**1. Database Creation**

**Step 1: Create the Database**

sql

CopyEdit

CREATE DATABASE ehr\_system;

USE ehr\_system;

✅ **Related to:**

* **General Requirement**: Establishing a database to manage EHRs, patients, and transactions.
* **Scalability (3.3.1)**: Forms the base structure for handling millions of transactions.

**2. EHR System Configuration**

**Step 3: Create the ehr\_systems Table**

sql

CopyEdit

CREATE TABLE ehr\_systems (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

name VARCHAR(255) UNIQUE NOT NULL,

base\_api\_url TEXT NOT NULL,

auth\_config JSON NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

✅ **Related to:**

* **3.1.1**: Maps EHR data fields.
* **3.1.3**: Supports extensibility for future EHR integrations.
* **6. Example of EHR Mapping**: This table stores mapping details for Athena, Allscripts, and other EHRs.

**3. Healthcare Providers**

**Step 2: Create the providers Table**

sql

CopyEdit

CREATE TABLE providers (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

name VARCHAR(255) NOT NULL,

ehr\_system\_id CHAR(36),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (ehr\_system\_id) REFERENCES ehr\_systems(id) ON DELETE SET NULL

);

✅ **Related to:**

* **3.1.1**: Maps hospital/clinic (providers) to an EHR.
* **3.1.5**: Helps manage EHR mappings for different hospitals/clinics.
* **6. Example of EHR Mapping**: Links providers to specific EHRs like Athena or Allscripts.

**4. Patients & Their Data**

**Step 4: Create the patients Table**

sql

CopyEdit

CREATE TABLE patients (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

provider\_id CHAR(36),

name VARCHAR(255) NOT NULL,

gender ENUM('Male', 'Female', 'Other'),

dob DATE,

address TEXT,

phone VARCHAR(20),

email VARCHAR(255),

emergency\_contact TEXT,

insurance\_provider VARCHAR(255),

insurance\_policy\_number VARCHAR(255),

primary\_care\_physician VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (provider\_id) REFERENCES providers(id) ON DELETE CASCADE

);

✅ **Related to:**

* **3.1.1**: Handles patient data.
* **3.1.2**: Ensures patients are linked to valid hospitals (providers).
* **3.1.3**: Allows scalable patient records across multiple EHRs.
* **3.1.8**: Can be extended for multi-language support.
* **6. Example of EHR Mapping**: Maps patient details to specific EHR field names.

**5. Survey Questions & EHR Field Mapping**

**Step 5: Create the questions Table**

sql

CopyEdit

CREATE TABLE questions (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

text TEXT NOT NULL,

category VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Step 6: Create the ehr\_mappings Table**

sql

CopyEdit

CREATE TABLE ehr\_mappings (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

ehr\_system\_id CHAR(36),

question\_id CHAR(36),

ehr\_field VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (ehr\_system\_id) REFERENCES ehr\_systems(id) ON DELETE CASCADE,

FOREIGN KEY (question\_id) REFERENCES questions(id) ON DELETE CASCADE

);

✅ **Related to:**

* **3.1.1**: Ensures the mapping of input data (questions) to EHR fields.
* **3.1.5**: Manages mappings per EHR.
* **6. Example of EHR Mapping**: Links specific questions to Athena and Allscripts fields.

**6. Storing Patient Responses**

**Step 7: Create the patient\_responses Table**

sql

CopyEdit

CREATE TABLE patient\_responses (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

patient\_id CHAR(36),

question\_id CHAR(36),

answer TEXT NOT NULL,

submitted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (patient\_id) REFERENCES patients(id) ON DELETE CASCADE,

FOREIGN KEY (question\_id) REFERENCES questions(id) ON DELETE CASCADE

);

✅ **Related to:**

* **3.1.1**: Stores patient input data.
* **3.1.2**: Ensures patient responses are correctly linked.
* **6. Example of EHR Mapping**: These answers are stored in fields like HISTORY\_MEDICAL\_PATIENT in Athena.

**7. Tracking EHR API Transactions**

**Step 8: Create the ehr\_transactions Table**

sql

CopyEdit

CREATE TABLE ehr\_transactions (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

patient\_id CHAR(36),

ehr\_system\_id CHAR(36),

status ENUM('PENDING', 'SUCCESS', 'FAILED') NOT NULL,

response JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (patient\_id) REFERENCES patients(id) ON DELETE CASCADE,

FOREIGN KEY (ehr\_system\_id) REFERENCES ehr\_systems(id) ON DELETE CASCADE

);

✅ **Related to:**

* **3.1.2**: Tracks transaction consistency and error handling.
* **3.1.6**: Helps measure API performance and scalability.
* **3.3.3**: Ensures fault tolerance via status tracking.
* **3.3.5**: Enables secure logging of sensitive API transactions.

**8. User Management**

**Step 9: Create the users Table**

sql

CopyEdit

CREATE TABLE users (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

role ENUM('Admin', 'Clinician') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

✅ **Related to:**

* **3.2.1**: Supports user management for the frontend tool.
* **3.3.5**: Implements authentication and role-based access control.

**9. Auditing & Logging Changes**

**Step 10: Create the audit\_logs Table**

sql

CopyEdit

CREATE TABLE audit\_logs (

id CHAR(36) PRIMARY KEY DEFAULT (UUID()),

user\_id CHAR(36),

ehr\_mapping\_id CHAR(36),

action VARCHAR(255) NOT NULL,

changes JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL,

FOREIGN KEY (ehr\_mapping\_id) REFERENCES ehr\_mappings(id) ON DELETE CASCADE

);

✅ **Related to:**

* **3.1.2**: Ensures all transactions and changes are logged.
* **3.3.3**: Supports service resiliency by tracking changes.
* **3.3.5**: Improves security via logging.

**10. Performance Optimization**

**Step 11: Create Indexes**

sql

CopyEdit

CREATE INDEX idx\_patient\_provider ON patients(provider\_id);

CREATE INDEX idx\_response\_patient\_question ON patient\_responses(patient\_id, question\_id);

CREATE INDEX idx\_transactions\_patient ON ehr\_transactions(patient\_id);

✅ **Related to:**

* **3.1.6**: Improves performance for large-scale queries.
* **3.3.1**: Helps optimize system for millions of users.

**Conclusion**

The SQL schema aligns **perfectly** with the **full-stack assignment** by ensuring:  
✅ Scalable **EHR integration**  
✅ **Accurate patient data mapping**  
✅ **Transaction consistency & security**  
✅ Support for **millions of concurrent users**

Would you like additional API design recommendations or frontend structure suggestions? 🚀