# Take-Home Assignment: Build a Mini Clinical Dashboard

## Overview

Create a full stack web app that displays patient demographics and upcoming appointments in a simple dashboard. The frontend should be built using React + TypeScript, and the backend using Python (FastAPI + Strawberry GraphQL). The system should ingest messy CSV data that is provided, validate and normalize it, store it in a relational database, and expose it via a GraphQL API.

## The Scenario

You've joined a team building a clinical system. You've been handed a CSV export of raw patient data from an external source, including patient demographics and appointments. The data has inconsistencies, missing values, and formatting issues.

Your task is to:

1. Clean and model the data into a normalized schema. Create a mutation for accepting the csv.
2. Store it in a relational database.
3. Build a GraphQL API that supports:
    - Listing all patients
    - Listing all appointments
    - Querying a single patient with their appointments
4. Build a simple React UI that:
    - Displays a table of patients with key fields
    - Allows clicking into a patient to see their appointment details

## Requirements

1. Data Ingestion

- Load a provided patients_and_appointments.csv file (provided)
- Parse, validate (e.g., missing fields, malformed phone/email/DOB), and transform it into clean relational tables:
    - Patients
    - Appointments
    - Plus relational tables and normalized tables as you see fit

2. Backend (Python or JavaScript)

- Use FastAPI and Strawberry GraphQL.
- Add GraphQL queries for:
  - patients
  - appointments
  - patient(uuid) → returns patient + nested appointments

3. Frontend (React + TypeScript)

- Use React, TypeScript, Apollo Client (bonus), and MUI (bonus)
- Build a responsive UI with two views:
  - Patient List View – Table showing patient name, DOB, phone, and appointment count.
  - Patient Detail View – When a user clicks a row, display their full info + a table of appointments.
- Use GraphQL queries to fetch data.

You do not need:

- Authentication
- API pagination for long lists
- Unit tests (bonus if included)

## Deliverables

- GitHub repo:
  - React Frontend in /frontend
  - Graph Backend in /backend
  - Sample messy data patients_and_appointments.txt
  - SQL queries for creating the database queries.sql
  - README.md with:
    - Setup instructions
    - Your comments on data modeling and validation choices

## Time Expectation

This assignment should take 2-4 hours to complete. It is okay to stub or simplify some parts—what we care about is how you reason, model, and communicate. Share the github repo path with us within 5 days of receiving the test.