

ADVANCED DATABASE SUBJECT

PHASE 2 PHYSICAL LEVEL DESIGN

Student made: 22127069 - Nguyễn Đặng Hoàng Đình

22127107 - Nguyễn Thế Hiền

Lecturer in charge: Hồ Thị Hoàng Vy

SEMESTER III – SCHOOL YEAR 2024-2025

Table of Contents

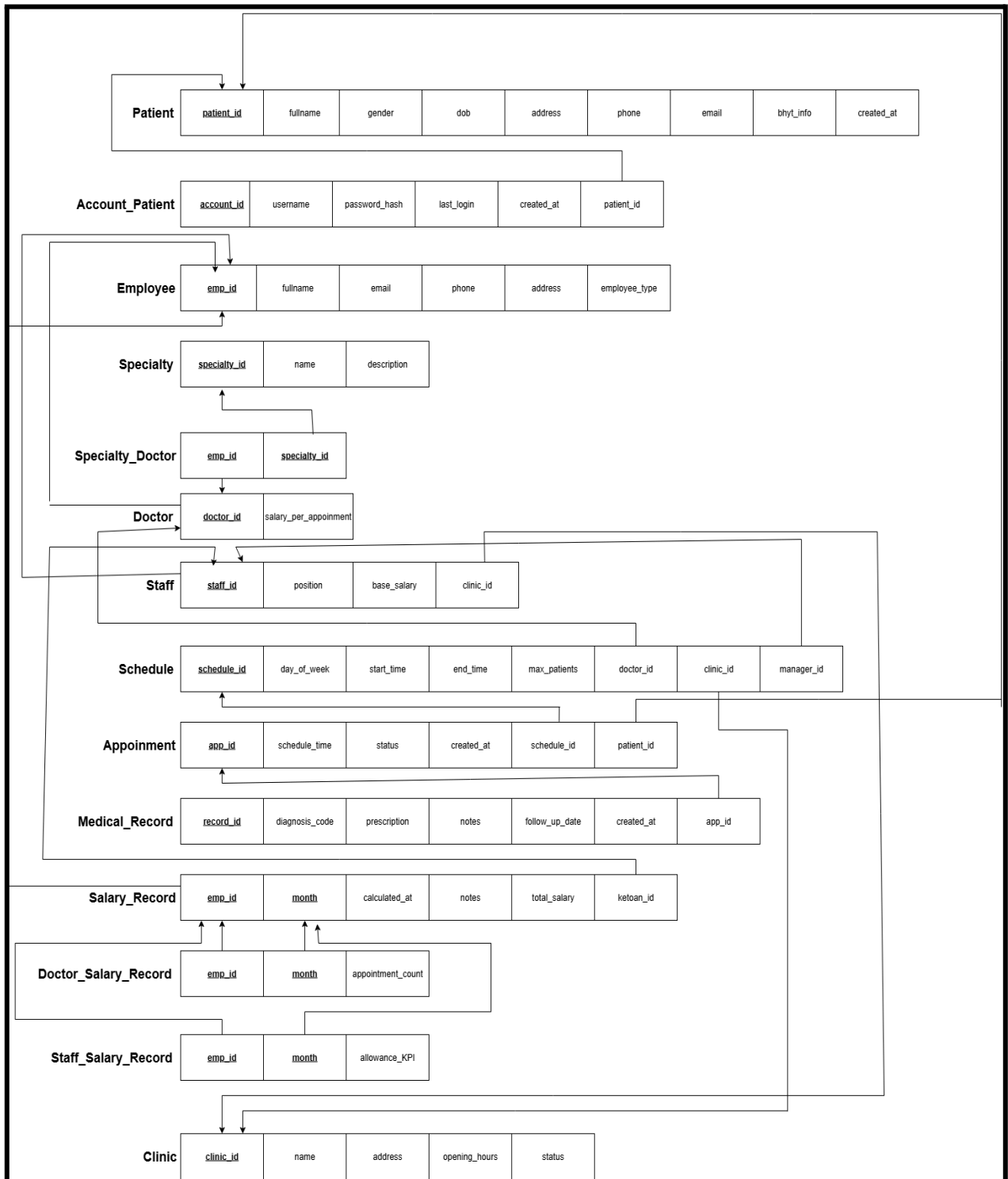
I. Relationship diagram.....	6
II. Create database.....	6
a. Tables.....	6
b. Stored procedures.....	6
c. Triggers.....	11
III. Analyse and implement performance improvement techniques	15
IV. Install the app.....	19

GROUP DETAILS INFORMATION TABLE

Group code:	21VP.ADB.03	
Number of members	2	
MSSV	Full name	Email
22127069	Nguyễn Đăng Hoàng Dinh	ndhdinh22@clc.fitus.edu.vn
22127107	Nguyễn Thế Hiển	nthien22@clc.fitus.edu.vn

Work assignment & completion evaluation table		
Work to be done	The performer	Level of completion
Create Database Analyze and Implement Performance Improvement Techniques Write Reports	22127069 - Nguyễn Đăng Hoàng Dinh	100%
Install the app Support Create Database Write a report	22127107 - Nguyễn Thế Hiển	100%

I. Relationship diagram



II. Create database

a. Tables

Create the tables designed in the relational schema above:

Create tables in turn:

- patients
- accounts
- specialties
- employees
- doctors
- doctor_specialties
- clinics
- staff
- schedules
- appointments
- medical_records
- salary_records
- doctor_salary_records
- staff_salary_records

b. Stored procedures

Stored Procedure: CalculateSalariesForMonth

Purpose:

This Stored Procedure calculates and stores monthly salaries for all employees in the system, including doctors, receptionists, managers, and others, based on the work data for the specified month.

Scripts:

```
CREATE PROCEDURE dbo.CalculateSalariesForMonth
    @Month CHAR(7), -- Format: 'YYYY-MM'
    @AccountantId INT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        BEGIN TRANSACTION;

        -- Kiểm tra accountant_id hợp lệ
        IF NOT EXISTS (
            SELECT 1 FROM staff
            WHERE staff_id = @AccountantId
```

```

        AND position = 'Accountant'
    )
BEGIN
    RAISERROR('Invalid accountant ID', 16, 1);
    ROLLBACK TRANSACTION;
    RETURN;
END

-- Tính lương cho từng employee
DECLARE @EmpId INT, @EmpType CHAR(1);
DECLARE emp_cursor CURSOR FOR
SELECT emp_id, employee_type FROM employees;

OPEN emp_cursor;
FETCH NEXT FROM emp_cursor INTO @EmpId, @EmpType;

WHILE @@FETCH_STATUS = 0
BEGIN
    DECLARE @TotalSalary DECIMAL(12, 2);

    IF @EmpType = 'D'
    BEGIN
        -- Tính Lương cho Bác sĩ
        DECLARE @AppointmentCount INT;
        DECLARE @SalaryPerAppointment DECIMAL(12, 2);

        SELECT @AppointmentCount = COUNT(*)
        FROM appointments a
        JOIN schedules s ON a.schedule_id = s.schedule_id
        WHERE s.doctor_id = @EmpId
        AND a.status = 'completed'
        AND CONVERT(CHAR(7), a.scheduled_time, 126) = @Month;

        SELECT @SalaryPerAppointment = salary_per_appointment
        FROM doctors
        WHERE doctor_id = @EmpId;

        SET @TotalSalary = @AppointmentCount *
@SalaryPerAppointment;
    
```

```

-- Insert/Update salary record
MERGE INTO salary_records AS target
USING (SELECT @EmpId AS emp_id, @Month AS month) AS
source
ON target.emp_id = source.emp_id AND target.month =
source.month
WHEN MATCHED THEN
    UPDATE SET total_salary = @TotalSalary,
               calculated_at = GETDATE(),
               accountant_id = @AccountantId
WHEN NOT MATCHED THEN
    INSERT (emp_id, month, total_salary,
accountant_id)
    VALUES (@EmpId, @Month, @TotalSalary,
@AccountantId);

-- Insert/Update doctor salary detail
MERGE INTO doctor_salary_records AS target
USING (SELECT @EmpId AS emp_id, @Month AS month) AS
source
ON target.emp_id = source.emp_id AND target.month =
source.month
WHEN MATCHED THEN
    UPDATE SET appointment_count = @AppointmentCount
WHEN NOT MATCHED THEN
    INSERT (emp_id, month, appointment_count)
    VALUES (@EmpId, @Month, @AppointmentCount);
END
ELSE IF @EmpType = 'S'
BEGIN
    -- Tính Lương cho Nhân viên
    DECLARE @BaseSalary DECIMAL(12, 2);
    DECLARE @AllowanceKPI DECIMAL(12, 2) = 500000; --
Example value

    SELECT @BaseSalary = base_salary
    FROM staff
    WHERE staff_id = @EmpId;

    SET @TotalSalary = @BaseSalary + @AllowanceKPI;

```



```

-- Insert/Update salary record
MERGE INTO salary_records AS target
USING (SELECT @EmpId AS emp_id, @Month AS month) AS
source
ON target.emp_id = source.emp_id AND target.month =
source.month
WHEN MATCHED THEN
    UPDATE SET total_salary = @TotalSalary,
               calculated_at = GETDATE(),
               accountant_id = @AccountantId
WHEN NOT MATCHED THEN
    INSERT (emp_id, month, total_salary,
accountant_id)
    VALUES (@EmpId, @Month, @TotalSalary,
@AccountantId);

-- Insert/Update staff salary detail
MERGE INTO staff_salary_records AS target
USING (SELECT @EmpId AS emp_id, @Month AS month) AS
source
ON target.emp_id = source.emp_id AND target.month =
source.month
WHEN MATCHED THEN
    UPDATE SET allowance_kpi = @AllowanceKPI
WHEN NOT MATCHED THEN
    INSERT (emp_id, month, allowance_kpi)
    VALUES (@EmpId, @Month, @AllowanceKPI);
END

FETCH NEXT FROM emp_cursor INTO @EmpId, @EmpType;
END

CLOSE emp_cursor;
DEALLOCATE emp_cursor;

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0

```

```

        ROLLBACK TRANSACTION;

    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
    DECLARE @ErrorState INT = ERROR_STATE();

    RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
END CATCH
END;
GO

```

Implementation process:

Calculate the salary for each employee

Procedure to review all employees and calculate salary in two types:

For doctors (employee_type = 'D'):

- **Calculation:** Number of completed appointments × Salary per appointment
- **Usage Data:**
 - Count the number of appointments with status = 'completed' in the month
 - Get the salary per appointment from the doctors' table
- **Formula:** total_salary = appointment_count × salary_per_appointment

For other employees (employee_type = 'S'):

- **Calculation method:** Basic salary + KPI allowance
- **Usage Data:**
 - Basic salary from the staff table
 - Default KPI allowance (eg, 500,000 VND)
- **Formula:** total_salary = base_salary + allowance_kpi

Save results:

Use the MERGE statement to:

- Update if there is already a salary record for that month
- Insert new if no record exists
- Save to both the main table (**salary_records**) and the detail table (**doctor_salary_records/staff_salary_records**)

c. Triggers

- **trg_appointments_validate_time**
 - **Purpose**

This trigger ensures data integrity and business compliance by checking that the appointment time falls within the doctor's time slot.
 - **Activation time:**
 - AFTER INSERT, UPDATE on the appointments table
 - Triggered after data change (new addition or update)
 - **Test conditions:**
 - **Check the day of the week:**
 - Flexible switching: Handles the issue of @@DATEFIRST settings being different between sessions
 - Standardize: Always ensure Monday = 1, Sunday = 7
 - Compare: The appointment date must match the working date in the schedule
 - **Check start time:**
 - Appointments cannot be scheduled before the doctor's office hours.
 - **Check end time:**
 - Appointments cannot be scheduled after closing time.
 - **Handling of violations:**

If any of these conditions are violated:

 - RAISERROR: Display clear error messages
 - ROLLBACK TRANSACTION: Undo all changes
 - RETURN: Stop execution immediately
 - **Professional meaning:**
 - Service quality assurance: Patients can schedule appointments only during business hours.
 - Optimize scheduling: Avoid doctors having to work overtime
 - Test Automation: No need for application-level validation

- **trg_appointments_check_max_patients**

- **Purpose:**

This trigger ensures that the maximum number of patients allowed in a doctor's working day is not exceeded, maintaining service quality and ensuring that doctors are not overloaded.

- **Activation time:**

- AFTER INSERT, UPDATE on the appointments table
 - Triggered after appointment data changes

- **Test logic:**

- Test scope:

```
SELECT COUNT(*) as total_count
FROM (
    -- Appointments đã có trong database
    SELECT a.app_id
    FROM appointments a
    WHERE a.schedule_id = i.schedule_id
    AND CAST(a.scheduled_time AS DATE) =
CAST(i.scheduled_time AS DATE)
    AND a.status NOT IN ('cancelled')
    AND a.app_id NOT IN (SELECT app_id FROM inserted)

    UNION ALL

    -- Appointments mới đang insert (trừ cancelled)
    SELECT i2.app_id
    FROM inserted i2
    WHERE i2.schedule_id = i.schedule_id
    AND CAST(i2.scheduled_time AS DATE) =
CAST(i.scheduled_time AS DATE)
    AND i2.status NOT IN ('cancelled')
) as combined
```

- **Count valid quantity:**

- Exclude cancelled appointments
 - Combine both existing and new appointments
 - Calculate by date and schedule

- **Compare with limit:**

- Compare total appointments with max_patients in the schedule
 - Indicates an error if the limit is exceeded

- **Handling of violations**

If the maximum quantity is exceeded:

- RAISERROR: Clear error messages
- ROLLBACK TRANSACTION: Undo all changes
- RETURN: Stop execution immediately
- **Business meaning:**
 - Ensuring the quality of examination: Doctors are not overloaded with work
 - Load Balancing: Reasonable distribution of patient numbers per day
 - Flexible: Allows cancellation without affecting limits
- **trg_appointments_check_patient_conflict**
 - **Purpose:**
This trigger ensures that each patient has only one active (uncompleted or uncanceled) appointment for the same doctor's schedule, avoiding duplicate bookings.
 - **Activation time:**
 - AFTER INSERT, UPDATE on the appointments table
 - Triggered after appointment data changes
 - **Test logic:**
 - **Test conditions:**
 - Only check appointments with active status ('booked', 'confirmed')
 - Ignore completed or canceled appointments
 - **Check for duplicates:**
 - Find appointments with the same patient and schedule
 - Has active status (not completed/cancelled)
 - Different app_id (not the record being processed)
 - **Xử lý khi vi phạm:**
If duplicates are found:
 - RAISERROR: Clear error messages to users
 - ROLLBACK TRANSACTION: Undo all changes
 - RETURN: Stop trigger execution immediately
 - **Ý nghĩa nghiệp vụ:**
 - Avoid duplication: Prevent multiple bookings for the same time slot.
 - Optimization: Ensuring fairness in appointment allocation.
 - Flexible: Allows booking a new schedule after completion or canceling the old schedule.

- **trg_appointments_create_medical_record**
 - **Purpose:**

This trigger automatically creates a medical record when an appointment is marked as complete, ensuring data consistency and supporting physician workflow.
 - **Activation time:**
 - AFTER UPDATE on the appointments table
 - Trigger only when the appointment status is updated
 - **Test logic:**
 - **Activation conditions:**
 - inserted: New status is 'completed'
 - deleted: Old status is NOT 'completed' (avoid multiple updates)
 - **Check for duplicates:**
 - Ensure no duplicate medical records are created for the same appointment.
 - Each appointment has only one medical record.
 - **Create default profile:**
 - app_id: Associated with completed appointment.
 - diagnosis_code: Default value 'UNKNOWN' (waiting for doctor to update).
 - created_at: Profile creation time.
 - **Professional meaning**
 - Automation: No need for manual profile creation
 - Completeness Guarantee: Every completed appointment has medical records
 - Save time: Reduce workload for doctors and staff

III. GROUP DETAILS INFORMATION TABLE

1. Doctor/staff view patient's record details

- Demographic information (patients).
- Appointments history (appointments)
- Medical records (medical_records) corresponding to “completed” appointments (if any).
- Related doctor and clinic information (doctors → employees, clinics).

Frequency: 10 - 20 times/patient/disease process

Implementation:

Enable performance statistics:

```
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
```

Execute the query:

```
DECLARE @PatientId INT = 155281;

SELECT p.patient_id, p.fullname, p.date_of_birth, p.gender, p.phone,
p.email,
      a.app_id, a.scheduled_time, a.status,
      mr.record_id, mr.diagnosis_code, mr.prescription,
mr.follow_up_date,
      e.emp_id AS doctor_emp_id, e.fullname AS doctor_name,
      c.clinic_id, c.name AS clinic_name
FROM patients p
LEFT JOIN appointments a ON a.patient_id = p.patient_id
LEFT JOIN medical_records mr ON mr.app_id = a.app_id
LEFT JOIN schedules s ON s.schedule_id = a.schedule_id
LEFT JOIN doctors d ON d.doctor_id = s.doctor_id
LEFT JOIN employees e ON e.emp_id = d.doctor_id
LEFT JOIN clinics c ON c.clinic_id = s.clinic_id
WHERE p.patient_id = @PatientId
ORDER BY a.scheduled_time DESC;
```

Result:

- **Tab Messages:**

```
SQL Server parse and compile time:
    CPU time = 15 ms, elapsed time = 22 ms.

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)
Table 'clinics'. Scan count 0, logical reads 2, physical reads 0
Table 'employees'. Scan count 0, logical reads 2, physical reads 0
Table 'doctors'. Scan count 0, logical reads 2, physical reads 0
Table 'schedules'. Scan count 0, logical reads 2, physical reads 0
Table 'medical_records'. Scan count 0, logical reads 4, physical reads 0
Table 'appointments'. Scan count 1, logical reads 37, physical reads 1
Table 'patients'. Scan count 0, logical reads 3, physical reads 0

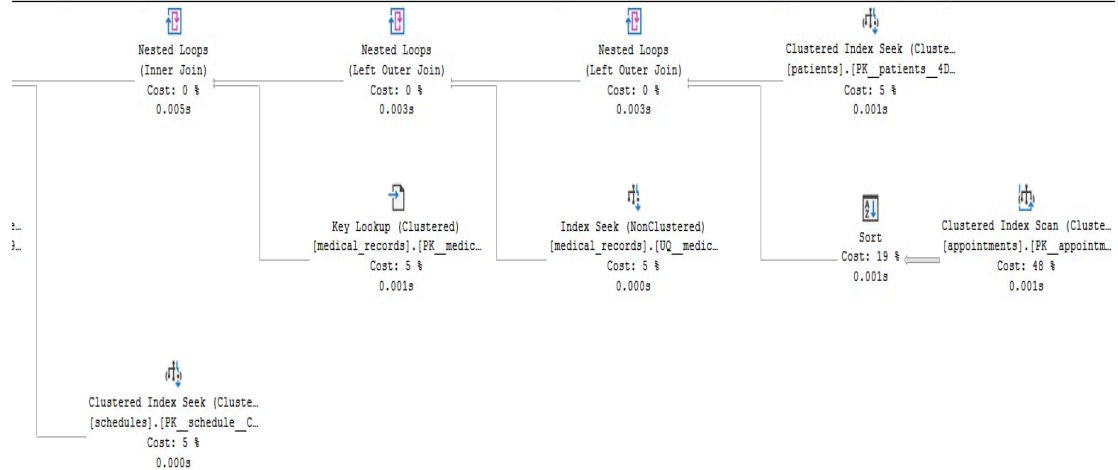
SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 4 ms.

Completion time: 2025-08-29T06:27:59.1258243+07:00
```

- **Actual Execution Plan:**

Query 1: Query cost (relative to the batch): 100%

```
SELECT p.patient_id, p.fullname, p.date_of_birth, p.gender, p.phone, p.email, a.app_id, a.scheduled_time, a.status, mr.record_id, mr.diagnosis_code, ...  
Missing Index (Impact 45.9365): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[appointments] ([patient_id])
```



Comment:

- SET STATISTICS IO/TIME shows appointments being scanned (Scan count 1, logical reads 37).
- Execution Plan shows Clustered Index Scan on appointments (accounting for the majority of the cost). There is Sort (ORDER BY) with a cost of ~19% and Nested Loops joins.
- Plan has a Missing Index warning: CREATE NONCLUSTERED INDEX ON [dbo].[appointments]([patient_id]) (impact 45.93) is recommended — meaning the optimizer thinks indexing on patient_id will improve significantly.
- There is also Key Lookup (Clustered) on medical_records — meaning the engine uses a nonclustered index on medical_records, but has to look up into the clustered to get the missing column.

Conclude:

The main bottleneck right now is that appointments are being scanned & sorted, and there are some lookups because existing indexes don't cover all the columns the query needs. Solution: create a proper covering index for appointments (to serve WHERE + ORDER BY) and create/tune an index for medical_records to avoid key lookups.

Improvement method:

- Create a covering nonclustered index on the appointments table:
- Reason:
 - Key (patient_id, scheduled_time DESC) allows index seek on patient and returns results in sorted order, helping to avoid Sort.

- INCLUDE the columns you return in SELECT (app_id, status, schedule_id) so the query doesn't have to look up into the clustered index anymore.

```
CREATE NONCLUSTERED INDEX IX_appointments_patient_scheduledtime
ON dbo.appointments (patient_id, scheduled_time DESC)
INCLUDE (app_id, status, schedule_id)
WITH (ONLINE = ON); -- nếu SQL Server edition hỗ trợ; nếu không, bỏ
phần WITH
GO
```

- Create a covering nonclustered index on the medical_records table (if there is not already a suitable index on app_id):
- Reason:
 - When medical_records is linked by mr.app_id = a.app_id, the index on app_id with diagnosis/prescription columns is INCLUDE to help the engine get medical record information directly from the index (avoiding Key Lookup into the clustered index medical_records).

```
CREATE NONCLUSTERED INDEX IX_medicalrecords_appid_covering
ON dbo.medical_records (app_id)
INCLUDE (record_id, diagnosis_code, prescription, follow_up_date)
WITH (ONLINE = ON);
GO
```

Results after improvement:

Tab Messages:

```
CPU time = 0 ms, elapsed time = 0 ms.

(1 row affected)
Table 'clinics'. Scan count 0, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'employees'. Scan count 0, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'doctors'. Scan count 0, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'schedules'. Scan count 0, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'medical_records'. Scan count 1, logical reads 2, physical reads 2, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'appointments'. Scan count 1, logical reads 2, physical reads 2, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0
Table 'patients'. Scan count 0, logical reads 3, physical reads 3, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0

(14 rows affected)

(1 row affected)

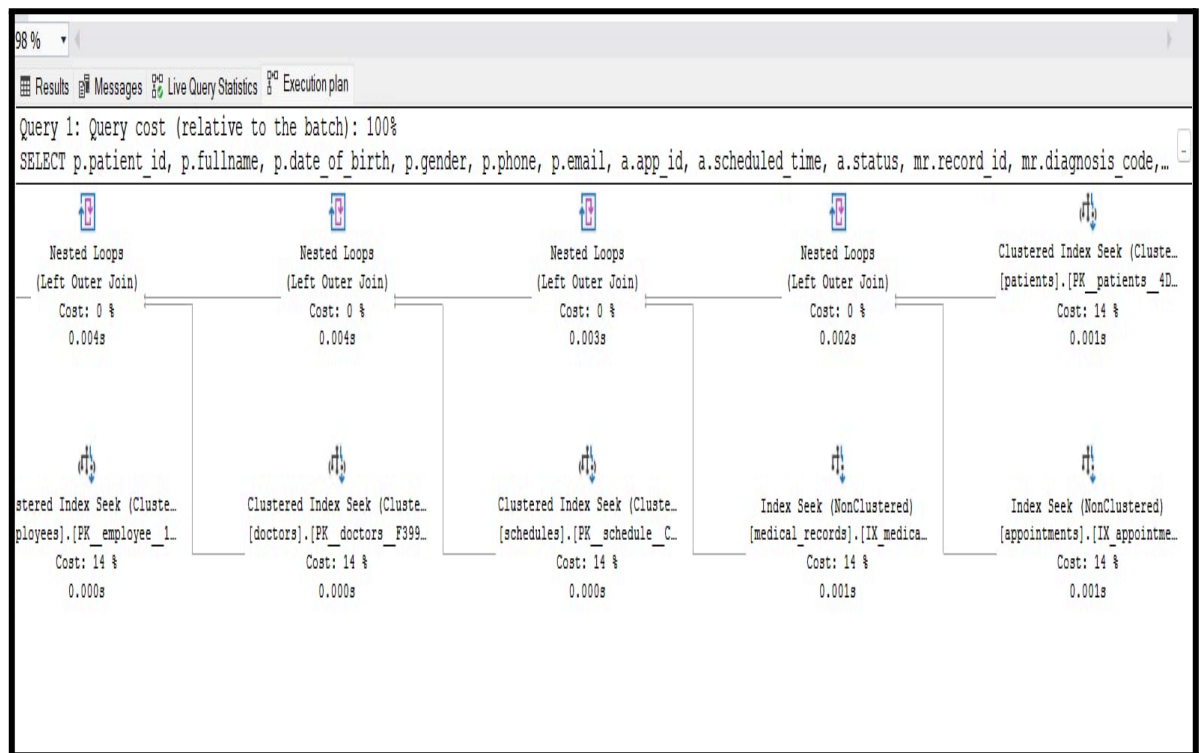
SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 119 ms.
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2025-08-23T07:11:53.7401431+07:00
```

Actual Execution Plan:



Comment on the results after installing the index:

- The execution plan shows **Index Seek** on appointments and also has **Index Seek/nonclustered** for medical_records — meaning SQL Server is using the new index instead of scanning.
- SET STATISTICS IO is very low (appointments logical reads = 2, medical_records logical reads = 2, patients = 3...) → I/O is significantly reduced.
- Elapsed time is very small (1 ms or 0 ms in the example) — fast response.

IV. Install the app

1. General introduction to the application

- The “**Clinic Management**” application is built to digitize the clinic management process, helping to optimize the management of doctors, patients, appointments, reports, and statistics. The system aims for a modern, friendly, easy-to-use interface for clinic staff and administrators.

2. System architecture & technology used (add details)

- The system is developed in a client-side (SPA) model using ReactJS. In this project, data is simulated (mockup data) instead of connecting directly to the real database. This helps

the development, testing, and demo process to be fast, suitable for a limited implementation time.

- Reasons for using mockup data:
 - Save time deploying backend and configuring the database.
 - Ensure that all business functions can be tested and demonstrated without depending on the real server.
 - Easily extendable, change the sample data to test different cases.
- If connecting to an actual Microsoft SQL Server:
 - The application will need to build an additional backend (API server) using Node.js, .NET Core, or another server-side technology.
 - Backend will use libraries like mssql (Node.js) or Entity Framework (C#) to connect and query data from SQL Server.
 - Frontend (ReactJS) will send requests (REST API or GraphQL) to the backend to get, add, edit, and delete actual data.
 - Need to configure the connection string, account, and SQL Server access permissions.

```
const sql = require('mssql');
const config = {
  user: 'sa',
  password: 'your_password',
  server: 'localhost',
  database: 'ClinicDB',
  options: { encrypt: true, trustServerCertificate: true }
};
sql.connect(config).then(pool => pool.request().query('SELECT * FROM Doctors'));
```

- In actual deployment, data will be stored, retrieved, and updated directly on SQL Server, ensuring consistency, security, and scalability.

3. System building process

- The development process includes the following steps:
 - Requirements analysis: Collect clinic management business requirements, identify main functions.
 - UI/UX design: Create wireframes, choose colors, layouts, and ensure a modern, responsive interface.
 - Build each module: Login, register, 2-layer authentication, manage doctors, patients, appointments, reports, and statistics.
 - Testing: Test each function, handle errors, optimize user experience.
 - Completion: Integrate modules, add advanced features, and complete the interface.

4. Detailed description of functions

- Login: Support 2-layer authentication (OTP/email code), clear error display, warning icon, loading effect, and show/hide password.
- Register: Check duplicate username/email, check password strength, show/hide password, loading, error/success notification.
- Doctor management: View list, advanced filtering (by specialty, status, KPI), view details, status history, rating, calendar schedule, analyze KPI with a chart.
- Patient management: View list, search, advanced filtering, and view detailed examination history.
- Appointment management: Create, edit, delete appointments, advanced filtering, export CSV.
- Reports & statistics: Quick statistics on the number of doctors, patients, appointments, KPIs, and visual charts.
- Notification (toast): Display a notification when the operation is successful/failed.
- Responsive interface: Compatible with computers and mobile devices.

5. User Interface (UI/UX)

- The interface is designed in a modern style, with harmonious colors and a clear layout. Components such as buttons, inputs, data tables, modals, and charts are all optimized for user experience. The system supports responsiveness, ensuring good usability on many devices..

6. Results achieved and evaluation

- The system fully meets the requirements of clinic management. Friendly interface, easy to use, good security with 2-layer authentication. Advanced functions such as filtering, statistics, charts, and toast notifications help improve management efficiency. The system is easy to expand, and can integrate additional modules such as drug management, payment, and hospital connection.

7. Illustration

a. Login Page

- Modern, friendly interface.
- Supports 2-layer authentication (OTP/email code) to increase security.
- Has the feature of showing/hiding password (eye icon).
- Displays incorrect input errors with warning icons and clear notifications.
- Loading effect when logging in or verifying OTP.



Clinic Management

Sign in to your account



☐ Remember me

[Forgot password?](#)

Sign In

Don't have an account? [Sign up](#)



Clinic Management

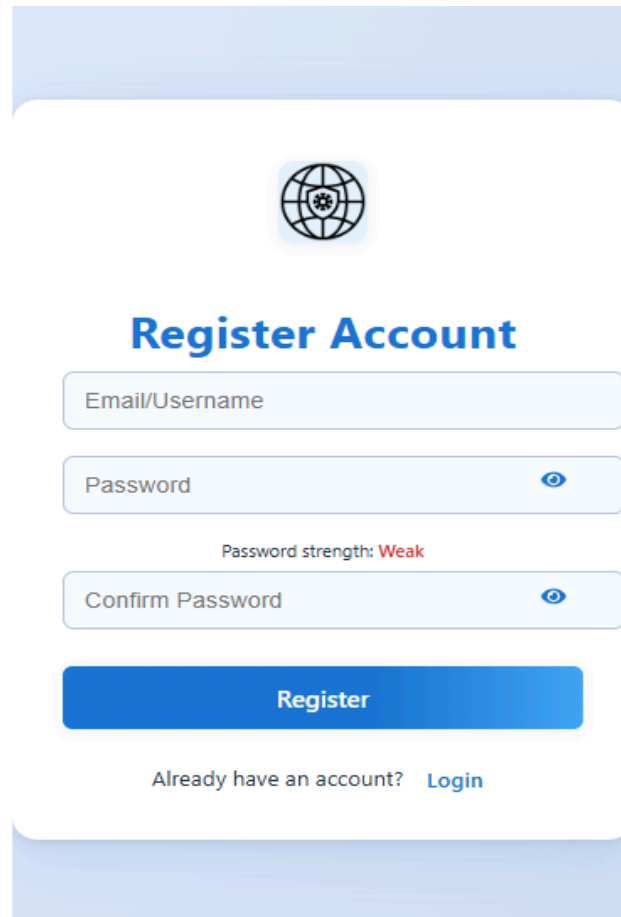
Sign in to your account

Verify OTP

Don't have an account? [Sign up](#)

b. Registration Page

- Check for duplicate email/username, report an error if it exists.
- Check password strength and display level (Weak/Medium/Strong).
- Has show/hide password and confirm password features.
- Loading effect when registering, successful or error notification.



The registration page features a light blue header and footer. At the top center is a circular logo with a globe and a gear. Below the logo is the title "Register Account" in bold blue text. The form consists of three input fields: "Email/Username", "Password", and "Confirm Password". The "Password" field has a strength indicator below it that reads "Password strength: Weak" in red. Each password field has a toggle icon (an eye) to the right. A blue "Register" button is positioned below the "Confirm Password" field. At the bottom, there is a link that says "Already have an account? Login".

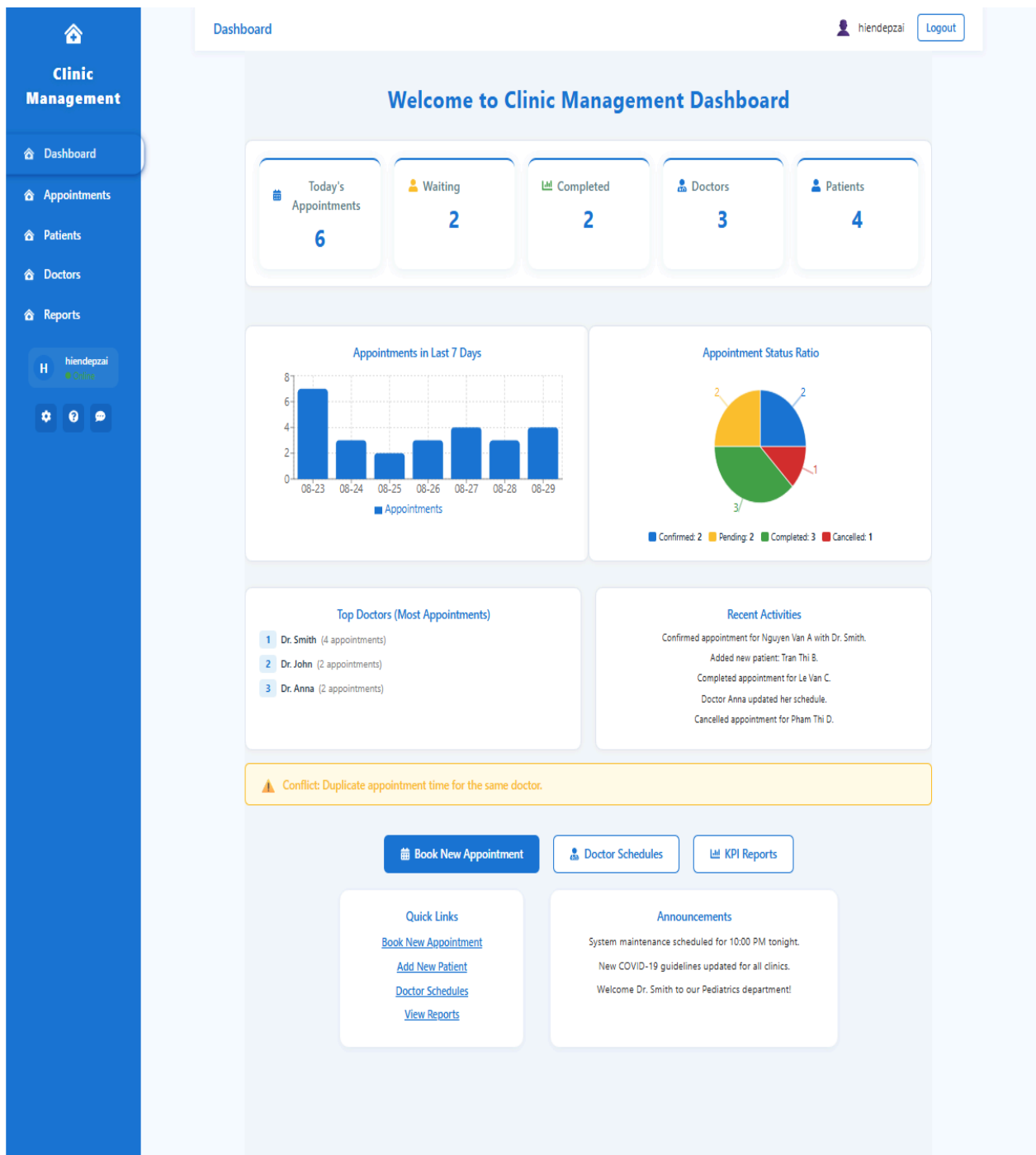
c. Dashboard overview

The Dashboard provides a clear and intuitive overview of clinic operations, designed for ease of use and quick access to key statistics:

- **Quick Statistics:** At the top, the dashboard displays the total number of doctors, patients, appointments, and their activity status. These figures are presented on distinct cards for immediate visibility.
- **Visual Insights:** The page features two side-by-side charts: "Appointments in Last 7 Days" (bar chart) and "Appointment Status Ratio" (pie chart), allowing users to quickly grasp trends and status distribution.
- **Top Doctors & Recent Activities:** Below the charts, the dashboard highlights the doctors with the most appointments and lists recent activities, keeping users informed of important updates

- **Alerts & Shortcuts:** Any conflicts or important notices are shown in a prominent alert box. Quick action buttons and links are provided for booking appointments, viewing schedules, and accessing reports.
- **Intuitive Layout:** The interface uses a clean, modern design with clear separation between sections, making information easy to follow and navigate.

Overall, the Dashboard page offers a user-friendly experience, enabling clinic staff to monitor operations efficiently and take action with minimal effort.



d. Doctor Management Page

The Doctors page provides a comprehensive and interactive interface for managing doctor profiles and performance:

- **Doctor List:** Displays all doctors with key information, including name, multi-specialty, status (active, on leave, inactive), KPI score, number of patients, and appointments. Each row includes quick actions for scheduling and viewing reviews.
- **Advanced Filters:** Users can filter the doctor list by specialty, status, KPI, number of patients, and appointments, making it easy to find specific profiles or analyze performance.
- **KPI Comparison Chart:** A bar chart at the top visually compares the monthly KPI scores of all doctors, helping staff quickly identify top performers and areas for improvement.
- **Doctor Details Modal:** Clicking on a doctor opens a detailed modal showing status history, ratings and reviews, work schedule in calendar format, and medical history. This allows for in-depth management and review of each doctor's activities.
- **Visual Analytics:** Additional charts display specialty ratio and status ratio, providing a quick overview of the distribution of specialties and current doctor statuses.
- **Toast Notifications:** The system provides instant feedback via toast notifications for successful or failed actions, ensuring users are always informed about the result of their operations.

Overall, the Doctors page offers powerful tools for clinic administrators to manage, analyze, and optimize doctor performance and scheduling, all within an intuitive and user-friendly interface.

Clinic Management

Dashboard

Appointments

Patients

Doctors

Reports

Hhiendepzai

▼Close

⚙️

🔔

💬

Dashboard

hiendepzaiLogout

KPI Comparison (This Month)

12

Smith

8

John

1

Anna

0

Lee

5

Mai

Doctors

Total: 5

Active: 3

On Leave: 1

Inactive: 1

Search doctor name...

All Specialties

All Status

Patients ≥...

Appointments ≥...

KPI ≥...

☐ Show inactive

Add Doctor

ID	Name	Specialty	Status	Patients	Appointments	KPI	Rating	Review	Actions
1	<div>D</div> Dr. Smith	Cardiology, Internal Medicine	Active	2	2	12	4.8	<div>★</div> Reviews	<div>📄</div> <div>👤</div> <div>Schedule</div> <div>🔔</div>
2	<div>D</div> Dr. John	Pediatrics	Active	1	1	8	4.5	<div>★</div> Reviews	<div>📄</div> <div>👤</div> <div>Schedule</div> <div>🔔</div>
3	<div>D</div> Dr. Anna	Dermatology	On Leave	1	1	1	4.2	<div>★</div> Reviews	<div>📄</div> <div>👤</div> <div>Schedule</div> <div>🔔</div>
5	<div>D</div> Dr. Mai	Neurology, Internal Medicine	Active	0	0	5	4.7	<div>★</div> Reviews	<div>📄</div> <div>👤</div> <div>Schedule</div> <div>🔔</div>

Doctor Overview (Radar Chart)

Smith

John

Anna

Mai

Specialty Ratio

● Cardiology: 1

● Internal Medicine: 2

● Pediatrics: 1

● Dermatology: 1

● Neurology: 1

Status Ratio

● Active: 3

● On Leave: 1

● Inactive: 0

Doctor Management

- Centralized management of doctor profiles and schedules.
- View appointment and patient history for each doctor.
- Manage and assign work shifts (schedule) for each doctor.
- Filter by specialty, status, and KPI.
- Data privacy: Only authorized staff can view/edit doctor info.

Quick Actions

- Search for doctors by name or specialty.
- View doctor details, appointment history, and work schedule.
- See number of patients, appointments, and KPI.

e. Patient Management

The Patients page offers a streamlined and centralized interface for managing patient records and activity:

- **Quick Statistics & Visuals:** At the top, the page displays total patients, chronic cases, and new patients for the month, along with a pie chart and a bar chart showing patient growth by month. This provides instant insight into patient demographics and trends.
- **New Patients Highlight:** A prominent card shows the number of new patients added this month, helping staff track recent registrations.
- **Advanced Filtering:** Users can filter the patient list by name, email, phone, ID, last visit date, doctor, visit count, and chronic status. This makes it easy to locate specific records or analyze patient groups.
- **Patient List & Actions:** The table lists all patients with key details (name, email, phone, visits, last visit). Quick action buttons allow users to view, download PDFs, edit, or delete records efficiently.
- **Patient Management Features:** The page supports centralized management of medical records, viewing visit history and schedules, and downloading medical documents. Data privacy is maintained, with access restricted to authorized staff.
- **Quick Actions:** Users can search for patients, view medical records, and see visit counts and last visit dates with just a few clicks.

Overall, the Patients page is designed for clarity and efficiency, enabling clinic staff to manage patient information, track activity, and perform essential actions with ease.

🏠

Clinic Management

🏠 Dashboard

🏠 Appointments

🏠 Patients

🏠 Doctors

🏠 Reports

H

hiendepzai

👤 Clinic

⚙️

🔒

💬

Dashboard

hiendepzai

Logout

Total

8

Chronic

3

New

2

Chronic

New

Other

Patient Growth by Month

8

6

4

2

0

2025-04

2025-06

2025-08

👤 New Patients This Month

2

Filters

No filters applied, showing all patients.

Name...

Email...

Phone...

ID...

dd/mm/yyyy

Visits ≥...

All Doctors

☐ Chronic only

Add Patient

ID	Name	Email	Phone	Visits	Last Visit	Actions
1	Nguyen Van A	vana@gmail.com	0901234567	3	2025-05-15	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>
2	Tran Thi B	thib@gmail.com	0902345678	3	2025-07-10	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>
3	Le Van C	vanc@gmail.com	0903456789	3	2025-06-30	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>
4	Pham Thi D	thid@gmail.com	0904567890	3	2025-05-22	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>
5	Hoang Minh G	minhg@gmail.com	0905678912	3	2025-07-11	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>
6	Nguyen Thi H	thih@gmail.com	0906789123	2	2025-06-20	<div>View</div> <div>PDF</div> <div>Edit</div> <div>Delete</div>

Patient Management

- Centralized management of patient medical records by patient code.
- View visit history, doctor, and re-exam schedule.
- Download PDF copy of medical record (if available).
- Data privacy: Only viewable by the patient (HIPAA-like).

Quick Actions

- Search for patients by name.
- View and download medical records.
- See visit count and last visit date.

f. Appointments Management

The Appointments page provides a comprehensive and user-friendly interface for managing clinic appointments:

- **Visual Statistics:** At the top, two charts display appointment data: a stacked bar chart shows appointment counts by status for each day, and a pie chart summarizes the ratio of appointment statuses (confirmed, pending, completed, canceled).
- **Appointment List & Actions:** The table lists all appointments with details such as patient name, doctor, date, time, and status. Action buttons allow users to view details, check in, or confirm appointments directly from the list.
- **Quick Booking & Search:** A prominent button enables fast booking of new appointments. Advanced search and filter options (by doctor, status, date) help users quickly find and manage appointments.
- **Appointments Calendar:** The integrated calendar provides a clear overview of scheduled appointments by day, week, or month. Users can click on dates to view appointment details and manage schedules efficiently.
- **Quick Actions & Tips:** Helpful tips and quick actions are provided at the bottom, guiding users to book, filter, and manage appointments, as well as keep patient and doctor information up to date.

Overall, the Appointments page is designed for clarity and efficiency, enabling clinic staff to manage, track, and update appointments with ease, while providing visual insights and actionable tools for daily operations.

g. Clinic Reports

The Clinic Reports page provides a comprehensive overview of clinic performance and activity, supporting data-driven decision making:

- **Export & Filtering:** Users can export report data to CSV and filter results by date range, doctor, service, and status, enabling flexible analysis and record-keeping.
- **Summary Section:** Key metrics are displayed at a glance, including total patients, total appointments, and total revenue, helping staff quickly assess clinic performance.
- **Doctor KPI Table:** A dedicated table shows each doctor's specialty, number of appointments, revenue generated, and average rating, allowing for easy comparison of individual performance.
- **Visual Analytics:** Two charts present monthly trends: "Appointments by Month" (bar chart) and "Revenue by Month" (line chart), giving users insight into clinic growth and financial performance over time.
- **Appointments Table:** A detailed table lists all appointments with date, patient, doctor, service, fee, and status, supporting thorough review and auditing of clinic activity.

Overall, the Clinic Reports page is designed for clarity and utility, enabling clinic administrators to monitor key metrics, analyze trends, and export data for further use, all within an intuitive and organized interface.

Clinic Management

Dashboard

Appointments

Patients

Doctors

Reports

Hhiendepzai

Online

Dashboard

hiendepzai

Logout

Clinic Reports

Export CSV

From: dd/mm/yyyy

To: dd/mm/yyyy

All Doctors

All Services

All Status

Summary

Total patients: 14

Total appointments: 14

Total revenue: 2.650.000 đ

Doctor KPI

Name	Specialty	Appointments	Revenue	Rating
Dr. Smith	Cardiology	6	1.210.000 đ	4.8
Dr. John	Pediatrics	4	640.000 đ	4.5
Dr. Anna	Dermatology	4	800.000 đ	4.7

Appointments by Month

Month	Appointments
2025-04	1
2025-05	3
2025-06	2
2025-07	4
2025-08	4

Revenue by Month

Month	Revenue (đ)
2025-04	200.000
2025-05	590.000
2025-06	360.000
2025-07	770.000
2025-08	730.000

Appointments Table

Date	Patient	Doctor	Service	Fee	Status
2025-08-29	Nguyen Van A	Dr. Smith	General Checkup	200.000 đ	Confirmed
2025-08-29	Tran Thi B	Dr. John	Eye Exam	150.000 đ	Pending
2025-08-30	Le Van C	Dr. Smith	General Checkup	200.000 đ	Completed
2025-08-30	Pham Thi D	Dr. Anna	Dermatology	180.000 đ	Cancelled
2025-04-12	Pham Van K	Dr. Smith	General Checkup	200.000 đ	Completed
2025-05-18	Tran Van I	Dr. John	ENT Exam	170.000 đ	Completed
2025-05-25	Le Thi J	Dr. Anna	Cardiology	220.000 đ	Completed
2025-05-05	Nguyen Van M	Dr. Smith	General Checkup	200.000 đ	Completed
2025-06-20	Tran Thi F	Dr. John	Eye Exam	150.000 đ	Completed
2025-06-10	Pham Thi H	Dr. Smith	General Checkup	210.000 đ	Completed
2025-07-15	Nguyen Van E	Dr. Smith	General Checkup	200.000 đ	Completed
2025-07-22	Le Van G	Dr. Anna	Dermatology	180.000 đ	Completed
2025-07-03	Tran Van N	Dr. John	ENT Exam	170.000 đ	Completed
2025-07-28	Le Thi O	Dr. Anna	Cardiology	220.000 đ	Completed

