Take-Home Assessment: Doctor Account Management, PDF Upload Portal, and Linking to Patient Profiles

Objective: To assess the candidate's ability to build a functional user authentication system for doctors, including account creation, login, PDF upload capabilities, and linking doctor profiles with patient profiles.

Task:

1. **Create a Combined Account Creation and Login Portal:**
    - Implement a form where doctors can sign up by entering their name, email, password, and specialty.
    - Allow existing users to log in using their email and password.
    - Validate the input data (e.g., email format, password strength).
    - Store the doctor's information in a secure database (e.g., DynamoDB or PostgreSQL).
2. **PDF Upload Functionality:**
    - Once logged in, allow doctors to upload PDF files.
    - Store the uploaded PDFs in a storage service (e.g., Amazon S3).
    - Associate uploaded PDFs with the logged-in doctor in the database.
3. **Linking Doctor Profiles with Patient Profiles:**
    - Allow doctors to link their profiles with patient profiles.
    - Implement functionality to assign or link a patient to a doctor.
    - Store the doctor-patient relationship in the database.
4. **Database Schema:**
    - Design a database schema for storing doctor accounts, patient accounts, uploaded PDFs, and doctor-patient relationships.
    - Ensure efficient retrieval and association of PDFs and patient profiles with the corresponding doctor.
5. **Frontend and Backend Implementation:**
    - Use a frontend framework like React for the UI.
    - Implement the backend using Node.js, Python, or any preferred language/framework.
    - Ensure that the frontend communicates with the backend securely (e.g., HTTPS).

Examples

1. Database Schema Design:

- **Doctors Table:**

```
CREATE TABLE Doctors (
    DoctorID SERIAL PRIMARY KEY,
```

```
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    PasswordHash VARCHAR(255),
    Specialty VARCHAR(100)
);
```

- **Patients Table:**

```
CREATE TABLE Patients (
    PatientID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    PasswordHash VARCHAR(255)
);
```

- **PDFs Table:**

```
CREATE TABLE PDFs (
    PDFID SERIAL PRIMARY KEY,
    DoctorID INT,
    FilePath VARCHAR(255),
    UploadDate TIMESTAMP,
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

- **Doctor-Patient Relationship Table:**

```
CREATE TABLE DoctorPatient (
    DoctorID INT,
    PatientID INT,
    PRIMARY KEY (DoctorID, PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
);
```

2. Sample Dummy Data:

- **Doctors:**

```
[
  {
    "DoctorID": 1,
    "Name": "Dr. John Smith",
    "Email": "john.smith@example.com",
    "PasswordHash": "hashedpassword123",
    "Specialty": "Cardiology"
  },
  {
    "DoctorID": 2,
    "Name": "Dr. Emily Johnson",
    "Email": "emily.johnson@example.com",
    "PasswordHash": "hashedpassword456",
    "Specialty": "Neurology"
  }
]
```


- **Patients:**


```
[
  {
    "PatientID": 1,
    "Name": "Alice Brown",
    "Email": "alice.brown@example.com",
    "PasswordHash": "hashedpassword789"
  },
  {
    "PatientID": 2,
    "Name": "Bob White",
    "Email": "bob.white@example.com",
    "PasswordHash": "hashedpassword012"
  }
]
```


- **PDFs:**


```
[
  {
```

```
    "PDFID": 1,
    "DoctorID": 1,
    "FilePath": "s3://your-bucket-name/1/pdf1.pdf",
    "UploadDate": "2023-07-01T12:34:56Z"
  },
  {
    "PDFID": 2,
    "DoctorID": 2,
    "FilePath": "s3://your-bucket-name/2/pdf2.pdf",
    "UploadDate": "2023-07-02T12:34:56Z"
  }
]
```


- **Doctor-Patient Relationships:**


```
[
  {
    "DoctorID": 1,
    "PatientID": 1
  },
  {
    "DoctorID": 2,
    "PatientID": 2
  }
]
```


Deliverables:

- Source code for the frontend and backend.
- Database schema design.
- Screenshots, video, or PowerPoint presentation demonstrating the output.

Evaluation Criteria:

- Functionality: Does the account creation, login process, PDF upload feature, and linking of doctor and patient profiles work as expected?
- Code Quality: Is the code well-organized and documented?
- User Experience: Is the UI user-friendly and responsive?

Additional Instructions:

- Feel free to make any assumptions necessary to complete the assessment.
- Use any examples or dummy data you find suitable.
- Utilize GenAI tools or any other resources that can help you.
- This assessment is open-ended; focus on showcasing your skills and creativity.