Analysis Document for Primary Care Clinics Management Application

:Technology Stack

- (**Backend:** C# with ASP.NET Core 9 (MVC
- **Frontend:** React with React Query for data fetching and state management
- **Database:** SQL Server
- **API Communication:** RESTful API

Overview .1

This application is designed to manage primary care clinics by streamlining the process of patient registration, appointment scheduling, medical file management, and treatment session tracking. It supports multiple user roles (Super Admin, Admin, Clinic Manager, Specialist, Patient's Employer, and Data Entry/Receptionist) with defined permissions and .responsibilities

:The solution is divided into three main projects

1. **:Entity & Database Project**
   - Handles the creation of database models, entity configurations, and .relationships
   - .Implements repository patterns to interact with the database
   - .Provides domain models and business logic abstraction
2. **:Medical File & Sessions Project**
   - .Manages patient records, treatment sessions, and associated attachments
   - Contains business logic for managing medical files, logging treatment .sessions, and handling file statuses
3. **:Appointments Management Project**
   - .Manages the scheduling of appointments and working hours
   - .Implements logic for booking, updating, and canceling appointments
   - .Links appointments to patients, clinics, and medical files

---

System Architecture .2

Layered Architecture .2.1

- **:Presentation Layer**

    - **:(Frontend (React**
        - Uses React components, routing, and state management via React Query.
        - .Consumes RESTful APIs for data operations
        - .Provides responsive UI for various user roles
- **:Business Logic Layer**
    - **:ASP.NET Core Controllers**
        - .Handle incoming API requests
        - .Enforce business rules and orchestrate between services
    - **:Services/Managers**
        - Encapsulate business operations (e.g., patient registration, file assignment, appointment scheduling).
- **:Data Access Layer**
    - **:Entity Framework Core 9**
        - .Manages data access through a Repository Pattern
        - .Maps domain entities to SQL Server tables
        - .Implements Unit of Work for transaction management

API Communication .2.2

- The RESTful API is designed to provide CRUD operations for patients, medical files, sessions, appointments, and other entities.
- React Query will be used on the front end for handling data fetching, caching, and synchronization with the server.

Functional Requirements .3

User Roles and Permissions .3.1

1. **:Super Admin**
    - .Full access to all features
    - Can manage user roles, system configurations, and data integrity across all modules.
2. **:Admin**
    - Manages clinics, adds new users, oversees patient records, tracks specialist performance, and handles referral processes.
    - Responsible for appointment scheduling and defining clinic-specific working hours and appointment slots.
3. **:Clinic Manager**
    - .Oversees clinic operations
    - .Assigns medical files to specialists
    - Monitors specialist performance and manages patient file transfers when necessary.
4. **:Specialist**
    - .Manages patient records and builds medical files

- Logs treatment sessions, writes assessments, uploads attachments, schedules appointments, and concludes treatments.
- Has the ability to refer patients to other departments if needed.

5. **Patient's Employer:**
   - Registers patients.
   - Monitors the treatment progress and appointment status.
   - Receives final reports and recommendations post treatment.
6. **Data Entry (Receptionist):**
   - Handles patient registration and creation of initial medical files.
   - Schedules appointments and maintains basic patient data.

3.2 Application Procedures

Patient Referral & Registration

- **Referral Source Determination:**
  - The system will capture whether a patient is referred by their employer or visits the clinic independently.
- **Registration Process:**
  - Receptionist registers the patient and creates a new medical file.
  - The file is linked with the patient's employee number (if applicable), employer details, and file opening date.

Clinic Manager Actions

- **File Assignment:**
  - Upon registration, the clinic manager assigns the patient's file to an appropriate specialist.
  - Schedules an initial consultation.
- **Performance Tracking:**
  - Monitors updates in patient files and evaluates specialist performance.
  - Reassigns files if necessary.

Specialist Actions

- **Medical File Management:**
  - Builds and updates the patient's medical file.
  - Records treatment sessions and evaluations.
- **Appointment Scheduling:**
  - Schedules further appointments, logs session details, uploads attachments, and concludes treatments.
  - May refer patients to other departments before closing the file.

Employer Actions

- **Monitoring:**
  - Tracks patient's appointment schedules and file statuses.
  - Receives final reports after file closure.
- **Post-Treatment:**

- After a specified period post file closure, access to patient records is restricted, though files remain stored.

Non-Functional Requirements .4

- **Scalability:**
  - The application should handle a growing number of users and data records.
  - Scalable architecture with efficient API handling via React Query.
- **Performance:**
  - Optimize API endpoints for quick response times.
  - Ensure efficient data queries with proper indexing in SQL Server.
- **Security:**
  - Implement role-based authorization and authentication.
  - Secure sensitive data both in transit (using HTTPS) and at rest.
  - Protect against common web vulnerabilities (e.g., SQL Injection, XSS).
- **Maintainability:**
  - Use a clean separation of concerns (e.g., Repository Pattern, MVC).
  - Implement thorough logging and exception handling.
- **Usability:**
  - Intuitive UI/UX design for each user role.
  - Responsive design for desktop and mobile access.

Database Design and Proposed System Tables .5

Below is an initial list of tables along with brief descriptions. Additional reference or lookup tables can be added as the application requirements evolve.

1. **Patients Table**
   - **Columns:** PatientID (PK), FirstName, LastName, DateOfBirth, Gender, ContactInfo, EmployerID (FK, if applicable), RegistrationType (Employer/Independent), etc.
   - **Purpose:** Stores personal and essential identification details.
2. **Medical Files Table**
   - **Columns:** FileID (PK), PatientID (FK), CreatedDate, StatusID (FK), AssignedSpecialistID (FK), etc.
   - **Purpose:** Contains patient medical records and file status information.
3. **Treatment Sessions Table**
   - **Columns:** SessionID (PK), FileID (FK), SessionDate, Description, SpecialistNotes, etc.
   - **Purpose:** Logs treatment details linked to a medical file.
4. **Attachments Table**
   - **Columns:** AttachmentID (PK), FileID (FK) or SessionID (FK), AttachmentTypeID (FK), FilePath, UploadedDate, etc.

- **Purpose:** Stores file attachments related to medical files or treatment sessions.

5. **Users Table**
   - **Columns:** UserID (PK), Username, PasswordHash, Email, RoleID (FK), ClinicID (FK, if applicable), etc.
   - **Purpose:** Maintains main user data and links to roles and clinics.
6. **Roles Table**
   - **Columns:** RoleID (PK), RoleName (e.g., Super Admin, Admin, Clinic Manager, Specialist, Employer, Receptionist), Description, etc.
   - **Purpose:** Defines user roles and associated permissions.
7. **Medical File Status Table**
   - **Columns:** StatusID (PK), StatusName (e.g., Under Treatment, Awaiting Next Appointment, Treatment Completed & File Closed), etc.
   - **Purpose:** Tracks the status of each medical file.
8. **Recommendations & Final Actions Table**
   - **Columns:** RecommendationID (PK), FileID (FK), RecommendationText, FinalDecisionDate, SpecialistID (FK), etc.
   - **Purpose:** Stores final assessments, recommendations, and actions taken when closing a file.
9. **Attachment Types Table**
   - **Columns:** AttachmentTypeID (PK), TypeName (e.g., Evaluation Document, Test Result, Prescription), etc.
   - **Purpose:** Defines the various types of attachments.
10. **Clinics Table**
    - **Columns:** ClinicID (PK), ClinicName, Address, ContactDetails, AdministrativeUnitID (FK), etc.
    - **Purpose:** Stores clinic-specific information.
11. **Administrative Units Table**
    - **Columns:** AdministrativeUnitID (PK), UnitName, Description, etc.
    - **Purpose:** Represents the administrative divisions or units associated with clinics.

Additional Considerations:

- **Audit Tables:** To track changes for sensitive tables (e.g., Medical Files, Appointments).
- **Lookup Tables:** For common data values such as appointment statuses, session types, etc.

Use Cases .6

Patient Registration & File Creation .6.1

- **Actor:** Receptionist / Data Entry
- **Description:**

- .(Register a new patient (either referred by employer or self-initiated
- .Automatically generate a medical file with linked details
- .(Store registration metadata (e.g., file creation date, referral type

Appointment Scheduling .6.2

- **Actor:** Admin, Clinic Manager, or Specialist
- **:Description**
  - .View available time slots based on defined working hours
  - .Schedule, update, or cancel appointments
  - .Link appointments to patient records and medical files

Medical File Management .6.3

- **Actor:** Specialist, Clinic Manager
- **:Description**
  - .Update patient medical files during treatment sessions
  - .Log treatment details, add attachments, and update file status
  - Close a medical file once treatment is complete, optionally storing final .recommendations

Monitoring and Reporting .6.4

- **Actor:** Employer, Admin, Clinic Manager
- **:Description**
  - .Employers can view the progress and status of their referred patients
  - Generate reports on treatment progress, appointment adherence, and overall .clinic performance
  - Post-treatment access control where employers can no longer track records .after a specified period

---

Integration with Frontend .7

- **:React Components**
  - .Each user role will have tailored dashboards
  - .Appointment booking and file management interfaces
  - .Forms for patient registration and treatment session logging
- **:React Query**
  - .Efficiently manage server state
  - .Handle caching and data refetching for real-time updates
  - .Provide optimistic updates for a seamless user experience
- **:Error Handling & Feedback**
  - .Implement global error boundaries
  - Provide immediate feedback to users on success or failure of operations .((e.g., form validations, API errors