Medimint - Project Proposal

This system aims to leverage blockchain technology for secure and efficient sharing of medical records between doctors and patients. The URI is stored on-chain and the metadata for the medical study can be stored on IPFS, which will reference to the patient information that is encrypted and securely stored off-chain.

Architecture Scheme for Medical Information Sharing System

Smart Contract Layer

- Smart Contract: The core smart contract handling the logic for token creation, transfer, and access control.
- Trusted Issuer Contract: Manages the entities (e.g., hospitals, doctors) authorized to issue or validate medical data NFTs.
- Deployer + Registry Smart Contract: Facilitate deployment of new contracts and maintain a registry for easy tracking and management.

Tokenization of Medical Records

- Medical Record Tokens: Each medical record (or a batch of related records) is represented as a unique token. These could be fungible (for standard records) or non-fungible (for unique, patient-specific records).

Access Control and Permissions

- Utilize transfer and balance functions for managing access to multiple records.

- Implement role-based access control within the smart contract to enforce who can access, transfer, or update a medical record token, etc

Security and Compliance

- Data Encryption: Encrypting sensitive medical data both on and off-chain.
- Compliance Layer: Ensuring that all interactions with the blockchain adhere to healthcare regulations like HIPAA.

Traceability

- Each action that involves the medical records will be traceable and visible on the blockchain explorer - mint NFT, share permission, revoke permission, etc

Blockchain Network

- Public/Private Blockchain: A public blockchain can be used, raising the need to gas tokens that needs to be available for each user. Alternatively, a private blockchain or account abstraction can be considered.

Note:

This streamlined proposal also demonstrates the feasibility of implementing the following functionalities using the Stellar network.

- HD Wallet Management: Implementation of Hierarchical Deterministic (HD) wallets.
- ECDSA Encryption: Integration of Elliptic Curve Digital Signature Algorithm (ECDSA) for secure transaction signing.
- NFT Minting: Basic functionality to mint Non-Fungible Tokens (NFTs) on the Stellar blockchain.
- Gasless Transactions with Bump Fees: Enabling transactions with no upfront fees for users, utilizing Stellar's bump fee mechanism.

Application - Backend (API + Workers)

We plan to build the system using Node.Js framework due to its reliability and versatility. Also, there is a supported package for handling DICOM format

System Management:

- AWS Cloud Integration: Leverage AWS services for scalability and reliability.
- Media Upload/Download, CDN, Caching: Ensure efficient handling of large files like medical images, using CDN for faster access and caching strategies for improved performance.

Encode/Decode DICOM:

- Handle medical imaging data effectively, ensuring compatibility with standard medical image formats.

Encrypt/Decrypt Resources:

- Implement strong encryption algorithms for data at rest and in transit, crucial for patient data privacy.

Wallet Management:

- Manage private keys (PK), sign messages and transactions, broadcast transactions. ● Lit Network Integration: For decentralized access control and data encryption.

User Management:

- Comprehensive user management including authentication, profile management, notifications, and email services.

DB Management:

- Manage local SQL databases, ensure efficient data migrations, regular maintenance, and robust backup strategies.

Access Control List:

- Implement robust authentication and authorization mechanisms, crucial for data security.

User Actions:

- Enable users to interact with medical records (create, share, view, revoke permissions, request access).

Blockchain Watchers (Workers):

- Monitor blockchain events, important for syncing on-chain events with the application.

Application - Frontend (React.JS)

UI/UX Design Integration:

- Integrate design, ensuring they are user-friendly and intuitive, especially given the complexity of the data and interactions.

Responsive Design:

- Ensure compatibility across various devices, particularly important for medical professionals and patients who may access the system through different platforms.

Accessibility:

- Design with accessibility in mind to accommodate users with different abilities.

Security Considerations:

- Implement security best practices in the frontend to prevent common vulnerabilities like XSS, CSRF, etc.
- Real-Time Updates:
- For notifications and blockchain event monitoring.

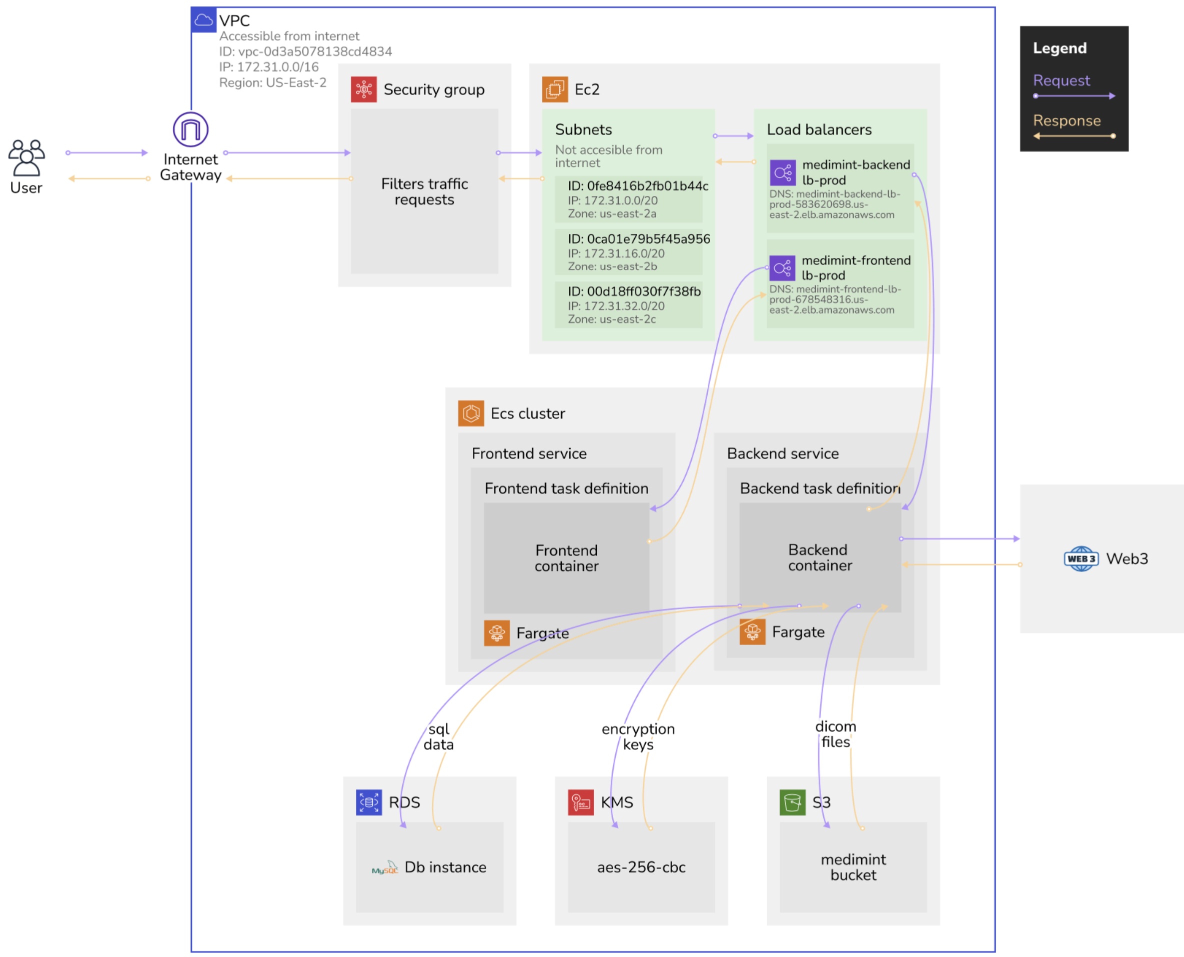
Additional Considerations

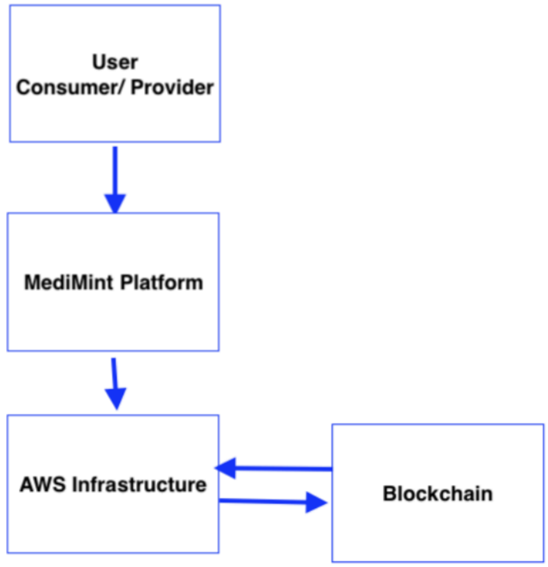
- Regulatory Compliance:
  - Ensure compliance with healthcare regulations like HIPAA and data protection laws.
- Testing and QA:
  - Manual testing to ensure reliability and user-friendliness, unit testing for smart contracts with 100% code coverage.
- Scalability:
  - Design the system to be scalable to handle increasing loads and future expansions.
- Documentation and Training:
  - Provide comprehensive documentation and training materials for end-users.

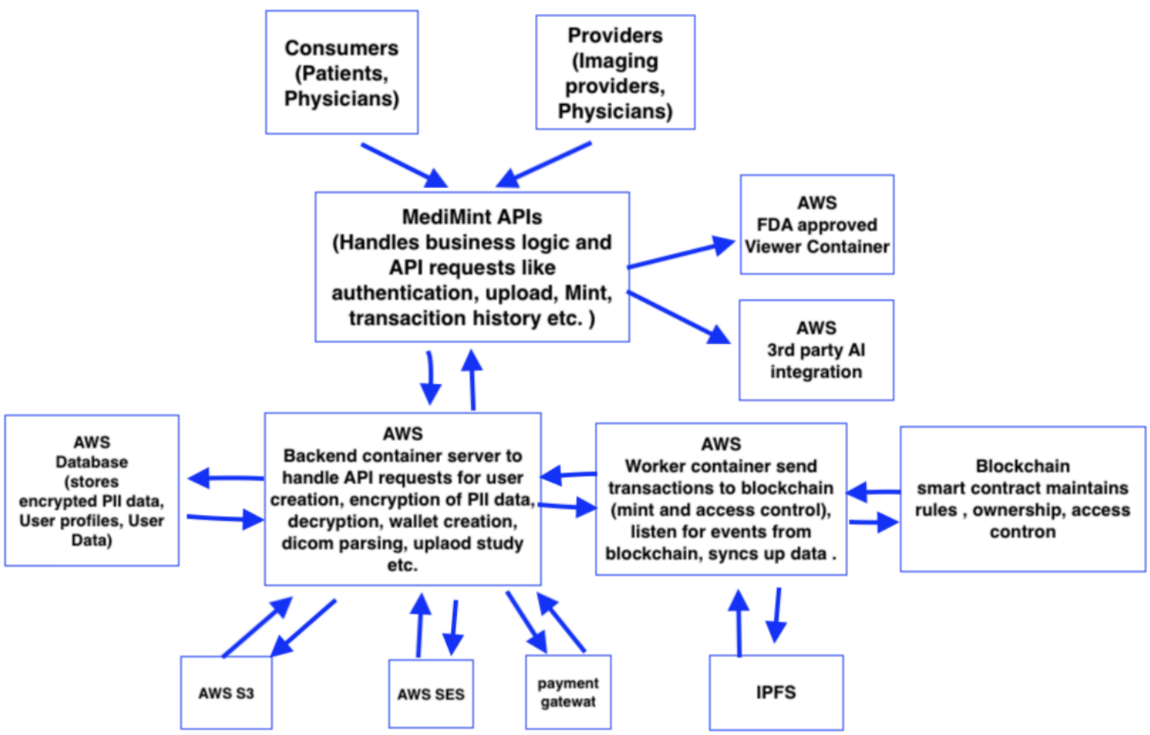
Architectural Diagrams:

AWS ARCHITECTURE

MEDIMINT PROJECT




User
Consumer/ Provider
MediMint Platform
AWS Infrastructure
Blockchain


Consumers (Patients, Physicians)
Providers (Imaging providers, Physicians)
MediMint APIs (Handles business logic and API requests like authentication, upload, Mint, transacition history etc.)
AWS FDA approved Viewer Container
AWS 3rd party AI integration
AWS Database (stores encrypted PII data, User profiles, User Data)
AWS Backend container server to handle API requests for user creation, encryption of PII data, decryption, wallet creation, dicom parsing, uplaod study etc.
AWS Worker container send transactions to blockchain (mint and access control), listen for events from blockchain, syncs up data .
Blockchain smart contract maintains rules , ownership, access contron
AWS S3
AWS SES
payment gatewat
IPFS