ENVIRONMENTAL SETUP

The **System Implementation** for the **Blockchain Interoperability with Cross-Chain Cryptography** system in healthcare will combine backend, and blockchain smart contract components to enable secure data exchange between different healthcare blockchains. It will also integrate the provided smart contracts into the implementation structure.

1. Backend Development with Node.js and Express.js

- **Server-side Implementation:**
  - The backend of the system is built using **Node.js** for executing server-side JavaScript and **Express.js** for managing API requests.
  - The backend is responsible for:
    - Processing user inputs for healthcare data.
    - Creating, tracking, and completing cross-chain transactions.
    - Interacting with Ethereum smart contracts to ensure secure and verifiable transactions.
- **APIs for Smart Contract Interactions:**
  - **Express.js** is used to create endpoints that interact with the **AtomicSwap** and **HealthcareData** smart contracts.
  - Example APIs:
    - `POST /api/patient/add`: Adds patient data to the blockchain.
    - `GET /api/patient/:address`: Retrieves patient data by address.
    - `POST /api/swap/create`: Creates a new atomic swap for healthcare data.
    - `POST /api/swap/complete`: Completes a swap with the correct secret.

2. Smart Contract Development with Solidity

Smart Contract 1: AtomicSwap.sol

- The `AtomicSwap` contract is responsible for enabling cross-chain healthcare data exchange.
- Healthcare providers and patients can initiate a swap to transfer data securely across chains.
- Key functionalities:
  - **createSwap**: Allows users to create a new swap with a hash and receiver address.

        - **completeSwap**: Allows the intended receiver to complete the swap with the correct secret.

Smart Contract 2: HealthcareData.sol

- The `HealthcareData` contract stores patient information securely on the blockchain.
- Patients can add and view their data, ensuring that records are tamper-proof and cross-chain interoperable.
- Key functionalities:
    - **addPatientData**: Allows patients to add their medical information.
    - **getPatientData**: Allows healthcare providers to query patient data based on an address.

3. Blockchain Interaction and Testing

- **Ethereum and Hardhat Testing:**
    - The blockchain system is deployed and tested on Ethereum using **Hardhat**.
    - **Mocha** and **Chai** are used for writing and running tests to verify that the healthcare data and atomic swap functionalities are working as expected.

4. Database Management with MongoDB

- **Patient Data Storage:**
    - MongoDB stores metadata of healthcare data (hashes, timestamps) to create an off-chain record of data exchanges.
    - Collections are created for storing patient data, healthcare providers, and transaction metadata.
- **API Integration:**
    - Backend APIs use MongoDB for retrieving and saving data related to patients and swaps.
    - Data in MongoDB complements the blockchain to store off-chain metadata, providing faster access when required.

5. Performance and Security

- **Security Mechanisms:**
    - Cross-chain cryptography is secured by **Elliptic Curve Cryptography (ECC)** for signing and verifying transactions.
    - The **AtomicSwap** contract ensures atomicity, meaning that either the data is fully transferred or the transaction is canceled.
- **Performance Optimization:**
    - Performance metrics such as transaction latency, block confirmation times, and data throughput are measured.
    - Smart contracts are optimized to reduce gas costs by minimizing storage usage.