Technical Documentation

QMC Hospital Management System

Technical Reference (System Architecture & Design)

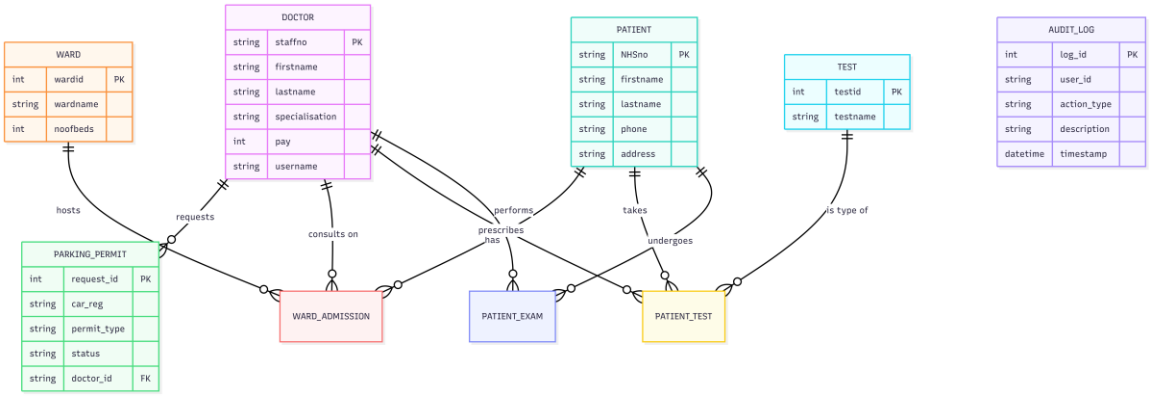
1) **Installation & Setup:**
   The Web-based interface is built using standard LMAP stack (Linux, PHP and MySQL(MariaDB))
   a) **Database Deployment:**
      - **Database:** Import the **hospital.sql** file into your MySQL server. This creates the necessary tables (doctor, patient, ward, tests, etc.) and inserts the initial admin account.
      - **Configuration:** Edit **db.inc.php** to set your local database username and password.
      - **Deployment:** Place all PHP and CSS files into your web server's public directory. No complex frameworks or dependencies are required.

2) **System Structure**
3) The system uses a straightforward Procedural PHP structure.
   a) **Authentication**: Uses PHP Sessions (session_start) to track logged-in users. A security check at the top of every file sends unauthenticated users back to the login page.
   b) **Separation of Concerns:**
      - db.inc.php: Holds database credentials (security).
      - main.css: Handles all styling (UI/UX).
      - "_home.php": The main dashboards acting as controllers.
      - Functional scripts (e.g., prescribe.php, admit_patient.php) handle specific tasks

4) Key Data Flows:
   a) **Admissions Logic:** When a doctor admits a patient, the system does not overwrite previous data. Instead, it inserts a new row into the wardpatientaddmission table. This makes sure that a full historical record of every ward a patient has ever visited is stored.
   b) **Audit Trail:** Security was a key requirement. A shadow logging system was implemented. Every time a user performs an INSERT or UPDATE query (like prescribing a test or approving parking), a secondary query triggers immediately to write the event to the audit_log table.

5) **Database Design (ERD)**:
   The database follows a normalized structure to prevent data redundancy.
   a) Entities & Relationships:

- **Doctor & Patients:** A Many-to-Many relationship exists here (doctors see many patients; patients see many doctors). this was resolved using transaction tables like patient_test and patientexamination.
- **Admissions:** The wardpatientaddmission table links Patients to Wards, recording the specific dates and times of stay.
- **Parking:** A One-to-Many relationship links a Doctor to their Parking Requests.
- **Tests:** The test table acts as a lookup list. When a patient gets a test, we link their ID to the Test ID in the patient_test table, adding a timestamp and result status.



6) **Design Rationale**:
Data integrity was prioritised over complexity. By using associative tables for admissions and tests, the hospital can generate reports on historical data rather than just seeing a snapshot of the current day. The addition of the audit_log table was a specific design choice to meet the security requirements of a medical environment.