Group Members:

Konyana M.L 202002723

Phalali T.J 202000316

Masike T. 202002500

Khonyane T. 202003604

Mafata T 201602230

System Architecture Design Document

The architecture of the Ntsoekhe Distributed Database Management System (DDBMS) will be designed to leverage Django integrated with a peer-to-peer (P2P) network framework, such as IPFS (InterPlanetary File System) or Dat, to establish a decentralized network of nodes. This approach will enable seamless communication and data synchronization among distributed nodes, ensuring scalability and fault tolerance.

Key Components of the System Architecture:

1. Peer-to-Peer Network Integration

Components:

- Node Communication: Nodes will communicate using secure P2P protocols like BitTorrent or Kademlia to exchange data and synchronize information.
- Data Distribution: Utilizing Django's sharding mechanism based on consistent hashing or Distributed Hash Table (DHT) will facilitate data distribution across the network.
- Network Topology: The system will operate without a central server, enabling each node to act autonomously and contribute to the overall network.

Benefits:

- Scalability: Nodes can be added or removed dynamically, allowing the system to scale effortlessly with increasing data volumes.
- Fault Tolerance: Redundant copies of data across nodes ensure data availability even in the event of node failures.
- Decentralization: Eliminates single points of failure and reduces reliance on a centralized infrastructure.

2. Communication and Synchronization

Secure and efficient communication channels will be established between Django instances running on distributed nodes. This ensures data integrity, confidentiality, and consistency across the network.

Strategies:

- Encryption: Implement SSL/TLS encryption to secure data transmission between nodes and protect against unauthorized access.

- Consistency Protocols: Use synchronization protocols to maintain data consistency and coherence across distributed components.
- Peer Discovery: Employ protocols like IPFS or Dat for peer discovery and management within the network.

Implementation:

- Each node will act as a peer, capable of both serving and retrieving data from other nodes.
- Communication protocols will be optimized for low latency and high throughput to support real-time data access and updates.

3. Security and Access Control

Robust security measures will be implemented to safeguard sensitive healthcare data and ensure compliance with privacy regulations.

Features:

- Authentication: Utilize Django's authentication system to control access to the DDBMS based on user roles and permissions.
- Authorization: Implement role-based access control (RBAC) to restrict user actions and data visibility.
- Data Encryption: Apply encryption techniques to protect data at rest and in transit, preventing unauthorized access.

Compliance:

The system will adhere to healthcare data privacy regulations by implementing stringent security measures and access controls.

Key Components of Database Schema Designs:

Decentralized Data Model:

- Tables: Define tables to store various types of healthcare data, including patient information, medical records, user roles, access controls, and audit trails.
- Entity-Relationships: Establish relationships between tables to maintain data integrity and facilitate efficient data retrieval and updates.

Data Replication and Redundancy:

- Replication Strategy: Implement data replication to ensure high availability and fault tolerance across distributed nodes.
- Redundant Copies: Maintain redundant copies of critical data to mitigate the risk of data loss due to node failures or network interruptions.

CRUD Operations Support:

- Table Structures: Design table structures to support CRUD (Create, Read, Update, Delete) operations across distributed nodes.
- Data Consistency: Implement transaction management and locking mechanisms to ensure data consistency during concurrent operations.

Data Encryption and Masking:

- Field-Level Encryption: Apply encryption techniques to sensitive fields (e.g., patient identifiers, health conditions) to protect patient privacy and comply with security standards.
- Data Anonymization: Incorporate data anonymization techniques to anonymize personal information where necessary to ensure compliance with privacy regulations.

Entity-Relationship Diagram (ERD)

Entities:

Patients:

Attributes:

- PatientID (Primary Key)
- FirstName
- LastName
- DateOfBirth
- Gender
- Phone
- Email

MedicalRecord:

Attributes:

- RecordID (Primary Key)
- PatientID (Foreign Key referencing Patients.PatientID)
- DateOfVisit
- Diagnosis
- Treatment
- RoleID (FK)

LabResult:

Attributes:

- ResultID (Primary Key)
- RecordID (Foreign Key referencing MedicalRecords.RecordID)
- TestName
- TestDate
- TestResult

UserRoles:

Attributes:

- RoleID (Primary Key)
- RoleName

AccessControls:

Attributes:

- UserID (Primary Key)
- UserName
- RoleID (Foreign Key referencing UserRoles.RoleID)

Relationships:

Patients to MedicalRecords (One-to-Many):

- Each patient (Patients.PatientID) can have multiple medical records (MedicalRecords.PatientID).

MedicalRecords to LabResult (One-to-Many):

- Each medical record (MedicalRecords.RecordID) can have multiple lab test results (LabResult.RecordID).

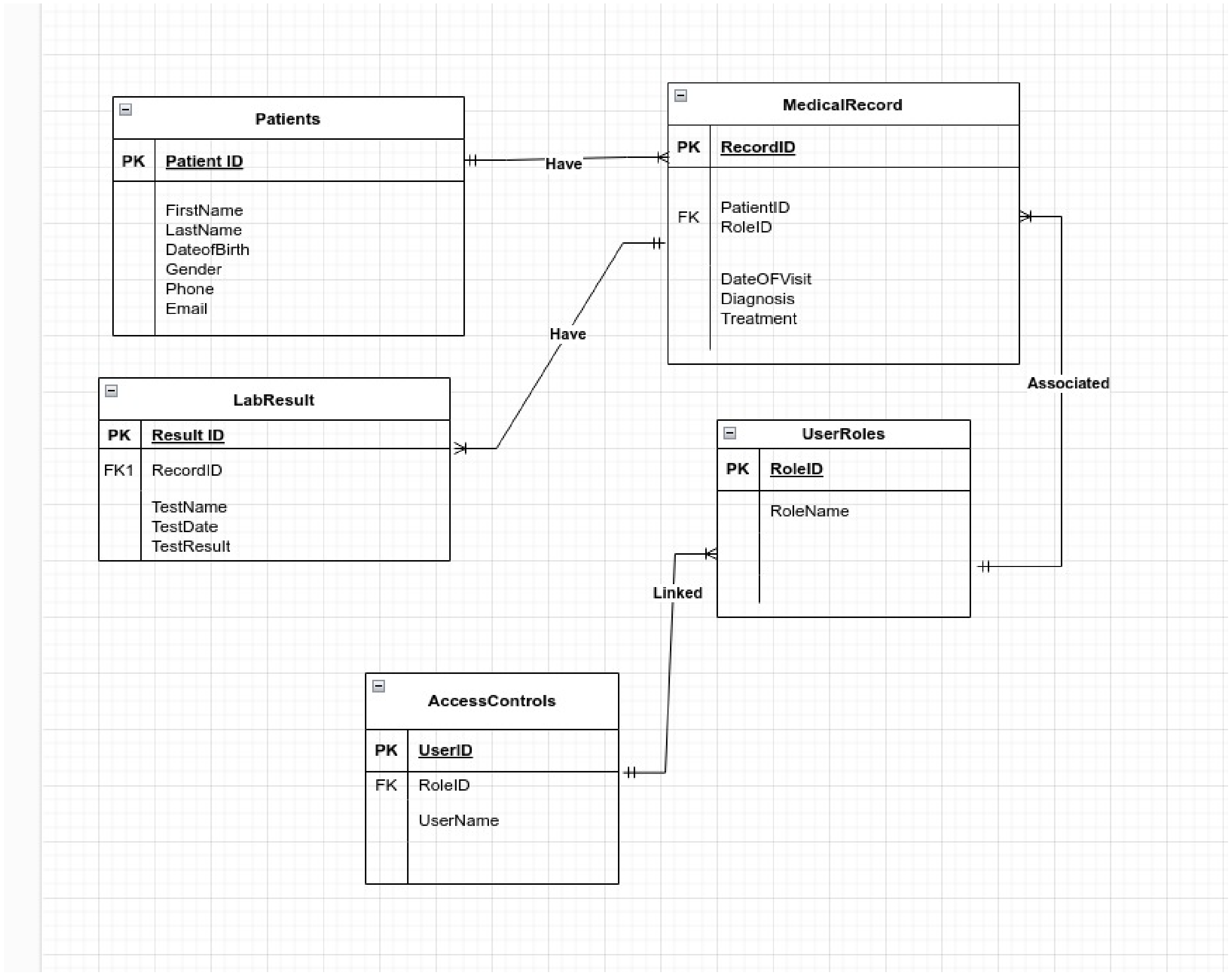
MedicalRecords to UserRoles (Many-to-One):

- Each medical record (MedicalRecords.RoleID) is associated with a specific role (UserRoles.RoleID), indicating the role responsible for the record.

AccessControls to UserRoles (One-to-Many):

- Each AccessControls entry (AccessControls.RoleID) is associated with exactly one UserRoles entry (UserRoles.RoleID), representing the role assigned to the user for access control purposes.
- However, a single UserRoles entry can be referenced by multiple AccessControls entries, allowing multiple users to have the same role (permissions).

ER diagram:


Patients
PK Patient ID
FirstName
LastName
DateofBirth
Gender
Phone
Email
Have
MedicalRecord
PK RecordID
FK PatientID
RoleID
DateOFVisit
Diagnosis
Treatment
Have
LabResult
PK Result ID
FK1 RecordID
TestName
TestDate
TestResult
Associated
UserRoles
PK RoleID
RoleName
Linked
AccessControls
PK UserID
FK RoleID
UserName