Smart Meeting Room System - Database Progress Report

Salah Awji

Date: August 2–3, 2025
Tech Stack Focus: SQL Server (T-SQL), Relational DB Design, Seeding
Backend Stack (in-progress): ASP.NET Core API, C#, Entity Framework

📌 Key Accomplishments

1. **Designed and Created the Full Relational Database Schema**
    a. Tables: `Users`, `Roles`, `Rooms`, `Features`, `RoomFeatures`, `Meetings`, `Moms`, `ActionItems`, `Attachments`, `Attendees.`
    b. Correct usage of:
        i. **Primary and Foreign Keys**
        ii. **ON DELETE CASCADE / SET NULL** rules
        iii. **Normalization** for reusability and referential integrity
2. **Seeded Data Across the Entire Schema**
    a. Used manual SQL `INSERT INTO` commands to seed initial rows across all tables
    b. Maintained proper foreign key relations while inserting
    c. Simulated a **realistic data flow** (users, rooms, meetings, room features, etc.)
    d. Prepared for app integration by thinking ahead about feature toggles and user roles
3. **Security Awareness**
    a. Identified and corrected an insecure pattern: storing passwords in plain text
    b. Planned to **hash passwords** in the backend before insertion
    c. Understood that passwords must not be modified or hashed in-place after creation
4. **Timestamps and Auditing**
    a. Added `createdAt` and `updatedAt` fields to important tables (e.g., `Users`, `Meetings`, `Roles`)
    b. Used `DEFAULT GETDATE()` for `createdAt` to auto-track record creation
    c. Discussed tracking `updatedAt` via backend logic on update

5. **Production-Ready Practices Introduced**
    a. Wrote seeding scripts with maintainability in mind
    b. Used `VARCHAR`, `INT`, `DATETIME`, and `BIT` types with intention
    c. Discussed and applied **cascading deletion logic**
    d. Clean, readable formatting and inline comments
6. **Git and Version Control**
    a. Prepped the DB for versioning by planning to commit seeded version to GitHub repo

🧠 Lessons Learned

- Databases aren't just about storing data — **they enforce integrity, manage security, and support app logic.**
- Planning cascade behavior in foreign keys prevents orphaned records or unintended data wipes.
- Real-world database work requires thinking about the future: who will use the data, how it will change, and what tools touch it.
- Passwords should **never** be stored in plain text — security starts at the database level, not just the UI.
- Manual seeding teaches relationships far better than random seed scripts — it's the best way to master schema design.

🔑 Important Keywords from Today

- `SQL Server, T-SQL, Primary Key, Foreign Key, ON DELETE CASCADE, GETDATE(),`
  `Data Seeding, Normalization, One-to-Many, Many-to-Many,`
  `JOIN Tables, Password Hashing, Referential Integrity,`
  `Relational Database Design, Role-Based Access, Timestamp Columns,`
  `Manual Inserts, Users/Roles/Meetings, Room Features`