reserve the rate for limited time

Business Document

Summary

Allow user to lock in a foreign exchange (forex) rate at a specified volume, price, currency pair and expiration date.

Action

When a user obtains a live rate, they can choose a "Reserve Rate" option and select how long they want the rate held (for example, 1, 5, and/or 15 minutes). Once confirmed:

- The system persists the locked rate and expiration timestamp.
- During the lock-in window, users are offered to cancel the reservation or to execute the trade at the guaranteed rate. Both actions are to be fulfilled with other business units.

Outcome

By reserving the rate, the user secures the exact exchange price for the selected duration.

- Notification of reservation (either success or failure) is to be sent. This could be in the form of a RESTful response or/and email.
- Upon the expiration, the reservation expires and the rate unlocks

Definition of Done

1. User interface presents a “Reserve Rate” button alongside each live quote.
2. Options (e.g., 1, 5, 15 minutes) are configurable and displayed.
3. System stores reservation records with start and end timestamps.
4. Expired reservations are automatically invalidated and cannot be executed.

Solution Design (As a guideline, not to be exact)

1. Database Design

Key Fields to Store:

1. **Reservation ID (Primary Key)**: Unique identifier for each reservation.
2. **User ID (Foreign Key)**: Links to the user who created the reservation.
3. **Token**: A unique token used for transaction tracking or authentication purposes.
4. **Currency Pair**: The currency pair being locked (e.g., EUR/USD).
5. **Volume**: The amount of currency to be exchanged.
6. **Lock-in Date**: Date and time when the reservation is made (i.e., the date the rate is locked).
7. **Expiration Date**: Date and time when the locked rate will expire.
8. **Spot Rate**: The current market exchange rate at the time of reservation.
9. **Forward Rate**: The future exchange rate agreed upon for the transaction.
10. **Status**: Tracks the current state of the reservation (e.g., active, expired, completed).
11. **Created At**: Timestamp of when the reservation was created.
12. **Updated At**: Timestamp of when the reservation was last updated.

Table: Reservations

Field Name	Data Type	Description
reservation_id	UUID	Unique identifier for the reservation (Primary Key)
user_id	UUID	Identifier linking to the user table
token	UUID	Unique token for transaction tracking
currency_pair	VARCHAR	The pair of currencies being locked
volume	DECIMAL	The amount of currency to exchange
lock_in_date	DATETIME	The date when the rate is locked
expiration_date	DATETIME	The date when the rate lock expires
spot_rate	DECIMAL	The spot rate at the time of reservation
forward_rate	DECIMAL	The forward rate agreed at the time of reservation
status	VARCHA	Status of the reservation (active, expired, etc.)

	R	
created_at	DATETIME	Timestamp when the reservation was created
updated_at	DATETIME	Timestamp of the last update to the reservation

2. Api

- `POST /api/rates/reserve`
  - Create a new rate reservation
  - Request: {userId, fromCurrency, toCurrency, amount}
  - Response: {reservationId, guaranteedRate, expirationTime}

- `GET /api/rates/reservations/{userId}`
  - Get user's active reservations

- `PUT /api/rates/reservations/expire`
- Batch expire old reservations (cron job)

3. Data Source

an api to get data from DB?