**Problem Statement:**

In today's world, cashless transactions are becoming increasingly popular, creating a need for a secure and efficient payment platform that can be trusted by­­ both customers and banks alike. Such a platform requires a robust backend architecture with solid db design to ensure (a) secure connections to banks via payment processors and (b) secure storage of bank-generated hash tokens in the gateway's database. The objective is to simplify the complex process that is online payments, building a framework that is secure, swift, and easy to use.

**Business Problem:**

In e-commerce, it is crucial to ensure that transactions occur securely and quickly without compromising sensitive information (gen. info/ssn/cvv/app-exc-pin when using external services on third party provided – *PayPal/Venmo*).

Data breaches can be brutal to both reputation and financial losses.

It is therefore essential to implement robust data security measures that mitigate risks of cyber-attacks.

**Proposal:**

A data-driven approach that leverages big data technologies and advanced analytics.

Upon registering with the payment application, customers receive a unique token generated by their bank for each transaction involving a specific merchant. This token is securely stored in the application's database and transmitted to the bank for verification during payment initiation. The transaction is approved only after successful token verification.

To safeguard user banking information, we have advocated implementation of encryption and tokenization techniques. Encryption provides protection for sensitive data like credit card numbers, while tokenization substitutes sensitive data with non-sensitive tokens. These methods effectively deter data breaches and secure confidential information.

We advise adopting a multi-layered security strategy to establish and maintain a secure network. This strategy includes firewalls, intrusion detection and prevention systems, and routine security evaluations, ensuring comprehensive protection for the payment platform against external threats and guaranteeing data encryption and security during transit.

To enable token-key authentication, we suggest incorporating a two-factor authentication (2FA) mechanism. 2FA necessitates users to provide a secondary authentication factor, such as a one-time password or biometric authentication, in addition to their standard password. This added layer of security ensures that only authorized users can access the payment platform.

For robust access control measures, we recommend employing role-based access control (RBAC). RBAC ensures that only authorized users can access sensitive data and execute critical operations. Additionally, RBAC helps mitigate insider threats by restricting access to sensitive data and operations exclusively to authorized personnel.

In conclusion, by harnessing big data technologies and sophisticated analytics, we can develop a secure, efficient, and user-friendly payment platform trusted by customers and banks alike. By integrating encryption and tokenization techniques, a multi-layered security approach, 2FA, and RBAC, we can safeguard the payment platform from both external and internal threats while keeping sensitive data secure at all times.

**What TAP** *(Transact Adhere Protocol)* **does?**

1. Performs valid transactions with appropriate error handling.

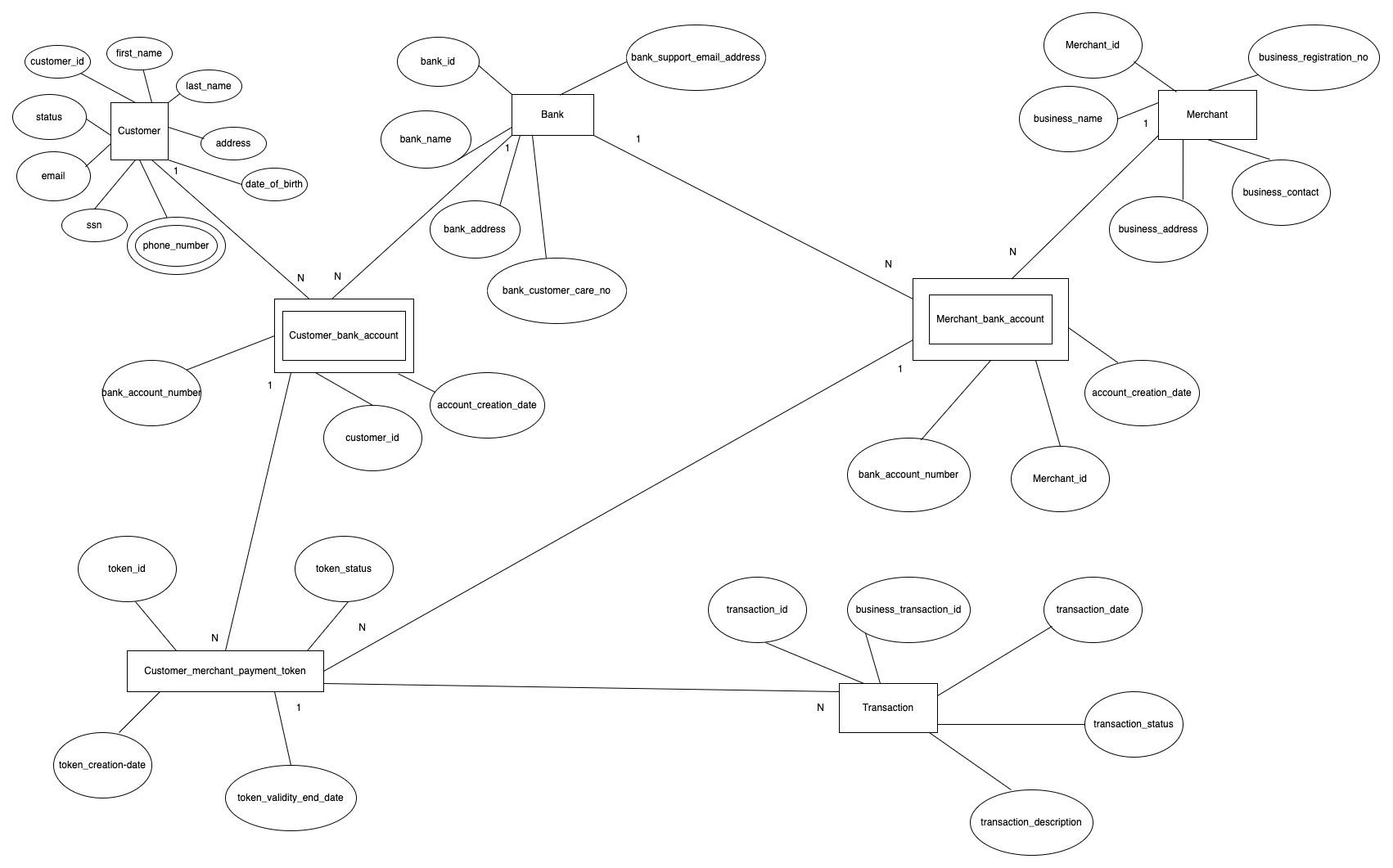
2. Maintains the abstraction of sensitive information (newer releases will enable integration of your APIs – enabling integration with your cloud – drop a ‘opt-in’ to my newsletter for updates on my repos).

3. Base to build-upon and maintain a secure network.

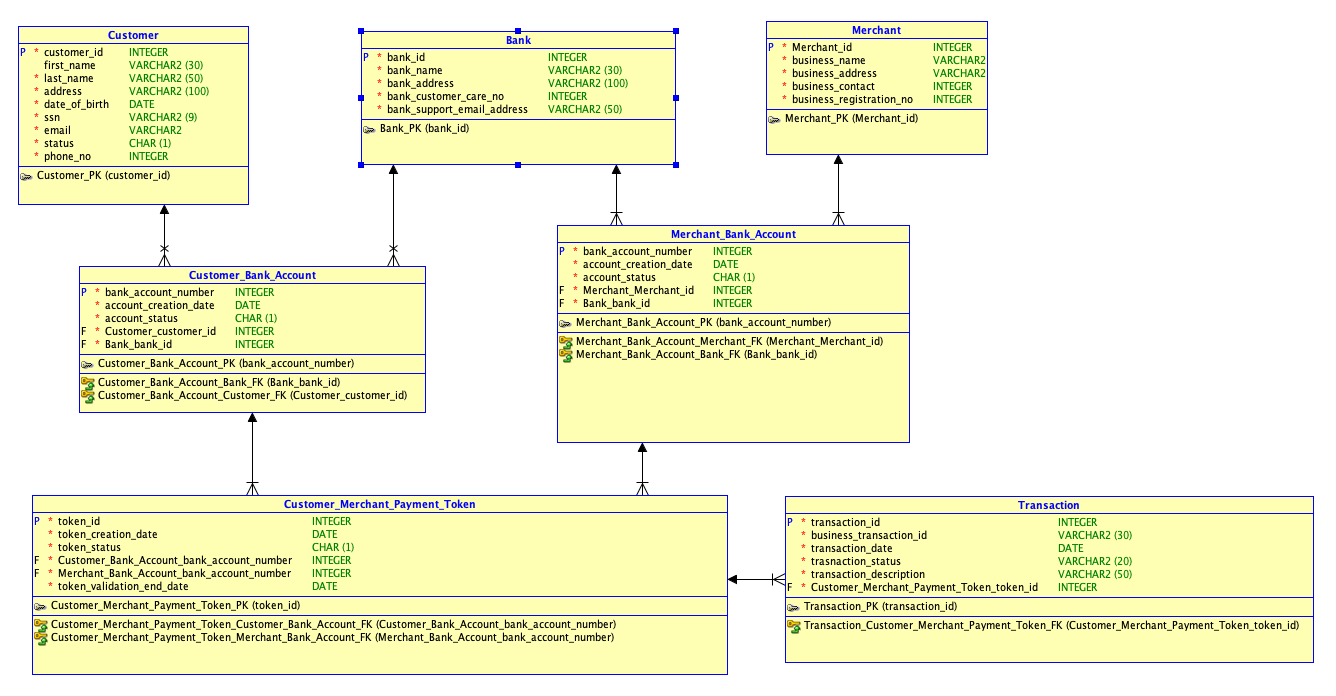
4. token-key authentication for appropriate hashing in transactions.

5. Implements strong access control measures.

Conceptual Model:-



ER Diagram:-



Document Entity and Attributes with Data Types defined:-

**Customer**

Primary Key - customer\_id

**Merchant**

Primary Key - merchant\_id

**Bank**

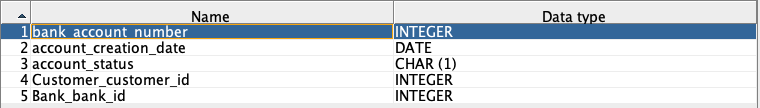
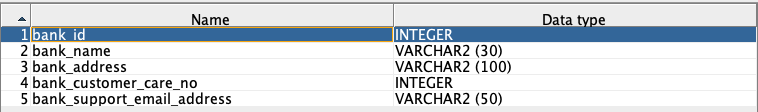
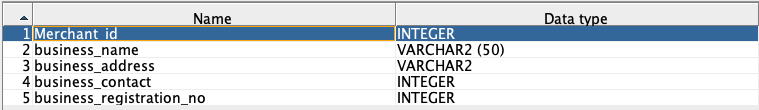
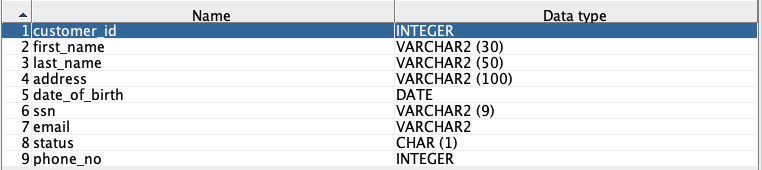
Primary Key - bank\_id

**Customer\_Bank\_Account**

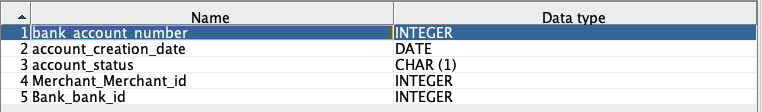
Primary Key - bank\_account\_number Foreign Key reference -

customer\_id from customer entity

bank\_id from bank entity



**Merchant\_Bank\_Account**



Primary Key - bank\_account\_number Foreign Key reference -

merchant\_id from merchantentity

bank\_id from bank entity

**Customer\_Merchant\_Payment\_Token**

Primary Key - token\_id Foreign Key reference -

customer\_bank\_account\_number from Customer\_Bank\_Account entity

merchant\_bank\_account\_number from Merchant\_Bank\_Account entity

**Transactions**

Primary Key - transaction\_id Foreign Key reference -

token\_id from Customer\_Merchant\_Payment\_Token entity

**BUSINESS RULES:**

1) A user can have multiple bank accounts across various banks.

2) Each bank will maintain only one account for a specific user.

3) A user is assigned a unique token ID for each distinct receiver.

4) Transactions will be declined if the token status is inactive or expired, which occurs when the token validity date has passed or the bank has not verified the token.

5) Each transaction is assigned a unique transaction ID that links to a single token ID, which is associated with two user IDs: one sender and one receiver.

6) The user's email ID will be unique across the users' table.

7) The 'type of user' attribute enforces a check constraint to ensure the proper entry of user-specific details and merchant/business information.

**VIEWS:**

1) Monthly tally of bank accounts created for a specific bank.

2) Total amount sent from and received by a bank account.

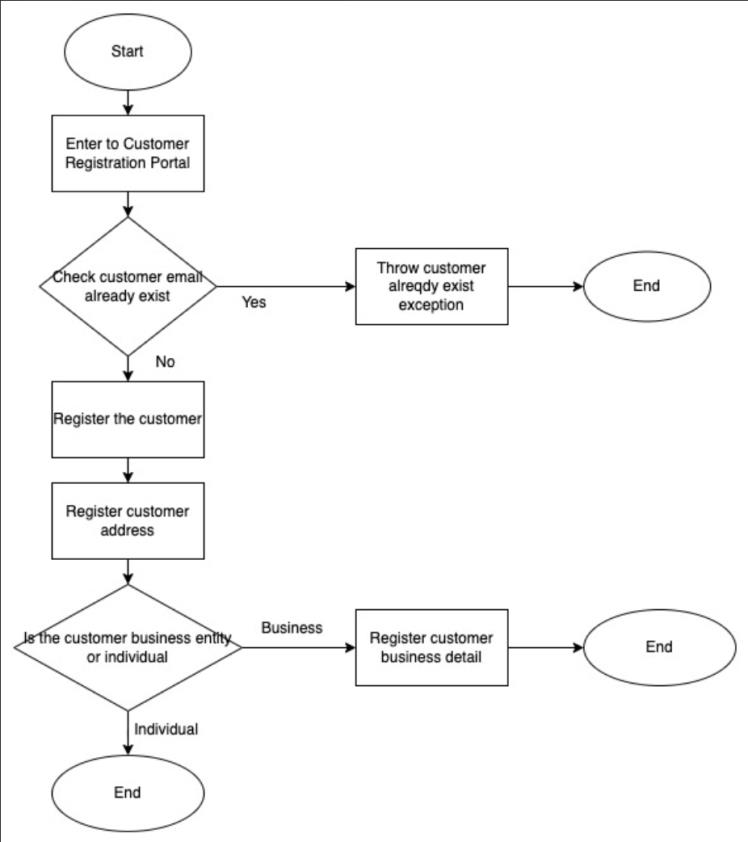
3) State-wise list of customers with active and inactive accounts.

4) Breakdown of transaction types: business-to-business, business-to-individual, individual-to-business, and individual-to-individual.

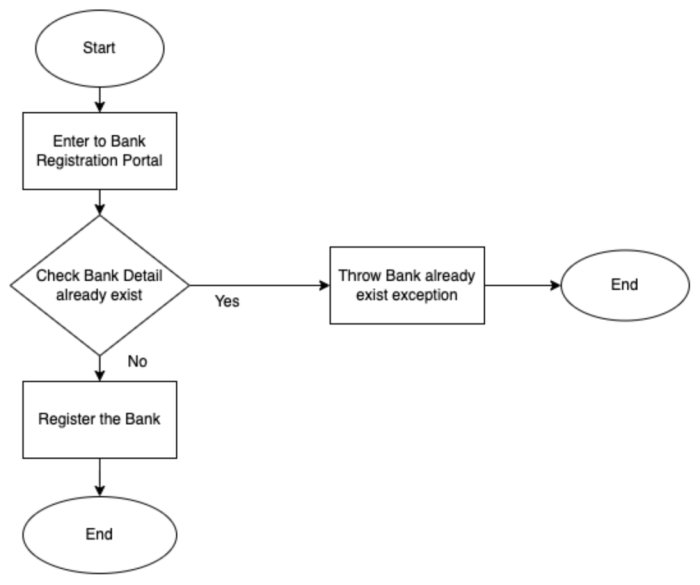
5) **[Internal view]** For customer support representatives, mask token IDs and account numbers to protect sensitive information.

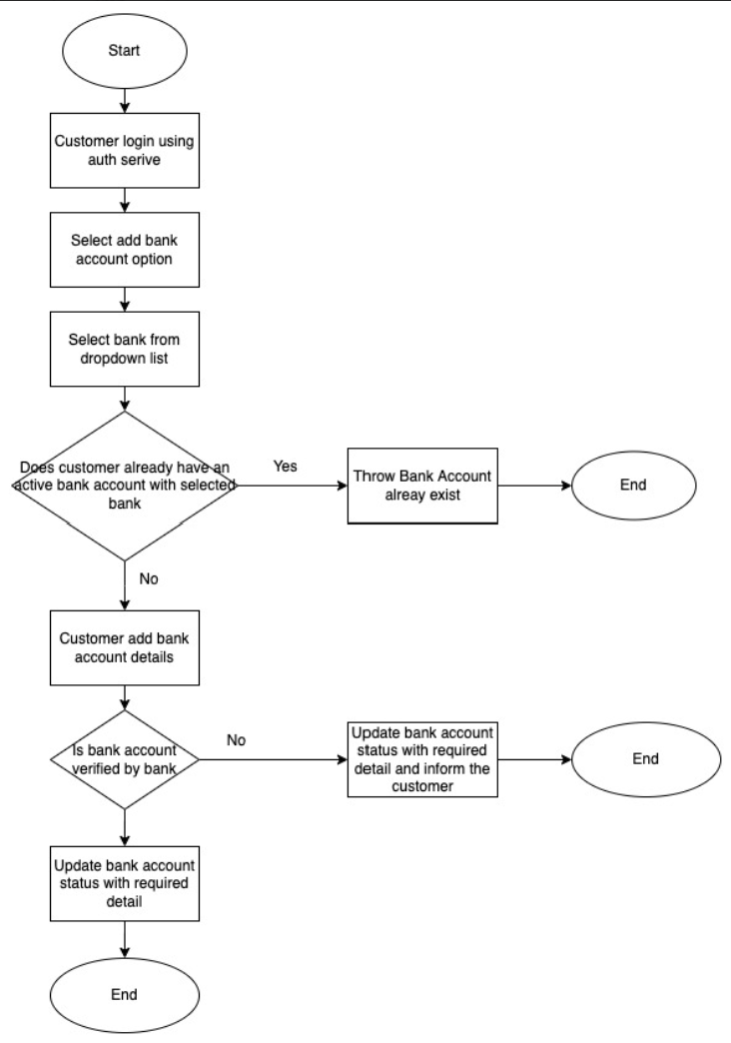
Data Flow Diagrams:

**[1] Customer Registration:**

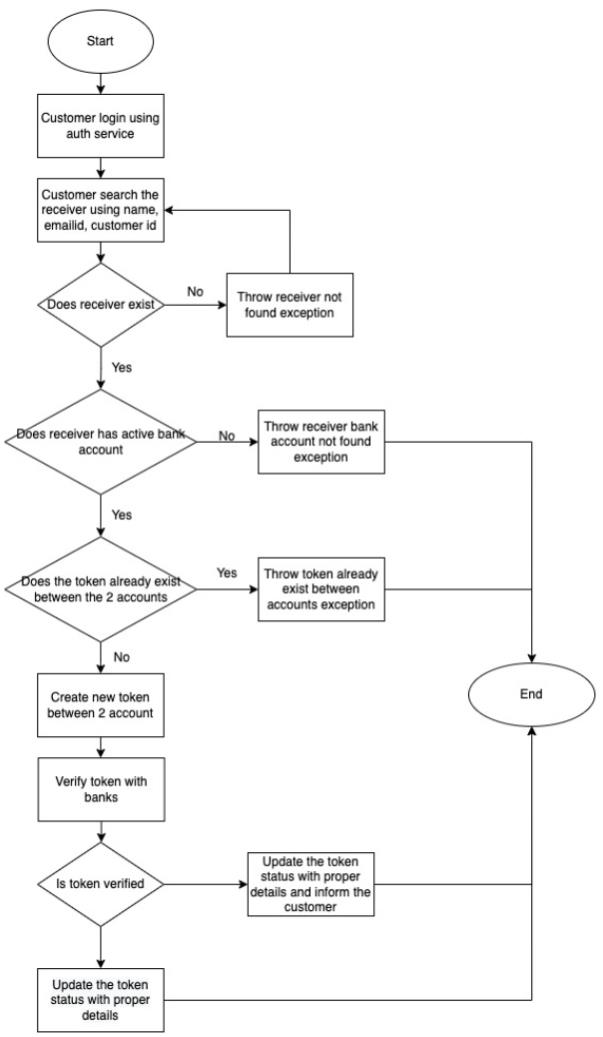


**[2] Bank Registration:**

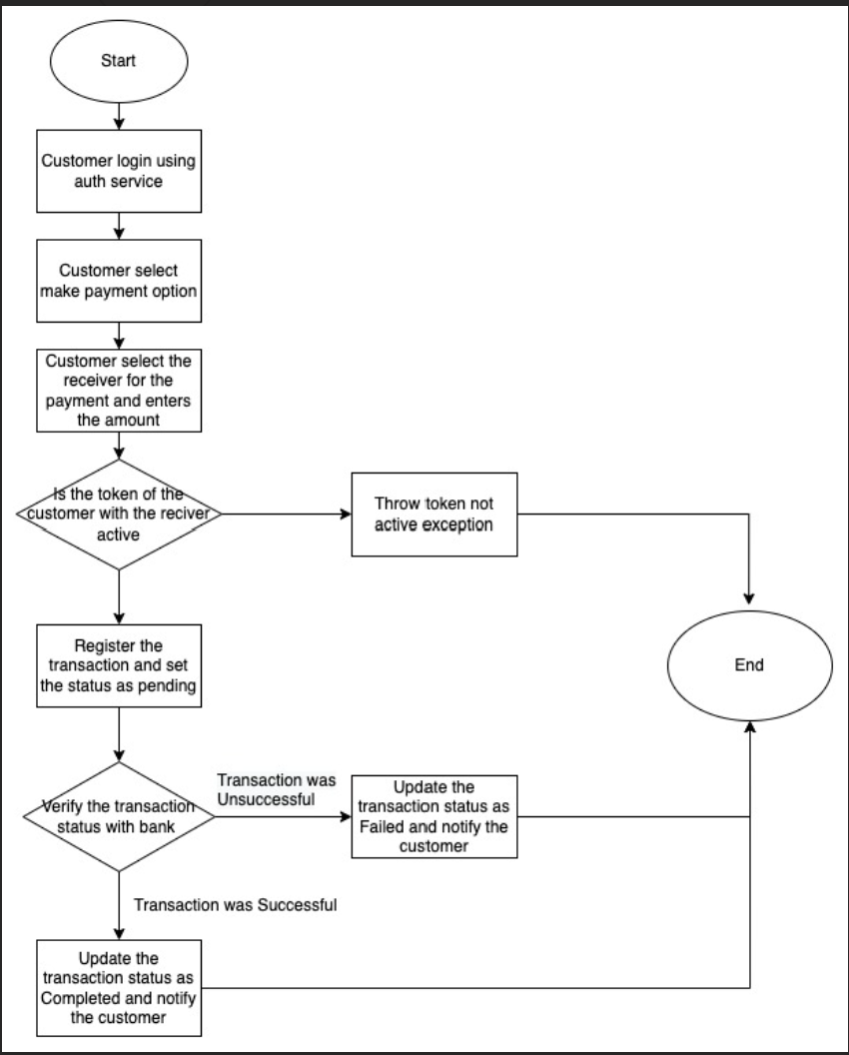


**[3] Bank account creation**

**[4] Sender Reciever Token Creation**



**[5] Send payment to receiver**



**---------------------------------------------------**

**---CONSTRAINTS---(SECURITY/OTHERWISE)---**

**---------------------------------------------------**

**A picture containing text, number, receipt, menu

Description automatically generated**