about controller

export const aboutController = {

index: {

handler: function (request, h) {

const viewData = {

title: "About ourHospital",

};

return h.view("about-view", viewData);

},

},

};

accounts controller

/\* eslint-disable func-names \*/

import { db } from "../models/db.js";

import { UserSpec, UserCredentialsSpec } from "../models/joi-schemas.js";

export const accountsController = {

index: {

auth: false,

handler: function (request, h) {

return h.view("main", { title: "Welcome to Department" });

},

},

showSignup: {

auth: false,

handler: function (request, h) {

return h.view("signup-view", { title: "Sign up for Department" });

},

},

signup: {

auth: false,

validate: {

payload: UserSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

return h.view("signup-view", { title: "Sign up error", errors: error.details }).takeover().code(400);

},

},

handler: async function (request, h) {

const user = request.payload;

await db.userStore.addUser(user);

return h.redirect("/");

},

},

showLogin: {

auth: false,

handler: function (request, h) {

return h.view("login-view", { title: "Login to Department" });

},

},

login: {

auth: false,

validate: {

payload: UserCredentialsSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

return h.view("login-view", { title: "Log in error", errors: error.details }).takeover().code(400);

},

},

handler: async function (request, h) {

const { email, password } = request.payload;

const user = await db.userStore.getUserByEmail(email);

if (!user || user.password !== password) {

return h.redirect("/");

}

request.cookieAuth.set({ id: user.\_id });

return h.redirect("/dashboard");

},

},

logout: {

handler: function (request, h) {

request.cookieAuth.clear();

return h.redirect("/");

},

},

async validate(request, session) {

const user = await db.userStore.getUserById(session.id);

if (!user) {

return { isValid: false };

}

return { isValid: true, credentials: user };

},

};

dashboard controller

import { db } from "../models/db.js";

import Joi from "joi";

import { HospitalSpec } from "../models/joi-schemas.js";

export const dashboardController = {

index: {

handler: async function (request, h) {

const loggedInUser = request.auth.credentials;

if (!loggedInUser) {

return h.redirect("/login");

}

const hospitals = await db.hospitalStore.getUserHospitals(loggedInUser.\_id);

console.log("Hospitals for user", loggedInUser.\_id, "→", hospitals);

const viewData = {

title: "Hospital Dashboard",

user: loggedInUser,

hospitals: hospitals,

};

return h.view("dashboard-view", viewData);

},

},

addHospital: {

validate: {

payload: HospitalSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

return h.view("dashboard-view", {

title: "Add Hospital Error",

errors: error.details

})

.takeover()

.code(400);

}

},

handler: async function (request, h) {

const loggedInUser = request.auth.credentials;

if (!loggedInUser || !loggedInUser.\_id) {

return h.redirect("/login");

}

const newHospital = {

userId: loggedInUser.\_id,

name: request.payload.name,

type: request.payload.type || "",

location: request.payload.location,

latitude: request.payload.latitude,

longitude: request.payload.longitude

};

await db.hospitalStore.addHospital(newHospital);

return h.redirect("/dashboard");

}

},

deleteHospital: {

handler: async function (request, h) {

await db.hospitalStore.deleteHospitalById(request.params.id);

return h.redirect("/dashboard");

},

},

deleteDepartment: {

handler: async function (request, h) {

await db.departmentStore.deleteDepartmentById(request.params.id);

return h.redirect("/dashboard");

},

},

};

department-controller

import { v4 } from "uuid";

import { db } from "../models/db.js";

import { DepartmentSpec, StaffSpec } from "../models/joi-schemas.js";

export const departmentController = {

index: {

handler: async function (request, h) {

const department = await db.departmentStore.getDepartmentById(request.params.id);

if (!department) {

return h.response("Department not found").code(404);

}

const hospital = await db.hospitalStore.getHospitalById(department.hospitalId);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

const staff = await db.staffStore.getStaffByDepartmentId(department.\_id);

const viewData = {

title: "Department",

hospital: hospital,

department: department,

staff: staff,

};

return h.view("department-view", viewData);

},

},

addDepartment: {

validate: {

payload: DepartmentSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

console.log("Validation Error:", error.details);

return h.view("hospital-view", {

title: "Add department error",

errors: error.details

}).takeover().code(400);

},

},

handler: async function (request, h) {

console.log("Params received:", request.params);

const hospital = await db.hospitalStore.getHospitalById(request.params.hospitalId);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

const newDepartment = {

hospitalId: hospital.\_id,

title: request.payload.title,

deptLocation: request.payload.deptLocation

};

await db.departmentStore.addDepartment(newDepartment);

console.log("New department added:", newDepartment);

return h.redirect(`/hospital/${hospital.\_id}`);

},

},

addStaff: {

validate: {

payload: StaffSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

return h.view("department-view", { title: "Add staff error", errors: error.details }).takeover().code(400);

},

},

handler: async function (request, h) {

const department = await db.departmentStore.getDepartmentById(request.params.id);

if (!department) {

return h.response("Department not found").code(404);

}

const hospital = await db.hospitalStore.getHospitalById(department.hospitalId);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

const newStaff = {

role: request.payload.role,

name: request.payload.name,

vignette: request.payload.vignette,

departmentId: department.\_id,

};

await db.staffStore.addStaff(newStaff);

return h.redirect(`/hospital/${hospital.\_id}/department/${department.\_id}`);

},

},

deleteStaff: {

handler: async function (request, h) {

const department = await db.departmentStore.getDepartmentById(request.params.id);

if (!department) {

return h.response("Department not found").code(404);

}

await db.staffStore.deleteStaff(request.params.staffid);

return h.redirect(`/hospital/${department.hospitalId}/department/${department.\_id}`);

},

},

deleteDepartment: {

handler: async function (request, h) {

const hospital = await db.hospitalStore.getHospitalById(request.params.hospitalId);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

await db.departmentStore.deleteDepartmentById(request.params.id);

return h.redirect(`/hospital/${hospital.\_id}`);

},

},

};

hospital-controller

import { db } from "../models/db.js";

import Joi from "joi";

import { HospitalSpec } from "../models/joi-schemas.js";

export const hospitalController = {

index: {

handler: async function (request, h) {

const hospital = await db.hospitalStore.getHospitalById(request.params.id);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

const departments = await db.departmentStore.getDepartmentsByHospitalId(request.params.id);

const viewData = {

title: "Hospital",

hospital: hospital,

departments: departments,

};

return h.view("hospital-view", viewData);

},

},

addHospital: {

validate: {

payload: HospitalSpec,

options: { abortEarly: false },

failAction: function (request, h, error) {

return h.view("dashboard-view", { title: "Add hospital error", errors: error.details })

.takeover()

.code(400);

},

},

handler: async function (request, h) {

await db.hospitalStore.addHospital(request.payload);

return h.redirect("/dashboard");

},

},

deleteHospital: {

handler: async function (request, h) {

const hospital = await db.hospitalStore.getHospitalById(request.params.id);

if (!hospital) {

return h.response("Hospital not found").code(404);

}

// Cleanup: Delete associated departments before deleting hospital

await db.departmentStore.deleteDepartmentsByHospitalId(request.params.id);

await db.hospitalStore.deleteHospitalById(request.params.id);

return h.redirect("/dashboard");

},

},

};

department-json-store.js

import { v4 } from "uuid";

import { db } from "./store-utils.js";

import { staffJsonStore } from "./staff-json-store.js";

export const departmentJsonStore = {

async getAllDepartments() {

await db.read();

return db.data.departments;

},

async addDepartment(department) {

await db.read();

db.data.departments = db.data.departments || [];

department.\_id = v4();

db.data.departments.push(department);

await db.write();

return department;

},

async getDepartmentById(id) {

await db.read();

const department = db.data.departments.find((d) => String(d.\_id) === String(id)) || null;

if (!department) return null;

department.staff = await staffJsonStore.getStaffByDepartmentId(department.\_id) || [];

return department;

},

async getUserDepartments(userid) {

await db.read();

return db.data.departments.filter((department) => department.userid === userid);

},

async deleteDepartmentById(id) {

await db.read();

const index = db.data.departments.findIndex((department) => department.\_id === id);

db.data.departments.splice(index, 1);

await db.write();

},

async deleteAllDepartments() {

db.data.departments = [];

await db.write();

},

async getDepartmentsByHospitalId(hospitalId) {

await db.read();

return db.data.departments.filter((d) => d.hospitalId === hospitalId);

}

};

hospital-json-store.js

import { v4 } from "uuid";

import { db } from "./store-utils.js";

import { staffJsonStore } from "./staff-json-store.js";

import { departmentJsonStore } from "./department-json-store.js";

export const hospitalJsonStore = {

async getAllHospitals() {

await db.read();

return db.data.hospitals || [];

},

async addHospital(hospital) {

await db.read();

db.data.hospitals = db.data.hospitals || [];

hospital.\_id = v4();

if (!hospital.userid) {

throw new Error("Missing userid in addHospital()");

}

db.data.hospitals.push(hospital);

await db.write();

return hospital;

},

async getHospitalById(id) {

await db.read();

const hospital = db.data.hospitals.find((h) => String(h.\_id) === String(id)) || null;

if (hospital) {

hospital.type = hospital.type || "Other";

hospital.departments = await departmentJsonStore.getDepartmentsByHospitalId(id) || [];

for (let dept of hospital.departments) {

dept.staff = await staffJsonStore.getStaffByDepartmentId(dept.\_id) || [];

}

}

return hospital;

},

async getUserHospitals(userid) {

await db.read();

return (db.data.hospitals || []).filter((hospital) => hospital.userid === userid);

},

async deleteHospitalById(id) {

await db.read();

const index = db.data.hospitals.findIndex((hospital) => hospital.\_id === id);

if (index !== -1) {

db.data.hospitals.splice(index, 1);

await db.write();

}

},

async deleteAllHospitals() {

db.data.hospitals = [];

await db.write();

},

};

staff json store

import { v4 } from "uuid";

import { db } from "./store-utils.js";

export const staffJsonStore = {

async getAllStaffs() {

await db.read();

return db.data.staff || [];

},

async getStaffsByHospitalId(hospitalId) {

await db.read();

return (db.data.staff || []).filter((staff) => staff.hospitalId === hospitalId);

},

async addStaff(departmentId, staff) {

await db.read();

if (!db.data.staff) db.data.staff = [];

staff.\_id = v4();

staff.departmentid = departmentId;

db.data.staff.push(staff);

await db.write();

return staff;

},

async getStaffByDepartmentId(id) {

await db.read();

return (db.data.staff || []).filter((staff) => staff.departmentid === id);

},

async getStaffById(id) {

await db.read();

return (db.data.staff || []).find((staff) => staff.\_id === id);

},

async deleteStaff(id) {

await db.read();

if (!db.data.staff) return;

const index = db.data.staff.findIndex((staff) => staff.\_id === id);

if (index !== -1) {

db.data.staff.splice(index, 1);

await db.write();

}

},

async deleteAllStaffs() {

db.data.staff = [];

await db.write();

},

async updateStaff(staff, updatedStaff) {

staff.role = updatedStaff.role;

staff.name = updatedStaff.name;

staff.vignette = updatedStaff.vignette;

await db.write();

},

};

store-utilis.js

import { JSONFilePreset } from "lowdb/node";

export const db = await JSONFilePreset("src/models/json/db.json", {

users: [],

departments: [],

staffs: [],

hospitals: [],

});

user json store

import { v4 } from "uuid";

import { db } from "./store-utils.js";

export const userJsonStore = {

async getAllUsers() {

await db.read();

return db.data.users;

},

async addUser(user) {

await db.read();

user.\_id = v4();

db.data.users.push(user);

await db.write();

return user;

},

async getUserById(id) {

await db.read();

return db.data.users.find((user) => user.\_id === id);

},

async getUserByEmail(email) {

await db.read();

return db.data.users.find((user) => user.email === email);

},

async deleteUserById(id) {

await db.read();

const index = db.data.users.findIndex((user) => user.\_id === id);

db.data.users.splice(index, 1);

await db.write();

},

async deleteAll() {

db.data.users = [];

await db.write();

},

};

db.js

import { userMemStore } from "./mem/user-mem-store.js";

import { departmentMemStore } from "./mem/department-mem-store.js";

import { staffMemStore } from "./mem/staff-mem-store.js";

import { hospitalMemStore } from "./mem/hospital-mem-store.js";

import { userJsonStore } from "./json/user-json-store.js";

import { departmentJsonStore } from "./json/department-json-store.js";

import { staffJsonStore } from "./json/staff-json-store.js";

import { hospitalJsonStore } from "./json/hospital-json-store.js";

import { connectMongo } from "./mongo/connect.js";

import { userMongoStore } from "./mongo/user-mongo-store.js";

import { departmentMongoStore } from "./mongo/department-mongo-store.js";

import { staffMongoStore } from "./mongo/staff-mongo-store.js";

import { hospitalMongoStore } from "./mongo/hospital-mongo-store.js";

export const db = {

userStore: null,

departmentStore: null,

staffStore: null,

hospitalStore: null,

init(storeType) {

switch (storeType) {

case "json":

this.userStore = userJsonStore;

this.departmentStore = departmentJsonStore;

this.staffStore = staffJsonStore;

this.hospitalStore = hospitalJsonStore;

break;

case "mongo":

this.userStore = userMongoStore;

this.departmentStore = departmentMongoStore;

this.staffStore = staffMongoStore;

this.hospitalStore = hospitalMongoStore;

connectMongo();

break;

default:

this.userStore = userMemStore;

this.departmentStore = departmentMemStore;

this.staffStore = staffMemStore;

this.hospitalStore = hospitalMemStore;

}

},

};

joi schemas

import Joi from "joi";

export const UserSpec = {

firstName: Joi.string().required(),

lastName: Joi.string().required(),

email: Joi.string().email().required(),

password: Joi.string().length(5).required(),

};

export const UserCredentialsSpec = {

email: Joi.string().email().required(),

password: Joi.string().required(),

};

export const StaffSpec = {

role: Joi.string().required(),

name: Joi.string().required(),

vignette: Joi.string().required(),

};

export const DepartmentSpec = {

title: Joi.string().required(),

deptLocation: Joi.string().min(1).required()

};

export const HospitalSpec = {

name: Joi.string().min(1).required(),

type: Joi.string().min(1).required(),

location: Joi.string().required(),

latitude: Joi.number().required().min(-90).max(90),

longitude: Joi.number().required().min(-180).max(180)

};

about view

{{> menu active="about"}}

<section class="hero">

<div class="hero-body has-text-centered">

<p class="title">

<span style="color: Dodgerblue;">ourHospital</span> : Departments and their key staff

</p>

<p class="box">

<i class="fas fa-stethoscope fa-3x" style="color: darkorange;"></i>

<i class="fas fa-syringe fa-3x" style="color: deepskyblue;"></i>

<i class="fas fa-pills fa-4x" style="color: gold;"></i>

</p>

</div>

</section>

dashboard view

{{> menu active="dashboard"}}

<section class="section">

{{> list-hospitals}}

{{> add-hospital}}

</section>

department view

{{> menu}}

<section class="section">

<div class="subtitle">

<strong>Hospital:</strong> {{hospital.name}}

</div>

<div class="title">

{{department.title}}

</div>

{{> list-staffs}}

{{> add-staff}}

</section>

hospital view

{{> menu}}

<section class="section">

<h1 class="title">{{hospital.name}}</h1>

<h2 class="subtitle">Type: {{hospital.type}}</h2>

<h2 class="subtitle">Departments</h2>

{{> list-departments departments=departments hospital=hospital }}

<!-- Add Department Form -->

{{> add-department hospital\_id=hospital.\_id}}

</section>

log in view

{{> welcome-menu active="login"}}

<section class="section">

<h1 class="title">Log in</h1>

<form action="/authenticate" method="POST">

<div class="field">

<label class="label">Email</label> <input class="input" type="text" placeholder="Enter email" name="email">

</div>

<div class="field">

<label class="label">Password</label> <input class="input" type="password" placeholder="Enter Password" name="password">

</div>

<div class="field is-grouped">

<button class="button is-link">Submit</button>

</div>

</form>

</section>

main

{{> welcome-menu}}

<section class="section has-text-centered">

<h1 class="title is-3">Sign up or Log in...</h1>

<figure class="image is-inline-block" style="max-width: 400px; margin-top: 20px;">

<img src="/images/chi.jpg" alt="Hospital image" class="is-rounded">

</figure>

</section>

server-instance

import Hapi from "@hapi/hapi";

import Vision from "@hapi/vision";

import Inert from "@hapi/inert";

import HapiSwagger from "hapi-swagger";

import Cookie from "@hapi/cookie";

import Joi from "joi";

import Handlebars from "handlebars";

import path from "path";

import { fileURLToPath } from "url";

import dotenv from "dotenv";

import { db } from "./models/db.js";

import { apiRoutes } from "./api/api-routes.js";

import { webRoutes } from "./web-routes.js";

import { accountsController } from "./controllers/accounts-controller.js";

import { createRequire } from "module";

const require = createRequire(import.meta.url);

const Package = require("../package.json");

const \_\_filename = fileURLToPath(import.meta.url);

const \_\_dirname = path.dirname(\_\_filename);

dotenv.config();

export async function createServer() {

const server = Hapi.server({

port: process.env.PORT || 3000,

});

// Swagger config

const swaggerOptions = {

info: {

title: "OurHospital API Documentation",

version: Package.version,

},

};

// Register plugins

await server.register([

Inert,

Vision,

Cookie,

{

plugin: HapiSwagger,

options: swaggerOptions,

},

]);

// Joi validation

server.validator(Joi);

// View engine

server.views({

engines: { hbs: Handlebars },

relativeTo: \_\_dirname,

path: "./views",

layoutPath: "./views/layouts",

partialsPath: "./views/partials",

layout: true,

isCached: false,

});

// Auth setup

server.auth.strategy("session", "cookie", {

cookie: {

name: process.env.COOKIE\_NAME,

password: process.env.COOKIE\_PASSWORD,

isSecure: false,

},

redirectTo: "/",

validate: accountsController.validate,

});

server.auth.default("session");

// DB init

db.init("json");

// Register routes

server.route(apiRoutes);

server.route(webRoutes);

server.route({

method: 'GET',

path: '/images/{param\*}',

handler: {

directory: {

path: path.resolve(\_\_dirname, '../Public/images'), // Correct path

listing: false,

index: false

}

},

options: {

auth: false // Allow image access without login

}

});

return server;

}

server.js

import { createServer } from "./server-instance.js";

async function init() {

const server = await createServer();

await server.start();

console.log(`Server running at: ${server.info.uri}`);

}

init();

web routes

import { createServer } from "./server-instance.js";

async function init() {

const server = await createServer();

await server.start();

console.log(`Server running at: ${server.info.uri}`);

}

init();

package.json

{

"name": "ourHospital",

"version": "1.0.0",

"description": "",

"license": "ISC",

"author": "",

"type": "module",

"main": "src/server.js",

"scripts": {

"start": "node src/server.js",

"lint": "./node\_modules/.bin/eslint . --ext .js",

"test": "mocha \"test/\*\*/\*.js\" --exit",

"dev": "./node\_modules/.bin/nodemon src/server.js"

},

"engines": {

"node": ">=16"

},

"devDependencies": {

"axios": "^1.8.4",

"chai": "^5.2.0",

"eslint": "^8.57.1",

"eslint-config-airbnb-base": "^15.0.0",

"eslint-config-prettier": "^10.0.1",

"eslint-plugin-import": "^2.31.0",

"esm": "^3.2.25",

"mocha": "^11.1.0",

"nodemon": "^3.1.9",

"prettier": "^3.4.2",

"supertest": "^7.1.0"

},

"dependencies": {

"@hapi/boom": "^10.0.1",

"@hapi/cookie": "^12.0.1",

"@hapi/hapi": "^21.3.12",

"@hapi/inert": "^7.1.0",

"@hapi/vision": "^7.0.3",

"dotenv": "^16.4.7",

"handlebars": "^4.7.8",

"hapi-swagger": "^17.3.2",

"joi": "^17.13.3",

"lowdb": "^7.0.1",

"mongoose": "^8.12.1",

"uuid": "^11.1.0"

}

}

list-hospitals.hbs

{{#if hospitals.length}}

{{#each hospitals}}

<div class="box">

<article class="media">

<!-- Hospital Info Left Side -->

<div class="media-content">

<div class="content is-flex is-justify-content-space-between is-align-items-center">

<div>

<h2 class="title is-4">{{name}}</h2>

<p><strong>Type:</strong> {{type}}</p>

</div>

<div class="has-text-right">

<p><strong>Location:</strong> {{location}}</p>

<p><strong>Coordinates:</strong> {{latitude}}, {{longitude}}</p>

</div>

</div>

<!-- Action Buttons -->

<div class="buttons mt-3">

<a href="/hospital/{{\_id}}" class="button is-info">

<span class="icon is-small">

<i class="fas fa-folder-open"></i>

</span>

<span>Departments</span>

</a>

<a href="/dashboard/deletehospital/{{\_id}}" class="button is-danger">

<i class="fas fa-trash"></i>

<span>Delete</span>

</a>

</div>

</div>

<!-- Hospital Image Right Side -->

<div class="media-right">

<figure class="image is-96x96">

<img src="/images/Mater.jpg" alt="Hospital Image">

</figure>

</div>

</article>

</div>

{{/each}}

{{else}}

<p>No hospitals found.</p>

{{/if}}

add department

<form action="/hospital/{{hospital\_id}}/department/add" method="POST">

<!-- Department -->

<div class="field">

<label class="label">Department Title</label>

<input class="input" type="text" placeholder="Enter department title" name="title" required>

</div>

<!-- deptLocation -->

<div class="field">

<label class="label">Department Location</label>

<input class="input" type="text" placeholder="Enter department location" name="deptLocation" required>

</div>

<!-- Submit Button -->

<button class="button is-link">Add Department</button>

</form>

add hospital

<form action="/dashboard/addhospital" method="POST">

<div class="field is-horizontal">

<div class="field-body">

<!-- Hospital Name -->

<div class="field">

<label class="label">Hospital Name</label>

<div class="control">

<input class="input" type="text" placeholder="Enter hospital name" name="name" required>

</div>

</div>

<!-- Type -->

<div class="field">

<label class="label">Type</label>

<div class="control">

<div class="select">

<select name="type" required>

<option value="" disabled selected>Select Type</option>

<option value="National">National</option>

<option value="Regional">Regional</option>

<option value="Local">Local</option>

<option value="Other">Other</option>

</select>

</div>

</div>

</div>

<!-- Location -->

<div class="field">

<label class="label">Location</label>

<div class="control">

<input class="input" type="text" placeholder="Enter hospital location" name="location" required>

</div>

</div>

<!-- Latitude -->

<div class="field">

<label class="label">Latitude</label>

<div class="control">

<input class="input" type="text" placeholder="e.g., 53.3498" name="latitude" required>

</div>

</div>

<!-- Longitude -->

<div class="field">

<label class="label">Longitude</label>

<div class="control">

<input class="input" type="text" placeholder="e.g., -6.2603" name="longitude" required>

</div>

</div>

</div>

</div>

<!-- Submit -->

<div class="field">

<div class="control">

<button class="button is-link">Add Hospital</button>

</div>

</div>

</form>

add staff

<form class="box" action="/hospital/{{hospital.\_id}}/department/{{department.\_id}}/addstaff" method="POST">

<label>Enter Staff Member Details:</label>

<div class="field is-horizontal">

<div class="field-body">

<div class="field">

<input class="input" type="text" placeholder="Enter Role" name="role" required>

</div>

<div class="field">

<input class="input" type="text" placeholder="Enter Name" name="name" required>

</div>

<div class="field">

<input class="input" type="text" placeholder="Enter vignette here" name="vignette" required>

</div>

</div>

</div>

<button class="button is-primary">Add Staff</button>

</form>

list departments

<!-- Hospital Title + Back Button -->

<div class="level mb-4">

<div class="level-left">

<h2 class="title is-4">{{hospital.name}}</h2>

</div>

<div class="level-right">

<a class="button is-light" href="/dashboard">

<span class="icon"><i class="fas fa-arrow-left"></i></span>

<span>Back to Hospital List</span>

</a>

</div>

</div>

{{#each departments}}

<div class="box box-link-hover-shadow">

<!-- Title & Location on same line -->

<div class="level">

<div class="level-left">

<h2 class="title is-4">{{title}}</h2>

</div>

<div class="level-right">

<p class="has-text-right"><strong>Location:</strong> {{deptLocation}}</p>

</div>

</div>

<!-- Buttons -->

<div class="buttons">

<a href="/hospital/{{../hospital.\_id}}/department/{{\_id}}" class="button is-link is-light">

<span class="icon is-small">

<i class="fas fa-users"></i>

</span>

<span>Staff</span>

</a>

<a href="/hospital/{{../hospital.\_id}}/department/{{\_id}}/delete" class="button is-danger is-light is-small">

<span class="icon is-small">

<i class="fas fa-trash"></i>

</span>

<span>Delete</span>

</a>

</div>

</div>

{{/each}}

list-staff.hbs

<!-- Hospital Title + Back Button -->

<div class="level mb-4">

<div class="level-left">

<h2 class="title is-4">{{hospital.name}}</h2>

</div>

<div class="level-right">

<a class="button is-light" href="/dashboard">

<span class="icon"><i class="fas fa-arrow-left"></i></span>

<span>Back to Hospital List</span>

</a>

</div>

</div>

{{#each departments}}

<div class="box box-link-hover-shadow">

<!-- Title & Location on same line -->

<div class="level">

<div class="level-left">

<h2 class="title is-4">{{title}}</h2>

</div>

<div class="level-right">

<p class="has-text-right"><strong>Location:</strong> {{deptLocation}}</p>

</div>

</div>

<!-- Buttons -->

<div class="buttons">

<a href="/hospital/{{../hospital.\_id}}/department/{{\_id}}" class="button is-link is-light">

<span class="icon is-small">

<i class="fas fa-users"></i>

</span>

<span>Staff</span>

</a>

<a href="/hospital/{{../hospital.\_id}}/department/{{\_id}}/delete" class="button is-danger is-light is-small">

<span class="icon is-small">

<i class="fas fa-trash"></i>

</span>

<span>Delete</span>

</a>

</div>

</div>

{{/each}}