WORK ASSIGNMENTS FOR SCHEDOOL

Frontend

- **UI Design & Implementation *(May)***
  - Build a reservation form (inputs: name, date, time, duration, room, etc.)
  - Implement calendar/schedule view with filter options (by date, room, user, etc.)
  - Add controls for add, edit, and delete reservations
- **Backend Integration *(May)***
  - Directly call backend (JS modules) from frontend actions (no API layer)
- **Validation *(May)***
  - Validate form inputs (format, required fields, data type)
  - Display backend results: success messages, conflict warnings, or errors

Backend (Node.js with SQLite)

- **Database & Models *(Poon & Win)***
  - Define SQLite schema for reservations (tables: rooms, reservations, etc.)
  - Implement connection, basic CRUD and query helpers for SQLite
- **Reservation Management *(Poon)***
  - Add, edit, and delete reservation functions
  - Implement conflict checking (prevent overlapping/double bookings)
- **Additional Features *(Beam)***
  - Export function: export reservations with selected format
  - Rule-based recommendation system (e.g., suggest available slots or rooms)
- **Testing *(Win)***
  - Unit tests for CRUD, conflict detection, and export functions

Other Tasks

- **Documentation (*All member)***
  - Setup guide: how to run the program (dependencies, SQLite setup)
  - Developer guide: code structure, how frontend calls backend modules
  - User guide: how to use reservation form, filters, export, etc.
- **Test flow *(Beam)***
  - Define test cases for each user action (add, edit, delete, export, conflicts)
  - End-to-end test: simulate user flow from frontend to backend
- **Demo Data Preparation *(Beam)***
  - Seed SQLite with sample rooms, users, and reservations
  - Ensure demo data includes edge cases (overlapping times, near-full schedule)