كلية العلوم

قسـم الانظمة الطبية الذكية

Lec 9 & 10

Project Management Concepts and Methods

Subject: Electronic Health Records

Level: fourth

Lecturer: Asst. Lecturer Qusai AL-Durrah

Why Does EHR Implementation Fail?

The Uncomfortable Truth

EHR projects fail far more often due to **management problems** than technical ones. A perfectly coded database means nothing if the project runs over budget, exceeds scope, or loses stakeholder trust.

Good project management controls: **scope, time, cost, quality, risks,** and **communication** — all simultaneously.

Where We Are Now

Planning was covered in **Lecture 3 & 4** . Today we shift from planning to **execution and control** — applying formal PM tools directly to our running `EHR_DB` project in SSMS.

Today's tools: WBS · RACI · Risk Register · Change Control · Gantt Schedule

Learning Outcomes

01

Define a Project

Understand what makes EHR implementations uniquely high-risk projects in healthcare IT.

02

Build a WBS

Decompose an EHR deliverable into assignable, estimable tasks using a Work Breakdown Structure.

03

Assign Roles via RACI

Clearly define who is Responsible, Accountable, Consulted, and Informed for each task.

04

Risk Register & Change Control

Create a risk mitigation plan and apply change control to prevent scope creep.

05

Build a Schedule

Define milestones and task dependencies to produce a realistic implementation timeline.

PART A — CORE CONCEPTS

What Is a Project in Healthcare IT?

Temporary & Unique

A project is a **temporary effort** with a defined start and end, producing a unique outcome — such as an EHR module or a full clinical system deployment.

Not Operations

Projects **end**; operations **continue**. Once the EHR system is live, the project closes — but running and maintaining it is ongoing operations.

Multi-Dimensional Deliverables

EHR deliverables include **software + workflow redesign + user training + governance policies**. Technical success alone is not enough.

The Triple Constraint — The Iron Triangle

Three Forces in Constant Tension

⏱️ Time

Schedule and deadlines

💰 Cost

Budget and resources

📋 Scope

Features and requirements

Quality sits at the center — it is the outcome of how well these three are balanced.

EHR_DB Example: Adding `LabResults` + auditing late in the semester increases scope → you must either extend the deadline, reduce other features, or accept lower quality.

Project Stakeholders in an EHR Context

Stakeholders define requirements, provide feedback, and ultimately **approve acceptance**. Ignoring any group leads to failed adoption, even with perfect code.

Sponsor

Holds final decision authority. Approves budget, scope changes, and sign-off.

Clinical Users

Doctors and nurses — define clinical workflow requirements and validate usability.

Admin Users

Registration staff who handle patient intake. Define data entry workflows.

IT Team

Manages the database, security, infrastructure, and system integrations.

Patients

Have rights to privacy and data access. HIPAA compliance protects their data.

PART B — PM METHODS APPLIED

Case Study: Our Mini EHR_DB Project

What We've Built So Far

Core Tables

`Patients`, `Encounters`, `Diagnoses`, `MedicationOrders` — all with PK/FK constraints enforcing relational integrity.

Integrity Rules

CHECK constraints, NOT NULL, DEFAULT values — preventing unsafe or inconsistent data from entering the system.

Next Deliverables (Our Project Scope)

- **LabOrders** — new table with FK to Encounters
- **LabResults** — FK to LabOrders, CHECK on result ranges
- **Test data** — valid inserts + deliberate violations
- **Verification queries** — JOINs across all tables
- **Documentation** — 1-page rationale + packaged SQL script

These technical tasks become the input for our WBS, RACI, and schedule.

WBS: Work Breakdown Structure (EHR_DB Extension)

A WBS decomposes the project into tasks you can **assign, estimate, and schedule**. Every leaf node should be independently executable by one person.

1 **Requirements**
Define lab workflow: what fields, data types, and constraints are needed for `LabOrders` and `LabResults`.

2 **ERD Update**
Update the Entity-Relationship Diagram to show `LabOrders` → `LabResults` relationships (1-to-N).

3 **Implement Tables**
Write and execute `CREATE TABLE` with PK, FK, NOT NULL, CHECK, and DEFAULT.

4 **Valid Test Data**
Insert valid rows that satisfy all constraints across all related tables.

5 **Violation Tests**
Insert deliberate FK violations and CHECK failures — confirm the DB rejects them correctly.

6 **Verification Queries**
Write JOIN queries across `Patients → Encounters → LabOrders → LabResults` to validate data integrity.

7 **Documentation**
Write a 1-page design rationale explaining each constraint choice. Package the final SQL script.

Solved Example — Building a WBS

Scenario: The project manager asks you to extend the EHR_DB by adding an `Appointments` module. Build a WBS with at least 5 tasks.

#	WBS Task	Description	Predecessor
1	Requirements gathering	Identify fields: AppointmentID, PatientID, DoctorID, Date, Status	—
2	ERD update	Add Appointments entity; draw FK to Patients & Doctors	Task 1
3	Implement table in SSMS	CREATE TABLE Appointments with PK/FK/CHECK on Status values	Task 2
4	Insert valid test data	At least 5 rows with valid PatientIDs and future dates	Task 3
5	Violation tests	Test invalid Status value; test FK to non-existent PatientID	Task 4
6	Documentation	Write rationale for each constraint; package SQL script	All previose Tasks

Key takeaway: Each task has a clear predecessor — this chain of dependencies defines the **critical path**. No testing before build; no build before ERD approval.

Estimation, Dependencies & the Critical Path

Estimating Effort per WBS Task

WBS Task	Estimated Hours	Depends On
Requirements	1 hr	—
ERD Update	1 hr	Requirements
Implement Tables	2 hrs	ERD Update
Valid Test Data	1 hr	Tables Created
Violation Tests	1 hr	Test Data
Documentation	1 hr	All above

Milestones

🏁 M1: Design Approved

ERD reviewed and signed off — no more structural changes.

🏁 M2: Build Complete

All tables created and constraints verified in SSMS.

🏁 M3: Tests Passed

All violation tests confirmed to fail correctly. Queries return expected data.

Critical Path: The longest chain of dependent tasks. Delay any task on the critical path → delay the entire project.

Responsibility Assignment (RACI) Matrix

A Responsibility Assignment (RACI) Matrix is a project management tool that maps project tasks (work packages) to the roles involved, making it explicit who does the work, who owns the decision, who provides input, and who must be informed.

R — Responsible

The person(s) who **perform the task** (do the work).

A — Accountable

The **single role/person ultimately answerable** and approves/signs off (owns the outcome). Best practice: one A per task.

C — Consulted

People who provide **expert input** before the task is completed (two-way communication).

I — Informed

People who need to be **notified of results/updates** (one-way communication).

Why it matters in EHR projects: it prevents “everyone thought someone else was doing it,” especially for safety-critical items like access control, audit logging, and data migration.

Responsibility Assignment (RACI) Matrix

A RACI (responsible, accountable, consulted and informed) matrix eliminates ambiguity. Every task has **exactly one Accountable (A)** — the person answerable if it is not done. Multiple people may be Responsible (R), but only one owns accountability.

Task	R — Does it	A — Owns it	C — Advises	I — Informed
Approve lab workflow requirements	Clinical Lead	Project Sponsor	IT Architect	All Team
Create tables + constraints in SSMS	DB Developer	IT Lead	Clinical Lead	Project Sponsor
Write test inserts + violation tests	QA Reviewer	IT Lead	QA Reviewer	DB Developer
Review script quality	QA Reviewer	QA Lead	IT Lead	IT Architect
Sign off acceptance criteria	Project Sponsor	Project Sponsor	Clinical Lead, IT Lead	All Stakeholders

Solved Example — Building a RACI Matrix

Scenario: Your team is adding the `Appointments` module. Team roles: *Student Developer, Team Lead, Clinical Advisor, QA Tester, Course Instructor*. Complete the RACI matrix below.

Task	R	A	C	I
Gather appointment requirements	Student Dev	**Team Lead**	Clinical Advisor	Instructor
Design ERD update	Student Dev	**Team Lead**	Clinical Advisor	QA Tester
Implement Appointments table	Student Dev	**Team Lead**	Clinical Advisor	Clinical Advisor
Test valid + violation inserts	QA Tester	**QA Tester**	Team Lead	Student Dev
Review and approve SQL script	Team Lead	**Team Lead**	QA Tester	Instructor
Final submission sign-off	Team Lead	**Instructor**	Team Lead	All

Notice: The Student Dev is Responsible for implementation but the Team Lead remains Accountable — a key RACI rule: the person doing the work is rarely the same as the person accountable for it.

Quality Management — Definition of Done

A **Definition of Done (DoD)** sets objective, binary criteria. Either the deliverable meets them — or it doesn't. This removes subjective judgment and ensures consistency across team members.

DoD for EHR_DB Lab Extension

✅ Named Constraints

All CHECK, FK, and DEFAULT constraints have explicit names (e.g., `CK_Lab_Status`).

✅ FK Integrity

No orphan rows exist. Every `LabResult` references a valid `LabOrder`.

✅ CHECK Constraints Active

Invalid values (e.g., negative lab values) are blocked — confirmed by violation tests.

✅ Clean-Run Script

The SQL script runs top-to-bottom on a **fresh database** with zero errors.

Acceptance Tests

- At least **2 deliberate FK violations** must produce an error message
- At least **2 CHECK constraint violations** must be rejected
- A JOIN query across all 4 tables must return correct data
- Script must succeed on a DB with only the base tables pre-existing

PART C — RISK, CHANGE & COMMUNICATION

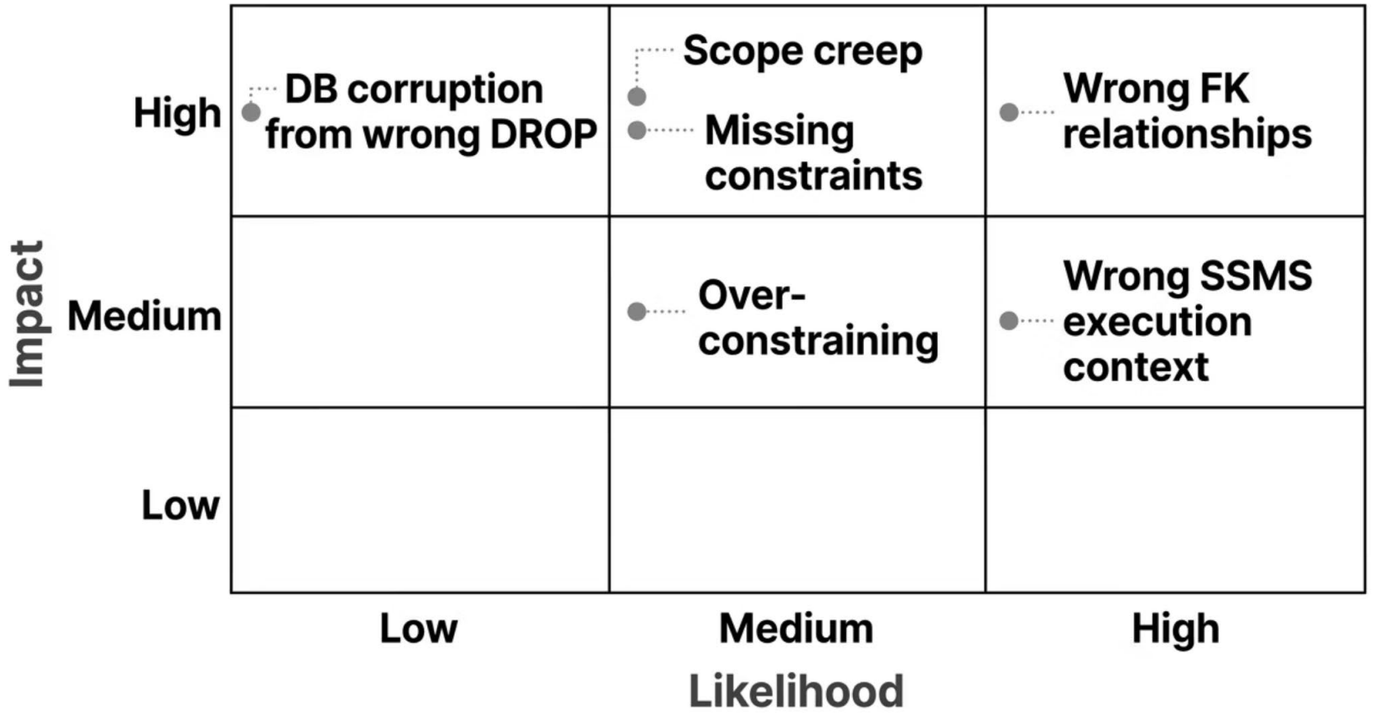
Risk Register — Top EHR_DB Risks

A risk register captures: **Risk Description** · **Likelihood** · **Impact** · **Mitigation Strategy** · **Owner**. Document it before problems occur, not after.

Risk	Likelihood	Impact	Mitigation	Owner
Wrong FK relationship design	High	High	ERD peer review before any CREATE TABLE	IT Lead
Over-constraining (blocks real data)	Medium	Medium	Use realistic CHECK ranges; test with actual clinical values	DB Developer
Missing constraints (silent data corruption)	Medium	High	Use constraint checklist at table creation	DB Developer
Wrong execution context in SSMS	High	Medium	Always begin script with `USE EHR_DB; GO`	DB Developer
Scope creep (extra tables/features added)	Medium	High	Apply formal change control for all new requirements	Project Sponsor

Risk Matrix — Prioritization Grid

The risk matrix visualizes **which risks to handle first**. High Impact + High Likelihood = immediate action required. Low Impact + Low Likelihood = monitor only.



Handle First

Wrong FK design (High/High) and Wrong SSMS execution context

Monitor Closely

Scope creep and Missing constraints — medium likelihood but high downstream damage.

Watch Only

Over-constraining — medium likelihood, medium impact; monitor and adjust CHECK ranges if real data is blocked.

Communication Plan

PM success depends on **proactive communication**. Problems discovered late are always more expensive than problems surfaced early through regular status updates.

📋 Weekly Status Report

Progress vs. plan, blockers encountered, next steps planned. Circulated to all stakeholders every week.

📝 Decision Log

All requirements decisions and relationship design choices are recorded with rationale and date. Prevents "who said what" disputes.

🐛 Issue Log

Every error found during testing is logged: description, root cause, resolution, and who fixed it. Forms part of the final documentation.

Key principle: Silent work is invisible work. A PM cannot manage what they cannot see. Regular communication turns individual efforts into a coordinated project.

PART D — SCHEDULE & DELIVERABLES

Timeline — Gantt-Style Implementation Schedule

Dependencies govern the sequence. Testing cannot begin before tables are built; queries cannot be verified before test data exists. The schedule below enforces these logical constraints.

Day	Milestone	Tasks	Dependency
Day 1	🏁 Design Approved	Finalize requirements; complete ERD update for LabOrders + LabResults	None (start here)
Day 2	Build Phase 1	Implement `LabOrders` table: PK, FK to Encounters, NOT NULL, CHECK constraints	ERD Approved
Day 3	Build Phase 2	Implement `LabResults` table: FK to LabOrders, CHECK on result values, DEFAULT	LabOrders complete
Day 4	🏁 Build Complete	Insert valid test data across all tables; execute deliberate violation tests	Both tables built
Day 5	🏁 Tests Passed	Write verification JOIN queries; finalize documentation; package SQL script	All tests confirmed

Critical path: Day 1 → Day 2 → Day 3 → Day 4 → Day 5. Any delay on this chain delays final submission.

Deliverables Checklist — What to Submit & Produce

A deliverable is only complete when it meets the **Definition of Done**. Use this checklist to self-assess before submission. Reproducibility is non-negotiable — every script must run on a clean database.

1

Updated ERD Diagram

Includes `LabOrders` and `LabResults` with correct 1-to-N cardinality notations and all FK relationships shown.

2

Full SQL Script

One packaged `.sql` file containing: `CREATE TABLE`, named constraints, valid `INSERT` statements, and deliberate violation tests.

3

Design Rationale

A 1-page document explaining *why* each constraint exists — what clinical or data integrity rule it enforces.

4

Test Evidence

Screenshots or copied SSMS error messages confirming that violation tests fail correctly and valid queries return expected results.

Reproducibility test: Delete your database, restore only the base schema, and run your script. If it succeeds end-to-end — your deliverable is complete.

Change Control Workflow

Without change control, every "quick addition" erodes the schedule. Every change — no matter how small — must follow this formal path before implementation.

Every Change Request Must State

- **Why** the change is needed (business or clinical justification)
- **Impact** on schedule, cost, and existing functionality
- **Risk** introduced by the change
- **Acceptance criteria** for the new feature

Real Example: "Add Imaging Table"

A teammate suggests adding an `ImagingOrders` table. This is **not "just adding it"** — it requires a change request. Impact analysis might reveal: +2 days of build time, new FK dependencies, additional test cases, and documentation updates.

Solved Example — Full Change Request

Scenario: During Day 3 of the project, a teammate suggests: "Let's add an `Appointments_Archive` table to store cancelled appointments — it'll only take an hour."

Required Component	Completed Change Request
1. Change Description	Add `Appointments_Archive` table to store cancelled/expired appointment records with columns: `ArchiveID` (PK), `AppointmentID` (FK → `Appointments`), `CancelledDate`, `Reason` (`VARCHAR 200`).
2. Justification / Business Need	Regulatory requirement — healthcare systems must retain cancelled appointment records for audit trails. Requested by Clinical Advisor after reviewing compliance checklist.
3. Impact Assessment	Scope impact — adds 1 new table + 1 FK constraint + 2 new test cases. Time impact — estimated +3 hours (design: 1hr, implement: 1hr, test: 1hr). Risk — low; no changes to existing tables.
4. Decision	**APPROVED** — with condition that the archive table is added in a separate script section clearly labelled `"Phase 2 Extension"` and does not affect the base submission deadline.

Notice: Even a “quick” 1-hour suggestion became a 3-hour task once properly scoped. This is why every change — no matter how small — must go through formal change control.

📝 Homework 9 — Project Management Pack

Submit: One PDF (maximum 2 pages) + one `.sql` script file. Due at the next lecture session.

Part 1 — WBS

Build a WBS with **at least 10 tasks** for adding `LabOrders` + `LabResults` to EHR_DB. Include task number, description, and predecessor for each task.

Part 2 — RACI Matrix

Create a RACI table with **at least 6 tasks**. Use realistic role names (student names). Ensure exactly one A per task.

Part 3 — Risk Register

Document **5 risks**. Each row must include: Risk Description, Likelihood (H/M/L), Impact (H/M/L), Mitigation Strategy, and Owner.

Part 4 — Change Request

Write a complete change request for *"Add Imaging Module"*: justify it, analyze schedule impact, state risks, and define acceptance criteria.

Thank you

Google Classroom