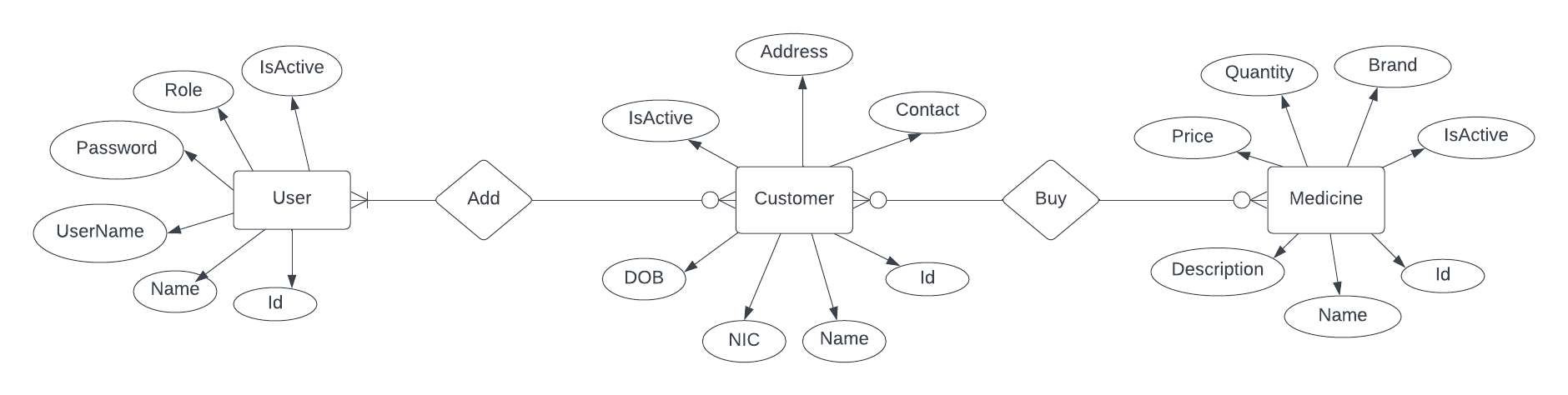
1. What would be the main API endpoints you would create when building this solution?

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **URL** | **Description** |
| /auth/login | POST | Authenticate the user by given username and password |
| /auth/check | GET | Check whether the user authenticated or not |
| /medicines | GET | Find all the medicines |
| /medicines/[id] | GET | Find medicine by Id |
| /medicines | POST | Create a medicine record |
| /medicines/[id] | PUT | Update a medicine record by Id |
| /medicines/[id]/delete | PUT | Soft delete a medicine record by Id |
| /medicines/[id] | DELETE | Permanently delete a medicine record by Id |
| /customers | GET | Find all customers |
| /customers/[id] | GET | Find a customer by Id |
| /customers | POST | Create a customer record |
| /customers/[id] | PUT | Update a customer record by Id |
| /customers/[id]/delete | PUT | Soft delete a customer record by Id |
| /customers/[id] | DELETE | Permanently delete a customer record by Id |
| /users | GET | Find all users |
| /users/[id] | GET | Find a user by Id |
| /users | POST | Create a user |
| /users/[id] | PUT | Update a user by Id |

1. What are the data models you would create to implement this solution?



ER Diagram

1. What are the frameworks / libraries you would use to build this solution?

|  |  |
| --- | --- |
| **Name of the framework / package** | **What would you accomplish using that?** |
| NestJs | Progressive back-end application framework |
| Prisma | ORM that manage and communicate with the database |
| Bcrypt | Hash and compare user passwords |
| Passport | Authentication middleware |

1. What are the main programs and tools you would use to build this solution?

|  |  |
| --- | --- |
| **Name of the program** | **What would you accomplish using that?** |
| VS Code | IDE for development |
| Postman | Test the API |
| DBeaver | Universal database management tool |

1. What is the approach you would use to implement User Permissions based on User Roles?

* Authenticate user using username and password or jwt tokens.
* Once successfully authenticated, add the user object including the user role to the request.
* After validating / authenticating the user, we can implement a basic RBAC (Role Base Access Control) mechanism to define user roles and privileges using Nest guards.
* Create a Role Enum representing roles in the system.

enum Role {

Owner = owner,

Manager = manager,

Cashier = cashier,

}

* Create a role decorator that allows specifying which roles can access specific endpoints.
* Create role guard which will compare role assigned to the current user that added to the request and the allowed roles given in the decorator that can access current endpoint that being processed.

import { Injectable, CanActivate, ExecutionContext } from '@nestjs/common';

import { Reflector } from '@nestjs/core';

@Injectable()

export class RolesGuard implements CanActivate {

constructor(private reflector: Reflector) {}

canActivate(context: ExecutionContext): boolean {

const allowedRoles = this.reflector.getAllAndOverride<Role[]>(ROLES\_KEY, [

context.getHandler(),

context.getClass(),

]);

if (!allowedRoles) true;

const { user } = context.switchToHttp().getRequest();

return allowedRoles.some((role) => user?.role === role);

}

}

* Now we can add the role guard to the endpoints and mention user roles that can access particular endpoint.

@Post()

@Roles(Role.Owner)

createMedicine(@Body() createMedicineDto: CreateMedicineDto) {

this.medicinesService.create(createMedicineDto);

}

@Put()

@Roles(Role.Owner, Role.Manager, Role.Cashier)

updateMedicine(@Body() updateMedicineDto: UpdateMedicineDto) {

this.medicinesService.create(updateMedicineDto);

}