other Comments | |

| | | |
|---|---|---|
| *USE CASE #* | CE4 | |
| *USE CASE Name* | InputTransactionDetails | |
| *ACTOR* | CurrencyExchangeAgent | |
| *Goal (1 phrase)* | To get the transaction details from the customer | |
| *Overview and scope* | This use case involves the Currency exchange agent to enter details of the customer such as Name, address, passport information. | |
| *Level* | Base case | |
| *Preconditions* | 1. The agent is logged into the system.<br>2. The user is registered in the system.<br>3. The customer request cash for currency exchange. | |
| *Postconditions in words (write in passive and past tense)* | 1. The user is registered in the system. | |
| *Trigger* | This is triggered when the customer wants to exchange the currency | |
| *Included Use Cases* | ConfirmFinalAmount<br>ConfirmCustomerName<br>VerifyCustomerDetails | |
| *Extending Use Cases* | None | |
| *MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence* | **Actor Action** | **System Action** |
| | 1. The Currency Exchange Agent logs into the system | 2.System asks for the final exchanging currency |
| | 3.The Agent selects the option to input transaction details | 4.The system validates the entered information for completeness and correctness |
| | 5. The Agent enters the transaction details including customer information, currency type, amount, and exchange rate | 6.The system calculates the total amount to be exchanged using the provided exchange rate |
| | | 7.The system displays a summary of the transaction details for confirmation |
| *Priority in scheduling* | High | |
| *Frequency* | Once per run | |
| *Business rules and data logic* | 1. Name must be in a valid format.<br>2. Name must not contain any special characters. | |
| *Other non-functional requirements* | | |
| *Superordinates* | | |

| | |
|---|---|
| ***Developer*** | Mohamed Ashiq Basheer Ahamed (mb4498) |
| ***Creation date and last modified date*** | 05/17/2024 |

| | | |
|---|---|---|
| *USE CASE #* | INC3 | |
| *USE CASE Name* | ConfirmFinalAmount | |
| *ACTOR* | Currency Exchange Agent | |
| *Goal (1 phrase)* | To confirm the final amount to be paid to a customer for currency exchange. | |
| *Overview and scope* | The system calculates the final amount and displays to the customer to finalize the amount based on the exchanged currency amount and the current exchange rate. | |
| *Level* | Include | |
| *Preconditions* | The currency exchange process is initiated by the cashier. | |
| *Postconditions in words (write in passive and past tense)* | The final amount to be paid by the customer is calculated and displayed. | |
| *Trigger* | This is triggered when the cashier selects the "Confirm final amount" option during the currency exchange process. | |
| *Included Use Cases* | NONE | |
| *Extending Use Cases* | NONE | |
| *MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence* | **Actor Action** | System |
| | 1.The Agent reviews the transaction details | 2.The system retrieves the current exchange rate (include from INC1) |
| | | 3.The system calculates the final amount including any commissions (include from INC2). |
| | | 4.The system displays the final amount to the CEA for confirmation. |
| | 5.The Agent selects the option to confirm the final amount. | 6.The system logs the confirmation of the final amount. |
| *Priority in scheduling* | High | |
| *Frequency* | Once per transaction | |
| *Business rules and data logic* | 1.The system must have the current rate of the currency. | |
| ***Other non-functional requirements*** | | |

| | |
|---|---|
| ***Superordinates*** | InputTransactionDetails |
| ***Developer*** | Mohamed Ashiq Basheer Ahamed (mb4498) |
| ***Creation date and last modified date*** | 05/18/2024 |
| *Other Comments* | |

| | | |
|---|---|---|
| *USE CASE #* | INC4 | |
| *USE CASE Name* | ConfirmCustomerName | |
| *ACTOR* | CurrencyExchangeAgent | |
| *Goal (1 phrase)* | Confirm the name of the customer during registration | |
| *Overview and scope* | This use case involves confirming the customer name when registering with the money exchange agent. | |
| *Level* | Include | |
| *Preconditions* | The agent must have entered the details of the customer. | |
| *Postconditions in words* | The customer's name is confirmed and validated. | |
| *Trigger* | The “inputTransactionDetails” triggers this use case. | |
| *Included Use Cases* | None | |
| *Extending Use Cases* | None | |
| *MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence* | **Actor Action** | **Actor Action** |
| | 1.The Agent requests the customer to provide identification. | 2.The system retrieves the customer's profile (include from CE2). |
| | 3.The Agent inputs the customer’s name as shown on the identification document. | 4.The system verifies the input name against the customer's profile. |
| | | 5.The system displays a confirmation message indicating the name matches the profile. |
| | | 6.The system logs the name confirmation. |
| *Priority in scheduling* | High | |
| *Frequency* | Once per transaction | |
| *Business rules and data logic* | The name of the customer should match the name on passport | |
| *Other non-functional requirements* | | |
| *Superordinates* | InputTransactionDetails | |
| *Developer* | Mohamed Ashiq Basheer Ahamed (mb4498) | |
| *Creation date and last modified date* | 05/18/2024 | |
| *Other Comments* | | |

| | | |
|---|---|---|
| *USE CASE #* | INC5 | |
| *USE CASE Name* | VerifyCustomerDetails | |
| *ACTOR* | Currency Exchange Agent | |
| *Goal (1 phrase)* | Confirm the customer's details during registration | |
| *Overview and scope* | This use case involves confirming the customer's details when registering with the money exchange agent. | |
| *Level* | Include | |
| *Preconditions* | The agent must have entered the details of the customer. | |
| *Postconditions in words (write in passive and past tense)* | The customer's passport details are confirmed and validated. | |
| *Trigger* | The "InputTransactionDetails" triggers this use case. | |
| *Included Use Cases* | None | |
| *Extending Use Cases* | None | |
| *MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence* | **Actor Action** | **System Action** |
| | 1.The Agent initiates the customer verification process. | 2.The system prompts the agent to confirm the customer's passport details |
| | 3.The Agent submits the customer's details including name, ID number, and nationality for verification. | 4.The system retrieves the customer's profile (include from CE2). |
| | | 5.The system checks the customer's details against the OFAC (Office of Foreign Assets Control) list. |
| | | 6.The system verifies the customer's details with the bank (include from BANK). |
| | | 7.The system displays a verification confirmation message indicating the customer's details are verified. |
| | | 8.The system logs the verification details. |
| *UNSUCCESSFUL SCENARIOS* | **Conditions** | **Actions** |
| | 1.The Agent submits the customer's details including | 2.The system checks the customer's details against the |

| | | |
|---|---|---|
| | name, ID number, and nationality for verification. | OFAC (Office of Foreign Assets Control) list. |
| | | 3.The system identifies a match with an entry on the OFAC list, indicating potential sanctions or issues. |
| | | 4.The system notifies the CEA to inform the customer that the transaction cannot be completed due to regulatory restrictions. |
| *Priority in scheduling* | High | |
| *Frequency* | Once per transaction | |
| *Business rules and data logic* | Passport number must be in a valid format.<br>Passport number must be unique for each customer. | |
| *Other non-functional requirements* | | |
| *Superordinates* | InputTransactionDetails | |
| *Developer* | Mohamed Ashiq Basheer Ahamed (mb4498) | |
| *Creation date and last modified date* | 05/18/2024 | |
| *Other Comments* | | |

| | |
|---|---|
| *USE CASE #* | CE3Ext3 |
| *USE CASE Name* | ScanOther |
| *ACTOR* | CurrencyExchangeAgent |
| *Goal (1 phrase)* | To capture and store information from other identification document. |
| *Overview and scope* | This use case involves the scanning and processing of identification documents other than passports, driving licenses, or national IDs. It ensures that alternative identification documents are captured accurately and their details extracted and stored in the customer's profile. |
| *Level* | Include |
| *Preconditions* | The Currency Exchange Agent (CEA) is logged into the system.<br>The scanning hardware is operational and connected to the system.<br>The customer has provided an alternative identification document. |
| *Postconditions in words (write in passive and past tense)* | The "Other" identification document was successfully scanned. |
| *Trigger* | The “inputTransactionDetails” triggers this use case. |

| | | |
|---|---|---|
| *Included Use Cases* | None | |
| *Extending Use Cases* | CE3<br>CE3Ext1<br>CE3Ext2 | |
| *MAIN SUCCESSFUL SCENARIO for this Use Case in numbered sequence* | **Actor Action** | **Actor Action** |
| | 1.The Agent selects the option to scan an "Other" type of identification document. | 2.The system activates the scanner and captures the image of the "Other" identification document. |
| | 3.The agent initiates the scan process by pressing the scan button. | 4.The system processes the scanned image to extract relevant details such as name, address, and document type. |
| | | 5.The system displays the extracted information on the screen for the agentto review. |
| | | 6.The system stores the scanned image and the extracted information in the customer's profile. |
| *Priority in scheduling* | High | |
| *Frequency* | Once per transaction | |
| ***Business rules and data logic*** | The system must support scanning of various alternative identification documents such as utility bills, bank statements, employee ID cards, and other accepted forms of identification. | |
| ***Other non-functional requirements*** | | |
| ***Superordinates*** | | |
| ***Developer*** | Mohamed Ashiq Basheer Ahamed (mb4498) | |
| ***Creation date and last modified date*** | 05/18/2024 | |
| *Other Comments* | | |

| | |
|---|---|
| ***USE CASE #*** | CE5 |
| ***USE CASE Name*** | Finalize Transaction by Cash |
| ***ACTOR*** | Currency Exchange Agent |
| ***Goal (1 phrase)*** | To process the final amount to be given to the traveler/customer by Cash |
| ***Overview and scope*** | This use case involves the initial medium of exchange the traveler/customer presents to the agent at the MEXS. The system calculates the amount to be exchanged based on the cash provided |

| | | |
|---|---|---|
| | by the traveler and proceeds with the transaction. This use case focuses on handling currency exchange transactions where the traveler pays with cash. | |
| ***Level*** | Base | |
| ***Preconditions*** | The Currency Exchange Agent is logged into the MEXS system.<br>The system is operational and has access to the latest currency exchange rates.<br>The traveler/customer request cash for currency exchange. | |
| ***Postconditions in words (write in passive and pas tense)*** | CurrencyExchange was created and associated with Transaction.<br>Attribute Transaction.method was created/updated.<br>Attributes CurrencyExchange.amount, CurrencyExchange.transactionId, CurrencyExchange.methodOfTransaction were created. | |
| **Trigger** | This event is triggered when the Traveler/customer selects cash as preferred medium of transaction | |
| **Included Use Cases** | Confirm Final Amount | |
| **Extending Use Cases** | Finalize Transaction by Check<br>Finalize Transaction by Card | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1. The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | |
| | Step 2. The traveler/customer presents the cash to the agent. | Step 3. System checks if the cash option is available |
| | | Step 4. System confirms the cash option is available |
| | Step 5. The agent verifies the authenticity of the cash and counts the amount provided by the traveler | |
| | Step 6. The agent enters the amount of cash provided into the MEXS system | Step 7. The system retrieves the current exchange rates and calculates the amount of local currency the traveler will receive based on the provided cash and the commission rate. |
| | | Step 8. INCLUDE 1 Confirm Final Amount |
| | Step 9. Agent confirms the transaction | Step 10. System outputs the final cash amount |
| | Step 11. Agent takes out cash and hands it to customer/traveler | Step 12. System stores transaction details |
| | | |

| | | |
|---|---|---|
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful*** variations of the *normal* execution path, including any extension points using EXTEND *eus_name*) | Step 1.- The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | **Step 2. EXTEND CE5Ext1** Finalize Transaction By Check |
| | | |
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful*** variations of the *normal* execution path, including any extension points using EXTEND *eus_name*) | Step 1.- The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | **Step 2. EXTEND CE5Ext2** Finalize Transaction By Card |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1.- Agent prompts customer for desired medium of transaction | |
| | Step 2. Traveler provides the desired medium of transaction | Step 3. System checks if the cash option is available |
| | | Step 4. Cash option is unavailable |
| | | Step 5. System aborts transaction |
| | | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1. The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | |
| | Step 2. The traveler/customer presents the cash to the agent. | Step 3. System checks if the selected option is available. |
| | | Step 4. Cash option is available |
| | Step 5. The agent verifies the authenticity of the cash and counts the amount provided by the traveler | |
| | Step 6. The traveler provides insufficient amount | Step 7. System aborts transaction |
| **Priority in scheduling** | High | |
| **Frequency** | Once per run | |
| **Business rules and data logic** | Customer information must be stored with each transaction per financial regulations. | |
| **Other non-functional requirements** | Responsive system performance for real-time transactions<br>Intuitive and easy to use interface | |
| **Superordinates** | | |
| **Developer** | David Nana Dwomoh Sarpong (dd993) | |

| | |
|---|---|
| **Creation date and last modified date** | 05/16/2024 |
| Other Comments | |

| | | |
|---|---|---|
| ***USE CASE #*** | CE5Ext1 | |
| ***USE CASE Name*** | Finalize Transaction by Check | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | To process the final amount to be given to the traveler/customer by Check | |
| ***Overview and scope*** | This use case involves the initial medium of exchange the traveler/customer presents to the agent at the MEXS. The system calculates the amount to be exchanged based on the cash provided by the traveler and proceeds with the transaction. This use case focuses on handling currency exchange transactions where the traveler receives a check. | |
| ***Level*** | <<Extended>> | |
| ***Preconditions*** | The Currency Exchange Agent is logged into the MEXS system.<br>The system is operational and has access to the latest currency exchange rates.<br>The traveler/customer requests check for currency exchange. | |
| ***Postconditions in words (write in passive and pa tense)*** | CurrencyExchange was created and associated with Transaction.<br>Attribute *Transaction.method* was created/updated.<br>Attributes *CurrencyExchange.amount*, *CurrencyExchange.transactionId*, *CurrencyExchange.methodOfTransaction* were created. | |
| **Trigger** | This event is triggered when the Traveler/customer selects check as preferred medium of transaction | |
| **Included Use Cases** | Confirm Final Amount | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1. The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | |
| | Step 2. The traveler/customer presents the cash to the agent. | Step 3. System checks if the check option is available |
| | | Step 4. System confirms the check option is available |
| | Step 5. The agent verifies the authenticity of the cash and counts the amount provided by the traveler | |
| | Step 6. The agent enters the amount of cash provided into the MEXS system | Step 7. The system retrieves the current exchange rates and calculates the amount of local currency the traveler will receive based on the provided cash and the commission rate. |

| | | |
|---|---|---|
| | | Step 8. INCLUDE Confirm Final Amount |
| | Step 9. Agent confirms the transaction | Step 12. System stores transaction details |
| | Step 11. Agent writes check and hands it to customer/traveler | |
| | | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1.- Agent prompts customer for desired medium of transaction | |
| | Step 2. Traveler provides the desired medium of transaction | Step 3. System checks if the check option is available |
| | | Step 4. Check option is unavailable |
| | | Step 5. System aborts transaction |
| | | |
| **Priority in scheduling** | High | |
| **Frequency** | Once per run | |
| **Business rules and data logic** | Customer information must be stored with each transaction per financial regulations. | |
| **Other non-functional requirements** | Responsive system performance for real-time transactions<br>Intuitive and easy to use interface | |
| **Superordinates** | CE4 Finalize Transaction by Cash | |
| **Developer** | David Nana Dwomoh Sarpong (dd993) | |
| **Creation date and last modified date** | 05/16/2024 | |
| Other Comments | | |

| | |
|---|---|
| ***USE CASE #*** | CE5Ext2 |
| ***USE CASE Name*** | Finalize Transaction by Card |
| ***ACTOR*** | Currency Exchange Agent |
| ***Goal (1 phrase)*** | To process the final amount to be given to the traveler/customer by Forex Card |
| ***Overview and scope*** | This use case involves the initial medium of exchange the traveler/customer presents to the agent at the MEXS. The system calculates the amount to be exchanged based on the cash provided by the traveler and proceeds with the transaction. This use case focuses on handling currency exchange transactions where the traveler receives a forex card. |
| ***Level*** | <<Extended>> |
| ***Preconditions*** | The Currency Exchange Agent is logged into the MEXS system.<br>The system is operational and has access to the latest currency exchange rates. |

| | | |
|---|---|---|
| | The traveler/customer requests a forex card for currency exchange. | |
| ***Postconditions in words (write in passive and past tense)*** | *CurrencyExchange was created and associated with Transaction. Attribute Transaction.method was* created/updated. *Attributes CurrencyExchange.amount, CurrencyExchange.transactionId, CurrencyExchange.methodOfTransaction were created.* | |
| **Trigger** | This event is triggered when the Traveler/customer selects forex card as preferred medium of transaction. | |
| **Included Use Cases** | Confirm Final Amount | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference "included use cases" in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1. The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | |
| | Step 2. The traveler/customer presents the cash to the agent. | Step 3. System checks if the check option is available |
| | | Step 4. System confirms the check option is available |
| | Step 5. The agent verifies the authenticity of the cash and counts the amount provided by the traveler | |
| | Step 6. The agent enters the amount of cash provided into the MEXS system | Step 7. The system retrieves the current exchange rates and calculates the amount of local currency the traveler will receive based on the provided cash and the commission rate. |
| | | Step 8. INCLUDE Confirm Final Amount |
| | Step 9. Agent confirms the transaction | |
| | Step 11. Agent puts forex card into system | Step 12. System deposits money on card |
| | | Step 13. System stores transaction details |
| | Step 14. Agent hands forex card to traveler. | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1. Agent prompts customer for desired medium of transaction | |
| | Step 2. Traveler provides the desired medium of transaction | Step 3. System checks if the check option is available |
| | | Step 4. Card option is unavailable |
| | | Step 5. System aborts transaction |
| | | |
| **Priority in scheduling** | High | |
| **Frequency** | Once per run | |

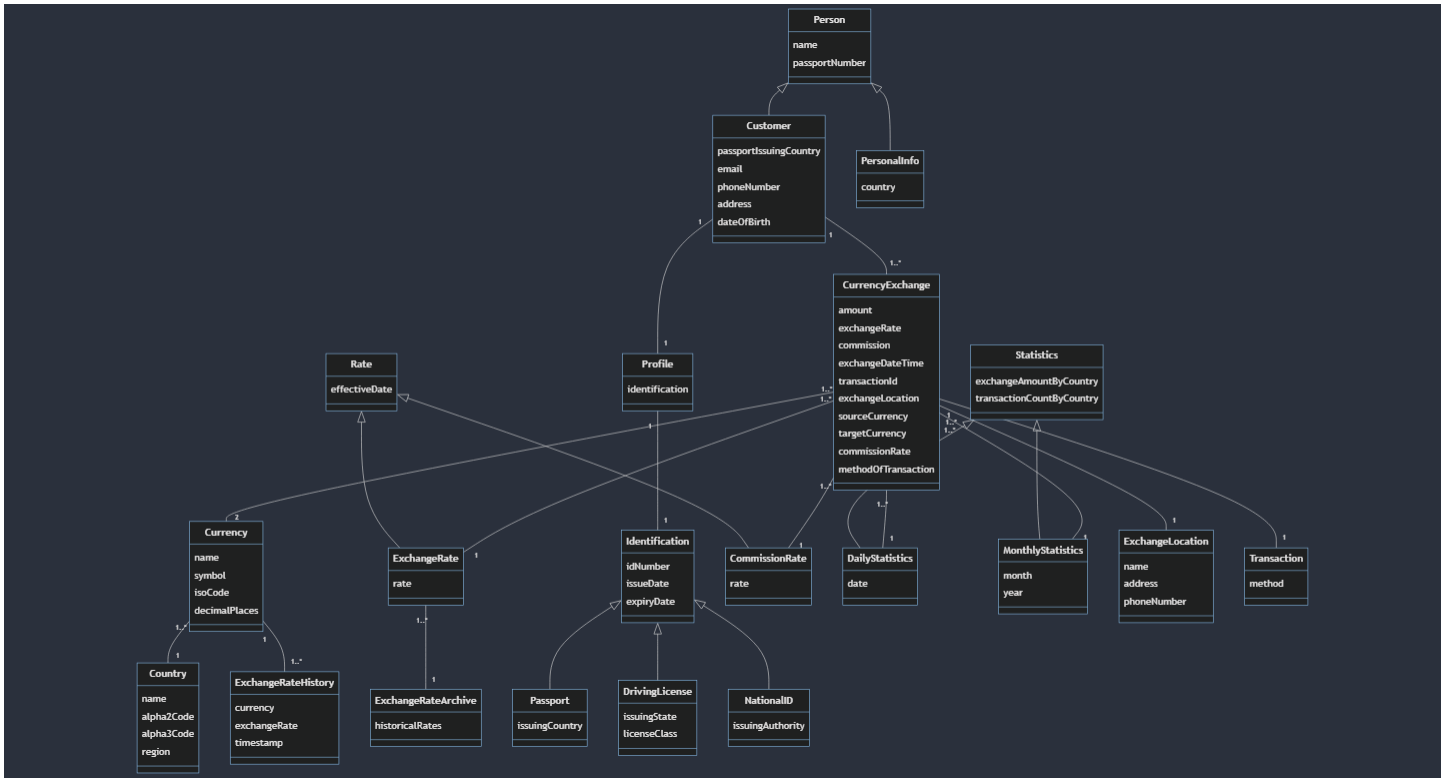
| | |
|---|---|
| **Business rules and data logic** | Customer information must be stored with each transaction per financial regulations. |
| **Other non-functional requirements** | Responsive system performance for real-time transactions<br>Intuitive and easy to use interface |
| **Superordinates** | CE5 Finalize Transaction by Cash |
| **Developer** | David Nana Dwomoh Sarpong (dd993) |
| **Creation date and last modified date** | 05/16/2024 |
| Other Comments | |

| | | |
|---|---|---|
| ***USE CASE #*** | INC6 | |
| ***USE CASE Name*** | Process Final Payment | |
| ***ACTOR*** | Currency Exchange Agent | |
| ***Goal (1 phrase)*** | To process the bank information of the traveler. | |
| ***Overview and scope*** | This use case describes the steps taken by the cashier to process the final payment in cash for a currency exchange transaction. | |
| ***Level*** | <<Included>> | |
| ***Preconditions*** | The customer has provided the desired amount of foreign currency in card/check.<br>The system has displayed the final amount to be paid in local currency.<br>The cashier has verified the accuracy of the transaction details. | |
| ***Postconditions in words (write in passive and pas tense)*** | The transaction is finalized.<br>The customer receives the exchanged foreign currency.<br>A receipt is printed for the customer.<br>The system updates the currency exchange rates and inventory.<br>CurrencyExchange was created and associated with Transaction.<br>Attribute Transaction.method was updated.<br>Attributes CurrencyExchange.amount, CurrencyExchange.transactionId, CurrencyExchange.methodOfTransaction were updated. | |
| **Trigger** | This event is triggered when the Traveler/customer gives the Agent card/check | |
| **Included Use Cases** | | |
| **Extending Use Cases** | | |
| ***MAIN SUCCESSFUL SCENARIO for this Use Case* in numbered sequence**<br><br>Reference “included use cases” in this section using INCLUDE *ius_name* | **Actor Action** | **System Action** |
| | Step 1. The cashier receives the cash payment from the customer. | Step 2. The system prompts the cashier to enter the received amount. |
| | Step 3. The cashier verifies the entered amount with the displayed final amount. | Step 4. The system validates the received amount against the final amount. |

| | | |
|---|---|---|
| | | Step 4. The system debits the customer's local currency account. |
| | Step 5. Agent confirms the transaction | Step 6. The system credits the system's internal account for the received local currency. |
| | | Step 7. The system credits the customer's account with the exchanged foreign currency. |
| | | Step 8. The system prints a receipt for the transaction. |
| | Step 9. The cashier provides the customer with the exchanged foreign currency with the chosen medium of choice and the receipt. | |
| | Step 11. Agent takes out cash and hands it to customer/traveler | Step 12. System stores transaction details |
| | | |
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful*** variations of the *normal* execution path, including any extension points using EXTEND *eus_name*) | Step 1.- The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | **Step 2. EXTEND CE5Ext1** Finalize Transaction By Check |
| | | |
| ***OTHER SUCCESSFUL SCENARIOS*** (Specify any ***successful*** variations of the *normal* execution path, including any extension points using EXTEND *eus_name*) | Step 1.- The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | **Step 2. EXTEND CE5Ext2** Finalize Transaction By Card |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1.- Agent prompts customer for desired medium of transaction | |
| | Step 2. Traveler provides the desired medium of transaction | Step 3. System checks if the cash option is available |
| | | Step 4. Cash option is unavailable |
| | | Step 5. System aborts transaction |
| | | |
| ***UNSUCCESSFUL SCENARIOS*** (*erroneous* situations*)* | Step 1. The Currency Exchange Agent greets the traveler/customer and asks for the currency to be exchanged. | |

| | | |
|---|---|---|
| | Step 2. The traveler/customer presents the cash to the agent. | Step 3. System checks if the selected option is available. |
| | | Step 4. Cash option is available |
| | Step 5. The agent verifies the authenticity of the cash and counts the amount provided by the traveler | |
| | Step 6. The traveler provides insufficient amount | Step 7. System aborts transaction |
| **Priority in scheduling** | High | |
| **Frequency** | Once per run | |
| **Business rules and data logic** | Customer information must be stored with each transaction per financial regulations. | |
| **Other non-functional requirements** | Responsive system performance for real-time transactions<br>Intuitive and easy to use interface | |
| **Superordinates** | | |
| **Developer** | David Nana Dwomoh Sarpong (dd993) | |
| **Creation date and last modified date** | 05/16/2024 | |
| Other Comments | | |

## 3. Class Model




### 3.1 Class Model Definitions

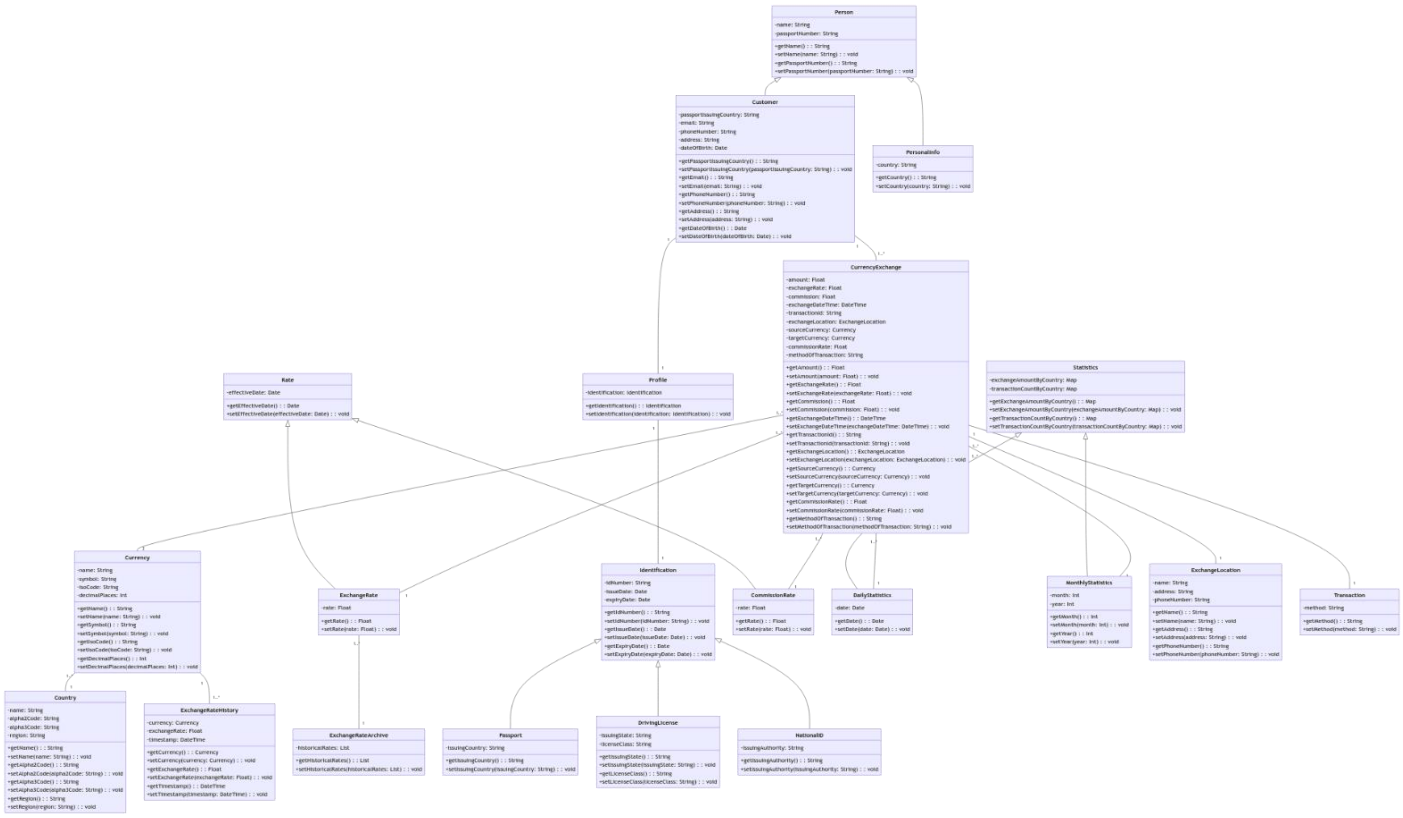
1. **PersonalInfo**: This class represents personal information about a person, specifically the country they are associated with.
2. **Profile**: This class holds identification details for a customer, linking to various forms of identification like Passport, DrivingLicense, or NationalID.
3. **Identification**: This class represents various forms of identification, characterized by an ID number, issue date, and expiry date.
4. **Passport**: This class is a type of Identification, specifically for passports, detailing the issuing country.
5. **DrivingLicense**: This class is a type of Identification, specifically for driving licenses, detailing the issuing state and license class.
6. **NationalID**: This class is a type of Identification, specifically for national IDs, detailing the issuing authority.
7. **CurrencyExchange**: This class represents the transaction details of currency exchange, including amount, exchange rate, commission, date/time of exchange, and transaction ID.
8. **Currency**: This class represents different currencies, characterized by name, symbol, ISO code, and decimal places.
9. **Rate**: This class acts as a base class for ExchangeRate and CommissionRate, capturing the effective date of the rate.

10. **ExchangeRate**: This class is a specialized Rate that captures the exchange rate between two currencies.
11. **CommissionRate**: This class is a specialized Rate that captures the commission rate applied during a currency exchange.
12. **ExchangeRateArchive**: This class keeps a record of historical exchange rates.
13. **ExchangeRateHistory**: This class represents the historical data of exchange rates for a specific currency, including the timestamp.
14. **Statistics**: This class captures statistical data on currency exchanges, such as the amount exchanged by country and transaction count by country.
15. **DailyStatistics**: This class represents statistics for currency exchanges on a daily basis, identified by the date.
16. **MonthlyStatistics**: This class represents statistics for currency exchanges on a monthly basis, identified by the month and year.
17. **Country**: This class represents a country, characterized by its name, alpha-2 code, alpha-3 code, and region.
18. **ExchangeLocation**: This class represents the location where currency exchanges take place, detailing the name, address, and phone number.
19. **Transaction**: This class represents the method used for a currency exchange transaction.

### 3.2 Selected Association Definitions

1. **CurrencyExchange "1..*" -- "2" Currency**: This association indicates that each CurrencyExchange transaction involves at least one source currency and one target currency. The multiplicity "1..*" for CurrencyExchange signifies that multiple currency exchanges can occur, each involving two currencies (one being exchanged for another).
2. **CurrencyExchange "1..*" -- "1" ExchangeRate**: Each CurrencyExchange is associated with one specific ExchangeRate. This means that multiple currency exchange transactions can refer to the same exchange rate, but each transaction is tied to exactly one rate.
3. **Currency "1..*" -- "1" Country**: Each Currency is associated with one Country. The multiplicity "1..*" on the Currency side signifies that multiple currencies can be associated with a single country, but each currency is tied to one country.

## 4. Design Class Diagram




## 4.1 Design Class Diagram Explanation

**Person**
Attributes:
name: The name of the person.
passportNumber: The person's passport number.

getName(): Retrieves the name of the person.
setName(name: String): Sets the name of the person.
getPassportNumber(): Retrieves the passport number.
setPassportNumber(passportNumber: String): Sets the passport number.

**Customer**
Attributes:
passportIssuingCountry: The country that issued the passport.
email: The email address of the customer.
phoneNumber: The customer's phone number.
address: The address of the customer.
dateOfBirth: The customer's date of birth.

**Operations:**
getPassportIssuingCountry(): Retrieves the passport issuing country.
setPassportIssuingCountry(passportIssuingCountry: String): Sets the passport issuing country.
getEmail(): Retrieves the email address.

setEmail(email: String): Sets the email address.
getPhoneNumber(): Retrieves the phone number.
setPhoneNumber(phoneNumber: String): Sets the phone number.
getAddress(): Retrieves the address.
setAddress(address: String): Sets the address.
getDateOfBirth(): Retrieves the date of birth.
setDateOfBirth(dateOfBirth: Date): Sets the date of birth.

**PersonalInfo**
Attributes:
country: The country information.
Operations:
getCountry(): Retrieves the country.
setCountry(country: String): Sets the country.

**Profile**
Attributes:
identification: Identification information associated with the profile.
Operations:
getIdentification(): Retrieves the identification information.
setIdentification(identification: Identification): Sets the identification information.

**Identification**
Attributes:
idNumber: Identification number.
issueDate: The date when the identification was issued.
expiryDate: The expiry date of the identification.
Operations:
getIdNumber(): Retrieves the identification number.
setIdNumber(idNumber: String): Sets the identification number.
getIssueDate(): Retrieves the issue date.
setIssueDate(issueDate: Date): Sets the issue date.
getExpiryDate(): Retrieves the expiry date.
setExpiryDate(expiryDate: Date): Sets the expiry date.
**Passport**
Attributes:
issuingCountry: The country that issued the passport.
Operations:
getIssuingCountry(): Retrieves the issuing country.
setIssuingCountry(issuingCountry: String): Sets the issuing country.

**DrivingLicense**
Attributes:
issuingState: The state that issued the driving license.
licenseClass: The class of the driving license.
Operations:
getIssuingState(): Retrieves the issuing state.
setIssuingState(issuingState: String): Sets the issuing state.
getLicenseClass(): Retrieves the license class.

setLicenseClass(licenseClass: String): Sets the license class.

**NationalID**
Attributes:
issuingAuthority: The authority that issued the national ID.
Operations:
getIssuingAuthority(): Retrieves the issuing authority.
setIssuingAuthority(issuingAuthority: String): Sets the issuing authority.

**CurrencyExchange**
Attributes:
amount: The amount of currency to be exchanged.
exchangeRate: The exchange rate applied.
commission: The commission fee for the exchange.
exchangeDateTime: The date and time of the exchange.
transactionId: The unique ID of the transaction.
exchangeLocation: The location where the exchange takes place.
sourceCurrency: The currency being exchanged from.
targetCurrency: The currency being exchanged to.
commissionRate: The rate of commission.
methodOfTransaction: The method used for the transaction.
Operations:
getAmount(): Retrieves the amount.
setAmount(amount: Float): Sets the amount.
getExchangeRate(): Retrieves the exchange rate.
setExchangeRate(exchangeRate: Float): Sets the exchange rate.
getCommission(): Retrieves the commission.
setCommission(commission: Float): Sets the commission.
getExchangeDateTime(): Retrieves the exchange date and time.
setExchangeDateTime(exchangeDateTime: DateTime): Sets the exchange date and time.
getTransactionId(): Retrieves the transaction ID.
setTransactionId(transactionId: String): Sets the transaction ID.
getExchangeLocation(): Retrieves the exchange location.
setExchangeLocation(exchangeLocation: ExchangeLocation): Sets the exchange location.
getSourceCurrency(): Retrieves the source currency.
setSourceCurrency(sourceCurrency: Currency): Sets the source currency.
getTargetCurrency(): Retrieves the target currency.
setTargetCurrency(targetCurrency: Currency): Sets the target currency.
getCommissionRate(): Retrieves the commission rate.
setCommissionRate(commissionRate: Float): Sets the commission rate.
getMethodOfTransaction(): Retrieves the method of transaction.
setMethodOfTransaction(methodOfTransaction: String): Sets the method of transaction.

**Currency**
Attributes:
name: The name of the currency.
symbol: The symbol of the currency.

isoCode: The ISO code of the currency.
decimalPlaces: The number of decimal places the currency supports.
Operations:
getName(): Retrieves the name of the currency.
setName(name: String): Sets the name of the currency.
getSymbol(): Retrieves the symbol.
setSymbol(symbol: String): Sets the symbol.
getIsoCode(): Retrieves the ISO code.
setIsoCode(isoCode: String): Sets the ISO code.
getDecimalPlaces(): Retrieves the number of decimal places.
setDecimalPlaces(decimalPlaces: Int): Sets the number of decimal places.

**Rate**
Attributes:
effectiveDate: The date the rate is effective from.
Operations:
getEffectiveDate(): Retrieves the effective date.
setEffectiveDate(effectiveDate: Date): Sets the effective date.

**ExchangeRate**
Attributes:
rate: The exchange rate value.
Operations:
getRate(): Retrieves the exchange rate.
setRate(rate: Float): Sets the exchange rate.

**CommissionRate**
Attributes:
rate: The commission rate value.
Operations:
getRate(): Retrieves the commission rate.
setRate(rate: Float): Sets the commission rate.

**ExchangeRateArchive**
Attributes:
historicalRates: A list of historical exchange rates.
Operations:
getHistoricalRates(): Retrieves the list of historical rates.
setHistoricalRates(historicalRates: List<ExchangeRate>): Sets the list of historical rates.

**ExchangeRateHistory**
Attributes:
currency: The currency associated with the exchange rate history.
exchangeRate: The historical exchange rate value.
timestamp: The timestamp of the historical rate.
Operations:
getCurrency(): Retrieves the currency.
setCurrency(currency: Currency): Sets the currency.

getExchangeRate(): Retrieves the exchange rate.
setExchangeRate(exchangeRate: Float): Sets the exchange rate.
getTimestamp(): Retrieves the timestamp.
setTimestamp(timestamp: DateTime): Sets the timestamp.

**Statistics**
Attributes:
exchangeAmountByCountry: A map of exchange amounts by country.
transactionCountByCountry: A map of transaction counts by country.
Operations:
getExchangeAmountByCountry(): Retrieves the exchange amount by country.
setExchangeAmountByCountry(exchangeAmountByCountry: Map<Country, Float>): Sets the exchange amount by country.
getTransactionCountByCountry(): Retrieves the transaction count by country.
setTransactionCountByCountry(transactionCountByCountry: Map<Country, Int>): Sets the transaction count by country.

**DailyStatistics**
Attributes:
date: The date for daily statistics.
Operations:
getDate(): Retrieves the date.
setDate(date: Date): Sets the date.

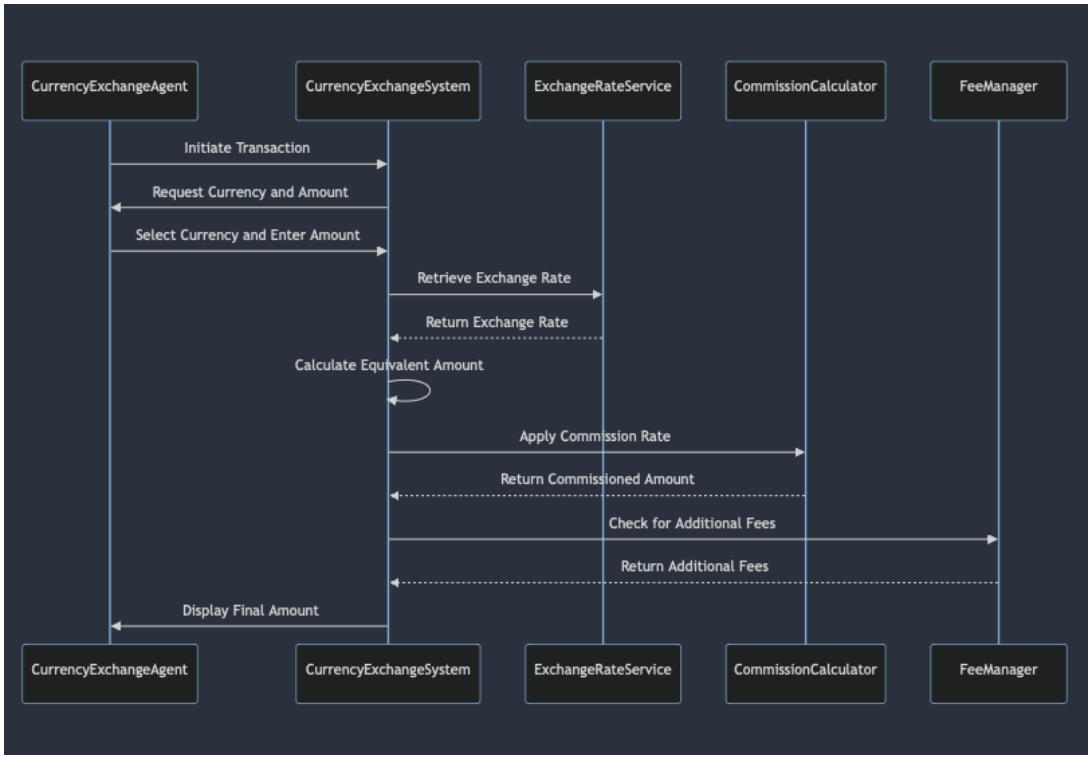
**MonthlyStatistics**
Attributes:
month: The month for monthly statistics.
year: The year for monthly statistics.
Operations:

getMonth(): Retrieves the month.
setMonth(month: Int): Sets the month.
getYear(): Retrieves the year.
setYear(year: Int): Sets the year.

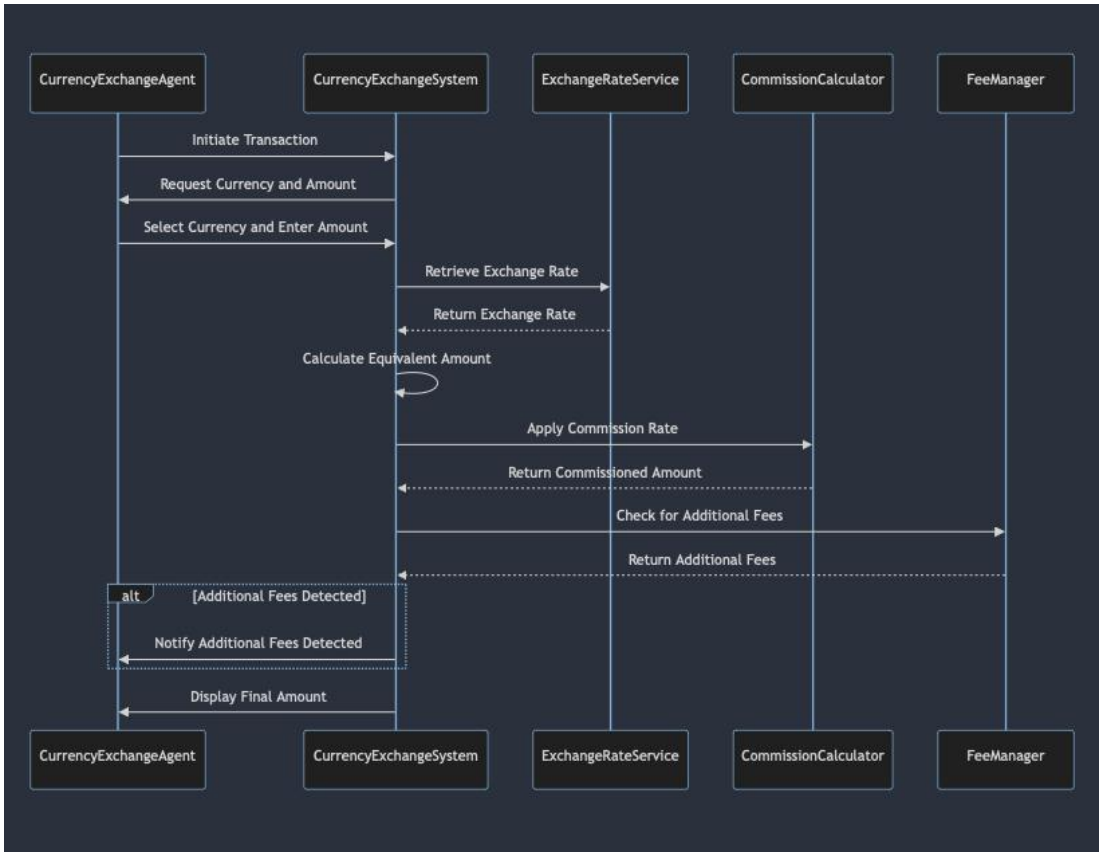
**Country**
Attributes:
name: The name of the country.
alpha2Code: The alpha-2 code of the country.
alpha3Code: The alpha-3 code of the country.
region: The region the country belongs to.
Operations:
getName(): Retrieves the name of the country.
setName(name: String): Sets the name of the country.
getAlpha2Code(): Retrieves the alpha-2 code.
setAlpha2Code(alpha2Code: String): Sets the alpha-2 code.

## 4.2 System Sequence Diagram

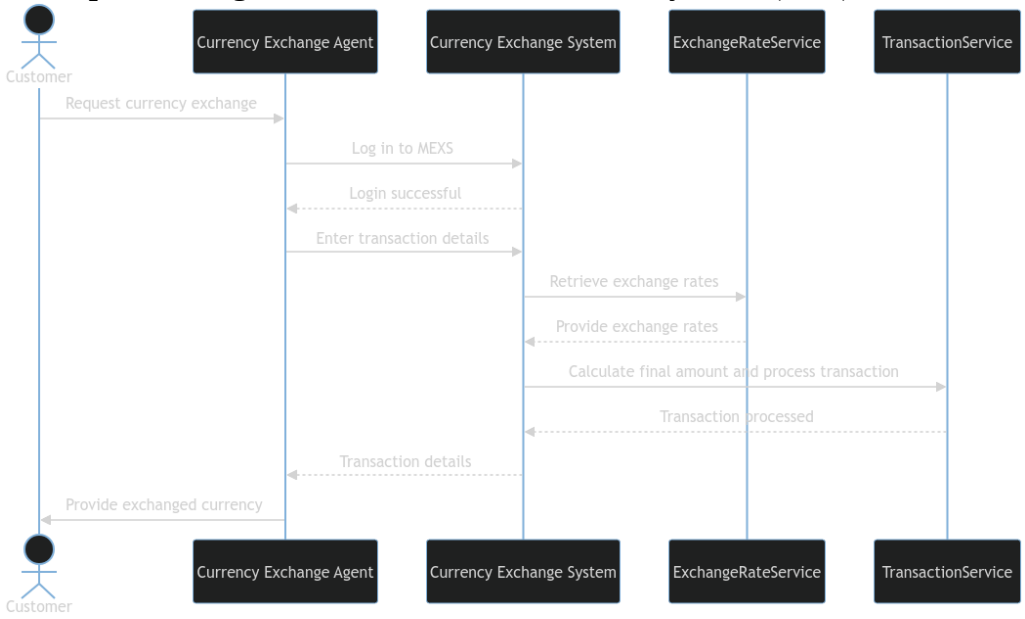
### System Sequence Diagram for Calculate Final Amount CE1- Anushka Awasthi



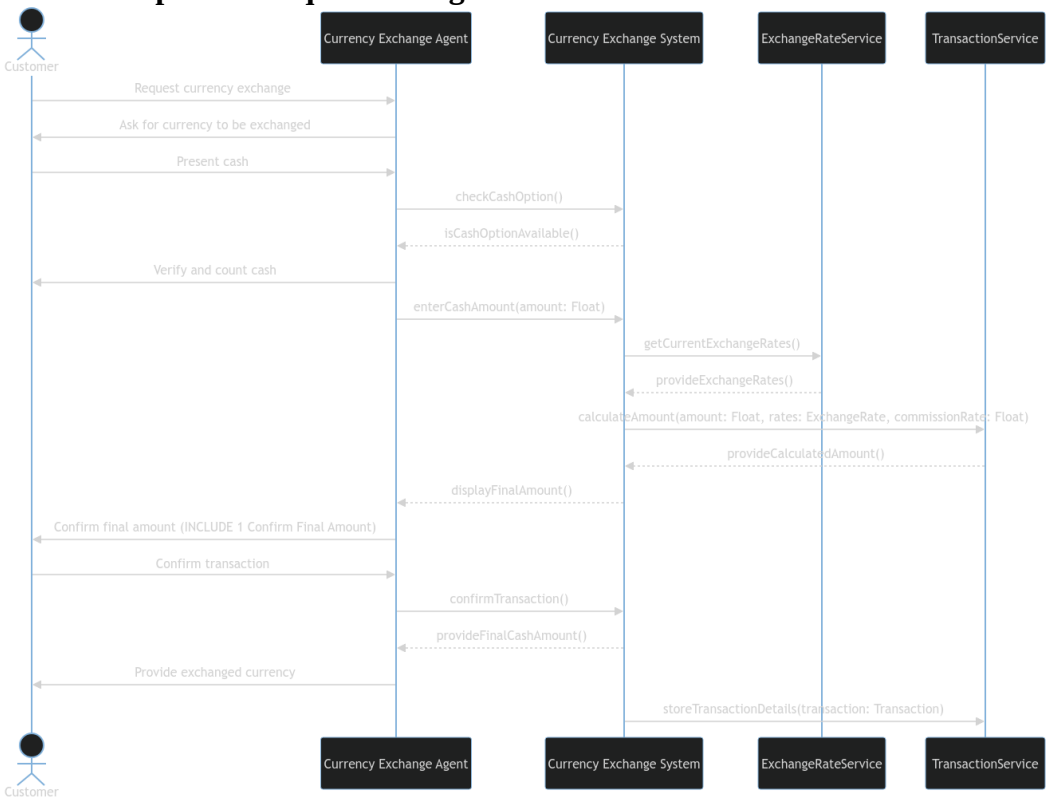

### Expanded Sequence Diagram for CE1- Anushka Awasthi



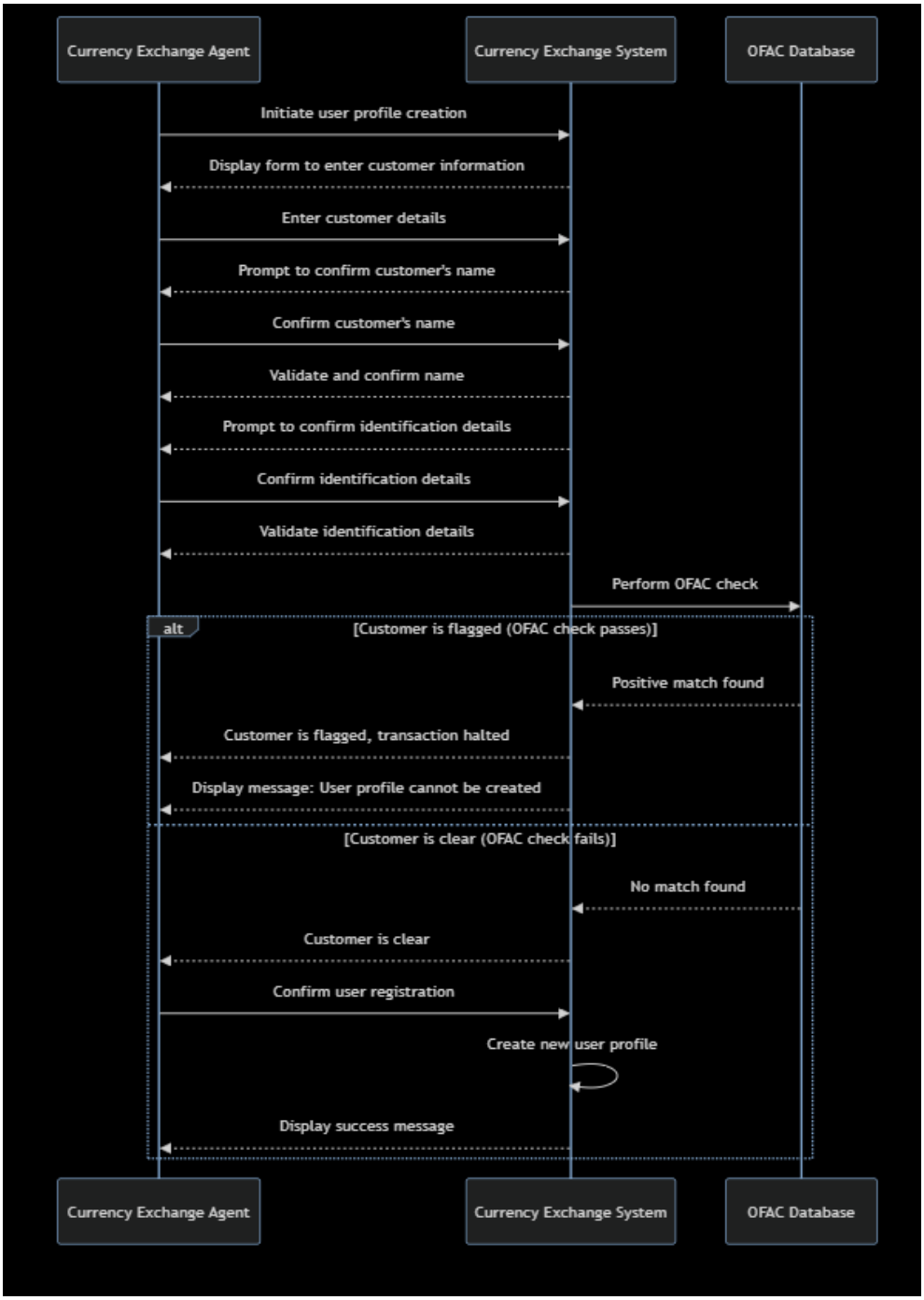
## System Sequence Diagram for Finalize Transaction by Cash (CE5) – David Nana




## Expanded Sequence Diagram for CE5 – David Nana

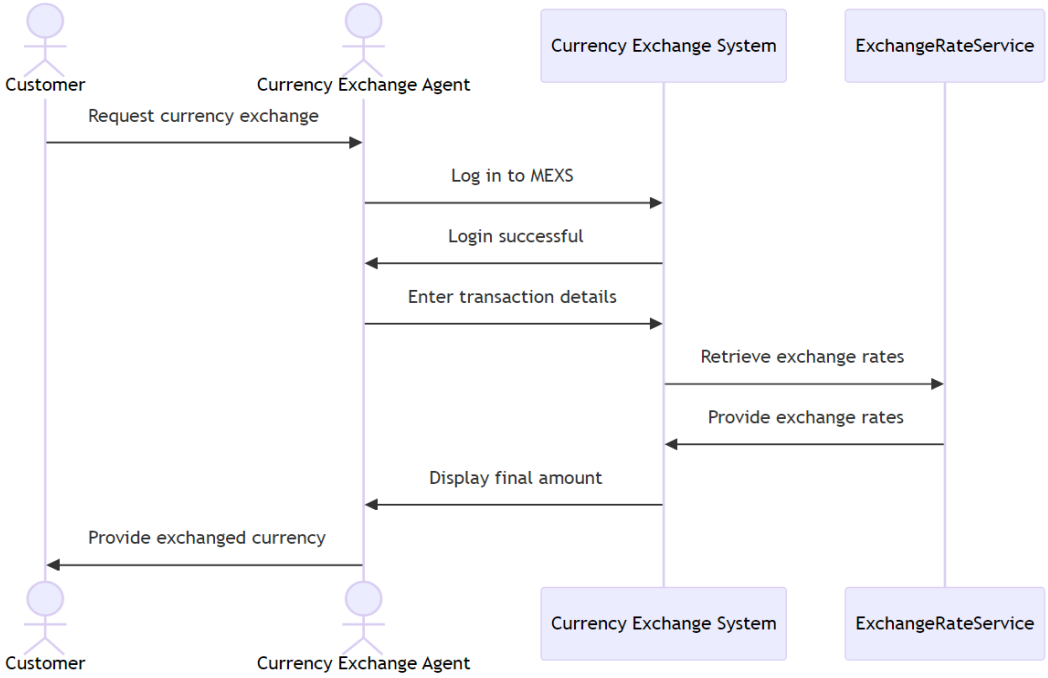


## System Sequence Diagram for Create User Profile – Rishabh Pandey

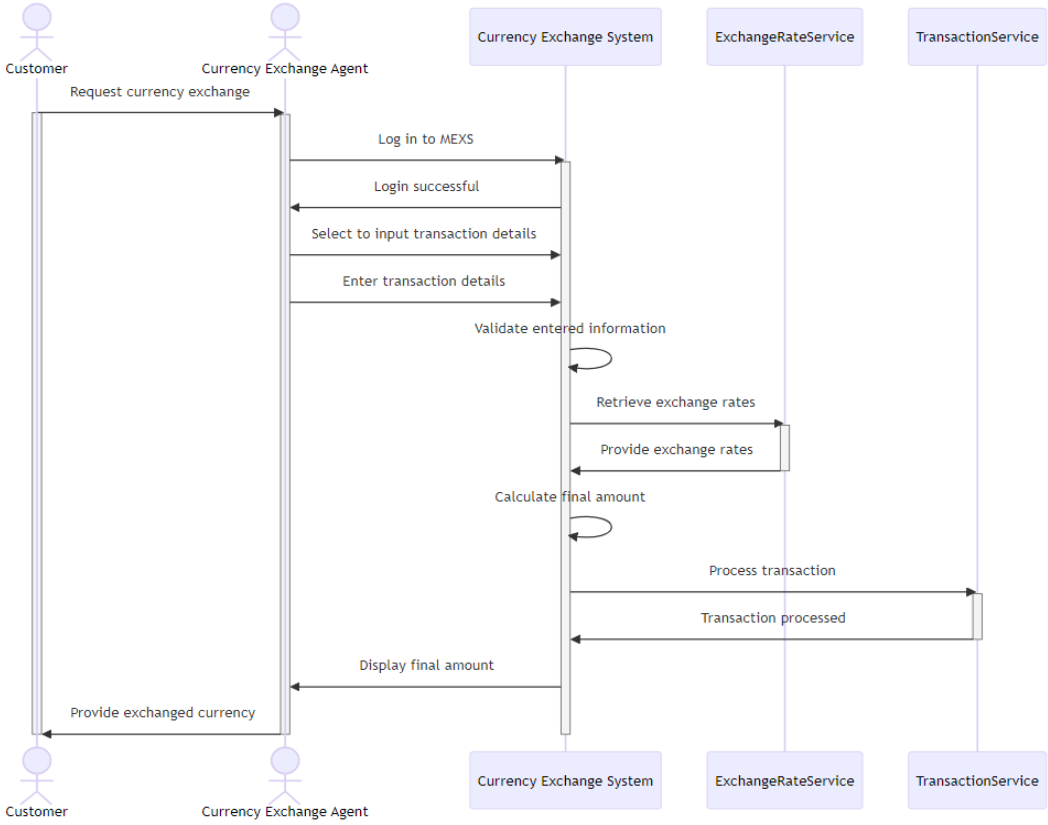


## Expanded Sequence Diagram for Create User Profile – Rishabh Pandey

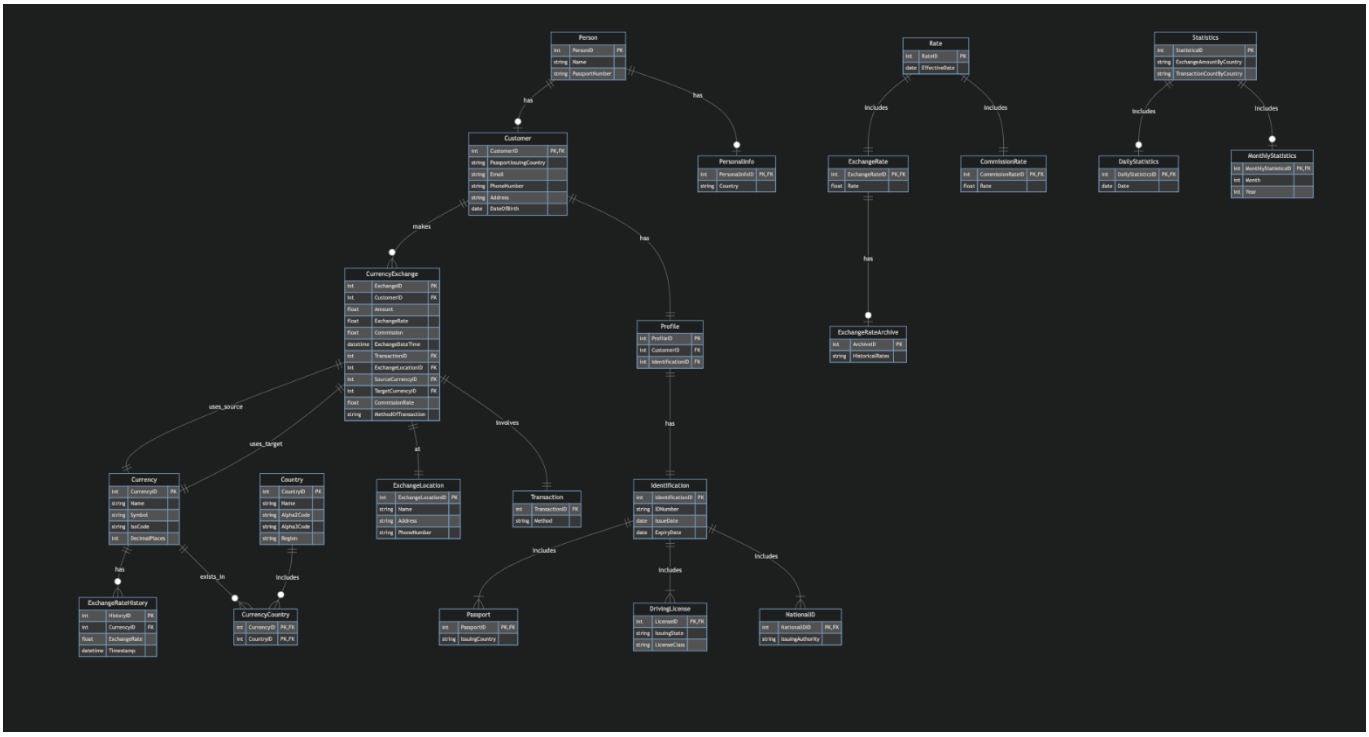
## System Sequence Diagram for CE4 - Input Transaction Details – Mohamed Ashiq




## Expanded Sequence Diagram for CE4 - Input Transaction Details - Mohamed Ashiq



## 5. Relational Database Schema




### 5.1 Relational Database Schema Explanation

The database schema consists of several interconnected tables designed to manage information related to persons, customers, identifications, currency exchanges, rates, and statistics. The Person table stores basic personal details, uniquely identified by PersonID, with attributes such as Name and PassportNumber. The Customer table, referencing PersonID, includes additional customer-specific details like PassportIssuingCountry, Email, PhoneNumber, Address, and DateOfBirth. Each customer can have a Profile linked to an Identification, which can be a Passport, DrivingLicense, or NationalID, each having attributes like IDNumber, IssueDate, and ExpiryDate.

The CurrencyExchange table tracks currency exchange transactions, including details such as Amount, ExchangeRate, Commission, ExchangeDateTime, and foreign keys to Customer, Transaction, ExchangeLocation, SourceCurrency, and TargetCurrency. Each exchange location and transaction method is recorded in the ExchangeLocation and Transactiontables respectively. Currency details are stored with attributes like Name, Symbol, IsoCode, and DecimalPlaces.

Rates are managed by the Rate table with an effective date, and specific rates are detailed in ExchangeRate and CommissionRate tables, each referencing the Rate table. Historical exchange rates are archived in the ExchangeRateArchive table, while the ExchangeRateHistory table logs exchange rates over time for different currencies. The Statistics table captures exchange amounts and transaction counts by country, with daily and monthly statistics recorded in DailyStatistics and MonthlyStatistics tables.

The Country table lists countries with attributes like Name, Alpha2Code, Alpha3Code, and Region, and a many-to-many relationship with Currency through the CurrencyCountrytable, linking currencies to countries where they are used. This schema provides a comprehensive framework for managing detailed currency exchange information, identification records, and associated statistics.