**(ChatGPT is used in this design)**

**System Overview**

The system is designed to accept broadcast requests, sign and broadcast transactions to a blockchain network, manage retries for failed transactions, and provide visibility into the status of transactions. The key components of the system include:

1. **API Gateway:** Serves as the entry point for internal services to submit transactions for broadcasting. It handles initial request validation and provides immediate feedback on request acceptance.
2. **Transaction Manager:** Coordinates the signing and broadcasting of transactions. It also manages retries and maintains the state of each transaction.
3. **Blockchain Interface:** Handles communication with blockchain nodes, including signing transactions and making RPC calls to broadcast signed transactions.
4. **Data Store:** Persists transaction states, logs, and outcomes to ensure reliability and enable analytics and administration.
5. **Admin Interface:** A web-based UI for administrators to view transaction statuses and manually retry failed broadcasts.

**Software Abstractions**

1. Transaction Request Object

Encapsulates the details of a broadcast request, including the message type, data payload and created time, etc. It provides methods for validation and conversion into a blockchain transaction format.

2. Transaction Signer

Responsible for signing transactions with the appropriate private key. It abstracts away the cryptographic operations, making it easy to support different signing mechanisms or keys if needed.

3. Blockchain Adapter

An abstraction over the blockchain communication, providing a unified interface for sending transactions regardless of the underlying blockchain specifics. This component handles the nuances of making RPC calls and interpreting responses.

4. Broadcast/Retry Management Mechanism

Implements logic for broadcasting/retrying failed transactions based on predefined criteria (e.g., time, maximum number of retries, exponential backoff). It integrates with the Transaction Manager to schedule retries. For example queueing mechanism can be applied.

5. Transaction State Machine

Manages the lifecycle of each transaction, including states such as **Pending**, **Signed**, **Success**, **Failed** and **Permanently Failed**. This component ensures that the system can accurately track and respond to the status of each transaction.

6. Data Repository

Interfaces with the data store, providing allowed CRUD operations for transaction records. It ensures data integrity and supports transactional consistency, especially across system restarts.

7. Admin Controller

Handles administrative requests from the Admin Interface, allowing for the manual retry of transactions and querying of transaction statuses.

**Process Flow**

1. **Transaction Submission:**
   * An internal service posts a transaction request to the TBS via the API Gateway.
   * The request is validated, and a new **Transaction Request Object** is created and persisted with an initial **Pending** state.
2. **Transaction Signing:**
   * The Transaction Manager picks up the pending transaction, and the **Transaction Signer** signs it, transitioning the state to **Signed**.
3. **Transaction Broadcasting:**
   * The **Blockchain Adapter** attempts to broadcast the signed transaction to the blockchain network.
   * On success, the transaction state is set to **Success**.
   * On failure or timeout, the transaction state is set to **Failed**, and the **Retry Mechanism** schedules it for a retry based on predefined rules. In failure queue case it will be added to the end of the queue.
4. **Failure Handling and Retries:**
   * The Transaction Manager monitors failed transactions and attempts retries as scheduled.
   * If retries exceed the maximum allowed, the transaction is marked as permanently failed.
   * If the broadcast service crushes and restart, all pending, signed and failure objects should be reloaded from persistent storage.
5. **Administration and Monitoring:**
   * The Admin Interface allows viewing transaction statuses and manually retrying failed transactions.
   * A background process updates the Admin Interface with the latest transaction states.