Steps to Develop a Crypto Exchange (From Scratch)

---

1. Requirement Analysis & Planning

Choose Exchange Type:

Centralized Exchange (CEX)

Decentralized Exchange (DEX)

Hybrid (combination of both)

---

2. Difference Between Centralized and Decentralized Exchange

Feature	Centralized Exchange (CEX)	Decentralized Exchange (DEX)
Control	Operated by a central authority	Operated via smart contracts, no central authority
User Funds	Custodial (platform holds users' funds)	Non-custodial (users control their private keys)
Speed	High-speed transactions using internal databases	Slower, as transactions happen on-chain
Liquidity	Generally higher due to centralized market-making	Depends on liquidity pools and users
Security Risk	Prone to hacking due to custodial nature	More secure as funds are user-controlled
KYC/AML	Usually mandatory	Optional or non-existent in most DEXs
Ease of Use	Easier for beginners	Requires knowledge of wallets and gas fees
Examples	Binance, Coinbase, Kraken	Uniswap, PancakeSwap, SushiSwap

---

3. Architecture & Tech Stack

- **Frontend:** React / Vue / Next.js
- **Backend:** Node.js (NestJS) / Go / Python
- **Blockchain Interface:** Ethers.js, Web3.js
- **Database:** PostgreSQL / MongoDB

- **Cache:** Redis
- **Message Queues:** Kafka / RabbitMQ
- **Infra:** Docker, Kubernetes

---

4. Core Modules to Build

a. User Management

- Signup/Login
- KYC integration: **Sumsub**, **Jumio**, **ShuftiPro**

b. Wallet System

- Generate unique wallet per user
- Use: **Fireblocks**, **BitGo**, **Coinbase Custody**

c. Blockchain Integration

- Handle deposits/withdrawals using listeners
- Multiple chain support (Ethereum, BSC, etc.)

d. Trading Engine (CEX only)

- Handle Limit, Market, and Stop orders
- Maintain in-memory order books

e. Fiat Integration

- Via **MoonPay**, **Ramp**, **Wyre**, **Razorpay**

f. Price Feeds

Use APIs like:

- CoinGecko
- CoinMarketCap
- Binance API
- CryptoCompare

g. Admin Dashboard

- Manage users, KYC, wallets, trades

h. Security

- Encrypt sensitive data
- 2FA with Google Authenticator/Authy
- DDoS, rate limiting, firewall

i. Notifications

- Email (SendGrid, Mailgun)
- SMS (Twilio)
- Push (Firebase)

When and Why You Need Web3 & Smart Contracts

Scenario	Web3/Smart Contract Usage
CEX backend	Web3 only to read/write blockchain data
Token Transfers	Web3 to interact with ERC20/BEP20 contracts
DEX/DeFi Platforms	Smart contracts to swap, stake, pool tokens
NFT platforms	Smart contracts to mint, transfer, list NFTs
Voting/Governance	Smart contracts for DAO-style proposals/voting

---

Fetching Networks Based on Coin/Token

Data Source	Description
CoinGecko API	/coins/{id} → Platforms field gives networks
CoinMarketCap	/cryptocurrency/info → Shows token networks
TokenLists.org	Curated token lists with network info
Chainlist.org	Lists EVM-compatible chains + chain IDs

Third-Party APIs & Services to Use

Purpose	Tool/API
Market Data	CoinGecko, CoinMarketCap
Wallet Management	Fireblocks, BitGo, Coinbase
KYC/AML	Sumsub, ShuftiPro, Jumio
Fiat Integration	MoonPay, Wyre, Ramp, Razorpay
Blockchain RPC	Infura, Alchemy, QuickNode
Smart Contract Tools	Hardhat, OpenZeppelin, Remix
Block Explorer API	Etherscan, BscScan

Purpose	Tool/API
Notification	Firebase, Twilio, SendGrid

Third-Party Services – Estimated Pricing

Wallet & Custody – Fireblocks

- Starts at approximately **$250/month** for basic asset management zoftwarehub.com+15getblock.io+15g2.com+15
- Enterprise-level pricing typically begins near **$10,000/month**, scaling with transaction volume and support

KYC/AML – Sumsub

- Basic plan: **$1.35 per verification**, with a **$149/month** minimum commitment zoftwarehub.com+4aventinelab.com+4sumsub.com+4
- Compliance plan: **$1.85 per check**, with a **$299/month** minimum aventinelab.com

Web3 Infrastructure – Alchemy

- Free tier: up to **100 million Compute Units (CUs)** per month support.coingecko.com+13alchemy.com+13alchemy.com+13
- PAYG: **$0.45 per million CUs** (up to 300M), then **$0.40 beyond** alchemy.com
- Growth plan: starts around **$49/month** alchemy-fitness.co.uk

Market Data – CoinGecko API

- Public (free) tier: ~**5–15 calls/min**, can increase to **30 calls/min** with a demo account support.coingecko.com
- Pro and Enterprise tiers: custom pricing, typically with higher rate limits support.coingecko.com+12coingecko.com+12support.coingecko.com+12

Proposed Database Schema

Here's a high-level schema capturing core entities for a CEX using Binance API, Fireblocks, etc.:

1. users

- `id PK`
- `email UNIQUE`
- `password_hash`
- `kyc_status ENUM('pending','verified','rejected')`
- `created_at, updated_at`

2. wallets

- id PK
- user_id FK → users.id
- network VARCHAR (e.g., 'ethereum', 'bsc')
- address VARCHAR
- private_key_encrypted
- created_at

3. assets

- id PK
- symbol VARCHAR (e.g., 'BTC', 'ETH')name VARCHAR
- network VARCHAR
- contract_address (nullable)
- decimals INT

4. balances

- id PK
- user_id FK → users.id
- asset_id FK → assets.id
- balance DECIMAL
- updated_at

5. orders

- id PK
- user_id FK → users.id
- asset_pair VARCHAR (e.g., 'ETH/BTC')
- side ENUM('buy','sell')type ENUM('limit','market','stop')
- price DECIMAL (nullable for market)
- amount DECIMAL
- filled DECIMAL DEFAULT 0
- status ENUM('open','partial','filled','cancelled')
- created_at, updated_at

6. trades

- id PK
- order_id FK → orders.id
- contra_order_id FK → orders.id
- price DECIMAL

- amount DECIMAL
- fee DECIMALtimestamp

7. deposits

- id PK
- user_id FK → users.id
- asset_id FK → assets.id
- tx_hash
- amount DECIMAL
- status ENUM('pending','confirmed','failed')
- network
- created_at, updated_at

8. withdrawals

- id PK
- user_id FK → users.id
- asset_id FK → assets.id
- tx_hash
- to_address VARCHAR
- amount DECIMAL
- fee DECIMAL
- status ENUM('pending','broadcast','confirmed','failed')
- network
- created_at, updated_at

9. fiat_transactions (if fiat integrated)

- id PK
- user_id FK → users.idtype ENUM('fiat_deposit','fiat_withdrawal')
- amount DECIMAL
- currency VARCHAR
- status ENUM('pending','completed','failed')
- provider VARCHAR (e.g., 'Stripe','Razorpay')
- created_at, updated_at

10. notifications

- id PK
- user_id FK → users.idtype ENUM('email','sms','push')
- message TEXT
- sent_at, status ENUM('queued','sent','failed')

Table Relationship Summary

Table	Linked To
wallets	users (user_id)
balances	users, assets
orders	users
trades	orders (order_id and other_order_id)
deposits	users, assets
withdrawals	users, assets
fiat_transactions	users
notifications	users
