SWIFT Integration and Monitoring System

Phase 1:

Phase 1.1: Requirements and Design (11 Points)

1. **Transaction Processing:** The system must be able to process financial transactions, including credit transfers, interbank transfers, and foreign exchange confirmations.
2. **Message Validation:** The system must validate incoming SWIFT and ISO 20022 messages against predefined rules to ensure data integrity.
3. **Alert Management:** The system must generate and manage alerts for system errors, security breaches, and other significant events.
4. **Reporting:** The system should provide reporting capabilities for transactions, message validation logs, and system activities.
5. **User and Role Management:** The system must manage user accounts and assign roles with specific access permissions to different functionalities and data.
6. **BIC Directory Management:** The system should allow for managing and accessing a directory of Bank Identifier Codes (BICs).
7. **Currency Management:** The system needs to manage currency exchange rates and ensure transactions are processed using active currencies.
8. **Account Management:** The system should handle International Bank Account Numbers (IBANs), including their status and associated account types.

9. **System Monitoring:** The system must log system activities and errors for auditing and troubleshooting purposes.

Phase 1.2. Functional Requirements (7-10 Points)

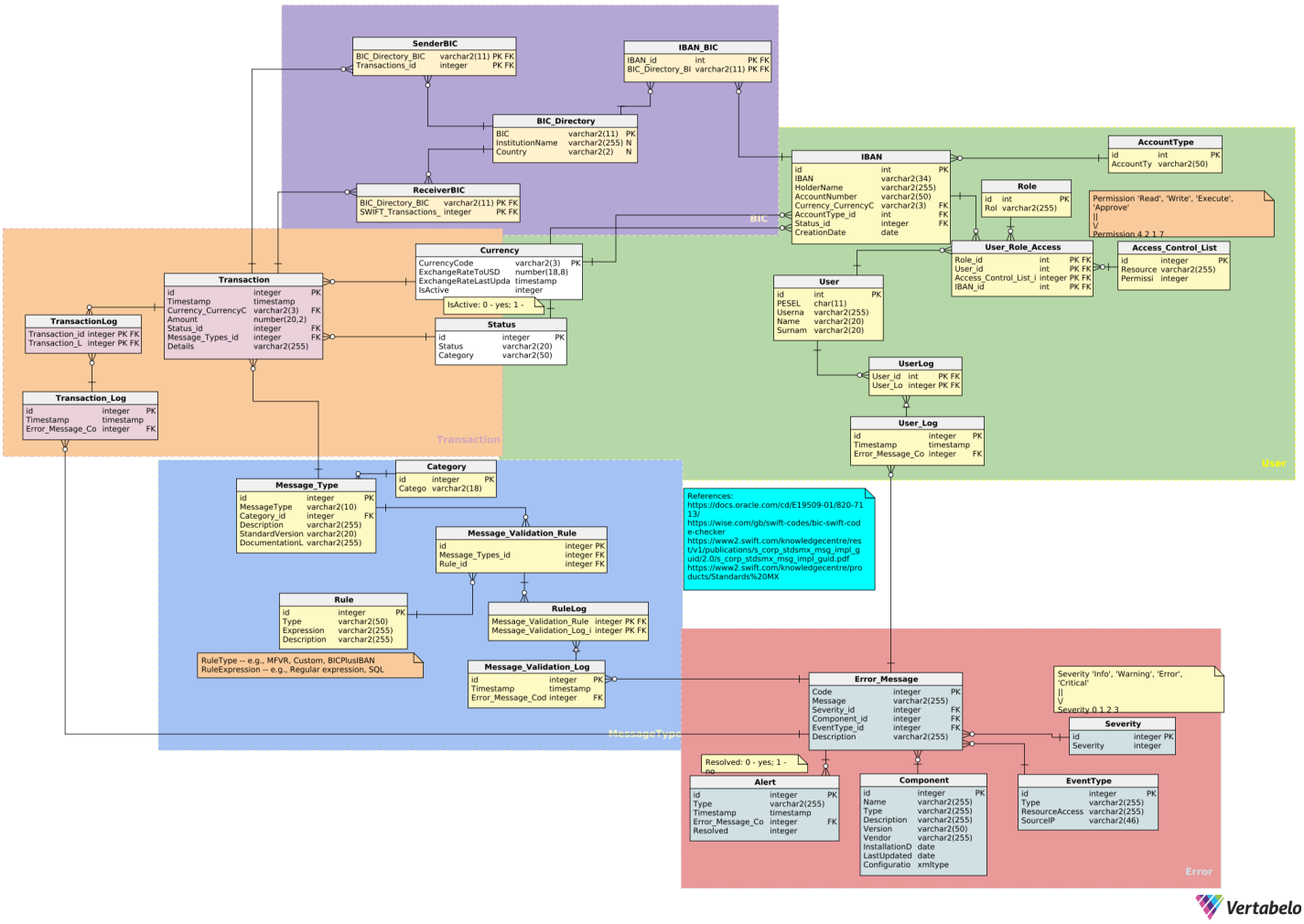
1. **Validate Incoming Messages:** The system shall validate incoming financial messages (MT and ISO 20022 formats) against defined validation rules.
2. **Process Credit Transfers:** The system shall process single customer credit transfer messages (MT103) and update transaction statuses.
3. **Record Transaction Details:** The system shall record all relevant details of processed transactions, including timestamp, currency, amount, and involved parties.
4. **Generate Alerts for Validation Failures:** The system shall generate alerts when incoming messages fail validation rules, including the error message code and timestamp.
5. **Search Transaction History:** The system shall allow users to search and view transaction history based on criteria such as date range, account number, or status.
6. **Manage User Roles and Permissions:** The system shall allow administrators to create and manage user roles and assign specific permissions to access system functionalities and data.
7. **Maintain BIC Directory:** The system shall allow authorized users to add, update, and view BIC directory information.
8. **Update Currency Exchange Rates:** The system shall allow authorized users to update currency exchange rates, including the last updated timestamp.
9. **Log User Login Activities:** The system shall log user login and logout activities, including timestamps and source IP addresses.
10. **Generate Message Validation Reports:** The system shall generate reports on message validation activities, including the number of successful and failed validations and the types of validation errors.

Phase 1.3. Entity Relationship Diagram (ERD) (Minimum 6 Entities)

here

PS: link to vertabelo

|
|
|
|
|
|
|
|
\/



Phase 2:

Phase 2.1: Database Implementation and Queries (19 Points)

Data Definition Language | Data Manipulation Language

```
CREATE TABLE "User" (
id INTEGER NOT NULL,
PESEL CHAR(11) NOT NULL,
Username VARCHAR2(255) NOT NULL,
Name VARCHAR2(20) NOT NULL,
Surname VARCHAR2(20) NOT NULL,
CONSTRAINT Users_pk PRIMARY KEY (id)
);


CREATE TABLE BIC_Directory (
id INTEGER NOT NULL,
BIC VARCHAR2(11) NOT NULL,
InstitutionName VARCHAR2(255) NULL,
Country VARCHAR2(2) NULL,
CONSTRAINT BIC_Directory_pk PRIMARY KEY (ID),
```

```
-- USER ----------------------------------------
INSERT INTO "User" (id, PESEL, Username, Name,
Surname) VALUES (1, '8001011234', 'akrava',
'Anastasiia', 'Kravchenko');

INSERT INTO Role (id, Role) VALUES (1,
'Administrator');

INSERT INTO Access_Control_List (id, "Resource",
Permission) VALUES (1, 'User Management', 7); --
Read + Write + Execute

INSERT INTO AccountType (id, AccountType) VALUES
(1, 'Savings');
```

```
CONSTRAINT BIC_Directory_uk UNIQUE (BIC)
);


CREATE TABLE IBAN (
id INTEGER NOT NULL,
IBAN VARCHAR2(34) NOT NULL,
HolderName VARCHAR2(255) NOT NULL,
AccountNumber VARCHAR2(50) NOT NULL,
Currency_CurrencyCode VARCHAR2(3) NOT NULL,
AccountType_id INTEGER NOT NULL,
Status_id INTEGER NOT NULL,
CreationDate DATE NOT NULL,
CONSTRAINT IBAN_pk PRIMARY KEY (id),
CONSTRAINT IBAN_uk UNIQUE (IBAN)
);


CREATE TABLE Transaction (
id INTEGER NOT NULL,
Timestamp TIMESTAMP NOT NULL,
Currency_CurrencyCode VARCHAR2(3) NOT NULL,
Amount NUMBER(20,2) NOT NULL,
Status_id INTEGER NOT NULL,
Message_Types_id INTEGER NOT NULL,
Details VARCHAR2(255) NOT NULL,
CONSTRAINT Transaction_pk PRIMARY KEY (id)
);


CREATE TABLE Error_Message (
Code INTEGER NOT NULL,
Message VARCHAR2(255) NOT NULL,
Severity_id INTEGER NOT NULL,
Component_id INTEGER NOT NULL,
EventType_id INTEGER NOT NULL,
Description VARCHAR2(255) NOT NULL,
CONSTRAINT Error_Message_pk PRIMARY KEY (Code)
);
```

```
INSERT INTO Status (id, Status, Category) VALUES (1,
'Pending', 'Transaction');

INSERT INTO Currency (CurrencyCode,
ExchangeRateToUSD, ExchangeRateLastUpdated,
IsActive) VALUES ('USD', 1.00000000,
TO_TIMESTAMP('2025-03-24 12:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 1);

INSERT INTO IBAN (id, IBAN, HolderName,
AccountNumber, Currency_CurrencyCode,
AccountType_id, Status_id, CreationDate) VALUES (1,
'GB82WEST12345698765432', 'John Smith',
'ACC123456', 'GBP', 2, 6, TO_DATE('2024-01-10',
'YYYY-MM-DD'));

INSERT INTO User_Role_Access (Role_id, User_id,
Access_Control_List_id, IBAN_id) VALUES (1, 1, 1, 1);

-- BIC ----------------------------------------
INSERT INTO BIC_Directory (ID, BIC, InstitutionName,
Country) VALUES (1, 'ABCDUSNYXXX', 'Bank of
America', 'US');

INSERT INTO IBAN_BIC (IBAN_id, BIC_Directory_id)
VALUES (1, 2);

-- TRANSACTION ----------------------------------------
INSERT INTO Category (id, Category) VALUES (1,
'MT1xx');

INSERT INTO Message_Type (id, MessageType,
Category_id, Description, StandardVersion,
DocumentationLink) VALUES (1, 'MT103', 1, 'Single
Customer Credit Transfer', '2023',
'https://www.swift.com/mt103');

INSERT INTO Transaction (id, Timestamp,
Currency_CurrencyCode, Amount, Status_id,
Message_Types_id, Details) VALUES (1,
TO_TIMESTAMP('2025-03-24 12:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 'USD', 150000.00, 2, 1, 'Payment for
services');

INSERT INTO SenderBIC (BIC_Directory_id,
Transactions_id) VALUES (1, 1);

INSERT INTO ReceiverBIC (BIC_Directory_id,
Transactions_id) VALUES (2, 1);

-- ERROR ----------------------------------------
INSERT INTO Severity (id, Severity) VALUES (1,
```

-- ONLY IMPORTANT MORE ON MY GitHub --

```
'Informational');

INSERT INTO Component (id, Name, Type,
Description, Version, Vendor, InstallationDate,
LastUpdated, Configuration) VALUES (1, 'Alliance
Access', 'Software', 'SWIFT messaging interface', '7.4',
'SWIFT', TO_DATE('2023-01-15', 'YYYY-MM-DD'),
TO_DATE('2023-06-20', 'YYYY-MM-DD'),
XMLType('<config><interface>TCP/IP</interface></co
nfig>'));

INSERT INTO EventType (id, Type, ResourceAccessed,
SourceIP) VALUES (1, 'System', 'Message Processing',
'127.0.0.1');

INSERT INTO Error_Message (Code, Message,
Severity_id, Component_id, EventType_id,
Description) VALUES (101, 'System Error
Encountered', 3, 1, 1, 'An unexpected system error
occurred during message parsing.');

INSERT INTO Alert (id, Type, Timestamp,
Error_Message_Code, Resolved, Status_id) VALUES (1,
'System Error', TO_TIMESTAMP('2025-03-24 10:00:00',
'YYYY-MM-DD HH24:MI:SS'), 101, 0, 10);

INSERT INTO Transaction_Log (id, Timestamp,
Error_Message_Code) VALUES (1,
TO_TIMESTAMP('2025-03-24 12:00:10', 'YYYY-MM-DD
HH24:MI:SS'), NULL);

INSERT INTO TransactionLog (Transaction_id,
Transaction_Log_id) VALUES (1, 1);

INSERT INTO User_Log (id, Timestamp,
Error_Message_Code) VALUES (1,
TO_TIMESTAMP('2025-03-24 09:00:00', 'YYYY-MM-DD
HH24:MI:SS'), NULL);

INSERT INTO UserLog (User_id, User_Log_id) VALUES
(1, 1);

INSERT INTO Message_Validation_Log (id, Timestamp,
Error_Message_Code) VALUES (1,
TO_TIMESTAMP('2025-03-24 12:30:00', 'YYYY-MM-DD
HH24:MI:SS'), 603);

INSERT INTO Rule (id, Type, Expression, Description)
VALUES (1, 'Regex', '^MT103.*', 'Message type must
start with MT103');

INSERT INTO Message_Validation_Rule (id,
```

```
Message_Types_id, Rule_id) VALUES (1, 1, 1);

INSERT INTO RuleLog (Message_Validation_Rule_id,
Message_Validation_Log_id) VALUES (1, 1);
```

Phase 2.2. DQL Instructions (10 Queries, 17 Points):

Click -----> Instructions <----- Click

```
-- 1. SELECT statement that joins at least two tables and contains WHERE clause
-- Find all transactions with an amount greater than 100000 USD.
SELECT
  t.id,
  t.Timestamp,
  t.Amount,
  c.CurrencyCode
FROM
  Transaction t
    JOIN
  Currency c ON t.Currency_CurrencyCode = c.CurrencyCode
WHERE
  t.Amount > 100000 AND c.CurrencyCode = 'USD';

-- 2. SELECT statement that joins at least two tables and contains WHERE clause
-- Find the names of users who have the 'Administrator' role.
SELECT
  u.Name,
  u.Surname
FROM
  "User" u
    JOIN
  User_Role_Access ura ON u.id = ura.User_id
    JOIN
  Role r ON ura.Role_id = r.id
WHERE
  r.Role = 'Administrator';

-- 3. SELECT statement that joins at least two tables and contains WHERE clause
-- List all alerts that are related to 'Security Breach' and are not yet resolved.
SELECT
  a.id,
  a.Timestamp,
  em.Message
FROM
  Alert a
    JOIN
  Error_Message em ON a.Error_Message_Code = em.Code
WHERE
  a.Type = 'Security Breach' AND a.Resolved = 0;




-- 4. SELECT statement that joins at least two tables and contains GROUP BY and HAVING clauses
```

```sql
-- Find the number of transactions for each currency where the count is greater than 1.
SELECT
  c.CurrencyCode,
  COUNT(t.id) AS NumberOfTransactions
FROM
  Transaction t
    JOIN
  Currency c ON t.Currency_CurrencyCode = c.CurrencyCode
GROUP BY
  c.CurrencyCode
HAVING
  COUNT(t.id) > 1;

-- 5. SELECT statement that joins at least two tables and contains GROUP BY and HAVING clauses
-- Find the average transaction amount for each message type, only for averages greater than 90000.
SELECT
  mt.MessageType,
  AVG(t.Amount) AS AverageTransactionAmount
FROM
  Transaction t
    JOIN
  Message_Type mt ON t.Message_Types_id = mt.id
GROUP BY
  mt.MessageType
HAVING
  AVG(t.Amount) > 90000;

-- 6. SELECT statement that joins at least two tables and contains GROUP BY and HAVING clauses
-- Find the number of message validation logs for each error message code, only showing codes with more than one log.
SELECT
  em.Message,
  COUNT(mvl.id) AS NumberOfValidationLogs
FROM
  Message_Validation_Log mvl
    JOIN
  Error_Message em ON mvl.Error_Message_Code = em.Code
GROUP BY
  em.Message
HAVING
  COUNT(mvl.id) > 1;




-- 7. SELECT statement with subquery
-- Find all IBANs that are associated with the 'Barclays Bank'.
SELECT
  i.IBAN
FROM
  IBAN i
WHERE
  i.id IN (SELECT IBAN_id FROM IBAN_BIC WHERE BIC_Directory_BIC = (SELECT BIC FROM BIC_Directory WHERE
InstitutionName = 'Barclays Bank'));

-- 8. SELECT statement with subquery
-- Find all users who have access to the 'Account Access' resource.
SELECT
  u.Name,
```

```
    u.Surname
FROM
    "User" u
WHERE
    u.id IN (SELECT User_id FROM User_Role_Access WHERE Access_Control_List_id = (SELECT id FROM Access_Control_List
WHERE "Resource" = 'Account Access'));




-- 9. SELECT statement with correlated subquery
-- Find all transactions that have an amount greater than the average amount for their currency.
SELECT
    t.id,
    t.Amount,
    c.CurrencyCode
FROM
    Transaction t
        JOIN
    Currency c ON t.Currency_CurrencyCode = c.CurrencyCode
WHERE
    t.Amount > (SELECT AVG(Amount) FROM Transaction WHERE Currency_CurrencyCode = t.Currency_CurrencyCode);

-- 10. SELECT statement with correlated subquery
-- Find all message validation logs that occurred after the timestamp of their corresponding alert.
SELECT
    mvl.id,
    mvl.Timestamp AS ValidationLogTimestamp,
    a.Timestamp AS AlertTimestamp
FROM
    Message_Validation_Log mvl
        JOIN
    Alert a ON mvl.Error_Message_Code = a.Error_Message_Code
WHERE
    mvl.Timestamp > a.Timestamp;
```