**Complete System Description: Remittance Platform with P2P Exchange**

This document outlines the system design for a remittance platform with peer-to-peer (P2P) exchange capabilities. The platform allows users to send money across borders, both using traditional currency exchanges and P2P-based transfers. The platform will have features like escrow services, KYC (Know Your Customer) verification, and the ability to connect users’ existing crypto wallets for remittances using stablecoins in the future.

**1. Traditional Remittance Flow (Normal Remittance System)**

In this basic remittance flow, users can transfer funds across countries using fiat currencies, such as GBP and NGN.

**Flow Overview:**

1. **User Registration and KYC Verification:**
   * Users sign up by providing basic personal details (name, contact info, etc.).
   * KYC (Know Your Customer) verification is done to ensure regulatory compliance. Documents like a national ID, proof of address, and a selfie may be required.
2. **User Adds Funds (Deposit):**
   * Users can add funds to their wallet using available payment methods (bank transfer, credit/debit cards, etc.).
   * The funds are credited to their wallet on the platform.
3. **User Initiates a Remittance:**
   * The sender (UserA) creates a remittance request by specifying:
     + The amount they wish to send.
     + The recipient’s details (name, contact, etc.).
     + The preferred currency exchange rate (e.g., GBP to NGN).
   * The system automatically calculates and displays the equivalent amount to the recipient.
4. **Exchange Processing:**
   * The platform automatically handles the currency exchange using a pre-defined exchange rate or the market rate.
   * The system deducts the equivalent amount from sender’s wallet and credits the recipient’s Account after conversion.
5. **Fund Transfer and Completion:**
   * After the exchange, the system debits UserA’s wallet with the converted funds and credits UserB’s wallet with the equivalent amount.
   * The transaction is completed, and the users receive notifications.

**2. Peer-to-Peer (P2P) Exchange System**

This is the advanced feature of the remittance platform, allowing users to exchange currencies directly with each other, bypassing traditional financial intermediaries. The system automatically matches buyers and sellers and executes the transaction when both parties have credited their wallets.

**Flow Overview:**

1. **User Registration and KYC Verification:**
   * Just like in the traditional remittance system, users go through KYC verification before participating in the P2P system.
2. **User Lists Offer (Seller Action)**:
   * **Sell Order Creation**: UserA (the seller) creates a sell order, specifying:
     + Amount of GBP they want to sell.
     + The exchange rate they are offering (e.g., 1 GBP = 500 NGN).
     + The available amount of GBP is credited to UserA’s wallet on the platform but is marked as “on hold.”
   * **Order Active Status**: The offer is now visible to other users in the marketplace. The offer is considered active unless cancelled by the seller.
3. **User Creates Buy Order (Buyer Action)**:
   * **Buy Order Creation**: UserB (the buyer) creates a buy order by specifying:
     + The amount of NGN they want to spend.
     + The exchange rate they are willing to accept (e.g., 1 GBP = 500 NGN).
     + The system will automatically attempt to match this order with a suitable sell offer based on exchange rates and amounts.
4. **Automatic Matching**:
   * The system’s **auto-matching engine** searches for the best matching offer based on:
     + **Exchange Rate**: The buyer’s desired exchange rate and the seller’s offered rate.
     + **Amount**: The amount of GBP the seller is offering and the amount of NGN the buyer wants to spend.
   * The system matches the buyer and seller based on these criteria and executes the transaction automatically.
5. **Transaction Execution**:
   * Once a match is made, the system handles the following:
     + **Escrow**: The GBP offered by the seller is placed in escrow until the transaction is successfully completed.
     + **Fund Deduction**: The system deducts NGN from UserB’s wallet (buyer) and places it in escrow.
     + **Fund Transfer**: Once both parties have credited their wallets, the platform automatically exchanges the funds:
       - UserB’s NGN is converted into GBP based on the agreed-upon exchange rate.
       - UserA receives the equivalent amount in NGN, and UserB receives the GBP amount.
     + **Completion**: Once the transaction is executed, both users are notified, and the system removes the hold on the funds.
6. **Order Matching Criteria**:
   * The system matches the buy and sell orders automatically based on the following conditions:
     + **Amount**: The buy and sell amounts must match.
     + **Exchange Rate**: The rates must align, or fall within an acceptable range, based on the buyer’s and seller’s preferences.
     + **Available Funds**: Both users must have sufficient funds in their wallets to complete the transaction.

**3. Escrow System (For P2P Transactions)**

To ensure the safety of both buyers and sellers, the platform uses an escrow system for P2P transactions.

* **Seller’s Funds**: When UserA lists their offer to sell GBP, the amount they wish to sell is locked in escrow. The funds are marked as "on hold" and cannot be used for other purposes until the transaction is completed or cancelled.
* **Buyer’s Funds**: When UserB places a buy order and a match is made, the NGN amount is also locked in escrow on the platform.
* **Transaction Finalization**:
  + When both parties have credited their wallets (i.e., both the seller has GBP, and the buyer has NGN), the system will automatically complete the exchange.
  + The funds are debited and credited between wallets, and the transaction is finalized.
  + If either party cancels or does not complete the transaction, the escrowed funds are refunded back to their respective wallets.
* **Dispute Resolution**: In the event of a dispute, the platform’s support team can step in to mediate and resolve the issue, ensuring that the funds are released correctly.

**4. Integration with Stablecoins and Crypto Remittances**

Future functionality will include the ability to make remittances via stablecoins (like USDT or USDC). This will allow users to send funds across borders more efficiently and with lower fees, especially for those who prefer to use crypto instead of fiat currency.

1. **Crypto Wallet Integration**:
   * Users can connect their existing crypto wallets (e.g., MetaMask, Trust Wallet) to the platform.
   * The platform will allow both fiat-to-crypto and crypto-to-fiat transactions. For instance, users can send stablecoins (like USDT) to others or convert them to local currencies (e.g., GBP to NGN).
2. **Stablecoin Transfers**:
   * When the stablecoin feature is implemented, users can send stablecoins directly from one wallet to another. The platform will handle the conversion between stablecoins and fiat currencies (e.g., converting USDT to NGN).
3. **Conversion between Fiat and Crypto**:
   * The system will allow users to convert between fiat currencies (e.g., GBP, NGN) and stablecoins directly on the platform. This will enable easy cross-border transactions with minimal fees and fast processing.

**5. User Interface & Experience**

The platform will have a user-friendly interface with the following features:

* **Dashboard**:
  + Overview of the user’s balances (fiat and crypto).
  + Active and past remittance transactions.
* **Transaction History**:
  + Detailed records of completed transactions, including exchange rates, amounts, and transaction status.
* **Offer Marketplace**:
  + A list of active P2P buy and sell offers, with the option to filter by currency pair, exchange rate, and transaction status.
* **Notifications**:
  + Real-time notifications for when a transaction is matched, when funds are received, and when the transaction is completed.

**6. Security and Compliance**

The platform will ensure the following:

* **KYC Compliance**: All users must complete KYC verification before using the P2P exchange or sending remittances.
* **Data Encryption**: All sensitive data, including user personal information and transaction details, will be encrypted.
* **Regulatory Compliance**: The platform will comply with local and international regulations governing remittance services and digital currencies, such as anti-money laundering (AML) and combating the financing of terrorism (CFT) guidelines.

**Conclusion**

This remittance platform combines traditional remittance services with P2P exchange features, providing users with flexibility, lower fees, and faster transaction processing. By integrating escrow and KYC systems, users are protected, and transactions are executed with confidence. Future enhancements will include support for stablecoin-based remittances, offering even more options for cross-border transfers.

**1. System Architecture**

To build this system, we’ll need:

* **Frontend**: User-facing interface (Web or Mobile)
* **Backend**: API service for business logic and transaction handling
* **Database**: For storing user information, transactions, offers, etc.
* **Payment Gateways**: For fiat deposits and withdrawals
* **Escrow Service**: For P2P transactions
* **Blockchain Integration**: For stablecoin transfers in the future
* **KYC Verification Service**: For regulatory compliance

**2. Backend Services and Flow**

**a) User Registration & KYC**

1. **API Endpoint**: /register
   * POST request: Accepts user details (email, password, phone number).
   * KYC check: For this phase, you can use third-party KYC services like [Onfido](https://onfido.com/) or [Jumio](https://www.jumio.com/).
   * Response: User registration success/failure, KYC status.

**b) Wallet Management**

1. **API Endpoint**: /wallets
   * POST request: Creates a wallet for the user.
   * GET request: Fetches wallet details (balance, transaction history).
   * PUT request: Adds funds (via credit card, bank transfer, or crypto).
   * Response: Wallet balance update.

**c) P2P Exchange**

1. **API Endpoint**: /p2p/createOffer
   * POST request: User creates a sell order (amount of GBP, rate, etc.).
   * Backend process: Lock the GBP amount in escrow.
   * Response: Success or error message.
2. **API Endpoint**: /p2p/buyOffer
   * POST request: User creates a buy order (amount of NGN, rate).
   * Backend process: Attempt to match the buy order with existing sell offers.
   * Response: Success or error message.
3. **API Endpoint**: /p2p/executeTransaction
   * POST request: Executes the matched transaction between buyer and seller.
   * Logic:
     + Check that both users have enough funds.
     + Lock funds in escrow until both parties are ready.
     + Exchange funds and transfer accordingly.
   * Response: Success or error message.

**d) Escrow Service**

The escrow system will handle the locking of funds for both buyer and seller during the transaction. The backend should:

* Lock funds in escrow once an order is placed.
* Ensure that funds are only released when both the buyer and seller have credited their wallets.

**3. Database Structure**

Here is a simplified schema for the backend:

* **Users** Table:
  + id (Primary Key)
  + email
  + password\_hash
  + kyc\_status (Pending, Verified, Rejected)
  + balance\_fiat (User’s balance in fiat)
  + balance\_crypto (User’s balance in crypto)
* **Wallets** Table:
  + user\_id (Foreign Key)
  + wallet\_type (Fiat, Crypto)
  + balance
  + currency
* **P2P Offers** Table:
  + id (Primary Key)
  + user\_id (Foreign Key to Users)
  + currency\_from
  + currency\_to
  + amount
  + exchange\_rate
  + status (Active, Matched, Completed)
* **Transactions** Table:
  + id (Primary Key)
  + buyer\_id (Foreign Key to Users)
  + seller\_id (Foreign Key to Users)
  + amount
  + status (Pending, Completed, Canceled)
  + escrow\_status (Locked, Released)

**4. Frontend Design**

For the frontend, you will need a dashboard where users can:

1. View wallet balances (both fiat and crypto).
2. Create buy or sell orders for P2P exchanges.
3. See active orders and pending transactions.
4. Display KYC status and request updates.
5. Display recent transactions and order history.

Technologies for the frontend:

* **Web**: React.js or Angular for a dynamic web interface.
* **Mobile**: Flutter or React Native for cross-platform mobile development.

**5. Payment Gateway Integration**

* For fiat deposits and withdrawals, you may integrate with popular payment gateways, such as:
  + **Stripe** or **PayPal** for card payments.
  + **Rave by Flutterwave** or **Paystack** for bank transfers (particularly in Nigeria).

**6. Blockchain Integration (For Future Stablecoin Support)**

Once stablecoin functionality is ready:

* Integrate with blockchain networks like Ethereum or Binance Smart Chain.
* Users can connect their existing wallets (MetaMask, Trust Wallet).
* Build APIs for handling crypto transactions (deposit/withdraw).

**7. Security**

* **Encryption**: Encrypt sensitive data, such as passwords and personal information (e.g., AES for data encryption).
* **Authentication**: Use JWT (JSON Web Tokens) or OAuth for session management and authorization.
* **Rate Limiting**: Implement rate-limiting mechanisms to prevent abuse and DDoS attacks.
* **SSL/TLS**: Ensure all data transmitted between the client and server is encrypted using HTTPS.

**8. Project Management & Task Breakdown**

To keep the project on track, break it down into tasks:

* **Phase 1: User Management**
  + User registration and authentication.
  + KYC service integration.
* **Phase 2: Wallet Management**
  + Integrate payment gateway for deposits.
  + Implement wallet functionalities (crediting, debiting).
* **Phase 3: P2P Exchange System**
  + Implement order creation, matching, and transaction handling.
  + Implement escrow service.
* **Phase 4: Stablecoin Integration (Future)**
  + Blockchain integration for crypto transactions.

**Next Steps:**

1. **Backend Setup**: Set up the server, choose a framework (e.g., Node.js, Django, etc.), and start implementing the user authentication and wallet management APIs.
2. **Frontend Development**: Start developing the user interface and connecting it to the backend APIs.
3. **Escrow System**: Develop the logic to lock and release funds during P2P transactions.

**1. Backend (API & Business Logic)**

Since the platform will handle financial transactions, we need a **secure, scalable, and maintainable** backend.

**Recommended Options**

1. **Node.js (with Express.js or NestJS)**
   * High performance with asynchronous processing.
   * Great for real-time applications (e.g., WebSockets for live transaction updates).
   * Large ecosystem with plenty of third-party libraries.
2. **Django (Python) or FastAPI**
   * Django is great for security (built-in authentication, CSRF protection).
   * FastAPI is highly efficient and fast for building APIs.
   * Python has strong libraries for financial calculations and ML-based fraud detection.
3. **Spring Boot (Java) [Alternative]**
   * Best for large-scale enterprise applications.
   * Strictly typed and secure but can be heavy for an MVP.

**My Recommendation: Node.js + NestJS**

* Scalable, efficient, and modern.
* Easy integration with payment gateways and blockchain APIs.

**2. Database (SQL vs NoSQL)**

Your system needs **strong consistency** for financial transactions, so an SQL database is ideal.

**Recommended Options**

1. **PostgreSQL**
   * Best for structured financial transactions.
   * Strong ACID compliance (Atomicity, Consistency, Isolation, Durability).
2. **MongoDB (For Non-Financial Data)**
   * Can store user profiles, chat messages, etc.
   * Best used as a secondary database (not for financial transactions).

**My Recommendation: PostgreSQL + Redis (for caching hot data)**

* **PostgreSQL** ensures reliable financial transactions.
* **Redis** helps with caching frequently accessed data (e.g., exchange rates).

**3. Frontend (User Interface)**

The platform will need both **Web & Mobile** apps.

**Recommended Options**

1. **React.js (for Web)**
   * Fast, scalable, and widely used.
   * Can be extended to **Next.js** for better SEO and performance.
2. **React Native (for Mobile)**
   * Cross-platform (iOS & Android) using a single codebase.
   * Easier maintenance compared to separate native apps.
3. **Flutter [Alternative]**
   * Good UI flexibility but may have fewer third-party financial libraries.

**My Recommendation: React.js (Web) + React Native (Mobile)**

* Reduces development time.
* Easier integration with the same backend API.

**4. Payment Gateway (For Fiat Transactions)**

You need **secure and reliable** gateways for fiat deposits and withdrawals.

**Recommended Options**

1. **Stripe** (Best for UK and International payments).
2. **Paystack / Flutterwave** (Best for Nigeria).
3. **Bank API Integration** (For direct bank deposits and withdrawals).

**My Recommendation:**

* Use **Stripe** for the UK and **Flutterwave/Paystack** for Nigeria.
* Later, integrate **bank APIs** for direct transactions.

**5. Escrow System (Core of P2P)**

The escrow system will **hold** funds in a secure account until the transaction is completed.

**Recommended Approach**

1. **Internal Escrow Database**
   * Use PostgreSQL to **lock** funds when an order is placed.
   * Funds are only released when the transaction is successful.
2. **Smart Contracts (For Future Crypto Escrow)**
   * Can be used to automate stablecoin escrow (on Ethereum, Solana, or BSC).
   * Initially, we use traditional escrow and later integrate smart contracts.

**My Recommendation: Start with a centralized escrow system using PostgreSQL**

* Later, move to **smart contracts** when you integrate stablecoins.

**6. Authentication & Security**

**Recommended Options**

1. **JWT (JSON Web Token)** for user authentication.
2. **OAuth 2.0** if integrating third-party logins.
3. **Argon2 or bcrypt** for password hashing.
4. **2FA (Two-Factor Authentication)** for added security.

**My Recommendation: JWT + bcrypt + 2FA**

* Ensures secure authentication.

**7. Blockchain Integration (For Future Stablecoins)**

Since you want to support **stablecoins & crypto transactions**, consider these networks:

**Recommended Blockchains**

1. **Ethereum (ERC-20 stablecoins like USDC, USDT, DAI)**
2. **Binance Smart Chain (BEP-20 tokens for lower fees)**
3. **Solana (Fast and cheap transactions, but still growing)**

**Wallet Integration**

* Use **MetaMask, Trust Wallet, or WalletConnect** for easy crypto deposits/withdrawals.
* Provide both **direct wallet-to-wallet** transfers and **fiat conversion options**.

**My Recommendation:**

* **Start with ERC-20 & BEP-20 stablecoins (USDT, USDC, BUSD).**
* Later, expand to **multi-chain support (Solana, Polygon, etc.)**.

**8. Infrastructure (Cloud & DevOps)**

For hosting and deployment, you need **scalability and security**.

**Recommended Options**

1. **AWS (Amazon Web Services)**
   * Best for enterprise-scale applications.
   * Use **AWS RDS** (Managed PostgreSQL) and **S3** (Storage).
2. **Google Cloud Platform (GCP)**
   * Similar to AWS but better for AI/ML integrations.
3. **DigitalOcean (For MVP Launch)**
   * Simpler, more cost-effective hosting for startups.

**My Recommendation: AWS (For full scale) or DigitalOcean (For MVP testing)**

**Finalized Tech Stack**

| **Component** | **Technology** |
| --- | --- |
| **Backend** | Node.js (NestJS) |
| **Database** | PostgreSQL + Redis |
| **Frontend (Web)** | React.js + Next.js |
| **Frontend (Mobile)** | React Native |
| **Payment Gateway** | Stripe + Paystack/Flutterwave |
| **Authentication** | JWT + bcrypt + 2FA |
| **Escrow** | Centralized (PostgreSQL), later Smart Contracts |
| **Blockchain (Future)** | Ethereum, BSC (Stablecoins) |
| **Cloud Hosting** | AWS or DigitalOcean |

**Next Steps**

1. **Set up the backend** (NestJS with PostgreSQL).
2. **Develop the wallet system** (Deposits, withdrawals, balances).
3. **Implement P2P escrow logic**.
4. **Build the frontend UI (Web & Mobile)**.