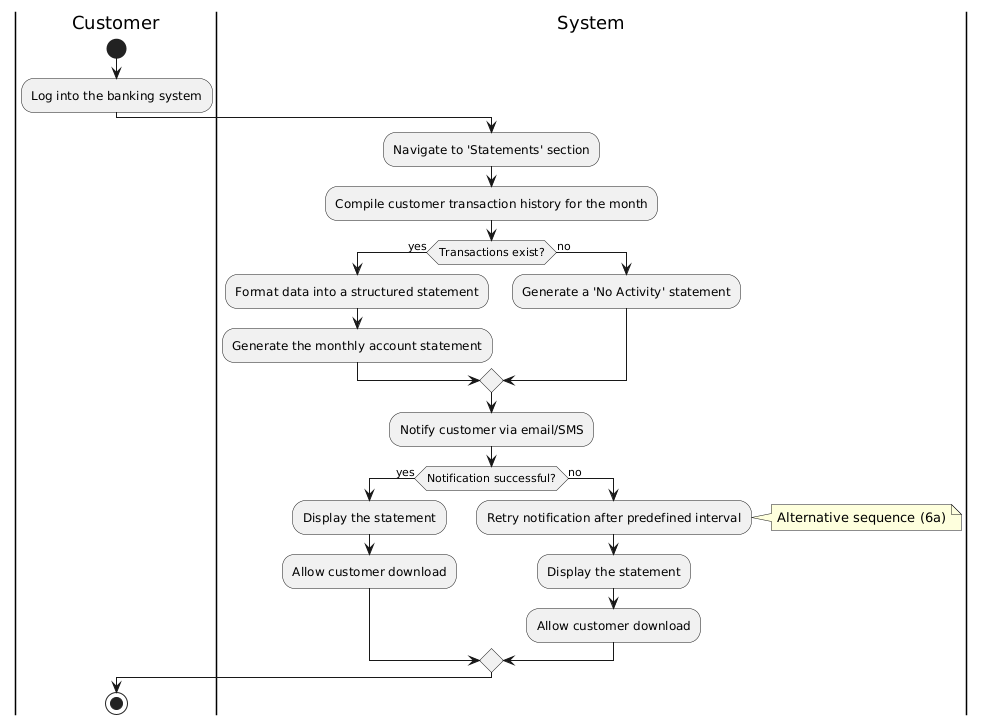
# UC\_17 Generate Monthly Account Statements

|  |  |
| --- | --- |
| UC Name | UC\_17 Generate Monthly Account Statements |
| Summary | This use case describes how the system generates and provides customers with their monthly account statements. |
| Dependency | None |
| Actors | Primary Actor: Customer |
| Preconditions | - The customer must have an active bank account. - The system must have transaction data for the customer. |
| Main Sequence | 1. Customer logs into the banking system. 2. Customer navigates to the 'Statements' section. 3. System compiles customer transaction history for the month. 4. System formats the data into a structured statement. 5. System generates the monthly account statement. 6. System notifies the customer via email/SMS. 7. System displays and allows the customer to download the statement. |
| Alternative Sequence | 3a. If no transactions exist, the system generates a 'No Activity' statement. 6a. If notification fails, the system retries after a predefined interval. |
| Non-functional Requirements | The statement generation should not exceed 10 seconds. Statements should be available in PDF format. The system must ensure secure access. |
| Postconditions | The customer receives a monthly statement. The statement is securely stored for future reference. |
|  |  |

# 



# UC\_18 Apply Security Measures for Online Transactions

|  |  |
| --- | --- |
| UC Name | UC\_18 Apply Security Measures for Online Transactions |
| Summary | This use case describes how the system enforces security measures for online banking transactions. |
| Dependency | None |
| Actors | Primary Actor: Customer |
| Preconditions | - The customer must have an active online banking account. - The system must have security protocols in place. |
| Main Sequence | 1. Customer initiates an online transaction. 2. System verifies authentication via login credentials. 3. System prompts for multi-factor authentication (MFA). 4. Customer provides required authentication (OTP, biometrics, etc.). 5. System validates authentication data. 6. System securely processes the transaction. 7. System notifies the customer of the transaction status. |
| Alternative Sequence | 3a. If MFA verification fails, the system denies the transaction. 5a. If authentication fails, the system logs the attempt and alerts the customer. |
| Non-functional Requirements | Transactions must be encrypted end-to-end. Authentication should not exceed 5 seconds. The system must support multiple security mechanisms. |
| Postconditions | Online transactions are securely processed. Unauthorized access attempts are logged. |
|  |  |

# 

A diagram of a system

AI-generated content may be incorrect.

# UC\_19 Perform Automatic Currency Conversion for International Transactions

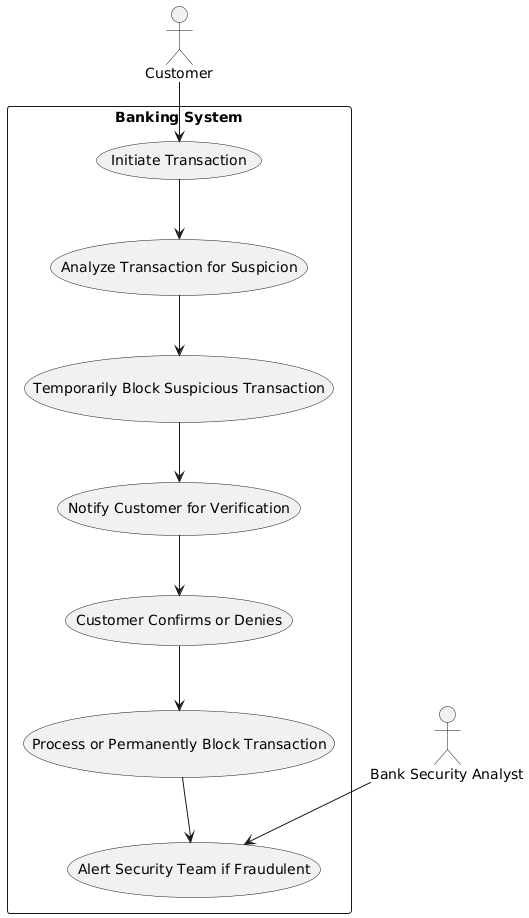
|  |  |
| --- | --- |
| UC Name | UC\_19 Perform Automatic Currency Conversion for International Transactions |
| Summary | This use case describes how the system automatically converts currencies for international transactions. |
| Dependency | None |
| Actors | Primary Actor: Customer |
| Preconditions | - The customer must have an active bank account. - The transaction must involve a foreign currency. |
| Main Sequence | 1. Customer initiates an international transaction. 2. System detects currency mismatch. 3. System fetches latest exchange rate. 4. System converts transaction amount to recipient's currency. 5. System processes transaction and deducts amount. 6. System notifies customer of transaction details and conversion rate. |
| Alternative Sequence | 3a. If exchange rate is unavailable, system retries after a delay. 5a. If conversion fails, the system cancels transaction and notifies customer. |
| Non-functional Requirements | Exchange rate retrieval should not exceed 3 seconds. System must use real-time exchange rates from a reliable source. System must provide transparent fee information. |
| Postconditions | Transaction is completed with correct currency conversion. Customer is informed of transaction details. |
|  |  |

# 

A diagram of a system

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| UC Name | UC\_20 Block Suspicious Transactions |
| Summary | This use case describes how the system identifies and blocks suspicious transactions. |
| Dependency | None |
| Actors | Primary Actor: Customer Secondary Actor: Bank Security Team |
| Preconditions | - The customer must have an active bank account. - The system must have fraud detection mechanisms. |
| Main Sequence | 1. Customer initiates a transaction. 2. System analyzes the transaction against fraud detection rules. 3. If flagged as suspicious, system temporarily blocks it. 4. System notifies customer and requests verification. 5. Customer confirms or denies transaction. 6. If confirmed, system processes transaction. 7. If denied, system permanently blocks it and alerts security team. |
| Alternative Sequence | 3a. If system incorrectly flags a transaction, customer can appeal for review. 5a. If customer does not respond, transaction remains blocked. |
| Non-functional Requirements | Suspicious transaction detection should occur in real-time. System must use machine learning or rule-based detection. Customer verification requests should not exceed 5 seconds. |
| Postconditions | Fraudulent transactions are prevented. Legitimate transactions are processed after verification. |

****

