Database Schema Documentation

Models Overview

Employee

- Main entity for storing employee information
- Each employee has one authentication record and belongs to one role
- Key fields:
  - `id`: Unique identifier (auto-increment)
  - `firstName`, `lastName`: Basic information
  - `shift`: Can be "morning", "evening", or "night"
  - `status`: Can be "active", "on leave", or "terminated"
  - `roleId`: Links to Role entity

Role

- Defines employee roles and permissions
- Has many-to-many relationship with Menu items
- Key fields:
  - `id`: Unique identifier
  - `name`: Role name (e.g., "admin", "mechanic", "manager")

Menu

- Represents navigation menu items and permissions
- Many-to-many relationship with Roles
- Key fields:
    - `id`: Unique identifier
    - `name`: Menu item name
    - `permissions`: Array of permission strings
    - `route`: Frontend route path
    - `icon`: Icon identifier
    - `order`: Display order

Auth

- Stores authentication credentials
- One-to-one relationship with Employee
- Key fields:
    - `id`: Unique identifier
    - `email`: Unique email address
    - `password`: Hashed password

Relationships

1. Role ←→ Menu (Many-to-Many)
    - Junction table: "role_menu"
2. Role → Employee (One-to-Many)
    - Employee has roleId foreign key
3. Employee → Auth (One-to-One)
    - Auth has employeeId foreign key

Example Data

```
// Example Role data
const roles = [
  { id: 1, name: "admin" },
  { id: 2, name: "manager" },
  { id: 3, name: "mechanic" }
];
```

```
// Example Menu data
const menus = [
  {
    id: 1,
    name: "Dashboard",
    permissions: ["view_dashboard"],
    route: "/dashboard",
    icon: "dashboard",
    order: 1
  },
  {
    id: 2,
    name: "Employee Management",
    permissions: ["manage_employees", "view_employees"],
    route: "/employees",
    icon: "people",
    order: 2
  }
];

// Example Employee data
const employees = [
  {
    id: 1,
    firstName: "John",
    lastName: "Doe",
    shift: "morning",
    workingHours: "08:00-17:00",
    specialization: "mechanic",
    status: "active",
    roleId: 3
  }
];

// Example Auth data
const auth = [
  {
    id: 1,
    email: "john.doe@example.com",
    password: "hashedPassword123",
    employeeId: 1
  }
];

// Example role_menu junction data
const roleMenu = [
  { roleId: 1, menuId: 1 },
  { roleId: 1, menuId: 2 },
  { roleId: 2, menuId: 1 }
];
```

Notes for Frontend Development

1. When creating new employees, ensure both Employee and Auth records are created
2. Menu access should be filtered based on user's role
3. Status changes will automatically update `statusChangedAt`
4. All dates are stored in ISO format
5. Phone numbers should be formatted consistently before saving