Hospital Appointment System

- Calisin, Maria Andrea
- De Guzman, Michelle
- Rojo, Matthew

FUNCTIONALITIES REQUIRED

1. User & Access Management

1. **User Registration / Profile**
    - Create accounts for:
        - **Patients** (individuals booking appointments)
        - **Doctors / Clinicians** (view/manage schedules, patient details)
        - **Front-Desk Staff / Receptionists** (assist patients, manage bookings)
        - **Administrators** (oversee system, manage users, settings)
    - Profile fields per role:
        - Patients: name, date of birth, gender, contact info, address, emergency contact, insurance details (optional)
        - Doctors: name, specialty, qualifications, contact info, working hours, consultation fees
        - Staff/Admin: name, role, contact info, privileges
    - Profile update: change password, update contact details, upload profile photo (optional)
    - Role assignment and permissions (e.g., `ROLE_PATIENT`, `ROLE_DOCTOR`, `ROLE_RECEPTIONIST`, `ROLE_ADMIN`)
    - "Remember me" cookies and session timeout enforcement
2. **Authentication & Authorization**
    - Login/Logout (Tomcat session creation/invalidation) for all user types
    - Role-based access control to restrict pages/actions (e.g., only doctors view their own schedules, only admins manage other users)
    - "Forgot Password" workflow (e-mail reset link with time-limited token)
    - Enforce strong password policies (e.g., minimum length, mixed character classes)
    - Hash passwords (BCrypt or similar) before storing in Derby
    - CSRF protection tokens on all POST forms
3. **Audit Trail (optional but recommended)**
    - Log critical user actions—login attempts, profile updates, appointment bookings or cancellations—with timestamp, user ID, and IP address
    - Maintain an `AuditLog` table for troubleshooting and compliance

2. Patient Management

1. **Patient Master Records**
    - Store personal details: Full Name, DOB, Gender, Contact (phone, e-mail), Address, Emergency Contact, Insurance Provider/Policy Number (if applicable)
    - Unique Patient ID/Record Number (generated)
    - Upload supporting documents (e.g., insurance card scan, ID) and track verification status
2. **Medical History & Demographics**
    - Capture basic medical history flags: chronic conditions (e.g., diabetes, hypertension), allergies, past surgeries
    - Link to external EMR modules (if integrated) or store summary notes in Derby

    - Track demographic fields: blood type, weight, height, primary physician
3. **Search & Filter Patients**
    - Search by Patient ID, name, phone number, or e-mail
    - Filter by age group (pediatric, adult, geriatric), insurance status, or chronic condition (e.g., “All diabetics”)
4. **Patient Status & Notifications**
    - Track status: Active, Inactive, Blocked (e.g., for unpaid bills), Deceased
    - Generate automated notifications: upcoming appointments, appointment reminders, missed appointments follow-up

3. Doctor / Clinician Management

1. **Doctor Profiles & Schedules**
    - Store doctor details: Name, Specialty/Department, Qualifications, Contact, Consultation Fees, Clinic Location
    - Define working days/hours per doctor (e.g., Dr. Smith: Mon/Wed/Fri, 9 AM–12 PM; Tue/Thu, 2 PM–5 PM)
    - Maintain schedule exceptions: vacations, conferences, on-call duty
2. **Availability & Time Slots**
    - Configure appointment slot length (e.g., 15 min, 30 min) per doctor/department
    - Block out unavailable slots (e.g., surgery, rounds, lunch)
    - Support “special clinics” (e.g., diabetic foot clinic every Tuesday) with custom time slot patterns
3. **Search & Filter Doctors**
    - Search by name, specialty, department, or location
    - Filter by availability (e.g., “Show all cardiologists with free slots this week”)
    - Show doctor ratings or feedback aggregates (if integrated)
4. **Doctor Dashboard**
    - View upcoming appointments (day/week view) with patient name, reason, and status
    - Quick access to patient records (Demographics, History) when clicking an appointment
    - Mark appointment status: Checked-In, In-Consultation, Completed, No-Show, Cancelled
    - Record quick notes or follow-up instructions tied to that appointment

4. Appointment Scheduling

1. **Appointment Booking (Patient Portal)**
    - Patients log in and choose:
        - Department or Specialty → Doctor (or “Any available in Department”)
        - Date range (e.g., next 7 days) → display available slots
    - Select available slot → confirm appointment; generate Appointment ID/Confirmation Code
    - At confirmation: capture reason for visit (e.g., “Annual check-up,” “Chest pain”), preferred language, any special needs
2. **Walk-In & Front-Desk Booking**
    - Receptionist/Staff portal: search patient by ID or register new patient → book appointment on behalf
    - Ability to override or manually insert appointments (e.g., emergency walk-in) at front desk
    - Immediate check-in for walk-ins (bypass slot requirement if urgent)
3. **Appointment Modification & Cancellation**
    - Patients or staff can reschedule (change date/time) or cancel if within allowed window (e.g., > 24 hours before)
    - Automatic release of slot back to availability grid on cancellation

    - Track cancellation reasons and frequency (e.g., “Patient no-show” triggers warning)
4. **Recurring Appointments & Follow-Ups**
    - For chronic patients, allow booking recurring slots (e.g., dialysis every Mon/Wed/Fri at 10 AM for next month)
    - System generates series of appointment entries; doctors can confirm or adjust each occurrence
    - Link follow-up notes so that when a clinician marks one completed, next is auto-scheduled per interval
5. **Appointment Status & Check-In Process**
    - Status flow: Scheduled → Confirmed → Checked-In → In-Consultation → Completed / No-Show / Cancelled
    - At check-in, front desk marks patient as “Checked-In” and prints a queue ticket or notifies nurse staff
    - Doctor’s portal updates status to “In-Consultation” when patient enters, then “Completed” when done
6. **Waitlist Management (Optional)**
    - If no slots available, patients can join a waitlist for a particular doctor/date
    - When a slot frees up or a cancellation occurs, system auto-notifies next waitlisted patient to confirm or decline
    - Manage waitlist queue per doctor or per department

5. Notifications & Reminders

1. **Automated E-Mail / SMS Reminders**
    - Send reminders 24 hours and 2 hours before scheduled appointment; include doctor name, time, location, and link to reschedule
    - Use JavaMail API for e-mails (HTML templates) and mock or real SMS gateway (e.g., Twilio) for SMS
2. **In-App Notifications / Alerts**
    - Dashboard alerts for patients: upcoming appointments, missed appointments follow-up
    - Dashboard alerts for doctors: new appointments booked, cancellations, urgent messages from front desk
    - Staff/admin alerts: overbookings, no-show trends, critical cancellations
3. **Push Notifications (Optional)**
    - If mobile module exists, send push via Web Push or mobile app integration for reminders and status changes
4. **Notification Templates & Preferences**
    - Store configurable templates in database:
        - Appointment Confirmation: `{PatientName}, your appointment with Dr. {DoctorName} is confirmed for {DateTime}`
        - Reminder: `Reminder: Your appointment tomorrow at {Time} with Dr. {DoctorName}`
        - Cancellation: `Your appointment on {DateTime} has been cancelled by {DoctorName}`
    - Allow patients to opt in/out of SMS vs. e-mail notifications, choose language

6. Doctor & Clinic Workflow

1. **Doctor Dashboard & Appointment Queue**
    - Show today’s schedule in chronological order with patient names and appointment types (New, Follow-Up)

    - Color-code appointments by status: Scheduled, Checked-In, No-Show, Completed
    - One-click access to patient profile and medical history (if integrated with EMR)
    - Buttons to mark “In-Consultation,” “Complete,” or “Cancel”
2. **Nurse / Triage Station**
    - Nurses see a waiting list of checked-in patients for a given doctor/department
    - Record vital signs (BP, temperature, weight) pre-consultation and link to appointment record
    - Assign priority (e.g., “High priority: severe chest pain”) so doctors can reorder queue
3. **Consultation Notes & Follow-Up Orders**
    - After consultation, doctor enters brief notes: diagnosis code, summary, prescribed investigations (lab tests, imaging), medication orders (if permitted)
    - Schedule follow-up appointment automatically if needed (e.g., “Review in 2 weeks”) or leave to front-desk staff
4. **Clinical Task Management (Optional)**
    - For multi-disciplinary clinics, create tasks for ancillary departments (e.g., “Lab: draw blood for CBC,” “Radiology: chest X-ray”)
    - Monitor task status: Pending → Completed → Results available → Notify doctor
5. **Overbooking & Exception Handling**
    - Allow doctors/staff to overbook a small number of slots per day if historically certain patients no-show (configurable threshold)
    - Flag overbooked slots and alert front desk to manage queue and patient expectations

7. Payment & Billing (Optional or Integrated)

1. **Consultation Fees & Billing Items**
    - Define consultation fee per doctor or per department (e.g., General Practitioner: $50, Specialist: $100)
    - Additional charges: urgent visit fee, late-day fee, weekend/holiday surcharge
    - Store these fees in a `BillingItems` table with effective dates
2. **Collecting Payments at Check-In or Check-Out**
    - Front desk captures payment at time of booking or at time of visit (configurable)
    - Support payment methods: cash, credit/debit card (mock integration), insurance co-pay
    - Record transaction in `Payments` table: PaymentID, PatientID, AppointmentID, amount, method, transaction timestamp, status
3. **Invoice / Receipt Generation**
    - Generate PDF/HTML invoice summarizing consultation fee and any additional charges (e.g., lab tests, imaging)
    - E-mail or print receipt for patient; store digital copy in `Invoices` table
4. **Insurance & Co-Pay Handling (Optional)**
    - Capture insurance details at patient registration (provider, policy number, coverage info)
    - At booking, compute estimated co-pay or coverage share and bill accordingly
    - Generate claim data for integration with external insurance systems (mock CSV export)
5. **Outstanding Balances & Payment Reminders**
    - Track unpaid invoices; if balance remains after appointment, flag patient and prevent future booking until cleared
    - Automated reminders via e-mail/SMS for outstanding balances after X days

8. Reporting & Analytics

1. **Appointment Reports**

    - Daily/Weekly/Monthly counts: total appointments scheduled, completed, cancelled, no-shows
    - Utilization rates per doctor: percentage of slots filled vs. available
    - Average wait time: time between scheduled appointment time and actual "In-Consultation" timestamp
2. **Patient Flow & No-Show Analysis**
    - No-show rate by department, doctor, day of week, or time slot
    - Peak booking hours and idle periods, to adjust staffing levels
    - Average lead time: time between booking and appointment date
3. **Revenue & Billing Reports (if billing integrated)**
    - Total revenue per doctor/department by period, average revenue per appointment
    - Payment breakdown by method (cash vs. card) and outstanding receivables
    - Insurance vs. self-pay revenue share (if insurance module enabled)
4. **Doctor Performance & Load**
    - Number of consultations per doctor per day/week, average consultation duration (based on timestamps)
    - Comparison of scheduled vs. actual consultations to identify overruns or downtime
5. **Custom/Ad-Hoc Report Builder**
    - UI for administrators to define custom filters (e.g., "List all no-shows for Dr. Patel in March 2025")
    - Export results to CSV, Excel, or PDF; schedule automated report e-mail delivery
6. **Dashboard & KPIs**
    - Role-based dashboards:
        - **Admin**: Overall appointment volume, revenue metrics, no-show trends, system usage
        - **Manager/Receptionist**: Today's schedule summary, pending cancellations, waitlist length
        - **Doctor**: Personal load (today's appointments), upcoming days' schedules, patient no-show counts
        - **Patient**: Upcoming appointments, recent visit history, outstanding balances
    - Drag-and-drop widgets to allow users to customize their dashboard layout; save in `DashboardConfig` table

9. Integration & Technical Infrastructure

1. **Database Schema (Derby)**
    - **Tables**:
        - `Users` (UserID, username, password_hash, role_id, status, created_at)
        - `Roles` (RoleID, role_name)
        - `Patients` (PatientID, UserID, full_name, dob, gender, contact_info, address, insurance_provider, insurance_policy, status)
        - `Doctors` (DoctorID, UserID, full_name, specialty, qualifications, contact_info, consultation_fee, status)
        - `ClinicLocations` (LocationID, name, address, phone)
        - `DoctorSchedules` (ScheduleID, DoctorID, day_of_week, start_time, end_time, slot_duration_minutes, location_id)
        - `Appointments` (AppointmentID, PatientID, DoctorID, scheduled_datetime, status, reason, created_at, updated_at)
        - `AppointmentStatuses` (e.g., Scheduled, Confirmed, CheckedIn, InConsultation, Completed, Cancelled, NoShow)
        - `WaitlistEntries` (WaitlistID, PatientID, DoctorID, requested_date, position, created_at)
        - `Notifications` (NotificationID, UserID, type, content, is_read, created_at)

        - `Payments` (PaymentID, PatientID, AppointmentID, amount, method, transaction_id, status, paid_at)
        - `Invoices` (InvoiceID, PaymentID, amount, issued_at, invoice_pdf_path)
        - `AuditLogs` (LogID, UserID, action_type, resource_type, resource_id, timestamp, ip_address)
    - Define primary/foreign keys and indexes on high-volume columns (e.g., `Appointments.scheduled_datetime`, `DoctorSchedules.DoctorID`)
    - Constraints:
        - Ensure no overlapping schedule entries for the same doctor/time
        - Enforce appointment slot availability based on `DoctorSchedules`
2. **JDBC Data Access Layer (DAO)**
    - DAO classes for each major entity (e.g., `UserDAO`, `PatientDAO`, `DoctorDAO`, `AppointmentDAO`, `PaymentDAO`)
    - Use `PreparedStatement` for all SQL operations to prevent SQL injection
    - Configure Tomcat JDBC connection pool in `context.xml` for performance and concurrency
3. **Servlets & JSP (or JSP + JSTL)**
    - **Controllers (Servlets)** examples:
        - `UserServlet` (login, registration, profile management)
        - `PatientServlet` (patient CRUD, search, verify insurance)
        - `DoctorServlet` (doctor CRUD, schedule management)
        - `ScheduleServlet` (view/modify doctor availability, block times)
        - `AppointmentServlet` (book, modify, cancel, check-in, complete)
        - `WaitlistServlet` (join, notify, remove waitlist entries)
        - `NotificationServlet` (list, mark as read)
        - `PaymentServlet` (process payments, record transactions, refunds)
        - `InvoiceServlet` (generate, download invoices)
        - `ReportServlet` (generate/export reports)
        - `DashboardServlet` (role-based dashboards)
    - **Views (JSPs)** examples:
        - `login.jsp`, `registerPatient.jsp`, `patientProfile.jsp`, `doctorProfile.jsp`, `adminDashboard.jsp`
        - `doctorList.jsp`, `doctorSchedule.jsp`, `scheduleForm.jsp`
        - `appointmentForm.jsp` (select doctor/date/time), `appointmentList.jsp`, `appointmentDetail.jsp`
        - `waitlistForm.jsp`, `waitlistList.jsp`
        - `paymentForm.jsp`, `invoiceView.jsp`, `paymentHistory.jsp`
        - `notificationList.jsp`
        - `reports.jsp`, `dashboard.jsp`
    - Follow MVC: Servlets → Service/DAO → Model → JSP; use JSTL tags (`<c:forEach>`, `<c:if>`, `<fmt:formatDate>`, `<fmt:formatNumber>`) for dynamic content
4. **Session & Cookie Management**
    - Store user role, user ID, and session-specific flags (e.g., current patient search filter) in session attributes
    - Enforce session timeout (e.g., 20 minutes of inactivity) in `web.xml` or programmatically
    - Use secure, HTTP-only cookies for session IDs and optional “Remember me” tokens
5. **Validation & Error Handling**
    - Client-side validation (JavaScript) for required fields (e.g., ensure appointment date/time is selected)
    - Server-side validation in servlets (e.g., check doctor’s availability before booking; check insurance details if required)
    - Custom error pages (`404.jsp`, `error.jsp`) with user-friendly messages; log stack traces internally

6. **Security**
    - Enforce HTTPS (Tomcat SSL configuration) so all data in transit is encrypted
    - Use `PreparedStatement` to prevent SQL injection; sanitize free-text inputs (appointment reasons, notes) to prevent XSS
    - Restrict URL patterns by role in `web.xml` (e.g., `/doctor/*` only for `ROLE_DOCTOR`, `/receptionist/*` only for `ROLE_RECEPTIONIST`, `/admin/*` only for `ROLE_ADMIN`)
    - Protect file upload directories (e.g., insurance scans, profile photos) by storing them outside the webroot and validating file types/sizes
7. **Deployment on Tomcat**
    - Define servlet mappings and security constraints in `WEB-INF/web.xml`
    - Place Derby driver JARs in `WEB-INF/lib`; choose embedded vs. network server mode based on concurrency needs
    - Configure JDBC DataSource in `META-INF/context.xml` or global `tomcat/conf/context.xml`
8. **Logging**
    - Use Log4j2 or `java.util.logging` to capture:
        - INFO: normal operations (e.g., "Patient 1002 booked Appointment 3005 with Doctor 2001 at 2025-06-10 10:00")
        - WARN: unusual events (e.g., "Attempt to book slot already taken for Appointment 3005")
        - ERROR: system failures or exceptions (e.g., DB connection errors, servlet exceptions)
    - Roll logs daily and archive older logs for audit and compliance
9. **Backup & Restore (optional but recommended)**
    - Schedule nightly Derby database backups via `SYSCS_UTIL.SYSCS_BACKUP_DATABASE`
    - Provide an admin UI or script to restore from a selected backup ZIP in case of data corruption or loss
10. **Unit & Integration Tests (optional)**
    - JUnit tests for DAO and service layers (e.g., ensure `AppointmentDAO.bookAppointment()` enforces slot availability and no overlaps)
    - Integration tests for servlets using embedded Tomcat or Mock frameworks (e.g., Mockito to simulate HTTP requests, session handling, and role enforcement)

10. Other Functionalities

1. **EMR / Clinical Data Integration (optional)**
    - If integrated with an EMR, provide links to full patient chart, lab results, prescriptions, and treatment plans from the appointment detail page
    - Use RESTful or JDBC integration to pull/push data between the Appointment System and EMR
2. **Telemedicine / Virtual Consultations (optional)**
    - Provide an option to book "Telehealth" appointments; generate secure video conference link (e.g., Zoom, Jitsi)
    - Store session link and allow doctor/patient to join from dashboard
    - Automatically send link via e-mail/SMS reminder 1 hour before
3. **Mobile-Friendly & PWA Enhancements**
    - Design JSP pages using a responsive framework (e.g., Bootstrap) so patients and staff can access from tablets or smartphones
    - Implement service workers for offline caching of critical pages (e.g., upcoming appointments)
4. **API Endpoints (optional)**
    - Expose RESTful services for:
        - Doctor availability (`GET /api/doctors/{id}/availability?date=2025-06-15`)

        - Book appointment (`POST /api/appointments`)
        - Get patient appointments (`GET /api/patients/{id}/appointments`)
        - Cancel appointment (`DELETE /api/appointments/{id}`)
    - Secure API calls with token-based authentication (e.g., JWT) and rate-limit requests
5. **Multi-Location / Multi-Branch Support (optional)**
    - Support multiple clinic locations or branches under the same hospital network
    - Allow patients to select preferred location when booking; maintain separate schedules per location
    - Enable cross-location transfers if a doctor practices at multiple sites
6. **Multi-Language & Localization (i18n)**
    - Use Java Resource Bundles (`messages_en.properties`, `messages_es.properties`) for UI text
    - Detect user's locale and render JSPs with `<fmt:formatMessage>` and `<fmt:formatDate>` tags
    - Format dates, times, and numbers according to locale preferences
7. **Accessibility Compliance (WCAG)**
    - Ensure JSPs use semantic HTML (e.g., proper labels for form fields, ARIA attributes) to support screen readers
    - Provide keyboard navigation and sufficient color contrast for visually impaired users
8. **Analytics Dashboard & KPIs**
    - Drag-and-drop widgets for quick insights:
        - "Today's Appointments by Department"
        - "Upcoming No-Show Follow-Ups"
        - "Revenue from Consultations" (if billing enabled)
        - "Doctor Utilization Rates"
    - Save user-specific dashboard layouts in a `DashboardConfig` table
9. **High Availability & Scalability Considerations**
    - Deploy in a clustered Tomcat environment behind a load balancer for fault tolerance
    - Run Derby in network server mode on a dedicated DB server, or migrate to a clustered RDBMS (e.g., PostgreSQL, MySQL) as usage grows
    - Implement caching (e.g., Ehcache) for frequently accessed data (doctor schedules, patient details) to reduce database load