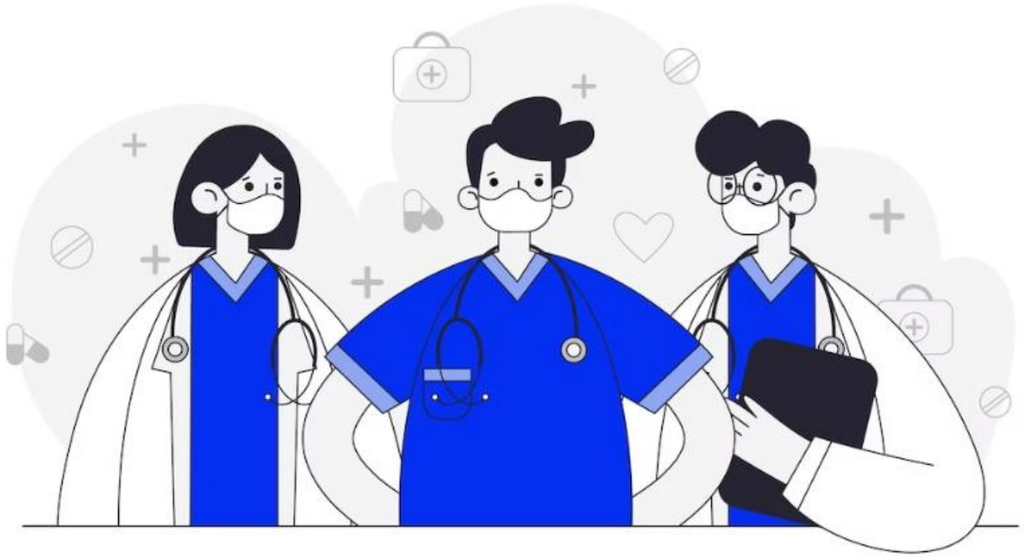
INT 6113 DATABASE MANAGEMENT SYSTEMS



Final Project: Hospital Patient Management System (HPMS)

- **By :** Tej Acharya
- **Date:** 4/30/2025
- **Project Overview:** A relational database application designed to manage core hospital operations (patients, appointments, visits, treatments, billing, staff). The system ensures efficient recording of hospital data with integrity and accessibility.

Agenda

1. Project Overview & Scenario
2. Database Design
3. Sample Data & SQL Functionality
4. System Interface Walkthrough
5. Advanced Features & Enhancements
6. Team Contributions
7. Q&A Session

PROJECT OVERVIEW & SCENARIO

The HPMS

The HPMS addresses the needs of a mid-sized hospital to track patient information, appointments, medical visits, and billing in one integrated system.

Scenario

Hospital staff often struggle with disjointed systems for scheduling, patient records, and billing. Our solution provides a unified platform to handle patient registrations, appointment booking, logging of visits and treatments, and generating bills.

Solution Scope

HPMS enables staff to manage patient data, schedule and record appointments/visits, assign treatments and medications, and process billing all within a single database-driven application. This improves data consistency and operational efficiency.

SYSTEM FEATURES

Secure Staff Login

Authentication for hospital staff with role-based access and session handling.

Patient Record Management

Add, update, and search patient demographic details easily.

Appointment Scheduling

Book future appointments linking patients with doctors and track their status (Scheduled/Completed/Cancelled).

Visit & Treatment Logging

Record each patient visit, including responsible doctor, room, treatments given, and any medications prescribed.

Billing & Payment Tracking

Automatically generate a billing record for each visit, recording total cost, payment status, and method (cash, card, insurance).

Room & Doctor Management

Keep an inventory of hospital rooms (ICU, Surgery, etc.) and their assigned doctors; manage doctor profiles and specializations.

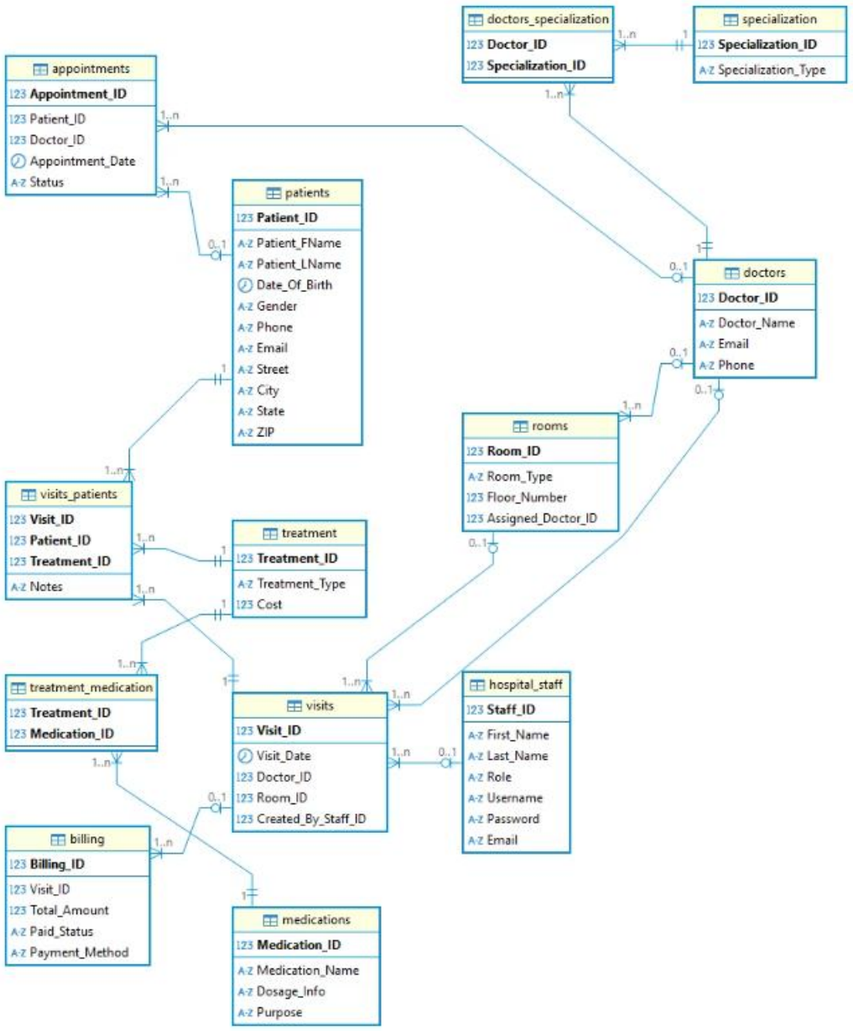
Analytics Dashboard

Provide at-a-glance metrics (total patients, total doctors, today's appointments) and quick links to key functions.

Live Search

Instant search functionality in UI for patients, visits, appointments to quickly retrieve records.

ERD Diagram & Key Design



1. **Entity-Relationship Diagram (ERD):** *(Normalized schema diagram illustrating all 13 tables and their relationships – PKs and FKs – is included on this slide.)*
2. **3NF Normalization:** All tables are structured in Third Normal Form – each non-key field depends only on the primary key, eliminating redundancy. This ensures data integrity and reduces duplicate data.
3. **Many-to-Many Handling:** Used junction tables to break down M:N relationships. For example, doctors_specialization links doctors ↔ specializations, visits_patients links visits ↔ patients (and treatments), and treatment_medication links treatments ↔ medications. This avoids data duplication while preserving complex relationships.
4. **Audit & Accountability:** The design includes fields like Created_By_Staff_ID in visits to log which staff member created a record. This provides an audit trail for actions (who recorded a visit) and enhances accountability.
5. **Modular Design:** Each entity (patients, doctors, visits, treatments, etc.) is in a separate table, making the system modular. New requirements (e.g., adding more details to treatments or a new entity) can be accommodated without impacting unrelated tables.

TABLE STRUCTURE & RELATIONSHIPS (1/3)

Patients: Patient_ID (PK)

stores patient demographics (first name, last name, DOB, gender, contact info, address).

– Relationships: One-to-many with Appointments (a patient can have many appointments). Many-to-many with Visits via the visits_patients junction (a patient can participate in many visits; each visit can include multiple patients).

Doctors: Doctor_ID (PK)

stores doctor's name and contact info.

– Relationships: Many-to-many with Specialization via doctors_specialization (a doctor can have multiple specialties; a specialty can have many doctors). One-to-many with Appointments (doctor has many appointments) and one-to-many with Visits (doctor performs many visits).

Specialization: Specialization_ID (PK)

contains specialty types (Cardiology, Pediatrics, etc.).

– Relationships: Linked to Doctors through doctors_specialization (no direct many-to-many in main tables, handled via junction table).

Doctors_Specialization: (junction table)

Composite PK (Doctor_ID, Specialization_ID) (both also FKs). Each record links one doctor to one specialization.

– This implements the many-to-many relationship between Doctors and Specializations without duplication – e.g., if a doctor has 2 specialties, there are 2 rows for that doctor in this table.

TABLE STRUCTURE & RELATIONSHIPS (2/3)

Appointments: Appointment_ID (PK); fields: Patient_ID (FK), Doctor_ID (FK), Appointment_Date, Status.

Represents a scheduled meeting between a patient and a doctor at a future date. Relationships: Each appointment links a Patient and a Doctor. Status tracks if it's Scheduled, Completed, or Cancelled. (When an appointment occurs, it usually results in a Visit record.)

Visits: Visit_ID (PK); fields: Visit_Date, Doctor_ID (FK), Room_ID (FK), Created_By_Staff_ID (FK).

Logs an actual patient visit on a date. Relationships: Each visit is performed by one Doctor and happens in one Room. It may involve multiple patients (via visits_patients). Created_By_Staff_ID links to Hospital_Staff (who entered the visit). Each visit has one corresponding Billing record.

Visits_Patients: (junction table) Composite PK (Visit_ID, Patient_ID, Treatment_ID) (all FKs).

Links which Patient(s) attended a visit and which Treatment(s) they received during that visit. Also includes a Notes field for any medical notes on that patient's treatment during the visit. (This table can have multiple entries per visit if multiple patients and/or multiple treatments are involved.)

Treatment: Treatment_ID (PK); fields: Treatment_Type (description of procedure), Cost.

Catalog of medical procedures or services (e.g., X-Ray, MRI, Physical Therapy). Relationships: Many-to-many with Medications via treatment_medication (a treatment can involve multiple meds; a medication can be used in multiple treatments). Linked into Visits through visits_patients (which specifies which treatments were given in a visit).

TABLE STRUCTURE & RELATIONSHIPS (3/3)

Medications: Medication_ID (PK); fields: Medication_Name, Dosage_Info, Purpose.

List of drugs that might be prescribed or administered. (E.g., Amoxicillin, 500mg, antibiotic.) Relationships: Many-to-many with Treatment via treatment_medication (which treatments include this medication).

Treatment_Medication: (junction table) Composite PK (Treatment_ID, Medication_ID) (both FKs).

Each record links one Treatment to one Medication involved in that treatment. This allows listing all meds used for a given treatment. (E.g., Treatment "Chemotherapy" might link to multiple medication IDs for the chemo drugs.)

Rooms: Room_ID (PK); fields: Room_Type (e.g., ICU, General, Surgery), Floor_Number, Assigned_Doctor_ID (optional FK).

Represents hospital rooms/wards. Relationships: One-to-many with Visits (a room hosts many visits). Assigned_Doctor_ID optionally links a doctor in charge of that room (e.g., a ward's primary doctor).

Hospital_Staff: Staff_ID (PK); fields: First_Name, Last_Name, Role (Admin/Nurse/Technician), Username, Password, Email.

Users of the system (hospital employees) with login credentials. Relationships: Referenced by Visits (Created_By_Staff_ID) to log who created each visit entry. Also used for system login/authentication but not directly linked to patient records otherwise.

Billing: Billing_ID (PK); fields: Visit_ID (FK), Total_Amount, Paid_Status, Payment_Method.

Financial record for a visit. Relationships: One-to-one with Visits (each visit has one bill). The billing entry captures the total charges for that visit and whether it's Paid, Pending, or Unpaid, along with how the patient paid (cash, credit, insurance, etc.).

Sample Data Evidence

- **Realistic Data Population:** The database is pre-populated with sample records to simulate hospital operations. We input ~100 patients, 35 doctors, 20 specializations, 100 visits, and associated appointments, treatments, etc. This volume demonstrates the system's ability to handle real-world usage.
- **Patients & Doctors:** Patients come from multiple cities/states with varied demographics. Doctors cover all 20 specialization areas (some doctors have 2 specialties). For example, there are 101 patients in the system (Patient IDs 1–100 from initial load, plus one added during testing) and 35 doctors (Doctor IDs 1–35).
- **Linked Records:** The data was generated to respect relationships – e.g. many patients have appointments and subsequent visits. Each doctor has several appointments and visits. Most patients have at least one visit, while a few (roughly 15–20) have none yet (e.g., new patients who haven't come in).
- **Example:** A patient *"Tyler Scott"* was added and then scheduled for an appointment with *Dr. Emily Walker* on 04/30/2025. When the visit occurred, it was logged with a visit record, treatments (e.g. Blood Test) were recorded in visits_patients, and a billing entry was generated. This end-to-end data flow is present for numerous scenarios in the sample dataset.
- **Data Integrity:** We verified that foreign key links are consistent (every Appointment has a valid Patient and Doctor, every Visit has a valid Doctor/Room/Staff, etc.). The presence of realistic, interlinked data provides evidence that the schema works for the intended use cases.

SQL Query Demonstration (1/3)

Query 1: Upcoming Appointments (Next 7 Days)

Uses JOINs to combine patient and doctor info with appointments, and a date filter.

```sql
SELECT p.Patient_FName, p.Patient_LName, d.Doctor_Name, a.Appointment_Date
FROM patients p
JOIN appointments a ON p.Patient_ID = a.Patient_ID
JOIN doctors d ON a.Doctor_ID = d.Doctor_ID
WHERE a.Status = 'Scheduled'
  AND a.Appointment_Date >= CURDATE()
ORDER BY a.Appointment_Date;
```

patients(+) 1

SELECT p.Patient_FName, p.Patient_LName, d.Doctor_Name, a.Appointment | Enter a SQL expression to filter results

	Patient_FName	Patient_LName	Doctor_Name	Appointment_Date
1	Charlotte	Patel	Dr. Ravi Lopez	2025-05-04 15:12:35
2	Olivia	Thomas	Dr. Priya Smith	2025-05-06 15:12:35
3	Ethan	Thomas	Dr. Daniel Garcia	2025-05-11 15:12:35
4	Olivia	Davis	Dr. David Smith	2025-05-12 15:12:35
5	Sophia	Martinez	Dr. John Davis	2025-05-13 15:12:35
6	Ava	Thomas	Dr. Daniel Anderson	2025-05-15 15:12:35
7	Charlotte	Thompson	Dr. Daniel Garcia	2025-05-15 15:12:35
8	Emma	Wilson	Dr. Priya Johnson	2025-05-17 15:12:35
9	Charlotte	Garcia	Dr. Linda Rodriguez	2025-05-18 15:12:35
10	Michael	Wilson	Dr. Ravi Lopez	2025-05-19 15:12:35
11	Michael	Garcia	Dr. William Singh	2025-05-19 15:12:35

Query 2: Top 5 Doctors by Number of Visits

Uses JOIN and GROUP BY to count visits per doctor.

```sql
SELECT d.Doctor_Name, COUNT(v.Visit_ID) AS VisitCount
FROM doctors d
JOIN visits v ON d.Doctor_ID = v.Doctor_ID
GROUP BY d.Doctor_ID
ORDER BY VisitCount DESC
LIMIT 5;
```

doctors 1

SELECT d.Doctor_Name, COUNT(v.Visit_ID) AS VisitCount FROM doctors d JC

	Doctor_Name	VisitCount
1	Dr. Jose Patel	6
2	Dr. Priya Rodriguez	5
3	Dr. David Lewis	5
4	Dr. Steven Davis	5
5	Dr. Nancy Martinez	5

SQL Query Demonstration (2/3)

Query 3: Patients with No Recorded Visits

Uses a subquery to find entities lacking related records.

```sql
SELECT Patient_FName, Patient_LName
FROM patients
WHERE Patient_ID NOT IN (
    SELECT DISTINCT Patient_ID
    FROM visits_patients
);
```

patients 1

SELECT Patient_FName, Patient_LName FROM patients WHERE

	Patient_FName	Patient_LName
1	James	Taylor
2	Noah	Brown
3	David	Patel
4	James	Miller
5	Michael	Wilson
6	Ethan	Perez
7	Olivia	Smith
8	Olivia	Martinez
9	Michael	Smith
10	Olivia	Thomas
11	David	Garcia
12	Tyler	Scott

Query 4: Doctors with Multiple Specializations

Uses GROUP BY and HAVING to find doctors linked to more than one specialty.

```sql
SELECT d.Doctor_Name, COUNT(ds.Specialization_ID) AS NumSpecialties
FROM doctors d
JOIN doctors_specialization ds ON d.Doctor_ID = ds.Doctor_ID
GROUP BY d.Doctor_ID
HAVING COUNT(ds.Specialization_ID) > 1;
```

doctors 1

SELECT d.Doctor_Name, COUNT(ds.Specialization_ID) AS NumSpecialties FR

	Doctor_Name	NumSpecialties
1	Dr. Daniel Anderson	3
2	Dr. Priya Rodriguez	2
3	Dr. Emily Walker	2
4	Dr. Angela Johnson	2
5	Dr. Ravi Lopez	2
6	Dr. David Garcia	3
7	Dr. Linda Rodriguez	2
8	Dr. Daniel Garcia	2
9	Dr. John Davis	3
10	Dr. Jose Walker	2
11	Dr. Priya Johnson	2
12	Dr. William Lee	3
13	Dr. Sarah Patel	2
14	Dr. David Lewis	3
15	Dr. Nancy Thomas	3
16	Dr. Emily Nguyen	2
17	Dr. Carlos Thomas	3
18	Dr. Jose Nguyen	3
19	Dr. William Singh	3
20	Dr. John Martinez	3
21	Dr. Mohamed Garcia	2
22	Dr. John Smith	3
23	Dr. Priya Smith	3
24	Dr. William Thomas	3
25	Dr. Angela Garcia	3
26	Dr. Fatima Patel	3

SQL Query Demonstration (3/3)

Query 5: Billing Summary by Payment Status

Uses aggregation (COUNT, SUM) and GROUP BY.

```sql
SELECT Paid_Status, COUNT(*) AS NumBills, SUM(Total_Amount) AS TotalAmount
FROM billing
GROUP BY Paid_Status;
```

billing 1

SELECT Paid_Status, COUNT(*) AS NumBills, SUM(Total_Amount) AS TotalAr | Enter a SQL expression to

	Paid_Status	NumBills	TotalAmount
1	Paid	34	176,937.96
2	Pending	33	167,734.68
3	Unpaid	33	150,731.46

Query 6: Most Common Specialization (Doctors)

Who has the most doctors? Uses JOIN, GROUP BY, ORDER, LIMIT.

```sql
SELECT s.Specialization_Type, COUNT(ds.Doctor_ID) AS NumDoctors
FROM specialization s
JOIN doctors_specialization ds ON s.Specialization_ID = ds.Specialization_ID
GROUP BY s.Specialization_ID
ORDER BY NumDoctors DESC
LIMIT 1;
```

specialization 1

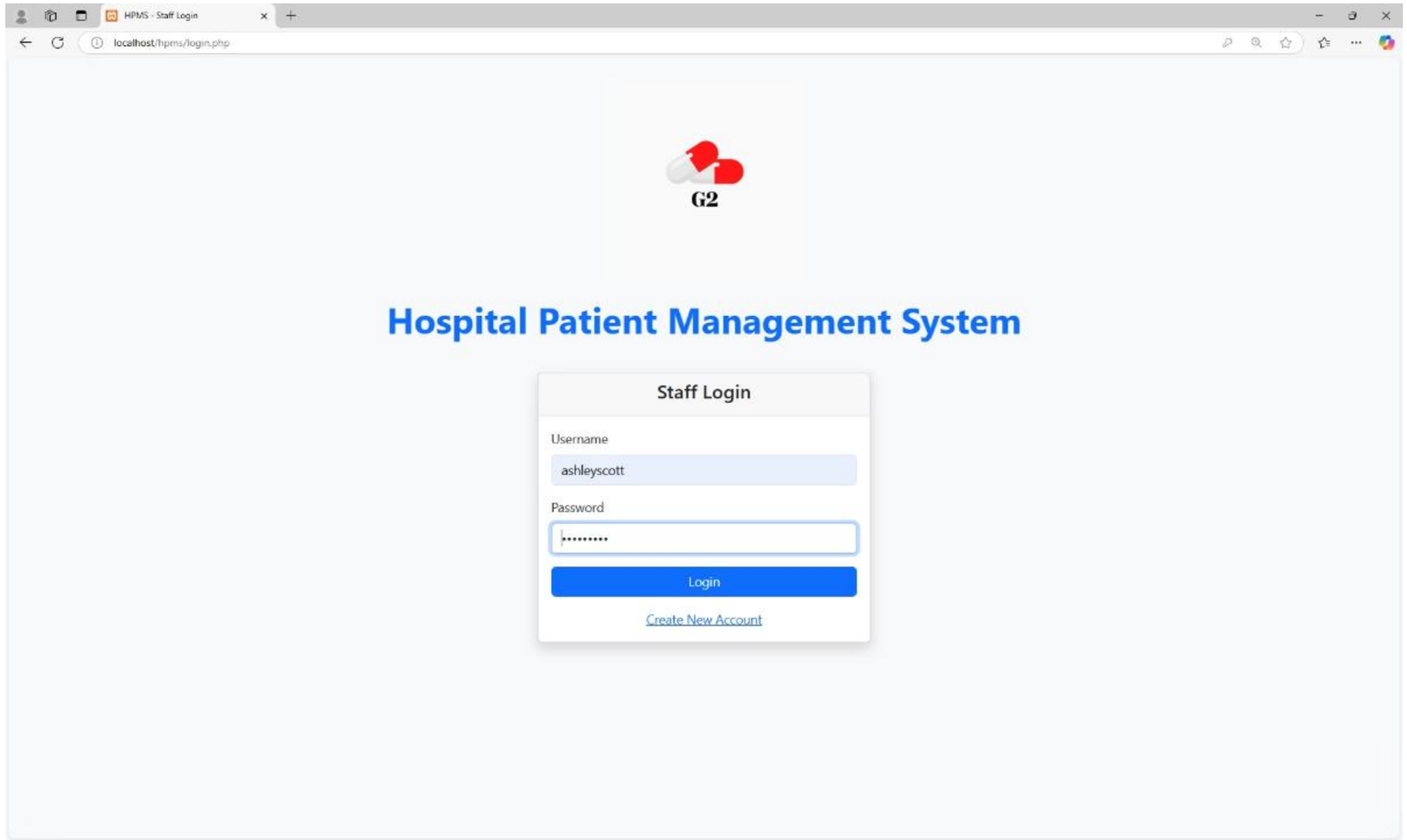
SELECT s.Specialization_Type, COUNT(ds.Doctor_ID) AS NumDoctors FROM | Enter a SQL expression to filt

	Specialization_Type	NumDoctors
1	Rheumatology	8

SQL CONCEPTS COVERED

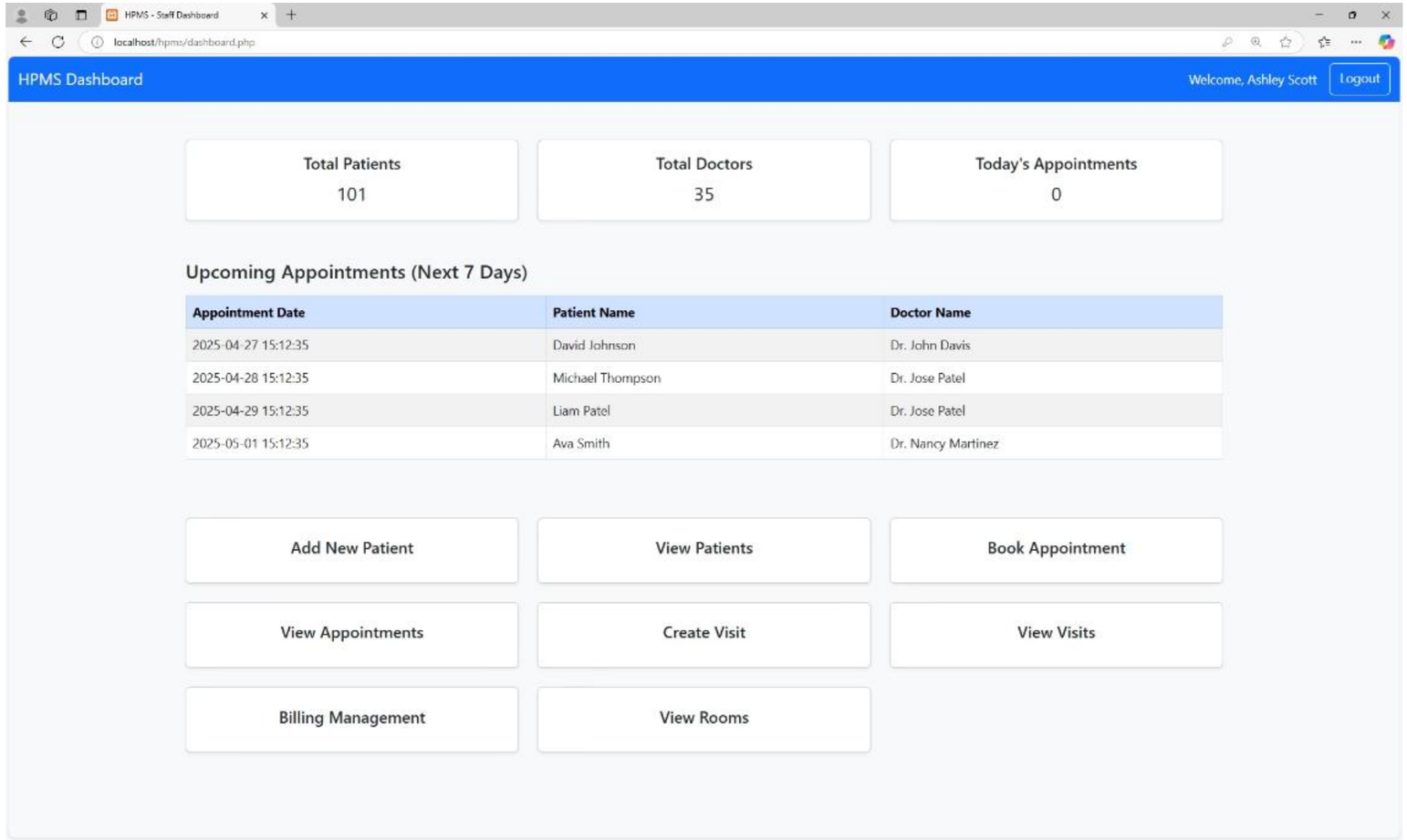
- **JOINs (Inner Joins):** Used extensively to combine data from multiple tables. For example, Q1 joins Patients→Appointments→Doctors; Q2 joins Doctors→Visits; Q6 joins Specialization→Doctors_Specialization. These ensure we can retrieve comprehensive information (like names instead of just IDs) from related tables.
- **Subqueries:** Utilized to filter results based on another query. Q3 uses a subquery to find patients not in the visits_patients table. This illustrates how to query for records lacking a relationship (using NOT IN with a sub-select).
- **Aggregation & GROUP BY:** Demonstrated in Q2, Q4, Q5, Q6. We used COUNT() to count records (visits per doctor, doctors per specialization, bills per status) and SUM() to total numeric values (billing amounts). Grouping by the appropriate field allowed us to get summary rows for each group.
- **HAVING Clause:** Shown in Q4 to filter grouped results (e.g., HAVING COUNT > 1 to find multi-specialty doctors). This is essential for conditions on aggregated data (as opposed to WHERE, which filters row-by-row).
- **ORDER BY & LIMIT:** We sorted results (e.g., most visits, most doctors) and limited output to top N as needed (Q2 and Q6). This helps in reporting top performances or key insights.
- **Data Integrity Checks:** (Implicitly, our queries also validate that relationships work – e.g., no join returned nulls unexpectedly – meaning the foreign keys and data consistency are sound.)

Staff login interface



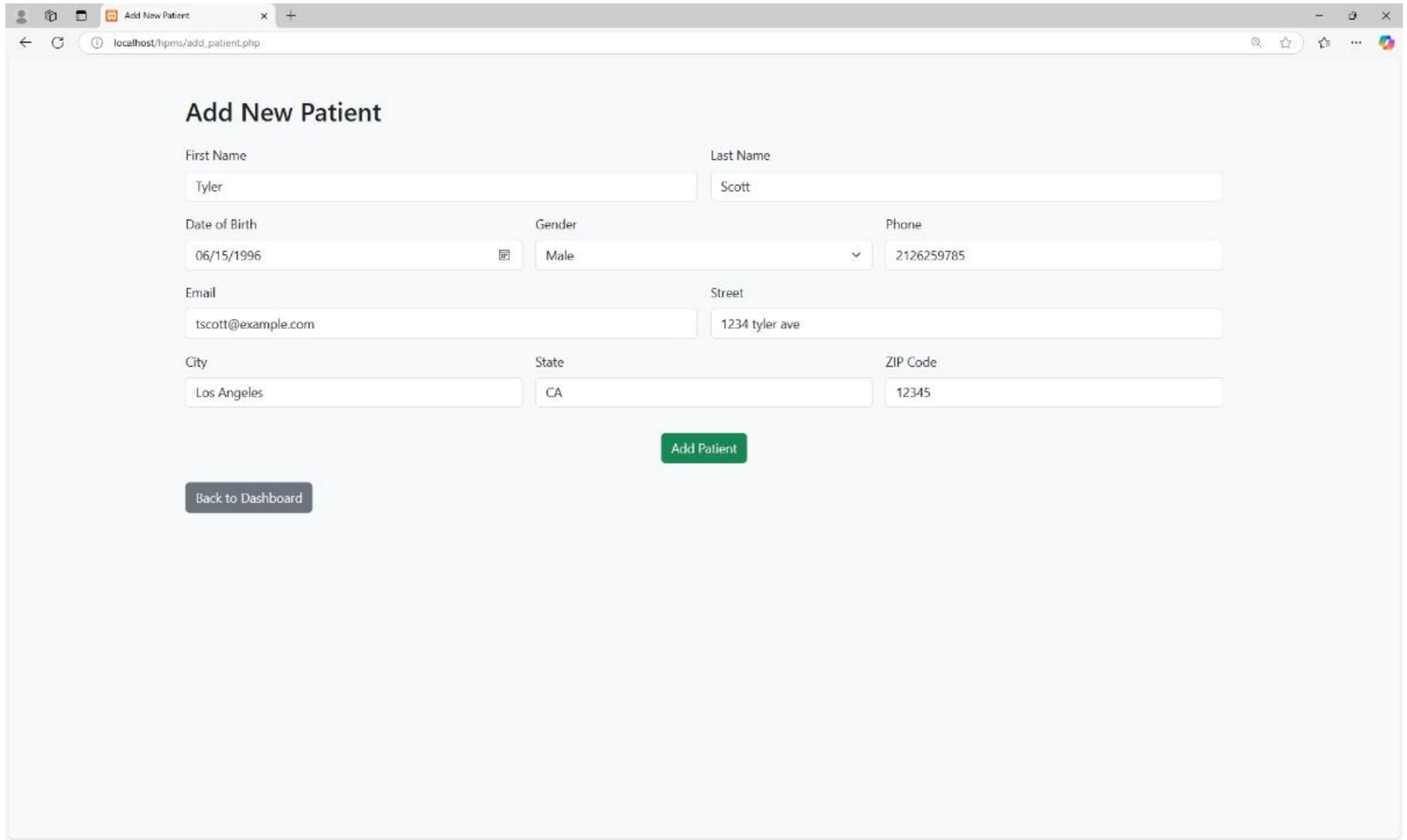
- Entry point for system access. Staff members authenticate using credentials from the hospital_staff table.
- Ensures only authorized users interact with patient and clinical data.
- New staff members whose details are not the hospital_staff tale, can create a new account and the necessary credentials in the hospital_staff table for future logins.

HOSPITAL ADMIN DASHBOARD



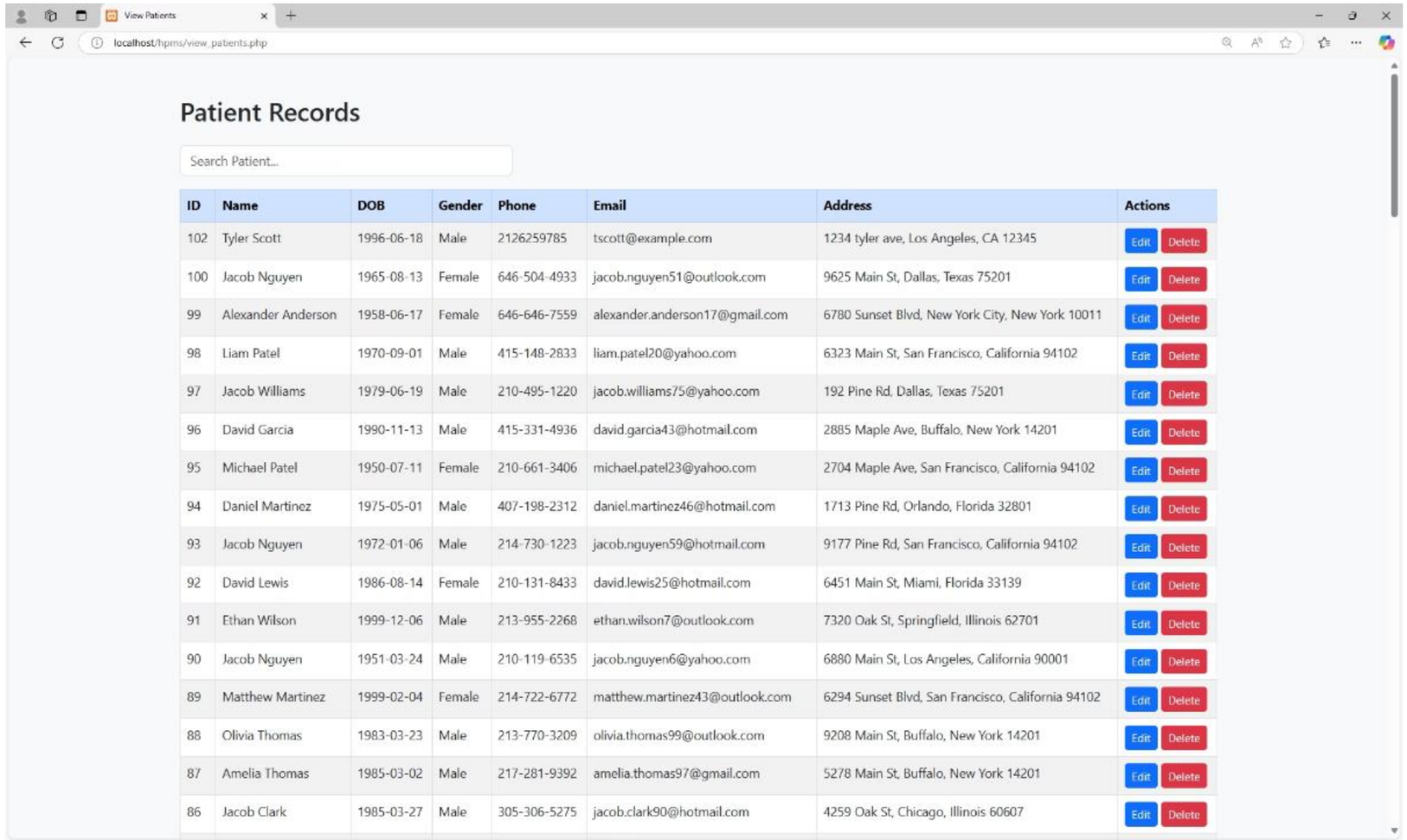
- Session is initialized upon login; username is used throughout the session (e.g., shown on dashboard).
- Displays KPIs like Total Patients, Total Doctors, Appointments Today.
- Shows upcoming appointments by joining appointments, patients, doctors.
- One-click navigation to key modules like Add Patient, Book Appointment, Billing, and Visit Logs.

Patient Registration Form



- Captures full demographic info into the patients table.
- Required fields: name, DOB, gender, contact, and address.
- On submission, triggers INSERT query into patients.

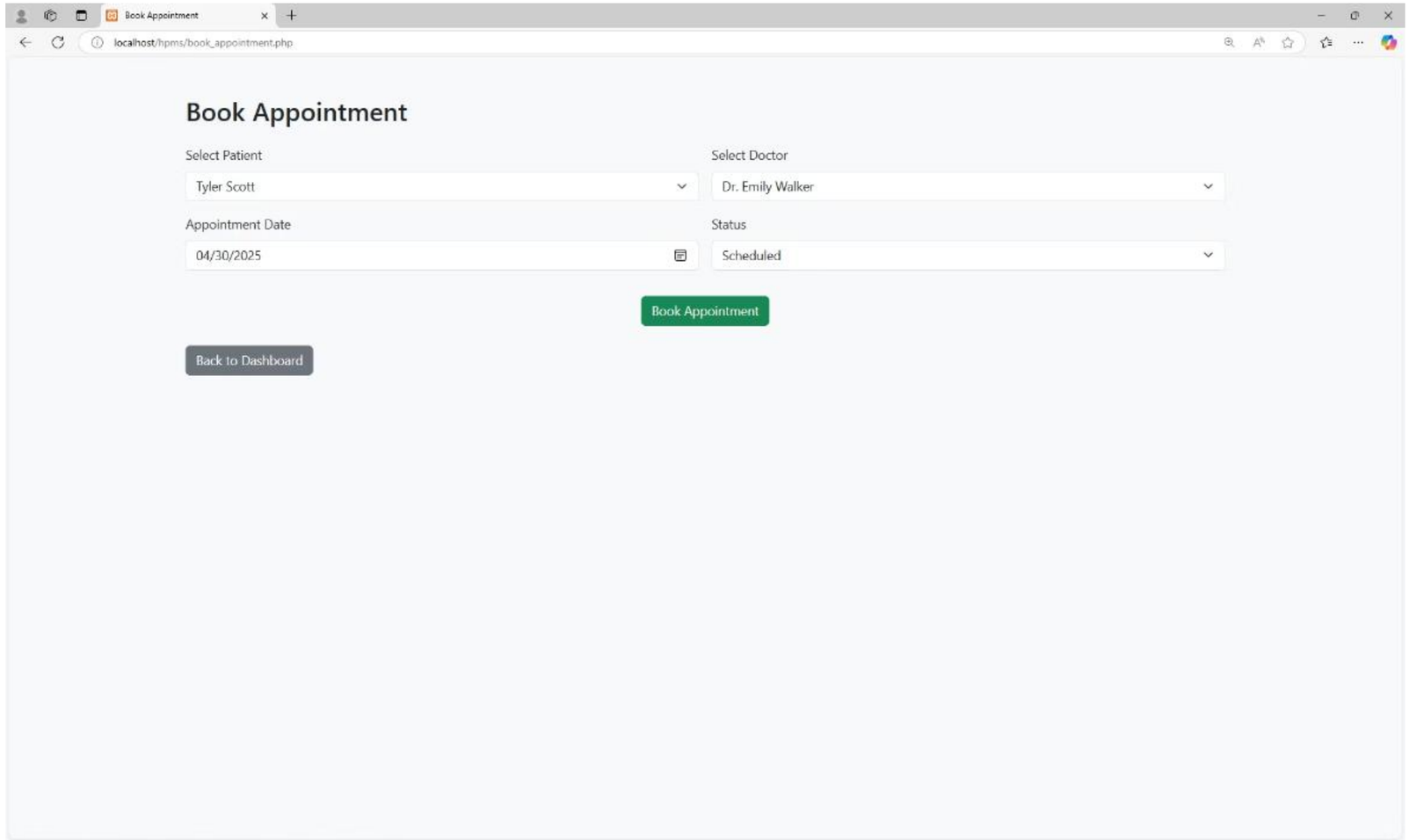
Patient Search, View, EDIT, and Delete



ID	Name	DOB	Gender	Phone	Email	Address	Actions
102	Tyler Scott	1996-06-18	Male	2126259785	tscott@example.com	1234 tyler ave, Los Angeles, CA 12345	Edit Delete
100	Jacob Nguyen	1965-08-13	Female	646-504-4933	jacob.nguyen51@outlook.com	9625 Main St, Dallas, Texas 75201	Edit Delete
99	Alexander Anderson	1958-06-17	Female	646-646-7559	alexander.anderson17@gmail.com	6780 Sunset Blvd, New York City, New York 10011	Edit Delete
98	Liam Patel	1970-09-01	Male	415-148-2833	liam.patel20@yahoo.com	6323 Main St, San Francisco, California 94102	Edit Delete
97	Jacob Williams	1979-06-19	Male	210-495-1220	jacob.williams75@yahoo.com	192 Pine Rd, Dallas, Texas 75201	Edit Delete
96	David Garcia	1990-11-13	Male	415-331-4936	david.garcia43@hotmail.com	2885 Maple Ave, Buffalo, New York 14201	Edit Delete
95	Michael Patel	1950-07-11	Female	210-661-3406	michael.patel23@yahoo.com	2704 Maple Ave, San Francisco, California 94102	Edit Delete
94	Daniel Martinez	1975-05-01	Male	407-198-2312	daniel.martinez46@hotmail.com	1713 Pine Rd, Orlando, Florida 32801	Edit Delete
93	Jacob Nguyen	1972-01-06	Male	214-730-1223	jacob.nguyen59@hotmail.com	9177 Pine Rd, San Francisco, California 94102	Edit Delete
92	David Lewis	1986-08-14	Female	210-131-8433	david.lewis25@hotmail.com	6451 Main St, Miami, Florida 33139	Edit Delete
91	Ethan Wilson	1999-12-06	Male	213-955-2268	ethan.wilson7@outlook.com	7320 Oak St, Springfield, Illinois 62701	Edit Delete
90	Jacob Nguyen	1951-03-24	Male	210-119-6535	jacob.nguyen6@yahoo.com	6880 Main St, Los Angeles, California 90001	Edit Delete
89	Matthew Martinez	1999-02-04	Female	214-722-6772	matthew.martinez43@outlook.com	6294 Sunset Blvd, San Francisco, California 94102	Edit Delete
88	Olivia Thomas	1983-03-23	Male	213-770-3209	olivia.thomas99@outlook.com	9208 Main St, Buffalo, New York 14201	Edit Delete
87	Amelia Thomas	1985-03-02	Male	217-281-9392	amelia.thomas97@gmail.com	5278 Main St, Buffalo, New York 14201	Edit Delete
86	Jacob Clark	1985-03-27	Male	305-306-5275	jacob.clark90@hotmail.com	4259 Oak St, Chicago, Illinois 60607	Edit Delete

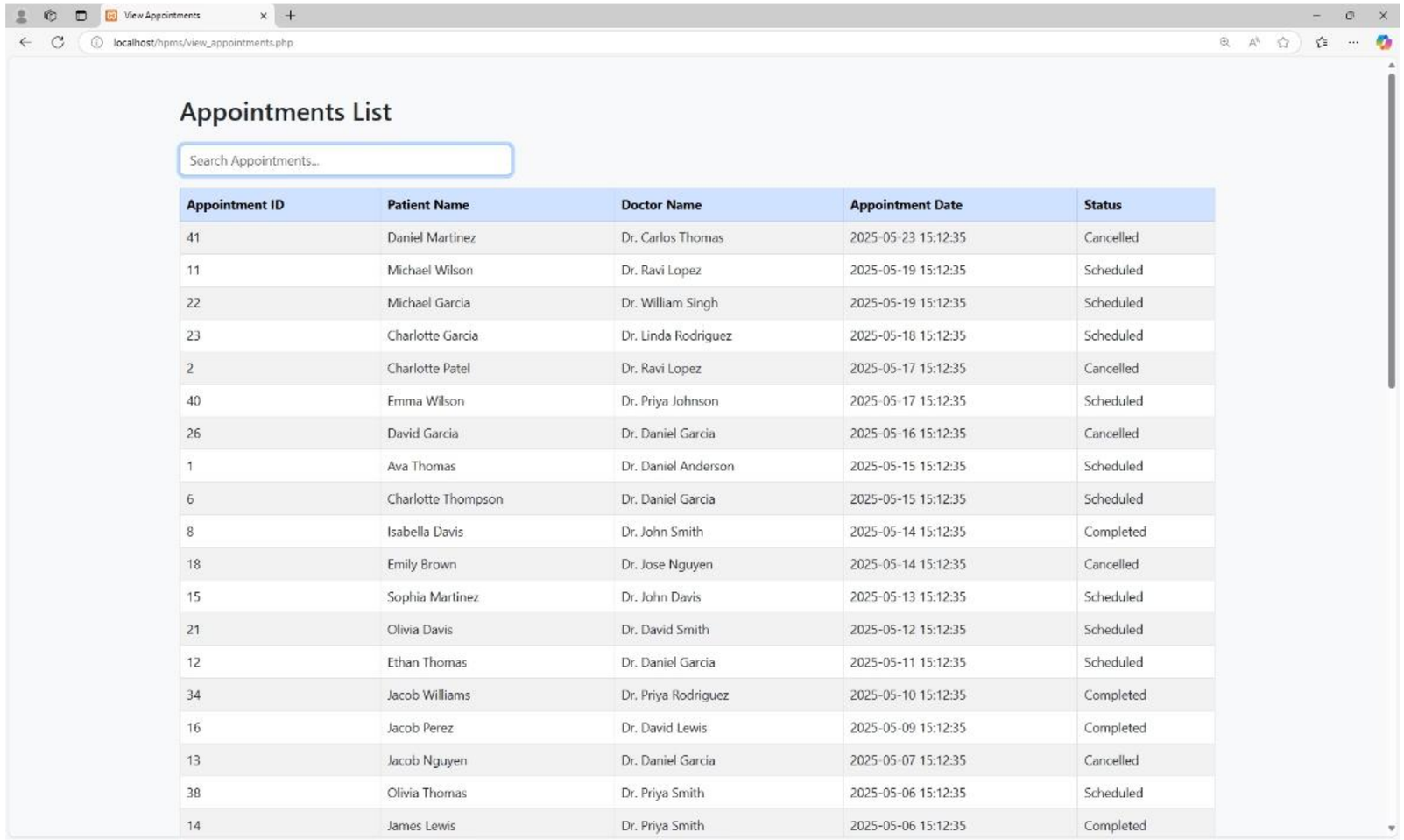
- Lists all registered patients in the system.
- Connected to the patients table with SELECT and optional filtering.
- Supports edit/delete options
- Supports patient search

Appointment Scheduling Form



- Books future visits by inserting records into appointments.
- Select Patient and Doctor from dropdowns (populated from their respective tables).
- Fields: date/time, status ("Scheduled" by default).

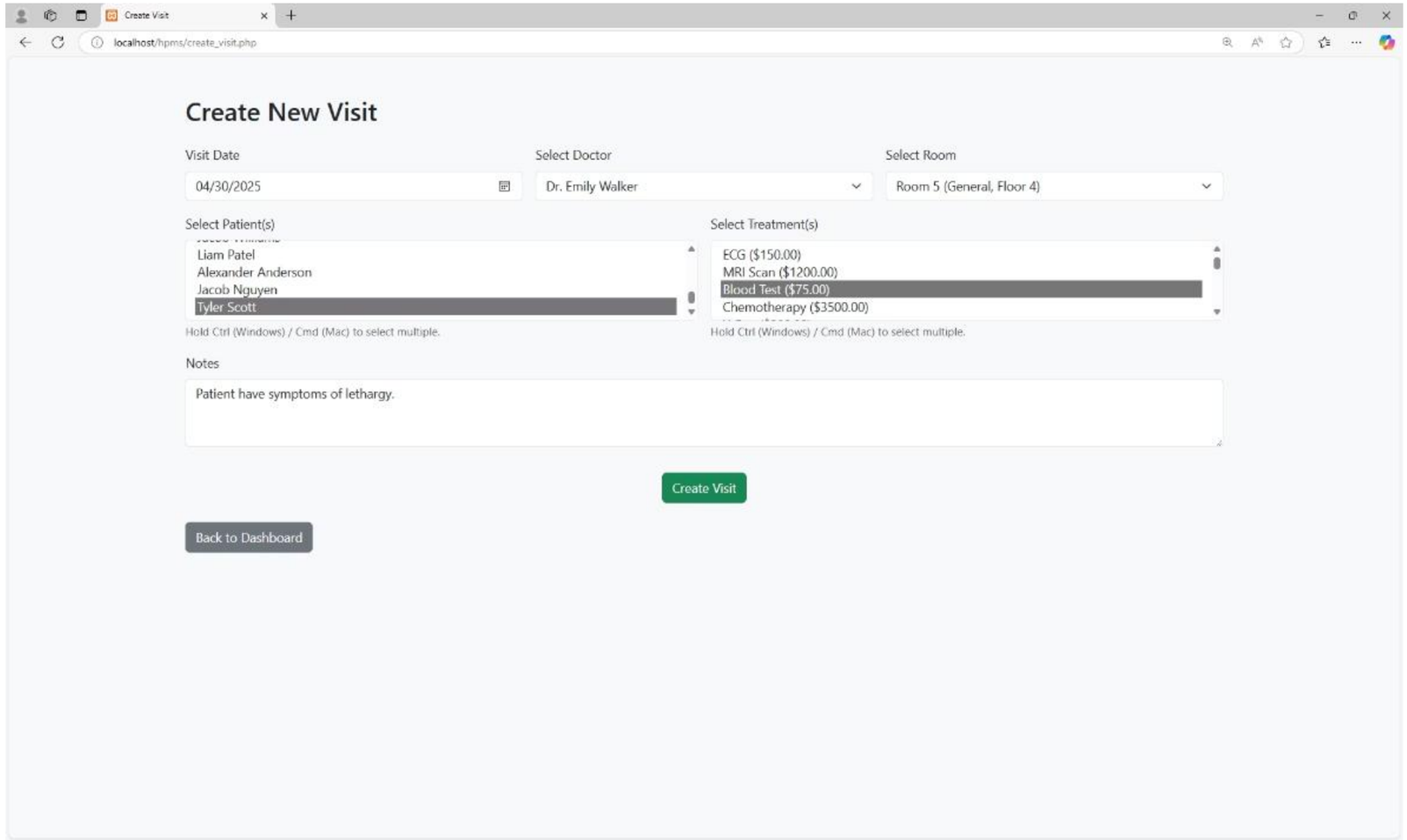
APPOINTMENT LIST & STATUS



Appointment ID	Patient Name	Doctor Name	Appointment Date	Status
41	Daniel Martinez	Dr. Carlos Thomas	2025-05-23 15:12:35	Cancelled
11	Michael Wilson	Dr. Ravi Lopez	2025-05-19 15:12:35	Scheduled
22	Michael Garcia	Dr. William Singh	2025-05-19 15:12:35	Scheduled
23	Charlotte Garcia	Dr. Linda Rodriguez	2025-05-18 15:12:35	Scheduled
2	Charlotte Patel	Dr. Ravi Lopez	2025-05-17 15:12:35	Cancelled
40	Emma Wilson	Dr. Priya Johnson	2025-05-17 15:12:35	Scheduled
26	David Garcia	Dr. Daniel Garcia	2025-05-16 15:12:35	Cancelled
1	Ava Thomas	Dr. Daniel Anderson	2025-05-15 15:12:35	Scheduled
6	Charlotte Thompson	Dr. Daniel Garcia	2025-05-15 15:12:35	Scheduled
8	Isabella Davis	Dr. John Smith	2025-05-14 15:12:35	Completed
18	Emily Brown	Dr. Jose Nguyen	2025-05-14 15:12:35	Cancelled
15	Sophia Martinez	Dr. John Davis	2025-05-13 15:12:35	Scheduled
21	Olivia Davis	Dr. David Smith	2025-05-12 15:12:35	Scheduled
12	Ethan Thomas	Dr. Daniel Garcia	2025-05-11 15:12:35	Scheduled
34	Jacob Williams	Dr. Priya Rodriguez	2025-05-10 15:12:35	Completed
16	Jacob Perez	Dr. David Lewis	2025-05-09 15:12:35	Completed
13	Jacob Nguyen	Dr. Daniel Garcia	2025-05-07 15:12:35	Cancelled
38	Olivia Thomas	Dr. Priya Smith	2025-05-06 15:12:35	Scheduled
14	James Lewis	Dr. Priya Smith	2025-05-06 15:12:35	Completed

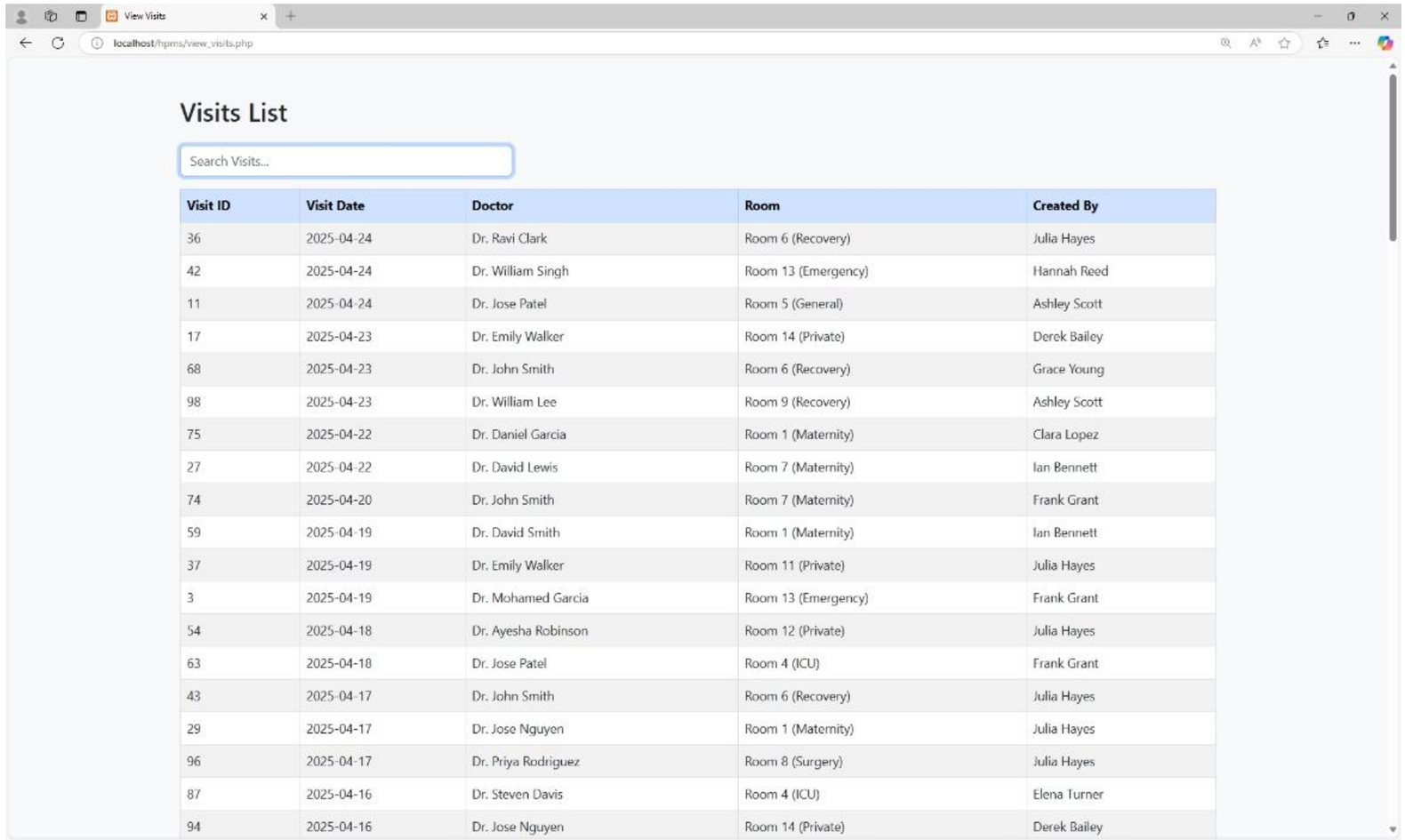
- Displays all scheduled/completed/cancelled appointments.
- Joins appointments, patients, and doctors to display names.
- Helps staff confirm bookings and view appointment history.

VISIT & TREATMENT LOGGING



- Used during patient check-in to log a clinical visit.
- Captures doctor, room, date, patients involved, treatments administered.
- Inserts into visits, visits_patients, and generates a billing record.

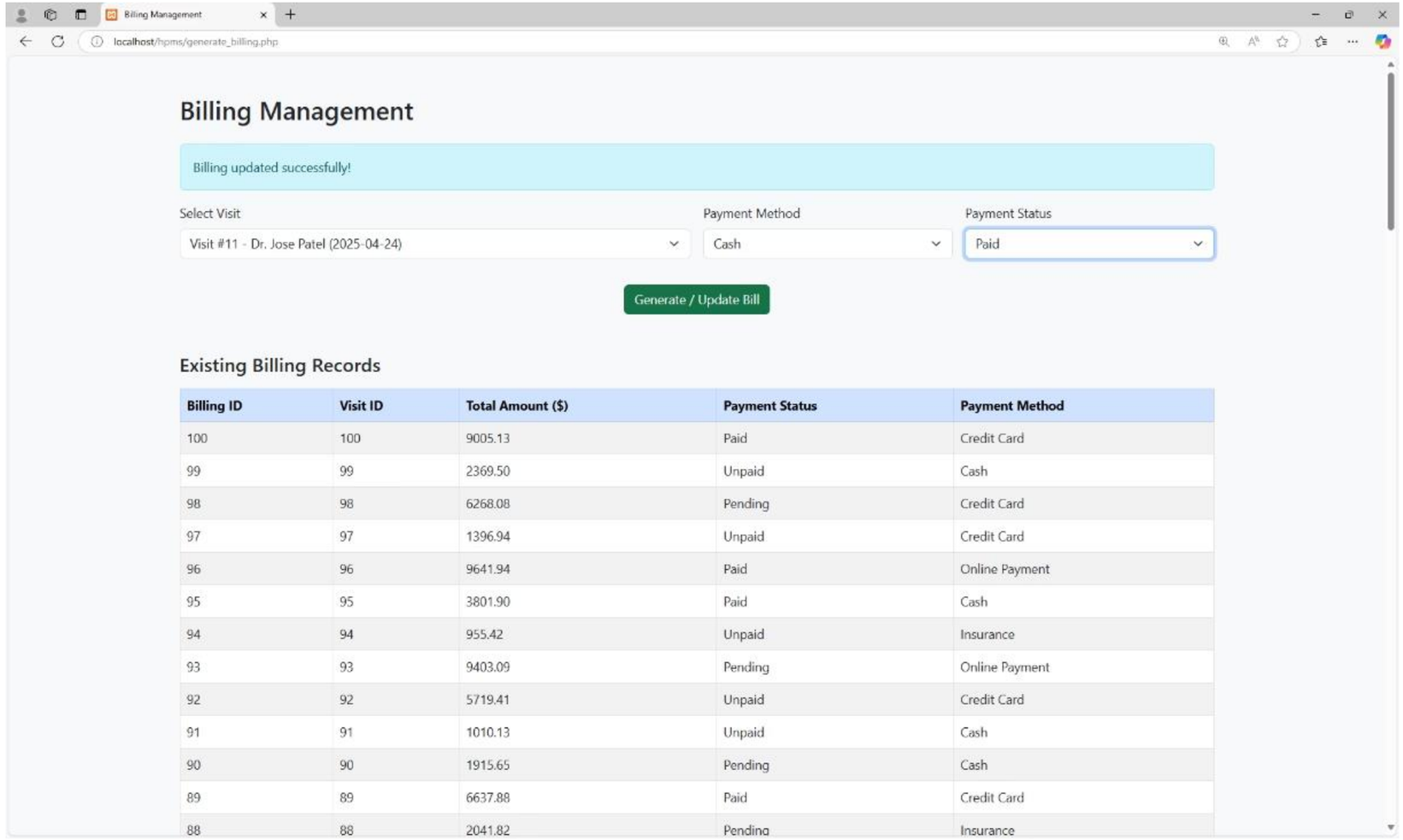
View Visits



Visit ID	Visit Date	Doctor	Room	Created By
36	2025-04-24	Dr. Ravi Clark	Room 6 (Recovery)	Julia Hayes
42	2025-04-24	Dr. William Singh	Room 13 (Emergency)	Hannah Reed
11	2025-04-24	Dr. Jose Patel	Room 5 (General)	Ashley Scott
17	2025-04-23	Dr. Emily Walker	Room 14 (Private)	Derek Bailey
68	2025-04-23	Dr. John Smith	Room 6 (Recovery)	Grace Young
98	2025-04-23	Dr. William Lee	Room 9 (Recovery)	Ashley Scott
75	2025-04-22	Dr. Daniel Garcia	Room 1 (Maternity)	Clara Lopez
27	2025-04-22	Dr. David Lewis	Room 7 (Maternity)	Ian Bennett
74	2025-04-20	Dr. John Smith	Room 7 (Maternity)	Frank Grant
59	2025-04-19	Dr. David Smith	Room 1 (Maternity)	Ian Bennett
37	2025-04-19	Dr. Emily Walker	Room 11 (Private)	Julia Hayes
3	2025-04-19	Dr. Mohamed Garcia	Room 13 (Emergency)	Frank Grant
54	2025-04-18	Dr. Ayesha Robinson	Room 12 (Private)	Julia Hayes
63	2025-04-18	Dr. Jose Patel	Room 4 (ICU)	Frank Grant
43	2025-04-17	Dr. John Smith	Room 6 (Recovery)	Julia Hayes
29	2025-04-17	Dr. Jose Nguyen	Room 1 (Maternity)	Julia Hayes
96	2025-04-17	Dr. Priya Rodriguez	Room 8 (Surgery)	Julia Hayes
87	2025-04-16	Dr. Steven Davis	Room 4 (ICU)	Elena Turner
94	2025-04-16	Dr. Jose Nguyen	Room 14 (Private)	Derek Bailey

- Shows a table of all past visits with doctor, date, room, and creator.
- Real-time search feature helps find visits by doctor or date.
- Backend joins visits, doctors, rooms, and hospital_staff.

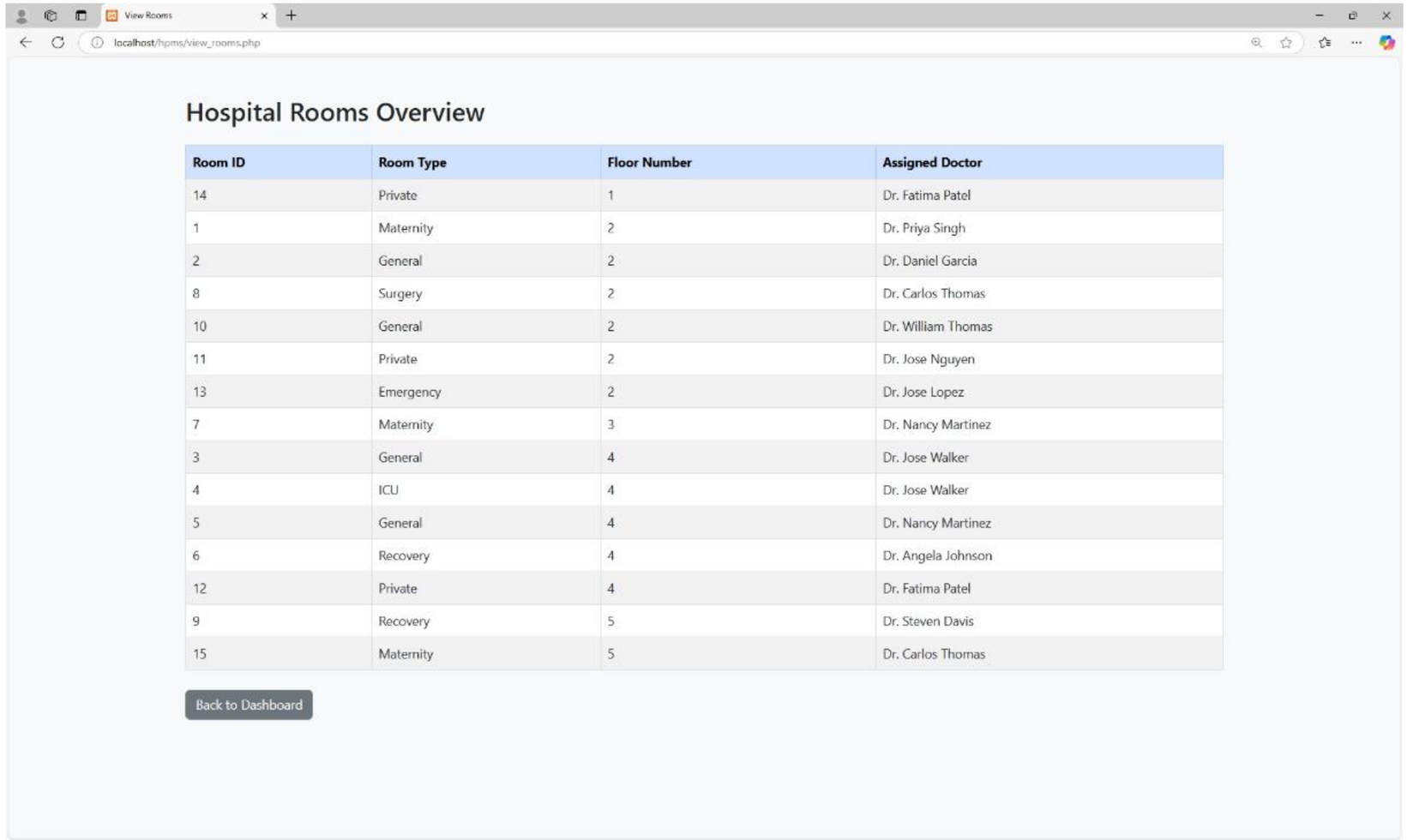
BILLING MANAGEMENT PANEL



Billing ID	Visit ID	Total Amount ($)	Payment Status	Payment Method
100	100	9005.13	Paid	Credit Card
99	99	2369.50	Unpaid	Cash
98	98	6268.08	Pending	Credit Card
97	97	1396.94	Unpaid	Credit Card
96	96	9641.94	Paid	Online Payment
95	95	3801.90	Paid	Cash
94	94	955.42	Unpaid	Insurance
93	93	9403.09	Pending	Online Payment
92	92	5719.41	Unpaid	Credit Card
91	91	1010.13	Unpaid	Cash
90	90	1915.65	Pending	Cash
89	89	6637.88	Paid	Credit Card
88	88	2041.82	Pending	Insurance

- Retrieves total cost for visit (based on treatments).
- Updates billing table: sets payment status, method (Cash/Card/Insurance).
- Displays current billing status with real-time updates.

ROOM MANAGEMENT



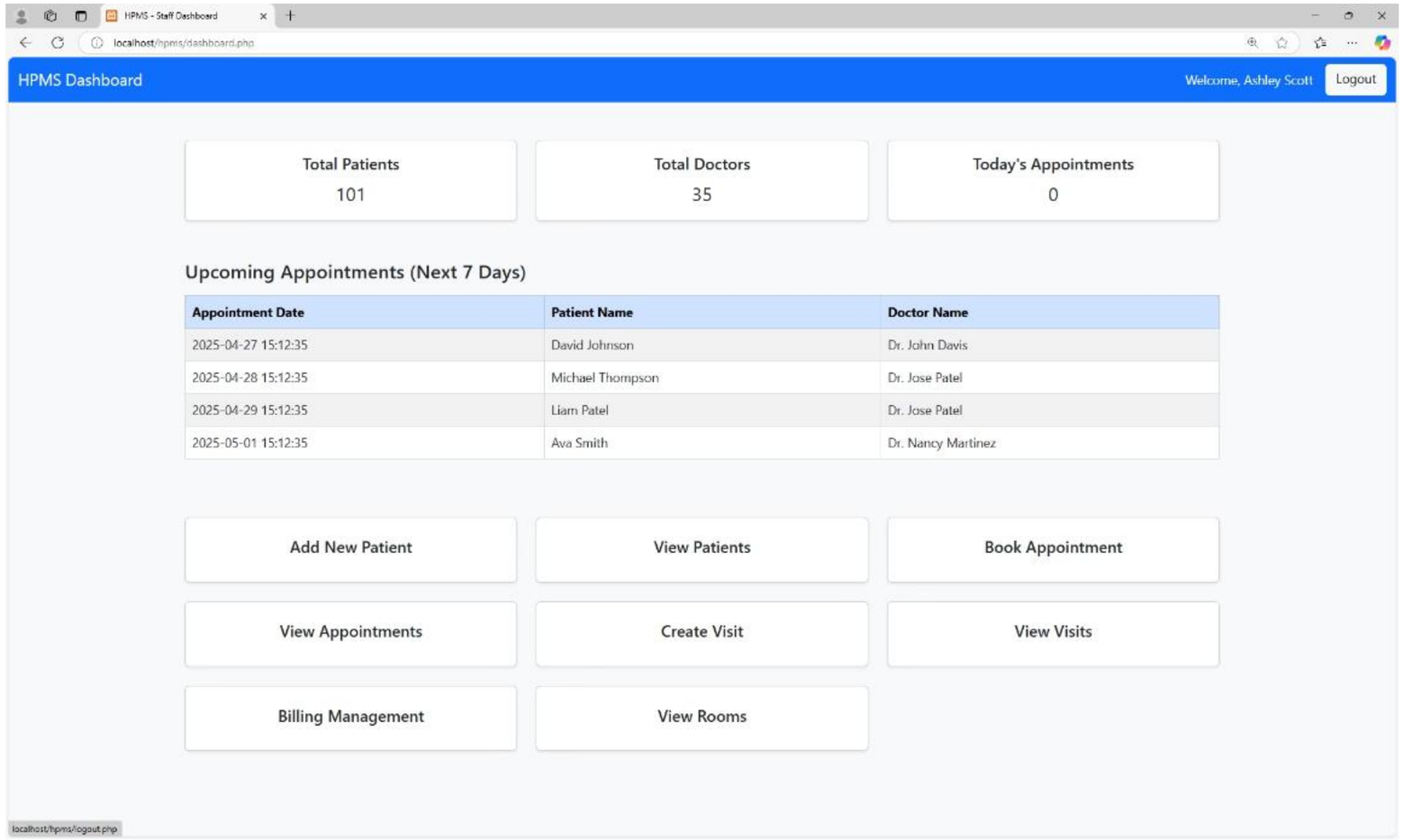
Hospital Rooms Overview

Room ID	Room Type	Floor Number	Assigned Doctor
14	Private	1	Dr. Fatima Patel
1	Maternity	2	Dr. Priya Singh
2	General	2	Dr. Daniel Garcia
8	Surgery	2	Dr. Carlos Thomas
10	General	2	Dr. William Thomas
11	Private	2	Dr. Jose Nguyen
13	Emergency	2	Dr. Jose Lopez
7	Maternity	3	Dr. Nancy Martinez
3	General	4	Dr. Jose Walker
4	ICU	4	Dr. Jose Walker
5	General	4	Dr. Nancy Martinez
6	Recovery	4	Dr. Angela Johnson
12	Private	4	Dr. Fatima Patel
9	Recovery	5	Dr. Steven Davis
15	Maternity	5	Dr. Carlos Thomas

Back to Dashboard

- Lists all hospital rooms with type and floor number.
- Linked to rooms table. Shows optional assigned doctor.
- Helps manage physical resources and occupancy.

SESSION LOGOUT



- Ends the current session and clears authentication.
- Redirects user to login screen.
- Prevents unauthorized access on shared computers.

Thank you for listening to my presentation. I welcome any questions about the Hospital Patient Management System!