KYC Application Architecture

# 1. Tech Stack

- Frontend: React for both mobile (React Native) and desktop (React.js).  
- Backend: Node.js with Express.js, using MongoDB or PostgreSQL for database, JWT for authentication.  
- Deployment: AWS Amplify/Netlify for web, Expo/React Native CLI for mobile, AWS/Google Cloud for backend.

# 2. Features

- User authentication (Sign-up, login, 2FA).  
- User profile with document upload (KYC documents).  
- Verification system (manual or third-party API).  
- Admin dashboard for verifying KYC submissions.  
- Notifications to inform users about their KYC status.

# 3. Steps

- Set up project environment using `create-react-app` for web and `React Native` or `Expo` for mobile.  
- Build UI components, including KYC forms, profile dashboard, and admin panel.  
- Create backend REST API with Express.js and handle file uploads with Multer.  
- Integrate manual or automated KYC verification systems like Jumio or Onfido.  
- Ensure proper security measures including HTTPS, encryption, and secure document storage.

# 4. Incorporating Blockchain for Security

Incorporating blockchain into your backend for the KYC application can significantly enhance security, transparency, and immutability. Here's how blockchain can be integrated into your KYC solution:

## Benefits of Using Blockchain for KYC

- \*\*Immutability\*\*: Ensures a tamper-proof record by preventing any alterations or deletions of data.  
- \*\*Decentralization\*\*: Storing KYC data across multiple nodes eliminates single points of failure.  
- \*\*Transparency and Auditability\*\*: Every action (e.g., KYC verification, document upload) can be recorded on the blockchain, creating an immutable audit trail.  
- \*\*User-Controlled Data\*\*: Users can control access to their KYC data via smart contracts.  
- \*\*Streamlining Compliance\*\*: Blockchain enables a single source of KYC information for multiple institutions.

## How to Incorporate Blockchain in Your KYC Application

### 1. Storing KYC Data on the Blockchain

Instead of storing documents directly on the blockchain (which can be expensive and slow), you can store cryptographic hashes (proof of documents), ensuring their integrity while storing actual files off-chain (e.g., AWS S3 or IPFS).

### 2. Using Smart Contracts

Smart contracts can be used to automate and manage the KYC process. These contracts can grant/revoke access to data, execute verification processes, and maintain an immutable audit trail.

### 3. Blockchain Platforms

Some common blockchain platforms for KYC applications:  
- \*\*Ethereum\*\*: Popular for smart contracts but has higher gas fees.  
- \*\*Hyperledger Fabric\*\*: Ideal for enterprise use cases with access control.  
- \*\*Corda\*\*: A blockchain platform focused on financial services.  
- \*\*Polygon\*\*: Layer 2 solution for Ethereum, offering lower fees.

### 4. Hybrid Approach

A hybrid approach balances security, cost, and performance by using traditional storage methods (AWS, IPFS) for documents and blockchain for integrity checks (storing document hashes) and smart contract automation.

## 5. Workflow

1. \*\*User Submission\*\*: KYC data is submitted, hashed, and uploaded to the blockchain while documents are securely stored off-chain.  
2. \*\*Verification\*\*: An admin verifies the submission, and the verification status is recorded on the blockchain.  
3. \*\*Access Control\*\*: Smart contracts handle data access requests.  
4. \*\*Audit & Compliance\*\*: Every action is traceable on the blockchain for auditing purposes.

## 6. Security Considerations

- \*\*Privacy\*\*: Encrypt sensitive data before storing it off-chain. Only store hashes or non-sensitive metadata on the blockchain.  
- \*\*Regulations\*\*: Ensure compliance with data privacy laws like GDPR.  
- \*\*Cost\*\*: Be mindful of transaction costs on public blockchains like Ethereum.