Enhancing Customer Engagement with Real-Time Email Notifications for Transaction Activities

Authors - Name: David Chike - Udeagha

Email: [chikedav@umd.edu]

Problem Statement

In the current configuration of the Testudo Bank application, users lack immediate and direct communication regarding transaction activities within their accounts. This gap could lead to delayed awareness of unauthorized transactions, insufficient real-time feedback on account status, and a general disconnection from financial activities. To address these issues, there is a need to implement a system that can provide instant notifications to users following each account transaction, thereby enhancing security, user engagement, and trust in the banking services provided by Testudo Bank.

Solution Requirements

1. **Immediate Email Alerts for Customers:**
    - This feature is crucial as it directly enhances the customer experience by providing instant notification after every transaction. It keeps customers informed and engaged with their account activity, promoting transparency and trust in the bank's services.
2. **Fraud Alerts for Administrators:**

- Immediate notifications for transactions flagged by fraud detection algorithms are vital for maintaining security. This feature allows bank administrators to quickly respond to potential fraud, minimizing financial loss and protecting both the customer's and bank's interests.

Chosen Solution: Internal Email System

High-Level Implementation

- **Build an internal email system** using SMTP servers directly managed within the bank's own infrastructure.
- **Integrate email queueing and sending** functionalities into the bank's existing systems to manage notifications associated with customer transactions.

Advantages

- **Control:** Complete autonomy over the email distribution process, ensuring enhanced security and privacy of sensitive data.
- **Cost-Effectiveness:** Lower ongoing expenses are possible, especially if transaction volumes remain manageable.
- **Customization:** Increased ability to tailor the system according to specific banking needs and preferences.

Disadvantages

- **Maintenance:** The system requires continuous upkeep, including managing server updates and resolving technical issues.
- **Scalability Challenges:** Expansion to accommodate a growing user base might necessitate substantial investments in infrastructure.
- **Deliverability Issues:** There is a risk of emails being flagged as spam or failing to reach recipients effectively, lacking the advanced features of professional Email Service Providers (ESPs).

Approach 2: Integration with an External Email Service Provider (ESP)

High-Level Implementation:

- Utilize a third-party Email Service Provider (ESP) such as SendGrid, Mailgun, or Amazon SES to handle email delivery.
- Set up API integration in the bank's backend system to trigger emails via the ESP's API when transactions occur.

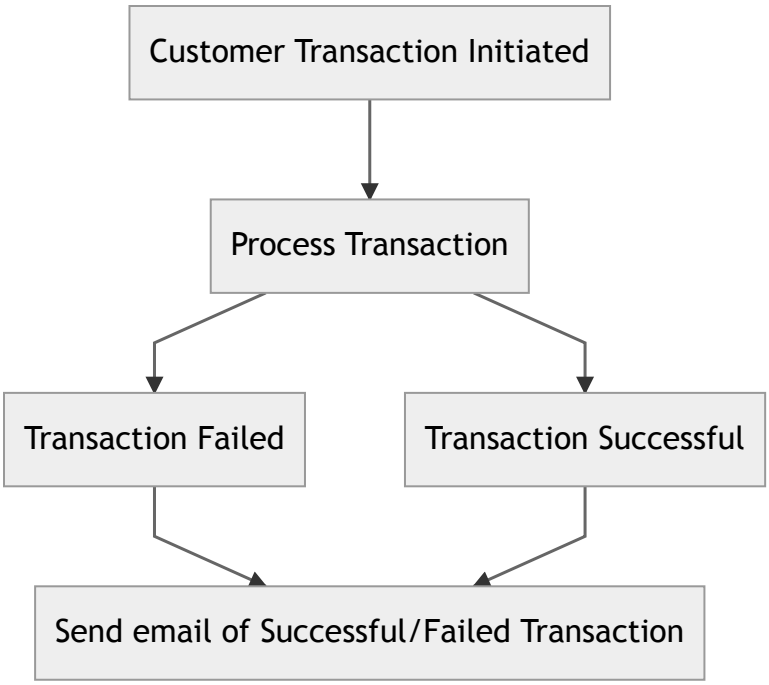
Pros:

- **Reliability:** ESPs offer high deliverability rates and are reliable for sending large volumes of email.
- **Scalability:** Easily scales to accommodate growth in transaction volumes and customer base without additional overhead on the bank's servers.
- **Features:** Provides advanced features such as analytics, email tracking, and comprehensive logs for troubleshooting.

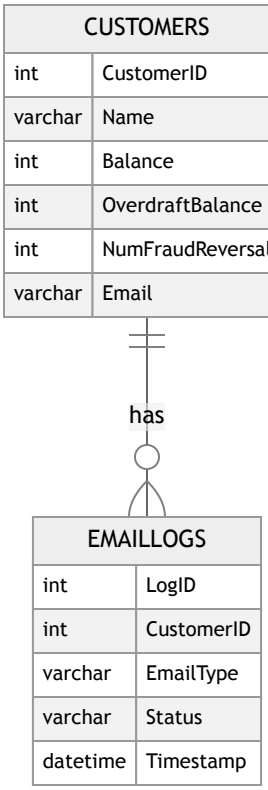
Cons:

- **Cost:** Can become costly as email volume grows, with pricing typically based on the number of emails sent.
- **Dependency:** Adds a dependency on a third-party service, which could affect service availability during outages.
- **Data Privacy:** Requires sharing customer data with a third-party, which could raise privacy and security concerns.

Flowchart Diagram



Updated DB Schema



1. **Customers Table:**

- **Addition of Email Column:**
  - `Email` : Stores the email address of each customer. This field is used to send transaction notifications.

2. **EmailLogs Table (New Table):**

   - **Columns:**
     - `LogID` : Primary key, auto-incremented.
     - `CustomerID` : Foreign key that links to `CustomerID` in the `Customers` table.
     - `EmailType` : Describes the type of email sent (e.g., 'Transaction Alert', 'Fraud Alert').
     - `Status` : Indicates the status of the email ('Sent', 'Failed').
     - `Timestamp` : Records the time when the email was sent.

Test Cases for Customers Table

- **Email Validity:**
  - **Test Case 1:** Verify that the `Email` column accepts only valid email formats.
  - **Test Case 2:** Ensure that each email in the `Email` column is unique.
  - **Test Case 3:** Confirm that the `Email` column is crucial for the new feature by testing its connection with transaction activities and user notifications.

Test Cases for EmailLogs Table

- **LogID Uniqueness and Auto-generation:**
  - **Test Case 1:** Validate that the `LogID` column is automatically generated and unique for each log entry.
- **CustomerID Integrity:**
  - **Test Case 2:** Check that the `CustomerID` column correctly traces back to the respective customer and maintains referential integrity by being a foreign key referencing the `Customers` table.
- **EmailType Categorization:**

    - **Test Case 3:** Test the functionality of the `EmailType` column to categorize emails sent for different purposes.
- **Status Tracking:**
    - **Test Case 4:** Ensure that the `Status` column accurately tracks the success or failure of email deliveries, aiding in troubleshooting issues.
- **Timestamp Accuracy:**
    - **Test Case 5:** Validate the `Timestamp` column's ability to capture the exact time an email is sent, which is crucial for logs and potential audits.