Assignment 3 (3 Hours) — Clinic System Enhancement with RBAC + Audit + Approvals

1) Objective

Enhance the existing Clinic/Diagnostic system by introducing **Role-Based Access Control (RBAC)** and operational controls typically required in healthcare workflows—such as **approval/review**, **audit logs**, and **controlled access to uploaded documents and visit data**.

Students are expected to reuse what they already built and extend it cleanly.

2) Roles & Access Rules (RBAC)

Roles to implement

1. **Admin**
2. **Doctor**
3. **LabTechnician**
4. **Receptionist**

Role must be stored per user and included in JWT claims (or equivalent).
All protected routes must enforce role-based authorization.

3) Feature Requirements

A) Admin Controls (User & Role Management)

Admin can:

1. View list of users
2. Change a user role
3. Activate/Deactivate users (blocked users cannot login)

Minimum endpoints

- GET /api/admin/users
- PUT /api/admin/users/{id}/role
- PUT /api/admin/users/{id}/status (active/inactive)

B) Visit Workflow Status + Approval

Each Visit must have a workflow status:

- **Draft** (created but not finalized)
- **Submitted** (ready for doctor review)
- **Approved** (doctor approved)

- **Rejected** (doctor rejected with reason)

Rules

- Receptionist can create visits in **Draft**
- Receptionist can move Draft → Submitted
- Only Doctor can Approve/Reject Submitted visits
- Approved/Rejected visits must not be editable by Receptionist (read-only)
- Rejection must store a **reason/comment**

Minimum endpoints

- PUT /api/visits/{id}/submit
- PUT /api/visits/{id}/approve
- PUT /api/visits/{id}/reject (body includes reason)

C) File Access Control (Sensitive Document Rules)

Uploaded files are sensitive medical data. Add access rules:

- **Receptionist**
    - Can upload files only for visits in Draft/Submitted
    - Can view file list for visits they created (optional) OR all visits (your choice—document it)
    - Cannot delete files after visit is Approved
- **LabTechnician**
    - Can upload lab reports only when visit is Submitted
    - Can delete/re-upload files only before approval
- **Doctor**
    - Can view/download all files
    - Cannot upload (optional) OR can upload “Doctor Notes” (bonus)
- **Admin**
    - Full access

You decide the exact rules where optional, but RBAC must be clearly implemented and documented.

Minimum endpoints

- POST /api/visits/{visitId}/files (multiple files)
- DELETE /api/files/{fileId} (must enforce role + visit status rules)

D) Audit Log (Who did what, when)

Maintain an audit log table/collection capturing:

- Action (e.g., VISIT_CREATED, VISIT_SUBMITTED, FILE_UPLOADED, VISIT_APPROVED, ROLE_CHANGED)
- Entity type (User / Patient / Visit / File)
- Entity id
- Performed by (user id)
- Timestamp
- Optional remarks (e.g., rejection reason)

Audit log must be written for:

- Registration/login optional (bonus)
- Visit create/update/submit/approve/reject
- File upload/delete
- Admin role/status changes

Minimum endpoint

- GET /api/audit?fromDate=&toDate=&action=&userId=
  Access:
    - Admin: all logs
    - Others: only their own logs (or restrict entirely; document it)

E) Enhanced Summary View (Role-aware)

Modify your visit-summary endpoint to behave differently per role:

- Receptionist: sees visits they created (or only Draft/Submitted—your choice)
- Doctor: sees Submitted visits first (sorted), then others
- LabTechnician: sees Submitted visits that have pending files OR missing lab files (optional logic)
- Admin: sees everything

Minimum endpoint

- GET /api/visit-summary (same route, different results based on role)

4) Required Validations & Constraints

- Status transition validation (cannot approve Draft, cannot submit Approved, etc.)
- Consistent HTTP codes:
    - 401 unauthenticated, 403 unauthorized, 409 invalid state transition (or 400)
- For rejection: reason is required
- Deactivated user cannot login

5) Deliverables

1. Updated source code
2. DB migration / schema updates (roles, status, audit logs)
3. README must include:
    - Roles and permissions matrix
    - How to test each role (sample users / credentials or steps)
    - API list + workflow steps (Draft → Submitted → Approved/Rejected)
4. (Optional) Postman collection / Swagger

6) Evaluation Criteria (High-level)

- Correct RBAC enforcement (not just UI hiding)
- Correct workflow transitions + lock-down after approval
- Audit logging completeness and correctness
- File access rules implemented reliably
- Clean design, readable code, good error handling

Bonus (If time remains)

- Password policy + account lockout after failed attempts
- Refresh tokens
- Pagination + filtering improvements in summary/audit
- Attach Doctor Notes (separate file type/category)

Roles & Permissions Matrix (Assignment 3)

Legend: ✅ Allowed | ❌ Not allowed | ⚠️ Allowed with conditions

1) Authentication & User Admin

Feature	Admin	Doctor	LabTechnician	Receptionist
Register	✅	✅	✅	✅
Login (JWT)	✅	✅	✅	✅
View Users	✅	❌	❌	❌
Change User Role	✅	❌	❌	❌
Activate/Deactivate User	✅	❌	❌	❌

Rules

- Deactivated user: ❌ login (return 403 with message “User inactive”)

2) Patients

Feature	Admin	Doctor	LabTechnician	Receptionist
Create Patient	✅	⚠️ (optional)	❌	✅
Update Patient	✅	⚠️ (optional)	❌	✅
Delete Patient (soft)	✅	❌	❌	⚠️ (optional)
View Patient List/Details	✅	✅	✅	✅

3) Visits (with Workflow Status)

Statuses: **Draft → Submitted → Approved/Rejected**

Feature	Admin	Doctor	LabTechnician	Receptionist
Create Visit	✅	⚠️ (optional)	❌	✅ (Draft only)
Update Visit	✅	⚠️ (notes only optional)	❌	⚠️ Only if Draft
Delete/Cancel Visit (soft)	✅	❌	❌	⚠️ Only if Draft
Submit Visit	✅	❌	❌	✅ (Draft → Submitted)
Approve Visit	✅	✅ (Submitted → Approved)	❌	❌
Reject Visit	✅	✅ (Submitted → Rejected + reason)	❌	❌
View Visit List/Details	✅	✅	✅	✅

Status rules

- Draft: editable by Receptionist/Admin
- Submitted: editable by Admin only (optional) but **not Receptionist**
- Approved/Rejected: read-only (except Admin can correct if you allow; document it)

4) Tests on Visit (Selected via cascading)

Feature	Admin	Doctor	LabTechnician	Receptionist
Add/Remove Tests	✅	⚠️ (optional)	❌	⚠️ Only if Draft
View Tests	✅	✅	✅	✅

5) Files (Multiple upload + access rules)

Feature	Admin	Doctor	LabTechnician	Receptionist
Upload Files	✅	⚠ (optional doctor notes)	⚠ Only if Submitted	⚠ Draft/Submitted
View File List	✅	✅	✅	⚠ (own visits or all—choose)
Download/View File	✅	✅	✅	⚠ (own visits or all—choose)
Delete File	✅	⚠ Only before Approved	⚠ Only before Approved	⚠ Only before Approved and visit not Approved

Recommended enforcement (simple & strict):

- No one except Admin can delete files after approval.
- LabTechnician upload only in Submitted.
- Receptionist upload only in Draft/Submitted.

6) Audit Logs

Feature	Admin	Doctor	LabTechnician	Receptionist
Write Audit Logs	✅	✅	✅	✅
View Audit Logs	✅ (all)	⚠ (own only)	⚠ (own only)	⚠ (own only)

Sample Postman Test Flow (10-minute evaluator run)

Seed / Setup Assumptions

Create 4 users:

- admin@clinic.com (Admin)
- doctor@clinic.com (Doctor)
- lab@clinic.com (LabTechnician)
- recep@clinic.com (Receptionist)

Also seed master data: departments/categories/tests.

Step 1 — Register or Login 4 roles

1. POST /api/auth/login for each user
   Save tokens:

- TOKEN_ADMIN
- TOKEN_DOCTOR
- TOKEN_LAB
- TOKEN_RECEP

Step 2 — Admin role/status checks

2. GET /api/admin/users (Admin token) → should succeed
3. Same endpoint with Doctor token → should return **403**
4. PUT /api/admin/users/{id}/status set lab user inactive
5. Attempt login with inactive lab user → **403**

(Re-activate lab user for next steps)

Step 3 — Receptionist creates patient + visit (Draft)

6. POST /api/patients (Receptionist)
7. POST /api/visits (Receptionist) with:

- patientId
- visitDate
- tests: [{departmentId, categoryId, testId}, ...] (1–2 tests)
  Expected: Visit created in **Draft**

8. POST /api/visits/{visitId}/files upload 2 files (Receptionist)
   Expected: ✅ allowed in Draft

Step 4 — Workflow submit and locking

9. PUT /api/visits/{visitId}/submit (Receptionist)
   Expected status: **Submitted**
10. PUT /api/visits/{visitId} try update (Receptionist)
    Expected: **403 or 409** (blocked since not Draft)

Step 5 — Lab uploads in Submitted

11. POST /api/visits/{visitId}/files upload 1 file (Lab token)
    Expected: ✅ allowed only in Submitted
12. Try lab upload in Draft visit (create another draft quickly)
    Expected: **403 or 409**

Step 6 — Doctor approves/rejects

13. PUT /api/visits/{visitId}/approve (Doctor)
    Expected: status **Approved**
14. Try DELETE /api/files/{fileId} using Receptionist or Lab after approval
    Expected: **403/409** (blocked)
15. GET /api/visits/{visitId}/full (Doctor)
    Expected: patient + visit + tests + files

Step 7 — Summary behavior differs by role

16. GET /api/visit-summary

- Receptionist: should see own visits (or per your rule)
- Doctor: should see Submitted on top (and/or all)
- Admin: sees all
- Lab: sees Submitted visits (or those needing lab files if implemented)

Step 8 — Audit verification

17. GET /api/audit (Admin) shows actions:

- VISIT_CREATED
- FILE_UPLOADED
- VISIT_SUBMITTED
- FILE_UPLOADED (lab)
- VISIT_APPROVED

18. GET /api/audit (Receptionist) shows only their actions (if own-only rule)

Suggested Minimal Request Bodies (for consistency)

Create Visit (example)

```
{
  "patientId": 101,
  "visitDate": "2026-01-02",
  "notes": "Complaints of chest pain",
  "tests": [
    { "departmentId": 1, "categoryId": 1, "testId": 1 },
    { "departmentId": 1, "categoryId": 2, "testId": 5 }
  ]
}
```

Reject Visit (example)

```
{
  "reason": "Missing required lab report / incomplete details"
}
```