Ai Nexus

Hospital Patient Management System

Patient Module — Features & UI Test Flow Guide

Version 1.1.0 · February 2026 · CONFIDENTIAL

Prepared by Ai Nexus Engineering Team

Table of Contents

1. Overview & System Architecture

The Patient Module is the foundational data layer of the Ai Nexus Hospital Management System. It provides HIPAA-compliant patient record management including registration, search, profile view, demographic updates, and status management. All PHI is encrypted at rest (AES-256) and every access is immutably audit-logged.

Layer	Technology	Purpose
Frontend	React 18 + TypeScript + Ant Design	SPA at http://localhost
API Gateway	Nginx (port 80)	Reverse-proxy; routes /api/* to backend
Backend	Spring Boot 3.4.1 / Java 17	REST API at hospital-api:8080
Security	Spring Security 6 + JWT (HMAC-SHA256)	Stateless auth + RBAC
Database	PostgreSQL 15	Patient records + immutable audit logs
Audit	AOP @AfterReturning + REQUIRES_NEW TX	All read/write actions logged
Encryption	AES-256-GCM (AttributeConverter)	PHI fields encrypted in DB

Base URL: http://localhost | **API Base:** http://localhost/api/v1 | **Health:** http://localhost/actuator/health

2. User Roles & Permissions

The module enforces Role-Based Access Control (RBAC) at both the API layer (Spring Security) and the UI layer (conditional rendering). Four roles are supported:

Role	Badge Colour	Register	View/Search	Edit	Activate/Deactivate
RECEPTIONIST	Blue	■	■	■	■
DOCTOR	Green	■	■	■	■
NURSE	Cyan	■	■	■	■
ADMIN	Red	■	■	■	■

How to login as any role: Go to http://localhost/login → enter any username → select role from dropdown → Sign In.

FEATURE 1

3. Authentication — Dev Login

A development-only login page issues a real HMAC-SHA256 JWT token so the UI can be tested without a production Auth Module. The token is stored in **localStorage** and sent as **Authorization: Bearer <token>** on every API call.

UI Test Steps

1	**Open browser**	Navigate to http://localhost — redirects to /login automatically
2	**Enter username**	Type any username, e.g. receptionist1, dr.smith, nurse.jane, admin
3	**Select role**	Choose from dropdown: RECEPTIONIST | DOCTOR | NURSE | ADMIN
4	**Read role description**	Description below dropdown explains permissions for selected role
5	**Click Sign In**	POST /api/v1/auth/dev-login → JWT token issued and stored
6	**Observe header**	Username + colour-coded role badge + Sign Out button appear in navbar
7	**Confirm redirect**	Automatically navigated to /patients (Patient List page)

Expected Results

- JWT token stored in localStorage under key 'token'
- Header shows: username | role badge (RECEPTIONIST=blue, DOCTOR=green, NURSE=cyan, ADMIN=red)
- Redirected to Patient List page
- Dev mode warning banner visible on login page

Test All 4 Roles

Username	Role	Expected Header Badge
receptionist1	RECEPTIONIST	Blue badge
dr.smith	DOCTOR	Green badge
nurse.jane	NURSE	Cyan badge
admin	ADMIN	Red badge

FEATURE 2

4. Patient Registration

Allows RECEPTIONIST and ADMIN roles to register new patients. The system generates a unique Patient ID (format: P + year + 3-digit sequence, e.g. **P2026001**), validates all fields, detects potential duplicate phone numbers, and records the registration user and timestamp.

4.1 Prerequisites

Login as **receptionist1 / RECEPTIONIST** or **admin / ADMIN** before testing.

4.2 Happy Path — Register a New Patient

1	**Navigate**	From Patient List, click **Register New Patient** button (top-right)
2	**Personal Info**	Fill: First Name = Sarah, Last Name = Connor, Date of Birth = 1985-07-04, Gender = Female
3	**Contact Info**	Phone = +1-555-200-0001, Email = sarah.connor@example.com, Address = 789 Elm St, City = Chicago, State = IL, ZIP = 60601
4	**Emergency Contact**	Name = John Connor, Phone = +1-555-200-0002, Relationship = Son
5	**Medical Info**	Blood Group = B+, Known Allergies = Sulfa drugs, Chronic Conditions = Asthma
6	**Submit**	Click **Register Patient** button
7	**Verify success**	Toast notification: 'Patient registered — ID: P2026XXX'
8	**Verify redirect**	Automatically navigated to patient detail page for the new patient

4.3 Form Sections

Section	Required Fields	Optional Fields
Personal Information	First Name, Last Name, Date of Birth, Gender	—
Contact Information	Phone Number	Email, Address, City, State, ZIP
Emergency Contact	—	Name, Phone, Relationship
Medical Information	—	Blood Group, Allergies, Conditions

4.4 Validation Testing

1	**Blank required field**	Click into First Name, tab away → red error: 'First name is required'
2	**Blank phone**	Click into Phone, tab away → red error: 'Phone number is required'
3	**Invalid phone format**	Type '12345' → red error: 'Phone: +1-XXX-XXX-XXXX, (XXX) XXX-XXXX, or XXX-XXX-XXXX'
4	**Invalid email**	Type 'notanemail' → red error: 'Invalid email address'

5	**Future date of birth**	Open date picker → future dates are greyed out and unselectable
6	**Submit empty form**	Click Register Patient with no data → all required fields highlight red

4.5 Duplicate Phone Warning

1	**Register patient 1**	Register with phone +1-555-200-0001 → success (P2026XXX)
2	**Register patient 2**	Register another patient with same phone +1-555-200-0001
3	**Observe warning**	Yellow warning toast: 'Another patient with this phone number may already exist'
4	**Confirm registration**	Second patient is registered successfully despite warning (both records saved)

4.6 RBAC — Register Button Hidden for DOCTOR/NURSE

- Login as dr.smith / DOCTOR → Register New Patient button is NOT visible on Patient List
- Login as nurse.jane / NURSE → Register New Patient button is NOT visible on Patient List
- Direct URL /patients/new as DOCTOR → POST /api/v1/patients returns 403 Forbidden

FEATURE 3

5. Patient Search & Filtering

The Patient List page provides live search with 300ms debounce across multiple fields, combined with dropdown filters for status, gender, and blood group. All authenticated roles can search and view the patient list.

5.1 Default State

- Page loads at http://localhost/patients
- Active patients shown by default (Status filter pre-set to 'Active')
- Table columns: Patient ID | Full Name | Age | Gender | Phone | Status | Action
- Pagination shows total count (e.g. '7 patients') and pages of 20 records each

5.2 Search Test Cases

1	**Search by name**	Type 'Connor' → only patients with Connor in name appear
2	**Search by Patient ID**	Type 'P2026001' → exact patient appears
3	**Search by phone**	Type '555-200' → patients with that phone prefix appear
4	**Search by email**	Type 'sarah.connor' → matching patient appears
5	**Clear search**	Click ✕ on search box → full list restored
6	**No results**	Type 'ZZZNOMATCH' → table shows 'No patients found'
7	**Live debounce**	Type quickly — results update 300ms after last keystroke, not per character

5.3 Filter Test Cases

1	**Status: Inactive**	Select Inactive from Status dropdown → only deactivated patients shown
2	**Status: clear**	Click ✕ on Status filter → all statuses shown
3	**Gender: Female**	Select Female → only female patients
4	**Blood Group: B+**	Select B+ → only B+ patients
5	**Combine filters**	Search 'Connor' + Gender 'Female' → only female Connors
6	**Clear all filters**	Click ✕ on each filter → full active list restored

5.4 Navigation from List

- Click any patient row → navigates to /patients/{id} (patient detail page)
- RECEPTIONIST/ADMIN rows show Edit button in Action column
- DOCTOR/NURSE rows show no Edit button in Action column

FEATURE 4

6. Patient Profile View

The patient detail page displays all patient information organised in four information cards plus a Record Information card. Every access to this page generates an immutable HIPAA READ audit log entry.

6.1 Test Steps

1	**Navigate**	From Patient List, click any patient row
2	**Personal Info card**	Verify: Status badge (green=Active / red=Inactive), DOB, Age (auto-calculated), Gender, Blood Group
3	**Contact Info card**	Verify: Phone, Email, Address, City, State, ZIP (shows '—' if empty)
4	**Emergency Contact**	Verify: Contact Name, Phone, Relationship (shows '—' if empty)
5	**Medical Info card**	Verify: Known Allergies, Chronic Conditions (shows '—' if empty)
6	**Record Info card**	Verify: Registered By, Registered At, Last Updated By, Last Updated At
7	**Back to List**	Click Back to List → returns to /patients

6.2 RBAC — Action Buttons by Role

Role	Edit Patient Button	Deactivate/Activate Button
RECEPTIONIST	■ Visible	■ Hidden
DOCTOR	■ Hidden	■ Hidden
NURSE	■ Hidden	■ Hidden
ADMIN	■ Visible	■ Visible

6.3 Status Badge Colour Coding

- ACTIVE patient → green dot + 'Active' text (never colour alone — WCAG 2.1 AA compliant)
- INACTIVE patient → red dot + 'Inactive' text

6.4 404 Page

1	**Navigate to invalid ID**	Visit http://localhost/patients/INVALID999
2	**Observe error**	Error alert: patient not found / 404 response from API

FEATURE 5

7. Edit Patient

RECEPTIONIST and ADMIN roles can update any patient's demographic information. The edit form is pre-populated with current data, applies the same validation rules as registration, and records who made the update and when.

7.1 Prerequisites

Login as **receptionist1 / RECEPTIONIST** or **admin / ADMIN**.

7.2 Happy Path — Edit Patient

1	**Open patient detail**	Navigate to any patient's detail page (click row from list)
2	**Click Edit Patient**	Click the Edit Patient button → navigates to /patients/{id}/edit
3	**Verify pre-fill**	Confirm all fields are pre-populated with current patient data
4	**Modify fields**	Change City to 'Springfield', add ', Diabetes' to Chronic Conditions
5	**Submit**	Click Save Changes button
6	**Verify success**	Toast: 'Patient updated'
7	**Verify redirect**	Returns to patient detail page
8	**Verify changes saved**	Updated City and Conditions visible in detail page
9	**Verify audit fields**	Record Info card shows: Last Updated By = your username, Last Updated At = now

7.3 Read-Only Fields (not editable)

- Patient ID — shown in page subtitle only, not in form
- Registration date — not present in edit form, only in detail view

7.4 Validation on Edit Form

1	**Clear First Name**	Delete first name, tab away → red error: 'First name is required'
2	**Invalid email**	Change email to 'bad-email' → red error: 'Invalid email address'
3	**Cancel**	Click Cancel → returns to detail page with no changes saved

FEATURE 6

8. Deactivate / Activate Patient

ADMIN-only feature. Patient records are never permanently deleted — they are deactivated (soft delete). Deactivation requires a confirmation modal. Activation is immediate. Both actions are fully audit-logged.

8.1 Prerequisites

Login as **admin / ADMIN**.

8.2 Deactivate a Patient

1	**Open an ACTIVE patient**	Navigate to any patient with green 'Active' status badge
2	**Click Deactivate**	Click the red **Deactivate Patient** button
3	**Confirmation modal**	Modal appears: 'Are you sure you want to deactivate this patient?'
4	**Click Cancel**	Modal closes, nothing changes — patient remains ACTIVE
5	**Click Deactivate again**	Click Deactivate Patient → modal appears again
6	**Confirm**	Click Deactivate in modal → action executes
7	**Verify status**	Status badge changes: green 'Active' → red 'Inactive'
8	**Verify button change**	Button label changes to 'Activate Patient'
9	**Toast message**	Success toast: 'Patient deactivated'
10	**Verify list filter**	Return to list (Active filter) → deactivated patient no longer appears

8.3 Activate a Patient

1	**Open an INACTIVE patient**	From list: clear Status filter → find patient with red 'Inactive' badge → click row
2	**Click Activate**	Click **Activate Patient** button — no confirmation needed
3	**Verify instant action**	No modal — activates immediately
4	**Verify status**	Status badge changes: red 'Inactive' → green 'Active'
5	**Toast message**	Success toast: 'Patient activated'
6	**Verify in list**	Return to list (Active filter) → patient appears again

8.4 RBAC — Deactivate Button Hidden for non-ADMIN

- RECEPTIONIST — Deactivate/Activate button not visible on detail page
- DOCTOR — Deactivate/Activate button not visible on detail page
- NURSE — Deactivate/Activate button not visible on detail page
- Direct API call PATCH /api/v1/patients/{id}/status as RECEPTIONIST → 403 Forbidden

FEATURE 7

9. Role-Based UI Visibility

The UI adapts dynamically based on the logged-in user's role. Buttons and controls are hidden — not just disabled — for unauthorized roles. This enforces the minimum necessary principle at the presentation layer (API layer enforces it independently).

UI Element / Page	RECEPTIONIST	DOCTOR	NURSE	ADMIN
Register New Patient button (List page)	■ Visible	■ Hidden	■ Hidden	■ Visible
Edit button per row (List page)	■ Visible	■ Hidden	■ Hidden	■ Visible
Edit Patient button (Detail page)	■ Visible	■ Hidden	■ Hidden	■ Visible
Deactivate Patient button (Detail page)	■ Hidden	■ Hidden	■ Hidden	■ Visible
Activate Patient button (Detail page)	■ Hidden	■ Hidden	■ Hidden	■ Visible
Patient list — view/search	■	■	■	■
Patient detail — view	■	■	■	■

How to Verify

1	**Login as RECEPTIONIST**	Verify: Register + Edit buttons visible. No Deactivate button.
2	**Logout → Login DOCTOR**	Verify: No Register, no Edit, no Deactivate buttons. Only view access.
3	**Logout → Login NURSE**	Verify: Same as DOCTOR — read-only view.
4	**Logout → Login ADMIN**	Verify: All buttons visible including Deactivate/Activate.

FEATURE 8

10. Sign Out

1	**Locate Sign Out button**	Top-right corner of the app header — visible when logged in
2	**Click Sign Out**	Click the Sign Out button (door icon)
3	**Token cleared**	localStorage 'token' key is removed
4	**Verify redirect**	Navigated to /login page
5	**Verify session ended**	Header no longer shows username or role badge
6	**Test protected URL**	Visit http://localhost/patients — API returns 401, error displayed
7	**Verify login required**	Must log in again to access any patient data

- JWT token removed from localStorage on sign out
- Subsequent API calls return 401 Unauthorized
- Login page shown — user must re-authenticate

FEATURE 9 — HIPAA

11. HIPAA Audit Trail

Every access to patient data generates an immutable audit log entry stored in PostgreSQL. The audit trail is enforced at the AOP layer (Spring @AfterReturning aspect) using a REQUIRES_NEW transaction to guarantee the log is written even after read-only service methods. Database-level triggers prevent any UPDATE or DELETE of audit records.

11.1 Audit Log Schema

Field	Type	Content
user_id	VARCHAR(100)	UUID of the logged-in user
username	VARCHAR(100)	Display username (e.g. receptionist1)
user_role	VARCHAR(50)	Role at time of action (RECEPTIONIST, DOCTOR, etc.)
action	VARCHAR(20)	READ | CREATE | UPDATE | DEACTIVATE | ACTIVATE
patient_id	VARCHAR(10)	Patient ID (e.g. P2026001) — no PHI stored
ip_address	VARCHAR(45)	Request IP address
occurred_at	TIMESTAMPTZ	UTC timestamp of the action

11.2 Actions That Generate Audit Entries

User Action in UI	API Endpoint	Audit Action
Register new patient	POST /api/v1/patients	CREATE
View patient detail page	GET /api/v1/patients/{id}	READ
Submit edit form	PUT /api/v1/patients/{id}	UPDATE
Confirm deactivation	PATCH /api/v1/patients/{id}/status	DEACTIVATE
Click Activate Patient	PATCH /api/v1/patients/{id}/status	ACTIVATE

11.3 How to Verify Audit Logs

Run the following command in your terminal after performing UI actions:

```
docker exec hospital-postgres psql -U hospital_user -d hospital_db \ -c "SELECT
to_char(occurred_at,'HH24:MI:SS') AS time, action, patient_id, username, user_role \ FROM audit_logs
ORDER BY occurred_at DESC LIMIT 15;"
```

11.4 Immutability Test

1	**Attempt UPDATE**	Run: UPDATE audit_logs SET action='TAMPERED' WHERE id=1;

2	**Verify rejection**	PostgreSQL raises: 'audit_logs are immutable (HIPAA requirement)'
3	**Attempt DELETE**	Run: DELETE FROM audit_logs WHERE id=1;
4	**Verify rejection**	Same immutability error — row cannot be deleted

HIPAA Note: Audit logs are retained for a minimum of 6 years per HIPAA §164.530(j). Records older than 6 years are moved to audit_logs_archive (also immutable) to keep the live table lean. No PHI is stored in the audit log — only patient IDs.

12. Non-Functional Requirements Coverage

NFR Category	Requirement	Implementation	Status
Performance	Search results < 2 seconds	Indexed search columns + debounce	■
Performance	Registration < 3 seconds	Async API + optimistic UI	■
Security	AES-256 encryption at rest for PHI	AttributeConverter on all PHI fields	■
Security	JWT-based stateless auth	HMAC-SHA256 JWT, 8hr expiry	■
Security	RBAC enforced at API level	Spring Security @PreAuthorize	■
Security	Audit logs immutable	DB trigger blocks UPDATE/DELETE	■
Security	Audit log 6-year retention	Archive table + Flyway migration	■
Usability	Inline validation on blur	Zod + react-hook-form mode:onBlur	■
Usability	300ms search debounce	useDebounce hook	■
Usability	Success notifications 5s auto-dismiss	Ant Design notification duration=5	■
Usability	Confirmation for destructive actions	ConfirmModal component	■
Usability	WCAG 2.1 AA — colour + text labels	StatusBadge: colour + text always	■
Data Integrity	Unique Patient ID	DB sequence + PatientIdGenerator	■
Data Integrity	Future DOB blocked	DatePicker disabledDate + Zod refine	■
Data Integrity	Records never permanently deleted	Soft delete via status only	■
Responsive	Mobile 320px to desktop 2560px	Ant Design grid system (Col/Row)	■

13. Complete Test Execution Checklist

Work through this checklist in order. Each item corresponds to a feature section above.

Login

- ■ Login as receptionist1 / RECEPTIONIST → blue badge in header
- ■ Login as dr.smith / DOCTOR → green badge
- ■ Login as nurse.jane / NURSE → cyan badge
- ■ Login as admin / ADMIN → red badge

Registration

- ■ Register Sarah Connor (all fields filled) → success toast + redirect
- ■ Blur empty First Name field → inline error shown
- ■ Enter invalid email → inline error shown
- ■ Try future date of birth → date picker blocks it
- ■ Register second patient with same phone → yellow duplicate warning
- ■ Login as DOCTOR → Register button not visible

Patient List

- ■ Active patients shown by default
- ■ Search by name → results filter live
- ■ Search by Patient ID → exact match found
- ■ Search by phone number → results filter
- ■ Apply Status = Inactive filter
- ■ Apply Gender filter
- ■ Apply Blood Group filter
- ■ Combine search + filter
- ■ Search with no match → 'No patients found'
- ■ Click patient row → navigates to detail

Profile View

- ■ All 5 cards visible: Personal, Contact, Emergency, Medical, Record Info
- ■ Active patient shows green status badge with 'Active' text
- ■ Inactive patient shows red status badge with 'Inactive' text
- ■ As DOCTOR/NURSE: no Edit or Deactivate buttons
- ■ As RECEPTIONIST: Edit button visible, no Deactivate button
- ■ As ADMIN: both Edit and Deactivate buttons visible
- ■ Back to List button returns to /patients

Edit Patient

- ■ Open edit form → all fields pre-populated
- ■ Patient ID not editable (shown in header only)
- ■ Change address and submit → success toast
- ■ Detail page shows updated data
- ■ Record Info shows Last Updated By and timestamp
- ■ Cancel discards changes

Status Mgmt

- ■ As ADMIN: Deactivate Patient → confirmation modal appears
- ■ Cancel modal → patient stays ACTIVE
- ■ Confirm deactivation → status changes to INACTIVE
- ■ INACTIVE patient not in Active filtered list
- ■ Activate patient → no confirmation, immediate
- ■ Status changes back to ACTIVE
- ■ As RECEPTIONIST: no Deactivate/Activate button visible

Audit Trail

- ■ After registration: CREATE entry in audit_logs
- ■ After viewing detail: READ entry in audit_logs
- ■ After editing: UPDATE entry in audit_logs
- ■ After deactivating: DEACTIVATE entry in audit_logs
- ■ After activating: ACTIVATE entry in audit_logs
- ■ Attempt UPDATE on audit_logs → immutability error

Sign Out

- ■ Click Sign Out → redirected to /login
- ■ Visit /patients after sign out → 401 error shown

14. Known Limitations & Out-of-Scope Items

The following items are **intentionally not implemented** in the Patient Module — they are explicitly deferred to other modules as defined in PRD §9 and §10:

Item	Owner Module	PRD Reference
Multi-Factor Authentication (MFA)	Auth Module	§7 HIPAA Technical Safeguards
15-minute session timeout	Auth Module	§7 HIPAA Technical Safeguards
Production HTTPS/TLS termination	Infrastructure/Nginx	§6 Security NFR
User management (create/assign roles)	Auth Module	§10 Dependencies
Appointment scheduling	Appointment Module	§9 Out of Scope
EMR / clinical notes	EMR Module	§9 Out of Scope
Billing and insurance	Billing Module	§9 Out of Scope
Reporting and analytics	Reporting Module	§9 Out of Scope

■■ **Production Deployment Blocker:** This module MUST NOT be deployed to production without a HIPAA-compliant Auth Module providing MFA and session timeout. The dev-login endpoint (DevAuthController) issues JWTs without MFA and must be removed before production deployment.

Document prepared by the Ai Nexus Engineering Team · Patient Module v1.1.0 · February 2026 · CONFIDENTIAL — Internal Use Only