Нижче наведено декілька порад і прикладів реалізації, як можна організувати систему ролей і доступів із врахуванням описаних вами ролей (власник СТО, керівництво, менеджер з закупівель, менеджер з клієнтами, майстри СТО, бухгалтер). Також продемонструю короткий фрагмент коду, який ілюструє базові ідеї створення та перевірки ролей у Node.js/Express, MongoDB та React.

**1. Підхід до системи ролей**

1. **Гнучкість ролей**. У моделі користувача (наприклад, у MongoDB) зберігаємо масив ролей. Таким чином:
   * Один користувач може мати декілька ролей одночасно.
   * У маленьких СТО одна людина може комбінувати в собі всі ролі.
   * Якщо знадобиться додати нову роль (наприклад, “Стажер” чи “Суперадмін”), це можна зробити, додавши її назву без зміни основної архітектури.
2. **Простота налаштування**.
   * У власному інтерфейсі адміністратора (або у власника СТО) можна реалізувати просту форму, де вибираються (чекбокси чи селекти) ролі для кожного користувача.
   * Тобто коли додаємо (або редагуємо) користувача, ми просто ставимо галочки: “Власник”, “Бухгалтер”, “Менеджер з клієнтами” тощо.
3. **Набір дозволів (permissions)**. Залежно від обраної ролі (чи ролей), у користувача будуть відкриті певні функціональні можливості:
   * **Власник СТО**: повний доступ (додавання, редагування, видалення всіх даних), створення або призначення інших ролей.
   * **Керівництво СТО**: може керувати замовленнями, переглядати звіти, але не обов’язково редагувати фінансові налаштування, якщо це прив’язано до окремої ролі бухгалтера.
   * **Менеджер із закупівель**: доступ до списку запчастин, замовлень у постачальників, формування заявок, контроль постачань.
   * **Менеджер по роботі з клієнтами**: доступ до історії клієнтів, записів на сервіс, комунікації з клієнтами.
   * **Майстри СТО**: можуть переглядати (і редагувати) статус робіт, виконані роботи, додавати інформацію про деталі ремонту.
   * **Бухгалтер**: формування фінансової звітності, виставлення рахунків, робота з документами на оплату, доступ до фінансових звітів.
4. **Можливість масштабування**. Якщо пізніше буде потреба у більш детальному розподілі, наприклад, у “Менеджера з клієнтами” має бути обмежений доступ, а у “Старшого менеджера” — розширений, то можна або додати роль “Старший менеджер”, або перейти до більш детальної схематики (ACL/permissions). Але для MVP досить простої моделі “Роль = набір можливостей”.

**2. Структура бази даних (MongoDB)**

Нижче приклад Mongoose-схеми для користувача з масивом ролей:

js

CopyEdit

// models/User.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

passwordHash: { type: String, required: true },

// Можна зберігати масив ролей

roles: [{ type: String, enum: ['OWNER', 'MANAGEMENT', 'PURCHASE\_MANAGER', 'CLIENT\_MANAGER', 'MECHANIC', 'ACCOUNTANT'] }],

// Додаткові поля: ім'я, прізвище, контактні дані тощо

firstName: String,

lastName: String,

email: String,

// ...

});

module.exports = mongoose.model('User', userSchema);

**Пояснення:**

* **roles** — масив рядків, у переліку enum можна передбачити всі ролі, які ми плануємо.
* Якщо в майбутньому ролей буде дуже багато, краще зберігати їх як прості рядки або мати окрему колекцію. Але для старту вистачить і enum.

**3. Перевірка ролей у бекенді (Express.js)**

Щоб робити перевірку доступу на рівні маршрутів (routes), можна створити middleware, яке перевірятиме, чи в користувача є хоча б одна потрібна роль:

js

CopyEdit

// middleware/checkRoles.js

function checkRoles(requiredRoles) {

return (req, res, next) => {

// req.user має бути заповнений після авторизації (наприклад, декодування JWT або сесії)

if (!req.user) {

return res.status(401).json({ message: 'Unauthorized' });

}

// Перевіряємо, чи користувач має хоча б одну роль із списку дозволених

const hasRole = requiredRoles.some(role => req.user.roles.includes(role));

if (!hasRole) {

return res.status(403).json({ message: 'Forbidden' });

}

next();

};

}

module.exports = { checkRoles };

**Приклад використання в маршруті:**

js

CopyEdit

const express = require('express');

const router = express.Router();

const { checkRoles } = require('../middleware/checkRoles');

// Наприклад, цей маршрут призначений для перегляду або редагування списку користувачів

router.get('/users',

checkRoles(['OWNER', 'MANAGEMENT']),

(req, res) => {

// Якщо скрипт дійшов сюди, значить у користувача є роль OWNER або MANAGEMENT

// ...отримуємо список користувачів

res.json({ message: 'List of users' });

}

);

module.exports = router;

Таким чином, коли користувач викликає цей маршрут, middleware checkRoles(['OWNER','MANAGEMENT']) перевіряє, чи у req.user.roles є хоча б одна зі вказаних ролей.

**4. Фронтенд (React) — візуальна частина**

1. **Налаштування ролей при створенні користувача**:
   * Припустимо, що в нас є сторінка “Manage Users” тільки для ролей OWNER або MANAGEMENT. Там можна заповнити форму, де є список чекбоксів:
     + Власник (OWNER)
     + Керівництво (MANAGEMENT)
     + Менеджер з закупівель (PURCHASE\_MANAGER)
     + Менеджер з клієнтами (CLIENT\_MANAGER)
     + Майстер СТО (MECHANIC)
     + Бухгалтер (ACCOUNTANT)
   * Дані з чекбоксів відправляємо в JSON на бекенд, де створюємо або оновлюємо користувача.
2. **Управління доступом у фронтенді**:
   * Після того, як користувач залогіниться, зберігаємо в localStorage (чи redux) jwt-токен.
   * Розкодовуємо (decode) токен, щоб витягнути з нього roles.
   * При переході на сторінки або відображенні певних кнопок/меню перевіряємо, чи roles.includes('OWNER') тощо.
   * Наприклад:

jsx

CopyEdit

// Десь у компоненті навігації

{userRoles.includes('OWNER') && (

<NavLink to="/admin-panel">Панель адміністратора</NavLink>

)}

* + Таким чином, кнопки чи посилання будуть видимі тільки для користувачів із потрібною роллю.

Звісно, пам’ятаємо, що **перевірка на фронтенді** — це лише зручність для користувача. Справжня безпека робиться на бекенді, тож там обов’язково мають бути middleware чи перевірки ролей, щоб ніхто не зміг обійти обмеження через прямі запити до API.

**5. Як спростити менеджмент ролей**

* **Створити простий “внутрішній” інтерфейс** для власника чи керівництва СТО, де вони можуть:
  1. Переглянути список усіх користувачів.
  2. Натиснути кнопку “Редагувати користувача”.
  3. Змінити набір ролей за допомогою чекбоксів.
  4. Натиснути “Зберегти”.
* **Автоматично видавати ролі за умовою** (необов’язково, але може бути зручно), наприклад:
  1. Якщо це новий користувач, який реєструється самостійно (клієнт), то ми йому за замовчуванням даємо роль “Клієнт” (якщо така потрібна).
  2. Якщо користувача створює “Власник” і обирає “Майстер”, одразу пропонувати налаштування доступу до ремонтних листів та записів.
* **Логування змін**. Якщо у майбутньому виникне потреба в додатковій безпеці, варто логувати, хто саме (і коли) змінив ролі користувача.