ER Triage System – Team Development Workflow

This document explains the correct workflow order for development in a team setting. It clarifies who should start first, how each part of the system depends on the other, and how to coordinate tasks effectively for the ER Triage System project.

1. The Development Pipeline

The system's general flow is: Frontend (GUI) → Backend (Services & Logic) → Database (Storage). However, when building the system, development usually starts in the reverse direction for better stability.

2. Correct Development Order

Step 1: Database Team (Starts First)

Goal: Create the foundation — the data structure that everyone will depend on.

Responsibilities:
• Design all tables, columns, relationships, and constraints.
• Define clear naming conventions for tables such as patients, vitals, triage_result, and audit_log.
• Populate sample data for testing.
• Provide an SQL schema file (e.g., ER_Triage_Schema.sql) to the backend team.

Why first? The backend logic and DAOs must know the table and column structure before they can connect and process data.

Step 2: Backend Team

Goal: Build the engine that interacts with the database and applies the triage logic.

Responsibilities:
• Implement DAO classes (PatientDAO, AuditLogDAO) that communicate with the database.
• Develop service classes (PatientService, TriageService, QueueService) containing core logic.
• Create controllers or servlets that expose API endpoints for the GUI to use.

Why second? The backend depends on the database structure. Once the schema is finalized, the backend can safely build queries and services.

Step 3: GUI (Frontend) Team

Goal: Build the user interface for nurses and doctors to interact with the system.

Responsibilities:
• Design forms and dashboards using HTML/CSS/JavaScript (or Java Swing if using desktop GUI).
• Connect UI forms to backend API endpoints (POST, GET, PATCH requests).
• Display triage results, patient queues, and alerts.

Why third? The frontend needs working backend endpoints to connect to. It's easier to design once the backend returns real JSON responses.

3. Example of Workflow in This Project

Order	Team	Task	Output
1️⃣	Database	Create tables: patients, vitals, audit_log	ER_Triage_Schema.sql
2️⃣	Backend	Build models, DAOs, services, and controllers	Working API & logic layer
3️⃣	GUI	Design intake form, queue dashboard	index.html / queue.html or Swing GUI
4️⃣	Testing	Run end-to-end test (Nurse → DB → Queue)	Validated workflow

4. Coordination and Overlap

• The backend and database teams can work in parallel if they agree on the database schema early.
• The GUI team can start designing layouts while waiting for backend APIs by using mock data.
• Final integration and testing always follow the order: Database → Backend → Frontend.

5. Recommended Team Roles (Your Group Example)

• A – Backend lead: Handles database connection, service logic, and API creation.
• D – Backend helper: Assists A with schema setup and logic testing.
• B – GUI developer: Creates the front-end interface and connects it to backend endpoints.
• C – Documentation and tester: Writes documentation, test cases, and verifies workflow integration.

6. Key Takeaway

The correct build sequence ensures smoother progress and fewer conflicts. Starting from the database establishes a solid structure for data storage, then the backend brings logic and automation, and finally, the GUI makes the system accessible and user-friendly. Following this order prevents miscommunication and allows each team to focus on their specialized layer.