Overview of the Bank Application

This is a secure bank application built using **Streamlit** for the user interface, **Supabase** for the backend database, and **AES encryption** for secure data storage and communication. The app allows users to securely register, log in, view their balance, and make transactions between users with AES encryption protecting sensitive data.

Key Components and Workflow

1. Supabase Integration

- **Supabase** is used as the backend database for storing user data (e.g., usernames, passwords, balance, and transaction records).
- It is initialized using `st.secrets` for secure access to the API keys and URL from Streamlit's secrets management.
- Functions like `create_user`, `get_user`, `update_balance`, and `create_transaction` interact with Supabase to store and retrieve data.

2. Encryption and Decryption

- **AES Encryption** (Advanced Encryption Standard) is used to securely store user balances and transaction details.
- The key for AES encryption is fetched from Streamlit's secrets management system (`st.secrets["encryption_key"]`), ensuring the key is of a valid size (16, 24, or 32 bytes).
- Encryption is done using CBC (Cipher Block Chaining) mode with a randomly generated initialization vector (IV).
- The data, like balance and transaction information, is padded to the appropriate length to ensure compatibility with AES encryption.

Key Functions:

- `encrypt_data`: Encrypts the given data using AES and returns the encrypted data in base64 format.
- `decrypt_data`: Decrypts base64-encoded encrypted data back to its original form.

3. Password Hashing

- **Bcrypt** is used to securely hash user passwords. This provides protection in case the database is compromised.
- During registration, passwords are hashed using `bcrypt.hashpw()`.
- During login, the entered password is checked against the stored hash using `bcrypt.checkpw()`.

4. JWT Authentication

- **JWT (JSON Web Tokens)** are used for user session management.
- When a user logs in successfully, a JWT token is generated with the username as the subject (`sub`) and an expiration time of 1 hour.
- The token is stored in the session state to maintain the user's logged-in status.

Key Functions:

- `create_jwt`: Creates and signs a JWT with the username and expiration time.
- `verify_jwt`: Decodes the JWT to verify its validity and extract the user's identity.

5. User Management

- **User Creation**: When a new user registers, their data (username, password, balance, IV) is encrypted and saved in Supabase.
- **User Retrieval**: When a user logs in, their data is fetched using their username.
- **Balance Updates**: Each user's balance is encrypted and stored in the database. Whenever a transfer occurs, the balances of both users are updated and encrypted before being saved.

Key Functions:

- `create_user`: Creates a new user in the database with a hashed password and encrypted balance.
- `get_user`: Retrieves a user from the database by username.

6. Transaction Management

- **Transfers**: Users can transfer money to others, and the balances are updated accordingly.
- **Transaction Record**: Each transfer is recorded with the sender, receiver, amount, and timestamp, all encrypted before being stored in the Supabase database.

Key Functions:

- `update_balance`: Updates a user's balance by decrypting, modifying, and re-encrypting the balance.
- `create_transaction`: Records a transaction with the sender and receiver usernames (encrypted), amount, and timestamp.

7. Streamlit Interface

- **User Login and Signup**: Users can sign up and log in through tabs. On successful login, the user is provided access to the main banking interface.
- **Main Banking Interface**: Users can see their balance, make transfers, and log out. Transfers are only allowed if the user has enough balance, and the receiver exists.
- **Session Management**: Session is maintained using JWT tokens, which are stored in `st.session_state`. When the session expires or the user logs out, the session state is cleared.

Detailed Workflow

1. **User Registration (Signup)**:
    - The user provides a username and password.
    - The application checks if the username already exists.
    - If not, the password is hashed, and an initial balance of 1000 is encrypted using AES.
    - A new user record is created in the database with the username, hashed password, encrypted balance, and IV (Initialization Vector).
2. **User Login**:
    - The user provides their username and password.
    - The application checks the username in the database.
    - If a matching user is found, the hashed password is checked against the stored hash.
    - If the password is correct, a JWT is generated, and the user is logged in.
3. **Main Banking Interface**:
    - After login, the user is redirected to the main interface where their balance is displayed.
    - The user can transfer funds to another user by entering their username and the transfer amount.
    - The balance is updated, and a transaction record is created with encrypted details.
    - The application ensures that the user has enough funds to transfer and prevents transfers to the user's own account.
4. **Logout**:
    - The user can log out, which clears the session and redirects them to the login page.

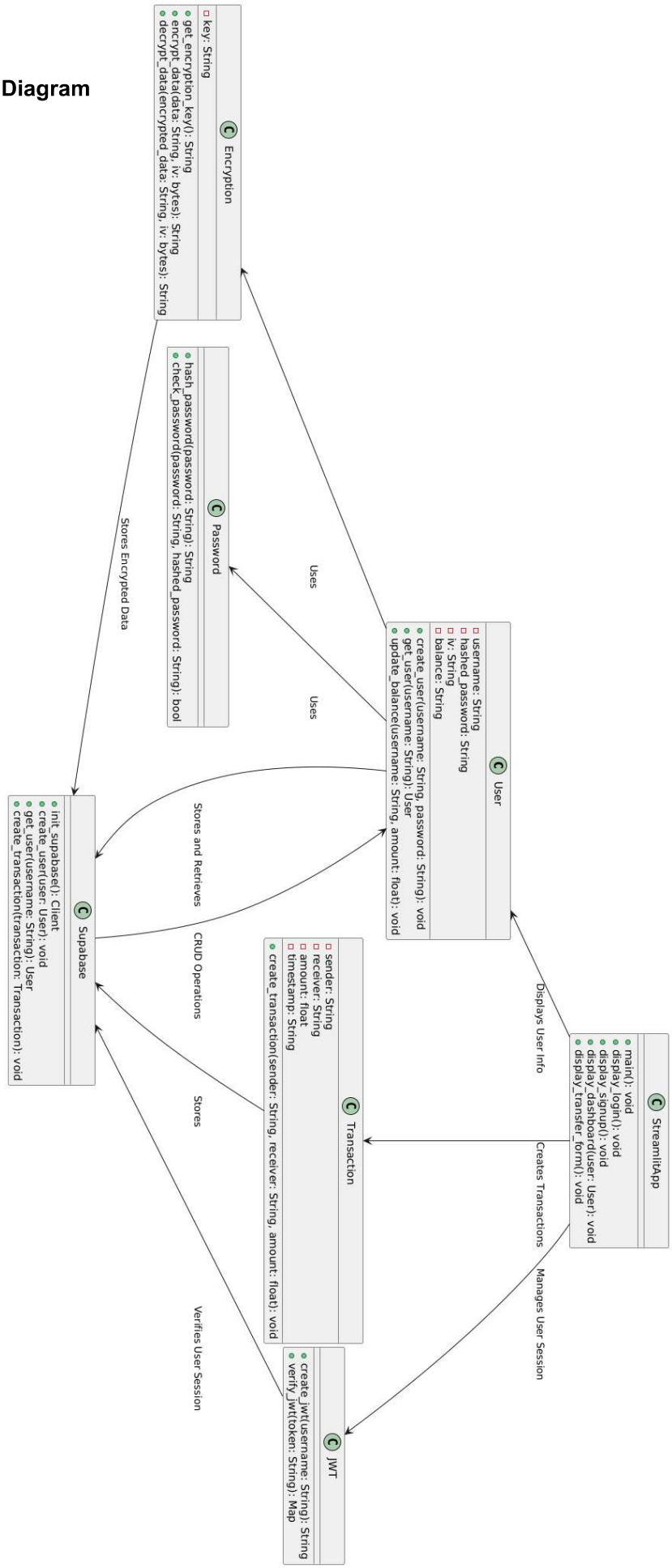
Security Features

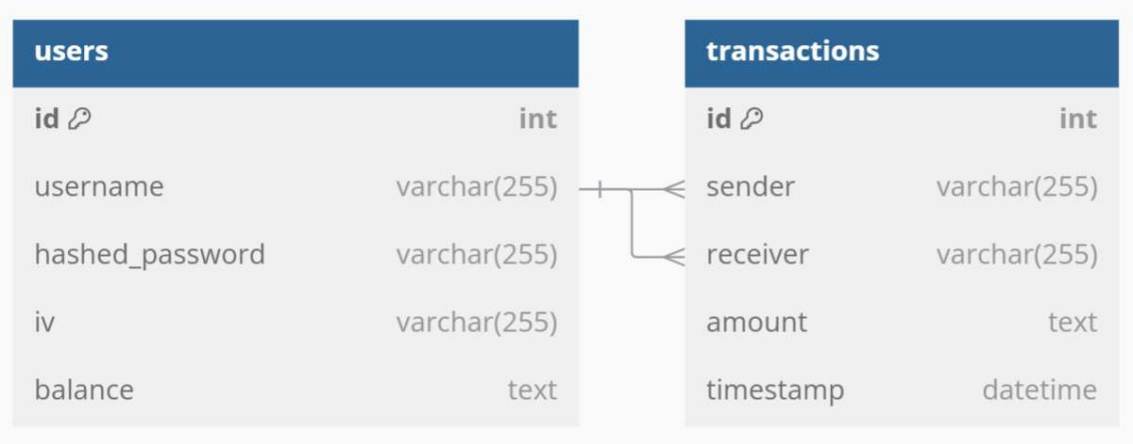
1. **Encryption**:
    - AES encryption ensures that sensitive information like balances and transaction details are stored securely in the database.
    - The encryption key is stored securely in `st.secrets`, preventing exposure in the code.
2. **Password Hashing**:
    - Bcrypt hashing makes it impossible to retrieve the original password, even if the database is compromised.
3. **JWT Authentication**:
    - JWT tokens protect user sessions and prevent unauthorized access.
    - Expiration times ensure that sessions are automatically invalidated after a set period.

Error Handling

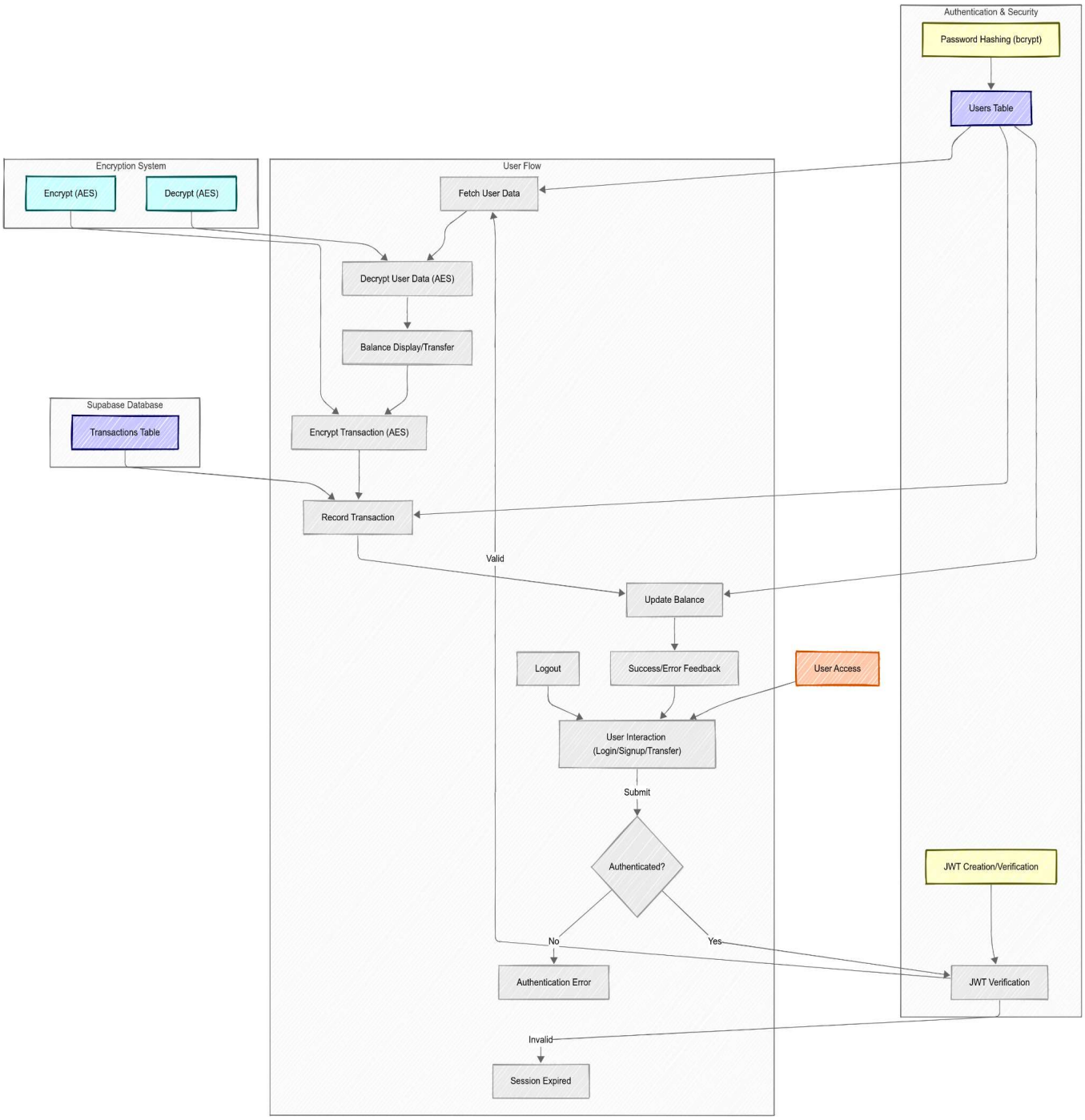
- **Invalid Credentials**: If the user provides an incorrect password, they receive an error message.
- **Insufficient Funds**: Users are prevented from transferring more than their available balance.
- **Invalid Transfer**: Transfers cannot be made to the user's own account, and if the receiver does not exist, an error message is shown.
- **Session Expiry**: If the JWT is invalid or expired, the user is logged out automatically.

UML Class Diagram





Database Design



Data Flow Diagram