TASK

// --- Backend (NestJS) ---

// src/app.module.ts

@Module({

imports: [

TypeOrmModule.forRoot({

type: 'postgres',

host: 'localhost',

port: 5432,

username: 'postgres',

password: 'password',

database: 'task\_db',

autoLoadEntities: true,

synchronize: true,

}),

UsersModule,

TasksModule,

AuthModule,

],

})

export class AppModule {}

// src/users/user.entity.ts

@Entity()

export class User {

@PrimaryGeneratedColumn() id: number;

@Column() name: string;

@Column({ unique: true }) email: string;

@Column() password: string;

@OneToMany(() => Task, task => task.createdBy) createdTasks: Task[];

@OneToMany(() => Task, task => task.assignedTo) assignedTasks: Task[];

}

@Entity()

export class Task {

@PrimaryGeneratedColumn() id: number;

@Column() title: string;

@Column() description: string;

@Column() dueDate: Date;

@Column() priority: string;

@Column() status: string;

@ManyToOne(() => User, user => user.createdTasks) createdBy: User;

@ManyToOne(() => User, user => user.assignedTasks) assignedTo: User;

}

// (only login example)

async validateUser(email: string, pass: string): Promise<any> {

const user = await this.usersService.findByEmail(email);

if (user && await bcrypt.compare(pass, user.password)) {

const { password, ...result } = user;

return result;

}

return null;

}

export default function Login() {

const router = useRouter();

const { register, handleSubmit } = useForm();

const onSubmit = async (data) => {

const res = await fetch('/api/login', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(data),

});

if (res.ok) router.push('/dashboard');

};

return (

<form onSubmit={handleSubmit(onSubmit)}>

<input placeholder="Email" {...register('email')} />

<input placeholder="Password" type="password" {...register('password')} />

<button type="submit">Login</button>

</form>

);

}

export default function Dashboard() {

const [tasks, setTasks] = useState([]);

useEffect(() => {

fetch('/api/tasks').then(res => res.json()).then(setTasks);

}, []);

return (

<div>

<h2>My Tasks</h2>

<ul>

{tasks.map(task => (

<li key={task.id}>{task.title} - {task.status}</li>

))}

</ul>

</div>

);

}