Dự án của tôi như sau:

HRM\_Project

Backend:

Config/db.js

const mongoose = require("mongoose");

const connectDB = async () => {

  try {

    await mongoose.connect(process.env.MONGO\_URI);

    console.log("MongoDB Connected");

  } catch (err) {

    console.error(err);

    process.exit(1);

  }

};

module.exports = connectDB;

Controllers

adminController.js

const User = require("../models/User");

const Employee = require("../models/Employee");

const Leave = require("../models/Leave");

const Salary = require("../models/Salary");

exports.getAllEmployees = async (req, res) => {

  const employees = await Employee.find().populate("userId", "username role");

  res.json(employees);

};

exports.addEmployee = async (req, res) => {

  try {

    const { username, password, name, department, position, role } = req.body;

    const newUser = new User({ username, password, role });

    await newUser.save();

    const newEmployee = new Employee({ userId: newUser.\_id, name, department, position });

    await newEmployee.save();

    res.json({ message: "Thêm nhân viên thành công" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

exports.getLeaveRequests = async (req, res) => {

  const leaves = await Leave.find().populate("userId", "username");

  res.json(leaves);

};

exports.approveLeave = async (req, res) => {

  const leave = await Leave.findByIdAndUpdate(req.params.id, { status: "approved" }, { new: true });

  res.json({ message: "Đã phê duyệt", leave });

};

exports.rejectLeave = async (req, res) => {

  const leave = await Leave.findByIdAndUpdate(req.params.id, { status: "rejected" }, { new: true });

  res.json({ message: "Đã từ chối", leave });

};

exports.addSalary = async (req, res) => {

  const { userId, amount, month } = req.body;

  const salary = new Salary({ userId, amount, month });

  await salary.save();

  res.json(salary);

};

exports.getSalaries = async (req, res) => {

  const list = await Salary.find().populate("userId", "username");

  res.json(list);

};

attendanceController.js

const Attendance = require("../models/Attendance");

// 📌 Check-in

const checkIn = async (req, res) => {

  try {

    const today = new Date().toISOString().split("T")[0]; // yyyy-mm-dd

    let attendance = await Attendance.findOne({

      userId: req.user.id,

      date: today

    });

    if (attendance) {

      return res.status(400).json({ error: "Bạn đã check-in hôm nay rồi" });

    }

    attendance = new Attendance({

      userId: req.user.id,

      date: today,

      checkIn: new Date()

    });

    await attendance.save();

    res.json({ message: "✅ Check-in thành công", attendance });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Check-out

const checkOut = async (req, res) => {

  try {

    const today = new Date().toISOString().split("T")[0];

    let attendance = await Attendance.findOne({

      userId: req.user.id,

      date: today

    });

    if (!attendance) {

      return res.status(400).json({ error: "Bạn chưa check-in hôm nay" });

    }

    if (attendance.checkOut) {

      return res.status(400).json({ error: "Bạn đã check-out rồi" });

    }

    attendance.checkOut = new Date();

    await attendance.save();

    res.json({ message: "✅ Check-out thành công", attendance });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Lấy lịch sử chấm công cá nhân

const getMyAttendance = async (req, res) => {

  try {

    const list = await Attendance.find({ userId: req.user.id }).sort({ date: -1 });

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Admin xem toàn bộ chấm công

const getAllAttendance = async (req, res) => {

  try {

    const list = await Attendance.find().populate("userId", "username role").sort({ date: -1 });

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

module.exports = {

  checkIn,

  checkOut,

  getMyAttendance,

  getAllAttendance

};

authController.jsconst bcrypt = require("bcryptjs");

const jwt = require("jsonwebtoken");

const User = require("../models/User");

const Employee = require("../models/Employee");

// Đăng ký

exports.register = async (req, res) => {

  try {

    const { username, password, name, department, position, role } = req.body;

    if (!username || !password || !name || !department || !position) {

      return res.status(400).json({ error: "Vui lòng nhập đầy đủ thông tin" });

    }

    const existing = await User.findOne({ username });

    if (existing) return res.status(400).json({ error: "Username đã tồn tại" });

    const hashedPassword = await bcrypt.hash(password, 10);

    const newUser = new User({ username, password: hashedPassword, role: role || "employee" });

    const savedUser = await newUser.save();

    const newEmployee = new Employee({ userId: savedUser.\_id, name, department, position });

    await newEmployee.save();

    res.json({ message: "Đăng ký thành công" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Đăng nhập

exports.login = async (req, res) => {

  try {

    const { username, password } = req.body;

    const user = await User.findOne({ username });

    if (!user) return res.status(400).json({ error: "Sai username hoặc password" });

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) return res.status(400).json({ error: "Sai username hoặc password" });

    const token = jwt.sign(

      { id: user.\_id, role: user.role },

      process.env.JWT\_SECRET || "secret",

      { expiresIn: "1d" }

    );

    res.json({ token, user: { id: user.\_id, username: user.username, role: user.role } });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

employeeController.js

const Employee = require("../models/Employee");

const User = require("../models/User");

const bcrypt = require("bcryptjs");

// 📌 Lấy danh sách nhân viên

exports.getEmployees = async (req, res) => {

  try {

    const employees = await Employee.find().populate("userId", "username role");

    res.json(employees);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Lấy 1 nhân viên theo ID

exports.getEmployee = async (req, res) => {

  try {

    const emp = await Employee.findById(req.params.id).populate("userId", "username role");

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    res.json(emp);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Tạo nhân viên mới (tạo cả User)

exports.createEmployee = async (req, res) => {

  try {

    const { username, password, name, department, position, role } = req.body;

    if (!username || !password || !name || !department || !position) {

      return res.status(400).json({ error: "Vui lòng nhập đầy đủ thông tin" });

    }

    // kiểm tra username trùng

    const existing = await User.findOne({ username });

    if (existing) return res.status(400).json({ error: "Username đã tồn tại" });

    // hash mật khẩu

    const hashed = await bcrypt.hash(password, 10);

    // tạo User

    const newUser = new User({

      username,

      password: hashed,

      role: role || "employee",

    });

    const savedUser = await newUser.save();

    // tạo Employee

    const newEmployee = new Employee({

      userId: savedUser.\_id,

      name,

      department,

      position,

    });

    await newEmployee.save();

    res.json({ message: "✅ Thêm nhân viên thành công", employee: newEmployee });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Cập nhật nhân viên

exports.updateEmployee = async (req, res) => {

  try {

    const { name, department, position, role } = req.body;

    const emp = await Employee.findById(req.params.id);

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    emp.name = name || emp.name;

    emp.department = department || emp.department;

    emp.position = position || emp.position;

    await emp.save();

    // cập nhật role trong User nếu có

    if (role) {

      await User.findByIdAndUpdate(emp.userId, { role });

    }

    res.json({ message: "✅ Cập nhật thành công", employee: emp });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Xóa nhân viên (và cả User liên kết)

exports.deleteEmployee = async (req, res) => {

  try {

    const emp = await Employee.findById(req.params.id);

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    await User.findByIdAndDelete(emp.userId);

    await emp.deleteOne();

    res.json({ message: "🗑️ Xóa nhân viên thành công" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

leaveController.js

const Leave = require("../models/Leave");

exports.createLeave = async (req, res) => {

  const { from, to, reason } = req.body;

  const leave = new Leave({ userId: req.user.id, from, to, reason });

  await leave.save();

  res.json(leave);

};

exports.getMyLeaves = async (req, res) => {

  const leaves = await Leave.find({ userId: req.user.id });

  res.json(leaves);

};

messageController.js

const Message = require("../models/Message");

exports.sendMessage = async (req, res) => {

  const { to, content } = req.body;

  const msg = new Message({ from: req.user.id, to, content });

  await msg.save();

  res.json(msg);

};

exports.getMessages = async (req, res) => {

  const list = await Message.find({

    $or: [

      { from: req.user.id, to: req.params.userId },

      { from: req.params.userId, to: req.user.id }

    ]

  }).populate("from to", "username");

  res.json(list);

};

salaryController.js

const Salary = require("../models/Salary");

const Employee = require("../models/Employee");

const Attendance = require("../models/Attendance");

const mongoose = require("mongoose");

// 📌 Admin: Lấy danh sách lương

exports.getAllSalaries = async (req, res) => {

  try {

    const employees = await Employee.find().populate("userId", "username role");

    let salaries = await Salary.find().populate("userId", "username role");

    // đảm bảo mỗi employee có 1 salary record

    const salaryMap = new Map(salaries.map(s => [s.userId?.\_id.toString(), s]));

    for (const emp of employees) {

      if (emp.userId && !salaryMap.has(emp.userId.\_id.toString())) {

        const fresh = new Salary({ userId: emp.userId.\_id, hourlyRate: 100000, amount: 0 });

        await fresh.save();

        salaries.push(await fresh.populate("userId", "username role"));

      }

    }

    // tổng giờ từ Attendance

    const totals = await Attendance.aggregate([

      { $group: { \_id: "$userId", totalHours: { $sum: { $ifNull: ["$totalHours", 0] } } } }

    ]);

    const totalMap = new Map(totals.map(t => [t.\_id.toString(), t.totalHours]));

    const result = salaries.map(s => {

      const userIdStr = s.userId?.\_id.toString();

      const emp = employees.find(e => e.userId?.\_id.toString() === userIdStr);

      const totalHours = totalMap.get(userIdStr) || 0;

      const amount = totalHours \* Number(s.hourlyRate || 0);

      return {

        \_id: s.\_id,

        userId: s.userId,

        username: s.userId?.username,

        name: emp?.name || "",

        department: emp?.department || "",

        position: emp?.position || "",

        totalHours,

        hourlyRate: s.hourlyRate,

        amount

      };

    });

    res.json(result);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 Admin: Cập nhật tiền giờ

exports.updateSalary = async (req, res) => {

  try {

    const { id } = req.params;

    const { hourlyRate } = req.body;

    const salary = await Salary.findById(id).populate("userId", "username role");

    if (!salary) return res.status(404).json({ error: "Không tìm thấy lương" });

    salary.hourlyRate = Number(hourlyRate);

    // tính lại tổng giờ

    const totals = await Attendance.aggregate([

      { $match: { userId: new mongoose.Types.ObjectId(salary.userId.\_id) } },

      { $group: { \_id: "$userId", totalHours: { $sum: { $ifNull: ["$totalHours", 0] } } } }

    ]);

    const totalHours = totals.length ? totals[0].totalHours : 0;

    salary.amount = totalHours \* salary.hourlyRate;

    await salary.save();

    res.json({ message: "✅ Cập nhật thành công", salary: { ...salary.toObject(), totalHours } });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// 📌 User: Xem lương cá nhân

exports.getMySalary = async (req, res) => {

  try {

    let s = await Salary.findOne({ userId: req.user.id }).populate("userId", "username role");

    if (!s) {

      s = new Salary({ userId: req.user.id, hourlyRate: 100000, amount: 0 });

      await s.save();

      s = await s.populate("userId", "username role");

    }

    const totals = await Attendance.aggregate([

      { $match: { userId: new mongoose.Types.ObjectId(req.user.id) } },

      { $group: { \_id: "$userId", totalHours: { $sum: { $ifNull: ["$totalHours", 0] } } } }

    ]);

    const totalHours = totals.length ? totals[0].totalHours : 0;

    const amount = totalHours \* Number(s.hourlyRate || 0);

    res.json([{

      \_id: s.\_id,

      userId: s.userId,

      username: s.userId?.username,

      totalHours,

      hourlyRate: s.hourlyRate,

      amount,

      date: s.date

    }]);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

Middleware

authMiddleware.js

const jwt = require("jsonwebtoken");

const User = require("../models/User");

module.exports = (roles = []) => {

  return async (req, res, next) => {

    try {

      const authHeader = req.headers.authorization;

      if (!authHeader || !authHeader.startsWith("Bearer ")) {

        return res.status(401).json({ error: "Không có token" });

      }

      const token = authHeader.split(" ")[1];

      const decoded = jwt.verify(token, process.env.JWT\_SECRET);

      // Tìm user trong DB

      const user = await User.findById(decoded.id);

      if (!user) {

        return res.status(401).json({ error: "User không tồn tại" });

      }

      req.user = user; // Gắn user đầy đủ vào request

      // Kiểm tra role nếu có truyền vào middleware

      if (roles.length && !roles.includes(user.role)) {

        return res.status(403).json({ error: "Không có quyền truy cập" });

      }

      next();

    } catch (err) {

      console.error("❌ Auth error:", err.message);

      res.status(401).json({ error: "Token không hợp lệ" });

    }

  };

};

errorHandler.js

function notFound(req, res, next) {

  const error = new Error(`Not Found - ${req.originalUrl}`);

  res.status(404);

  next(error);

}

function errorHandler(err, req, res, next) {

  const statusCode = res.statusCode === 200 ? 500 : res.statusCode;

  res.status(statusCode).json({

    message: err.message,

    stack: process.env.NODE\_ENV === "production" ? null : err.stack,

  });

}

module.exports = { notFound, errorHandler };

Models

Attendance.js

const mongoose = require("mongoose");

const AttendanceSchema = new mongoose.Schema({

  userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

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

  checkIn: { type: Date },

  checkOut: { type: Date },

  totalHours: { type: Number, default: 0 },

  status: { type: String, enum: ["Present", "Absent", "Leave"], default: "Present" },

});

module.exports = mongoose.model("Attendance", AttendanceSchema);

ChatRoom.js

const mongoose = require("mongoose");

const chatRoomSchema = new mongoose.Schema(

  {

    type: { type: String, enum: ["private", "group"], required: true },

    name: { type: String }, // tên nhóm, vd: "Phòng IT"

    department: { type: String }, // dùng cho group

    participants: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],

  },

  { timestamps: true }

);

module.exports = mongoose.model("ChatRoom", chatRoomSchema);

Document.js

const mongoose = require("mongoose");

const documentSchema = new mongoose.Schema({

  title: String,

  folder: { type: String, default: "Chung" }, // 📁 Thư mục tài liệu

  department: { type: String, default: "general" },

  filePath: String,

  fileType: String,

  uploadedBy: { type: mongoose.Schema.Types.ObjectId, ref: "User" },

  uploadedAt: { type: Date, default: Date.now },

  downloadedBy: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }]

});

module.exports = mongoose.model("Document", documentSchema);

Employee.js

const mongoose = require("mongoose");

const employeeSchema = new mongoose.Schema(

  {

    userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

    name: String,

    department: String,

    position: String,

    address: String,

    avatar: String

  },

  { timestamps: true }

);

module.exports = mongoose.model("Employee", employeeSchema);

Leave.js

const mongoose = require("mongoose");

const leaveSchema = new mongoose.Schema(

  {

    userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

    from: Date,

    to: Date,

    reason: String,

    status: { type: String, enum: ["pending", "approved", "rejected"], default: "pending" }

  },

  { timestamps: true }

);

module.exports = mongoose.model("Leave", leaveSchema);

Message.js

// backend/models/Message.js

const mongoose = require("mongoose");

const messageSchema = new mongoose.Schema(

  {

    roomId: { type: mongoose.Schema.Types.ObjectId, ref: "ChatRoom", required: true },

    sender: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

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

  },

  { timestamps: true }

);

module.exports = mongoose.model("Message", messageSchema);

Notification.js

const mongoose = require("mongoose");

const notificationSchema = new mongoose.Schema({

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

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

  target: { type: String, enum: ["all", "employee", "department"], default: "all" },

  targetValue: { type: String }, // userId hoặc department

  createdBy: { type: mongoose.Schema.Types.ObjectId, ref: "User" },

}, { timestamps: true });

module.exports = mongoose.model("Notification", notificationSchema);

PerformanceReview.js

const mongoose = require("mongoose");

const performanceReviewSchema = new mongoose.Schema({

  userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

  tasksCompleted: { type: Number, default: 0 }, // Số nhiệm vụ hoàn thành

  communication: { type: Number, default: 0 },  // Điểm giao tiếp

  technical: { type: Number, default: 0 },      // Điểm kỹ thuật

  attitude: { type: Number, default: 10 },      // Thái độ (0 hoặc 10)

  feedback: { type: String },                   // Nhận xét chi tiết

}, { timestamps: true });

module.exports = mongoose.model("PerformanceReview", performanceReviewSchema);

Salary.js

const mongoose = require("mongoose");

const salarySchema = new mongoose.Schema({

  userId: {

    type: mongoose.Schema.Types.ObjectId,

    ref: "User",

    required: true

  },

  hourlyRate: {

    type: Number,

    default: 100000 // mặc định 100k/giờ

  },

  amount: {

    type: Number,

    required: true,

    default: 0

  },

  date: {

    type: Date,

    default: Date.now

  },

  month: {

    type: String,

    required: false

  }

}, { timestamps: true });

module.exports = mongoose.model("Salary", salarySchema);

User.js

const mongoose = require("mongoose");

const userSchema = new mongoose.Schema(

  {

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

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

    role: { type: String, enum: ["admin", "manager", "employee"], default: "employee" }

  },

  { timestamps: true }

);

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

WorkSchedule.js

const mongoose = require("mongoose");

const workScheduleSchema = new mongoose.Schema({

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

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

  assignedTo: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

  startDate: { type: Date, required: true },

  endDate: { type: Date, required: true },

  createdAt: { type: Date, default: Date.now },

});

module.exports = mongoose.model("WorkSchedule", workScheduleSchema);

node\_modules

Routes

adminRoutes.js

const express = require("express");

const bcrypt = require("bcryptjs");

const mongoose = require("mongoose");

const multer = require("multer");

const path = require("path");

const fs = require("fs");

// ======================= MODELS =======================

const User = require("../models/User");

const Employee = require("../models/Employee");

const Leave = require("../models/Leave");

const Salary = require("../models/Salary");

const Attendance = require("../models/Attendance");

const PerformanceReview = require("../models/PerformanceReview");

const Document = require("../models/Document");

const WorkSchedule = require("../models/WorkSchedule");

const Notification = require("../models/Notification");

// ======================= MIDDLEWARE =======================

const auth = require("../middleware/authMiddleware");

const router = express.Router();

/\* ========================= NHÂN VIÊN ========================= \*/

// Lấy danh sách nhân viên

router.get("/employees", auth(["admin"]), async (req, res) => {

  try {

    const employees = await Employee.find().populate("userId", "username role");

    res.json(employees);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Thêm nhân viên

router.post("/employees", auth(["admin"]), async (req, res) => {

  try {

    const { username, password, name, department, position, role } = req.body;

    if (!username || !password || !name || !department || !position) {

      return res.status(400).json({ error: "Vui lòng nhập đầy đủ thông tin" });

    }

    const existing = await User.findOne({ username });

    if (existing) return res.status(400).json({ error: "Username đã tồn tại" });

    const hashed = await bcrypt.hash(password, 10);

    const newUser = new User({ username, password: hashed, role: role || "employee" });

    const savedUser = await newUser.save();

    const newEmployee = new Employee({

      userId: savedUser.\_id,

      name,

      department,

      position,

    });

    await newEmployee.save();

    const newSalary = new Salary({

      userId: savedUser.\_id,

      hourlyRate: 100000,

      amount: 0,

      date: new Date(),

    });

    await newSalary.save();

    res.json({

      message: "✅ Thêm nhân viên & tạo lương mặc định thành công",

      employee: newEmployee,

      salary: newSalary,

    });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Cập nhật nhân viên

router.put("/employees/:id", auth(["admin"]), async (req, res) => {

  try {

    const { name, department, position, role } = req.body;

    const emp = await Employee.findById(req.params.id);

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    emp.name = name || emp.name;

    emp.department = department || emp.department;

    emp.position = position || emp.position;

    await emp.save();

    if (role) await User.findByIdAndUpdate(emp.userId, { role });

    res.json({ message: "✅ Cập nhật thành công", employee: emp });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Xóa nhân viên

router.delete("/employees/:id", auth(["admin"]), async (req, res) => {

  try {

    const emp = await Employee.findById(req.params.id);

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    await User.findByIdAndDelete(emp.userId);

    await Salary.deleteMany({ userId: emp.userId });

    await Attendance.deleteMany({ userId: emp.userId });

    await emp.deleteOne();

    res.json({ message: "🗑️ Đã xóa nhân viên + dữ liệu liên quan" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= NGHỈ PHÉP ========================= \*/

router.get("/leave-requests", auth(["admin"]), async (req, res) => {

  try {

    const leaves = await Leave.find().populate("userId", "username");

    res.json(leaves);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.put("/leave-requests/:id/approve", auth(["admin"]), async (req, res) => {

  const leave = await Leave.findByIdAndUpdate(req.params.id, { status: "approved" }, { new: true });

  res.json({ message: "✅ Đã phê duyệt", leave });

});

router.put("/leave-requests/:id/reject", auth(["admin"]), async (req, res) => {

  const leave = await Leave.findByIdAndUpdate(req.params.id, { status: "rejected" }, { new: true });

  res.json({ message: "❌ Đã từ chối", leave });

});

/\* ========================= LƯƠNG ========================= \*/

router.get("/salary", auth(["admin"]), async (req, res) => {

  try {

    const employees = await Employee.find().populate("userId", "username role");

    let salaries = await Salary.find().populate("userId", "username role");

    const hasSalary = new Set(salaries.map(s => s.userId?.\_id.toString()));

    for (const emp of employees) {

      if (emp.userId && !hasSalary.has(emp.userId.\_id.toString())) {

        const fresh = new Salary({ userId: emp.userId.\_id, hourlyRate: 100000, amount: 0 });

        await fresh.save();

        salaries.push(await fresh.populate("userId", "username role"));

      }

    }

    const totals = await Attendance.aggregate([

      { $group: { \_id: "$userId", totalHours: { $sum: { $ifNull: ["$totalHours", 0] } } } }

    ]);

    const totalMap = new Map(totals.map(t => [t.\_id.toString(), t.totalHours || 0]));

    const result = salaries.map(s => {

      const emp = employees.find(e => e.userId?.\_id.toString() === s.userId?.\_id.toString());

      const totalHours = totalMap.get(s.userId?.\_id.toString()) || 0;

      const amount = totalHours \* (s.hourlyRate || 0);

      return {

        \_id: s.\_id,

        userId: s.userId,

        name: emp?.name,

        department: emp?.department,

        position: emp?.position,

        totalHours,

        hourlyRate: s.hourlyRate,

        amount,

      };

    });

    res.json(result);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= CHẤM CÔNG ========================= \*/

router.get("/attendance", auth(["admin"]), async (req, res) => {

  const list = await Attendance.find().populate("userId", "username role").sort({ date: -1 });

  res.json(list);

});

/\* ========================= ĐÁNH GIÁ HIỆU SUẤT ========================= \*/

router.get("/performance", auth(["admin"]), async (req, res) => {

  try {

    const reviews = await PerformanceReview.find().populate("userId", "username role");

    res.json(reviews);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.post("/performance", auth(["admin"]), async (req, res) => {

  try {

    const review = new PerformanceReview(req.body);

    await review.save();

    res.json({ message: "✅ Đã thêm đánh giá", review });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.put("/performance/:id", auth(["admin"]), async (req, res) => {

  try {

    const updated = await PerformanceReview.findByIdAndUpdate(req.params.id, req.body, { new: true })

      .populate("userId", "username");

    if (!updated) return res.status(404).json({ error: "Không tìm thấy đánh giá" });

    res.json({ message: "✅ Đã cập nhật đánh giá", review: updated });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.delete("/performance/:id", auth(["admin"]), async (req, res) => {

  try {

    const deleted = await PerformanceReview.findByIdAndDelete(req.params.id);

    if (!deleted) return res.status(404).json({ error: "Không tìm thấy đánh giá" });

    res.json({ message: "🗑️ Đã xóa đánh giá" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= QUẢN LÝ TÀI LIỆU ========================= \*/

// cấu hình multer

const storage = multer.diskStorage({

  destination: (req, file, cb) => {

    const department = req.body.department || "general";

    const dir = `uploads/documents/${department}`;

    if (!fs.existsSync(dir)) fs.mkdirSync(dir, { recursive: true });

    cb(null, dir);

  },

  filename: (req, file, cb) => {

    cb(null, Date.now() + "-" + file.originalname);

  },

});

const upload = multer({ storage });

// 📌 lấy danh sách tài liệu

router.get("/documents", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      filter = { department: emp?.department || "general" };

    }

    const docs = await Document.find(filter)

      .populate("uploadedBy", "username")

      .sort({ uploadedAt: -1 });

    res.json(docs);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// 📌 upload

router.post("/documents", auth(["admin"]), upload.single("file"), async (req, res) => {

  try {

    if (!req.file) return res.status(400).json({ error: "Vui lòng chọn file" });

    const doc = new Document({

      title: req.body.title,

      department: req.body.department || "general",

      filePath: req.file.path,

      fileType: req.file.mimetype,

      uploadedBy: req.user.id,

    });

    await doc.save();

    res.json({ message: "✅ Upload thành công", doc });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// 📌 tải file

router.get("/documents/download/:id", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    const doc = await Document.findById(req.params.id);

    if (!doc) return res.status(404).json({ error: "Không tìm thấy tài liệu" });

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      if (!emp || emp.department !== doc.department) {

        return res.status(403).json({ error: "🚫 Không có quyền tải tài liệu này" });

      }

    }

    const filePath = path.resolve(doc.filePath);

    if (!fs.existsSync(filePath)) return res.status(404).json({ error: "File không tồn tại" });

    res.setHeader(

      "Content-Disposition",

      `attachment; filename="${doc.title}${path.extname(doc.filePath)}"`

    );

    res.setHeader("Content-Type", doc.fileType);

    res.sendFile(filePath);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// 📌 xóa tài liệu

router.delete("/documents/:id", auth(["admin"]), async (req, res) => {

  try {

    const doc = await Document.findById(req.params.id);

    if (!doc) return res.status(404).json({ error: "Không tìm thấy tài liệu" });

    if (fs.existsSync(doc.filePath)) fs.unlinkSync(doc.filePath);

    await doc.deleteOne();

    res.json({ message: "🗑️ Đã xóa tài liệu" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= LỊCH LÀM VIỆC ========================= \*/

// Lấy danh sách lịch (Admin)

router.get("/work-schedule", auth(["admin"]), async (req, res) => {

  try {

    const schedules = await WorkSchedule.find()

      .populate("assignedTo", "username role")

      .sort({ startDate: -1 });

    res.json(schedules);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Thêm mới lịch

router.post("/work-schedule", auth(["admin"]), async (req, res) => {

  try {

    const { task, department, assignedTo, startDate, endDate } = req.body;

    if (!task || !assignedTo || !startDate || !endDate)

      return res.status(400).json({ error: "Vui lòng nhập đầy đủ thông tin" });

    const schedule = new WorkSchedule({

      task,

      department,

      assignedTo,

      startDate,

      endDate,

    });

    await schedule.save();

    res.json({ message: "✅ Đã thêm lịch làm việc", schedule });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Cập nhật lịch

router.put("/work-schedule/:id", auth(["admin"]), async (req, res) => {

  try {

    const { task, department, assignedTo, startDate, endDate } = req.body;

    const updated = await WorkSchedule.findByIdAndUpdate(

      req.params.id,

      { task, department, assignedTo, startDate, endDate },

      { new: true }

    );

    if (!updated) return res.status(404).json({ error: "Không tìm thấy lịch" });

    res.json({ message: "✅ Cập nhật lịch thành công", schedule: updated });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Xóa lịch

router.delete("/work-schedule/:id", auth(["admin"]), async (req, res) => {

  try {

    const deleted = await WorkSchedule.findByIdAndDelete(req.params.id);

    if (!deleted) return res.status(404).json({ error: "Không tìm thấy lịch" });

    res.json({ message: "🗑️ Đã xóa lịch làm việc" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= THÔNG BÁO ========================= \*/

router.get("/notifications", auth(["admin"]), async (req, res) => {

  const noti = await Notification.find()

    .populate("createdBy", "username")

    .sort({ createdAt: -1 });

  res.json(noti);

});

router.post("/notifications", auth(["admin"]), async (req, res) => {

  try {

    const { title, message, target, targetValue } = req.body;

    const noti = new Notification({

      title,

      message,

      target,

      targetValue,

      createdBy: req.user.id,

    });

    await noti.save();

    res.json({ message: "✅ Đã gửi thông báo", noti });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.put("/notifications/:id", auth(["admin"]), async (req, res) => {

  try {

    const updated = await Notification.findByIdAndUpdate(req.params.id, req.body, { new: true });

    if (!updated) return res.status(404).json({ error: "Không tìm thấy thông báo" });

    res.json({ message: "✅ Cập nhật thông báo thành công", noti: updated });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

router.delete("/notifications/:id", auth(["admin"]), async (req, res) => {

  try {

    const deleted = await Notification.findByIdAndDelete(req.params.id);

    if (!deleted) return res.status(404).json({ error: "Không tìm thấy thông báo" });

    res.json({ message: "🗑️ Đã xóa thông báo" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

attendanceRoutes.js

const express = require("express");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

const Attendance = require("../models/Attendance");

const Employee = require("../models/Employee");

// ========================= LẤY LỊCH SỬ CHẤM CÔNG =========================

router.get("/", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      filter.userId = req.user.id;

    }

    const records = await Attendance.find(filter)

      .populate("userId", "username role")

      .sort({ date: -1 });

    res.json(records);

  } catch (err) {

    console.error("❌ Lỗi lấy chấm công:", err);

    res.status(500).json({ error: err.message });

  }

});

// ========================= CHECK-IN =========================

router.post("/check-in", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    const today = new Date().toISOString().split("T")[0];

    const existed = await Attendance.findOne({ userId: req.user.id, date: today });

    if (existed) return res.status(400).json({ error: "Hôm nay bạn đã Check-in rồi." });

    const record = new Attendance({

      userId: req.user.id,

      date: today,

      checkIn: new Date(),

      status: "Present",

    });

    await record.save();

    res.json({ message: "✅ Check-in thành công", record });

  } catch (err) {

    console.error("❌ Lỗi check-in:", err);

    res.status(500).json({ error: err.message });

  }

});

// ========================= CHECK-OUT =========================

router.post("/check-out", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    const today = new Date().toISOString().split("T")[0];

    const record = await Attendance.findOne({ userId: req.user.id, date: today });

    if (!record) return res.status(404).json({ error: "Bạn chưa Check-in hôm nay." });

    if (record.checkOut) return res.status(400).json({ error: "Bạn đã Check-out rồi." });

    record.checkOut = new Date();

    record.totalHours =

      (new Date(record.checkOut) - new Date(record.checkIn)) / 1000 / 60 / 60;

    await record.save();

    res.json({ message: "✅ Check-out thành công", record });

  } catch (err) {

    console.error("❌ Lỗi check-out:", err);

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

authRoutes.js

const express = require("express");

const { register, login } = require("../controllers/authController");

const router = express.Router();

router.post("/register", register);

router.post("/login", login);

module.exports = router;

employeeRoutes.js

// backend/routes/employeeRoutes.js

const express = require("express");

const path = require("path");

const fs = require("fs");

const bcrypt = require("bcryptjs");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

// Models

const User = require("../models/User");

const Employee = require("../models/Employee");

const Document = require("../models/Document");

const WorkSchedule = require("../models/WorkSchedule");

const Notification = require("../models/Notification");

const PerformanceReview = require("../models/PerformanceReview");

const Attendance = require("../models/Attendance");

/\* ========================= HỒ SƠ CÁ NHÂN ========================= \*/

// Lấy hồ sơ của chính mình

router.get("/me", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const emp = await Employee.findOne({ userId: req.user.id }).populate("userId", "username role");

    if (!emp) return res.status(404).json({ error: "Không tìm thấy hồ sơ" });

    res.json(emp);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Cập nhật hồ sơ của chính mình (CHỈ CHO PHÉP ĐỔI TÊN)

router.put("/profile", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const emp = await Employee.findOne({ userId: req.user.id });

    if (!emp) return res.status(404).json({ error: "Không tìm thấy hồ sơ" });

    const name = typeof req.body.name === "string" ? req.body.name.trim() : "";

    if (!name) return res.status(400).json({ error: "Tên hiển thị không được để trống" });

    emp.name = name;

    await emp.save();

    res.json({ message: "✅ Cập nhật thông tin thành công", employee: emp });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Đổi mật khẩu

router.put("/change-password", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const { oldPassword, newPassword } = req.body || {};

    if (!oldPassword || !newPassword) {

      return res.status(400).json({ error: "Thiếu mật khẩu cũ hoặc mật khẩu mới" });

    }

    if (String(newPassword).length < 6) {

      return res.status(400).json({ error: "Mật khẩu mới phải từ 6 ký tự" });

    }

    // Lấy user hiện tại

    const user = await User.findById(req.user.id);

    if (!user) return res.status(404).json({ error: "Không tìm thấy tài khoản" });

    // Kiểm tra mật khẩu cũ

    const ok = await bcrypt.compare(oldPassword, user.password);

    if (!ok) return res.status(400).json({ error: "Mật khẩu cũ không đúng" });

    // Cập nhật mật khẩu mới

    user.password = await bcrypt.hash(newPassword, 10);

    await user.save();

    res.json({ message: "✅ Đổi mật khẩu thành công" });

  } catch (err) {

    console.error("❌ change-password error:", err);

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= DANH SÁCH NHÂN VIÊN (phục vụ chat) ========================= \*/

router.get("/employees", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const { departmentOnly } = req.query; // departmentOnly=true -> chỉ lấy cùng phòng

    let filter = {};

    if (departmentOnly === "true" && req.user.role !== "admin") {

      const me = await Employee.findOne({ userId: req.user.id });

      filter = { department: me?.department || "\_\_none\_\_" };

    }

    const employees = await Employee.find(filter).populate("userId", "username role");

    res.json(employees);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Danh sách phòng ban (distinct)

router.get("/departments", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const deps = await Employee.distinct("department");

    res.json(deps.filter(Boolean));

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= TÀI LIỆU (theo phòng ban) ========================= \*/

// Liệt kê tài liệu user được phép xem (phòng ban của mình + "general")

router.get("/documents", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      const dept = emp?.department || "general";

      filter = { department: { $in: [dept, "general"] } };

    }

    const docs = await Document.find(filter)

      .populate("uploadedBy", "username")

      .sort({ uploadedAt: -1 });

    res.json(docs);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Tải tài liệu (chỉ admin hoặc đúng phòng ban)

router.get("/documents/download/:id", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const doc = await Document.findById(req.params.id);

    if (!doc) return res.status(404).json({ error: "Không tìm thấy tài liệu" });

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      const dept = emp?.department || "\_\_none\_\_";

      if (!(doc.department === "general" || doc.department === dept)) {

        return res.status(403).json({ error: "🚫 Không có quyền tải tài liệu này" });

      }

    }

    const filePath = path.resolve(doc.filePath);

    if (!fs.existsSync(filePath)) return res.status(404).json({ error: "File không tồn tại" });

    res.setHeader(

      "Content-Disposition",

      `attachment; filename="${doc.title}${path.extname(doc.filePath)}"`

    );

    res.setHeader("Content-Type", doc.fileType || "application/octet-stream");

    res.sendFile(filePath);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= LỊCH LÀM VIỆC (cá nhân + phòng ban) ========================= \*/

router.get("/work-schedule", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      const dept = emp?.department || "\_\_none\_\_";

      filter = {

        $or: [

          { assignedTo: req.user.id }, // task gán cho cá nhân

          { department: dept },        // task gán cho phòng ban

        ],

      };

    }

    const list = await WorkSchedule.find(filter)

      .populate("assignedTo", "username")

      .sort({ startDate: 1 });

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= THÔNG BÁO (lọc theo đích) ========================= \*/

router.get("/notifications", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    if (req.user.role === "admin") {

      const all = await Notification.find().populate("createdBy", "username").sort({ createdAt: -1 });

      return res.json(all);

    }

    const emp = await Employee.findOne({ userId: req.user.id });

    const dept = emp?.department || "\_\_none\_\_";

    const list = await Notification.find({

      $or: [

        { target: "all" },

        { target: "user", targetValue: req.user.id },

        { target: "department", targetValue: dept },

      ],

    })

      .populate("createdBy", "username")

      .sort({ createdAt: -1 });

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= HIỆU SUẤT: xem của chính mình ========================= \*/

router.get("/performance", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const list =

      req.user.role === "admin"

        ? await PerformanceReview.find().populate("userId", "username")

        : await PerformanceReview.find({ userId: req.user.id }).populate("userId", "username");

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= BÁO CÁO (gửi lên) ========================= \*/

// Dùng Notification như hộp thư đến admin

router.post("/report", auth(["employee", "manager"]), async (req, res) => {

  try {

    const { title, content } = req.body;

    if (!title || !content) return res.status(400).json({ error: "Thiếu tiêu đề hoặc nội dung" });

    const noti = new Notification({

      title: `Báo cáo: ${title}`,

      message: content,

      target: "all", // hoặc target riêng admin nếu bạn muốn

      createdBy: req.user.id,

    });

    await noti.save();

    res.json({ message: "✅ Gửi báo cáo thành công", report: noti });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\* ========================= CHẤM CÔNG ========================= \*/

const dayKey = (d = new Date()) => {

  const y = d.getUTCFullYear();

  const m = String(d.getUTCMonth() + 1).padStart(2, "0");

  const day = String(d.getUTCDate()).padStart(2, "0");

  return `${y}-${m}-${day}`;

};

// Lấy danh sách chấm công của chính mình

router.get("/attendance", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") filter = { userId: req.user.id };

    const list = await Attendance.find(filter)

      .populate("userId", "username")

      .sort({ date: -1 });

    res.json(list);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

// Check-in

router.post("/attendance/check-in", auth(["employee", "manager"]), async (req, res) => {

  try {

    const key = dayKey();

    const existing = await Attendance.findOne({ userId: req.user.id, date: key });

    if (existing) return res.status(400).json({ error: "⚠️ Bạn đã điểm danh hôm nay" });

    const att = new Attendance({

      userId: req.user.id,

      date: key,

      checkIn: new Date(),

      totalHours: 0,

    });

    await att.save();

    res.json({ message: "✅ Check-in thành công", attendance: att });

  } catch (err) {

    console.error("❌ Lỗi check-in:", err);

    res.status(500).json({ error: err.message });

  }

});

// Check-out

router.post("/attendance/check-out", auth(["employee", "manager"]), async (req, res) => {

  try {

    const key = dayKey();

    const att = await Attendance.findOne({ userId: req.user.id, date: key });

    if (!att) return res.status(400).json({ error: "⚠️ Bạn chưa check-in hôm nay" });

    if (att.checkOut) return res.status(400).json({ error: "⚠️ Bạn đã check-out rồi" });

    att.checkOut = new Date();

    if (att.checkIn) {

      att.totalHours = Math.max(0, (att.checkOut - att.checkIn) / 3600000);

    }

    await att.save();

    res.json({ message: "✅ Check-out thành công", attendance: att });

  } catch (err) {

    console.error("❌ Lỗi check-out:", err);

    res.status(500).json({ error: err.message });

  }

});

// Peers cho chat (dept/all)

router.get("/peers", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const scope = String(req.query.scope || "dept").toLowerCase(); // 'dept' | 'all'

    let filter = {};

    if (req.user.role === "admin") {

      filter = {};

    } else {

      const me = await Employee.findOne({ userId: req.user.id }).populate("userId", "username");

      if (!me) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

      filter = scope === "all" ? {} : { department: me.department };

    }

    const list = await Employee.find(filter)

      .populate("userId", "username")

      .sort({ name: 1 });

    const result = list.map((e) => ({

      \_id: e.\_id,

      userId: e.userId?.\_id?.toString(),

      username: e.userId?.username || "",

      name: e.name || "",

      department: e.department || "",

    }));

    res.json(result);

  } catch (err) {

    console.error("❌ /employees/peers error:", err);

    res.status(500).json({ error: err.message });

  }

});

// Nhân viên cùng phòng ban (phục vụ chat)

router.get("/my-department", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const me = await Employee.findOne({ userId: req.user.id }).populate("userId", "username role");

    if (!me) return res.status(404).json({ error: "Không tìm thấy hồ sơ nhân viên" });

    const sameDept = await Employee.find({

      department: me.department,

      userId: { $ne: req.user.id }, // loại chính mình

    }).populate("userId", "username role");

    res.json(sameDept);

  } catch (err) {

    console.error("❌ /employees/my-department error:", err);

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

leaveRoutes.js

const express = require("express");

const { createLeave, getMyLeaves } = require("../controllers/leaveController");

const auth = require("../middleware/authMiddleware");

const router = express.Router();

router.post("/", auth(["employee", "manager"]), createLeave);

router.get("/me", auth(["employee", "manager"]), getMyLeaves);

module.exports = router;

messageRoutes.js

const express = require("express");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

const ChatRoom = require("../models/ChatRoom");

const Message = require("../models/Message");

const Employee = require("../models/Employee");

/\*\*

 \* 📜 Lấy danh sách phòng chat

 \*/

router.get("/rooms", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const myId = req.user.id;

    const emp = await Employee.findOne({ userId: myId });

    const dept = emp?.department || "Chưa phân phòng";

    // Danh sách phòng private

    const privateRooms = await ChatRoom.find({

      type: "private",

      participants: myId,

    })

      .populate("participants", "username")

      .sort({ updatedAt: -1 });

    // Tạo hoặc lấy phòng ban

    let deptRoom = await ChatRoom.findOne({ type: "group", department: dept });

    if (!deptRoom) {

      deptRoom = await new ChatRoom({

        type: "group",

        name: `Phòng ${dept}`,

        department: dept,

        participants: [myId],

      }).save();

    }

    res.json({ privateRooms, deptRoom });

  } catch (err) {

    console.error("❌ Lỗi load rooms:", err);

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 🔸 Tạo hoặc mở phòng private giữa 2 user

 \*/

router.post("/rooms/private", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const { otherUserId } = req.body;

    const myId = req.user.id;

    let room = await ChatRoom.findOne({

      type: "private",

      participants: { $all: [myId, otherUserId] },

    }).populate("participants", "username");

    if (!room) {

      room = await new ChatRoom({

        type: "private",

        participants: [myId, otherUserId],

      }).save();

      await room.populate("participants", "username");

    }

    res.json(room);

  } catch (err) {

    console.error("❌ Lỗi tạo phòng:", err);

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 💬 Lấy lịch sử tin nhắn theo roomId

 \*/

router.get("/:roomId", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const { roomId } = req.params;

    const messages = await Message.find({ roomId })

      .populate("sender", "username")

      .sort({ createdAt: 1 });

    res.json(messages);

  } catch (err) {

    console.error("❌ Lỗi lấy tin nhắn:", err);

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* ✉️ Gửi tin nhắn (lưu luôn DB)

 \*/

router.post("/", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    const { roomId, content } = req.body;

    if (!roomId || !content) return res.status(400).json({ error: "Thiếu dữ liệu" });

    const room = await ChatRoom.findById(roomId);

    if (!room) return res.status(404).json({ error: "Không tìm thấy phòng chat" });

    const msg = new Message({

      roomId,

      sender: req.user.id,

      content,

    });

    await msg.save();

    // Cập nhật thời gian hoạt động của phòng

    room.updatedAt = new Date();

    await room.save();

    const populatedMsg = await msg.populate("sender", "username");

    res.json(populatedMsg);

  } catch (err) {

    console.error("❌ Lỗi gửi tin nhắn:", err);

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

notificationRoutes.js

const express = require("express");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

const Notification = require("../models/Notification");

const Employee = require("../models/Employee");

/\*\*

 \* 📢 Lấy danh sách thông báo cho từng user

 \* - Admin: thấy toàn bộ thông báo

 \* - Nhân viên: thấy thông báo của phòng ban mình + thông báo chung

 \*/

router.get("/", auth(["admin", "employee", "manager"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      const dept = emp?.department || "general";

      filter = {

        $or: [

          { target: "all" },

          { target: "department", targetValue: dept },

          { target: "user", targetValue: req.user.id },

        ],

      };

    }

    const list = await Notification.find(filter)

      .populate("createdBy", "username")

      .sort({ createdAt: -1 });

    res.json(list);

  } catch (err) {

    console.error("❌ Lỗi lấy thông báo:", err);

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 📨 Gửi thông báo (chỉ admin)

 \* target:

 \*  - all: toàn bộ

 \*  - department: phòng ban cụ thể

 \*  - user: cá nhân

 \*/

router.post("/", auth(["admin"]), async (req, res) => {

  try {

    const { title, message, target, targetValue } = req.body;

    const noti = new Notification({

      title,

      message,

      target: target || "all",

      targetValue: targetValue || null,

      createdBy: req.user.id,

    });

    await noti.save();

    res.json({ message: "✅ Đã gửi thông báo", noti });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 🗑️ Xóa thông báo (admin)

 \*/

router.delete("/:id", auth(["admin"]), async (req, res) => {

  try {

    const deleted = await Notification.findByIdAndDelete(req.params.id);

    if (!deleted) return res.status(404).json({ error: "Không tìm thấy thông báo" });

    res.json({ message: "🗑️ Đã xóa thông báo" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

reportRoutes.js

// backend/routes/reportRoutes.js

const express = require("express");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

const Attendance = require("../models/Attendance");

const PerformanceReview = require("../models/PerformanceReview");

const WorkSchedule = require("../models/WorkSchedule");

const Document = require("../models/Document");

const Employee = require("../models/Employee");

/\*\*

 \* 📊 Báo cáo tổng hợp cá nhân

 \*/

router.get("/me", auth(["employee", "manager"]), async (req, res) => {

  try {

    const userId = req.user.id;

    // Lấy thông tin nhân viên

    const emp = await Employee.findOne({ userId }).select("name department");

    if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

    // 🕒 Tổng giờ làm

    const attendance = await Attendance.find({ userId });

    const totalHours = attendance.reduce((sum, a) => sum + (a.totalHours || 0), 0);

    // 📅 Nhiệm vụ được giao

    const tasksCompleted = await WorkSchedule.countDocuments({ assignedTo: userId });

    // 📈 Hiệu suất trung bình

    const perf = await PerformanceReview.find({ userId });

    const avgScore =

      perf.length > 0

        ? perf.reduce((sum, p) => sum + (p.score || 0), 0) / perf.length

        : 0;

    // 📂 Tài liệu đã tải (nếu bạn có trường downloadedBy)

    const docsDownloaded = await Document.countDocuments({ downloadedBy: userId });

    res.json({

      name: emp.name,

      department: emp.department,

      totalHours,

      tasksCompleted,

      avgScore: Number(avgScore.toFixed(2)),

      docsDownloaded,

    });

  } catch (err) {

    console.error("❌ Lỗi tổng hợp báo cáo:", err);

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

salaryRoutes.js

const express = require("express");

const { getAllSalaries, updateSalary, getMySalary } = require("../controllers/salaryController");

const auth = require("../middleware/authMiddleware");

const router = express.Router();

// Admin quản lý toàn bộ lương

router.get("/", auth(["admin"]), getAllSalaries);

router.put("/:id", auth(["admin"]), updateSalary);

// User xem lương của mình

router.get("/me", auth(["employee", "manager", "admin"]), getMySalary);

module.exports = router;

userRoutes.js

const express = require("express");

const WorkSchedule = require("../models/WorkSchedule");

const auth = require("../middleware/authMiddleware");

const mongoose = require("mongoose");

const router = express.Router();

// Nhân viên xem lịch của mình hoặc phòng ban

router.get("/work-schedule/my", auth(["employee", "manager"]), async (req, res) => {

  const userId = new mongoose.Types.ObjectId(req.user.id);

  const list = await WorkSchedule.find({

    $or: [

      { assignedTo: userId },

      { department: req.user.department }

    ]

  }).sort({ startDate: 1 });

  res.json(list);

});

// Nhân viên cập nhật trạng thái công việc

router.put("/work-schedule/:id/status", auth(["employee", "manager"]), async (req, res) => {

  const { status } = req.body;

  const task = await WorkSchedule.findOneAndUpdate(

    { \_id: req.params.id, assignedTo: req.user.id },

    { status },

    { new: true }

  );

  if (!task) return res.status(403).json({ error: "Không có quyền cập nhật" });

  res.json({ message: "✅ Đã cập nhật trạng thái", task });

});

module.exports = router;

workScheduleRoutes.js

// backend/routes/workScheduleRoutes.js

const express = require("express");

const router = express.Router();

const auth = require("../middleware/authMiddleware");

const WorkSchedule = require("../models/WorkSchedule");

const Employee = require("../models/Employee");

/\*\*

 \* 📅 Nhân viên xem lịch làm việc của mình hoặc phòng ban

 \*/

router.get("/", auth(["employee", "manager", "admin"]), async (req, res) => {

  try {

    let filter = {};

    if (req.user.role !== "admin") {

      const emp = await Employee.findOne({ userId: req.user.id });

      if (!emp) return res.status(404).json({ error: "Không tìm thấy nhân viên" });

      filter = {

        $or: [

          { department: emp.department },

          { assignedTo: req.user.id },

        ],

      };

    }

    const schedules = await WorkSchedule.find(filter)

      .populate("assignedTo", "username")

      .sort({ startDate: 1 });

    res.json(schedules);

  } catch (err) {

    console.error("❌ Lỗi lấy lịch làm việc:", err);

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 🧾 Admin thêm mới lịch làm việc

 \*/

router.post("/", auth(["admin"]), async (req, res) => {

  try {

    const { task, department, assignedTo, startDate, endDate } = req.body;

    if (!task || !startDate || !endDate)

      return res.status(400).json({ error: "Thiếu thông tin lịch làm việc" });

    const schedule = new WorkSchedule({

      task,

      department,

      assignedTo,

      startDate,

      endDate,

      createdAt: new Date(),

    });

    await schedule.save();

    res.json({ message: "✅ Đã thêm lịch làm việc", schedule });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* ✏️ Admin chỉnh sửa lịch

 \*/

router.put("/:id", auth(["admin"]), async (req, res) => {

  try {

    const updated = await WorkSchedule.findByIdAndUpdate(req.params.id, req.body, { new: true });

    if (!updated) return res.status(404).json({ error: "Không tìm thấy lịch" });

    res.json({ message: "✅ Cập nhật lịch thành công", schedule: updated });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

/\*\*

 \* 🗑️ Admin xóa lịch

 \*/

router.delete("/:id", auth(["admin"]), async (req, res) => {

  try {

    const deleted = await WorkSchedule.findByIdAndDelete(req.params.id);

    if (!deleted) return res.status(404).json({ error: "Không tìm thấy lịch" });

    res.json({ message: "🗑️ Đã xóa lịch làm việc" });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

});

module.exports = router;

Uploads/documents  
server.js

// backend/server.js

const express = require("express");

const mongoose = require("mongoose");

const dotenv = require("dotenv");

const cors = require("cors");

const http = require("http");

const { Server } = require("socket.io");

dotenv.config();

const app = express();

const server = http.createServer(app);

// ===== SOCKET.IO =====

const io = new Server(server, { cors: { origin: "\*" } });

io.on("connection", (socket) => {

  console.log("🔌 Socket connected:", socket.id);

  // client emit 'join' với { userId, department }

  socket.on("join", ({ userId, department }) => {

    if (userId) socket.join(`user:${userId}`);

    if (department) socket.join(`dept:${department}`);

  });

  // Gửi tin nhắn realtime theo roomId hoặc department

  socket.on("send\_message", (payload) => {

    if (payload.type === "private" && payload.toUserId) {

      io.to(`user:${payload.toUserId}`).emit("receive\_message", payload);

      if (payload.fromUserId) {

        io.to(`user:${payload.fromUserId}`).emit("receive\_message", payload);

      }

    } else if (payload.type === "group" && payload.department) {

      io.to(`dept:${payload.department}`).emit("receive\_message", payload);

    } else if (payload.roomId) {

      io.to(payload.roomId).emit("receive\_message", payload);

    } else {

      io.emit("receive\_message", payload);

    }

  });

  socket.on("disconnect", () => {

    console.log("🔌 Socket disconnected:", socket.id);

  });

});

// ===== MIDDLEWARE =====

app.use(cors({ origin: process.env.CORS\_ORIGIN || "\*" }));

app.use(express.json());

// Static files

app.use("/uploads", express.static("uploads"));

// ===== KẾT NỐI MONGODB =====

mongoose

  .connect(process.env.MONGO\_URI)

  .then(() => console.log("✅ MongoDB Connected"))

  .catch((err) => {

    console.error("❌ MongoDB connection error:", err.message);

    process.exit(1);

  });

// ===== ROUTES =====

app.get("/", (req, res) => res.json({ ok: true, service: "HRM Backend" }));

// 🔹 Các route cơ bản

app.use("/api/auth", require("./routes/authRoutes"));

app.use("/api/employees", require("./routes/employeeRoutes"));

app.use("/api/attendance", require("./routes/attendanceRoutes"));

app.use("/api/leave-requests", require("./routes/leaveRoutes"));

app.use("/api/messages", require("./routes/messageRoutes"));

// 🔹 Các route mở rộng

app.use("/api/notifications", require("./routes/notificationRoutes"));

app.use("/api/work-schedule", require("./routes/workScheduleRoutes"));

app.use("/api/report", require("./routes/reportRoutes")); // ⚡ ĐÃ CHỈNH: /api/report (không phải /api/reports)

app.use("/api/messages", require("./routes/messageRoutes"));

// 🔹 Route dành cho admin

app.use("/api/admin", require("./routes/adminRoutes"));

// 🔹 Route lương (nếu có)

try {

  app.use("/api/salary", require("./routes/salaryRoutes"));

} catch {}

// ===== MIDDLEWARE LỖI =====

const { notFound, errorHandler } = require("./middleware/errorHandler");

app.use(notFound);

app.use(errorHandler);

// ===== START SERVER =====

const PORT = process.env.PORT || 5000;

server.listen(PORT, () => console.log(`🚀 Server running on port ${PORT}`));

module.exports = { io };

.env

PORT=5000

MONGO\_URI=mongodb+srv://HRM\_Project\_DB:0Kv7p0miy2pFviF1@cluster0.jwwdgus.mongodb.net/HRM\_Project\_database

JWT\_SECRET=supersecret

CORS\_ORIGIN=http://localhost:3000

Frontend

node\_modules

package.json

package-lock.json

Public/index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>HRM Project</title>

    <meta name="viewport" content="width=device-width, initial-scale=1" />

  </head>

  <body>

    <div id="root"></div>

  </body>

</html>

Src/api/axios.js

import axios from "axios";

const api = axios.create({

  baseURL: "http://localhost:5000/api" // khi deploy đổi thành URL backend trên Render

});

// Đính kèm token cho mọi request

api.interceptors.request.use(config => {

  const token = localStorage.getItem("token");

  if (token) config.headers.Authorization = `Bearer ${token}`;

  return config;

});

export default api;

Src

Components

Protected.js

import React, { useContext } from "react";

import { Navigate } from "react-router-dom";

import { AuthContext } from "../context/AuthContext";

function Protected({ children, roles }) {

  const { user } = useContext(AuthContext);

  if (!user) return <Navigate to="/" replace />;

  if (roles && !roles.includes(user.role)) return <Navigate to="/" replace />;

  return children;

}

export default Protected;

SidebarMenu.js

import React, { useContext } from "react";

import { Link } from "react-router-dom";

import { AuthContext } from "../context/AuthContext";

function SidebarMenu({ role }) {

  const { logout } = useContext(AuthContext);

  return (

    <div className="list-group">

      {role === "admin" ? (

        <>

          <Link to="/admin/dashboard" className="list-group-item">🏠 Dashboard</Link>

          <Link to="/admin/hr-management" className="list-group-item">👨‍💼 Quản lý nhân sự</Link>

          <Link to="/admin/leave-approval" className="list-group-item">📌 Phê duyệt nghỉ phép</Link>

          <Link to="/admin/salary-management" className="list-group-item">💰 Quản lý lương</Link>

          <Link to="/admin/attendance-management" className="list-group-item">⏱ Quản lý chấm công</Link>

          <Link to="/admin/performance-review" className="list-group-item">📊 Đánh giá hiệu suất</Link>

          <Link to="/admin/document-management" className="list-group-item">📂 Quản lý tài liệu</Link>

          <Link to="/admin/work-schedule" className="list-group-item">📅 Lịch làm việc</Link>

          <Link to="/admin/notifications" className="list-group-item">🔔 Thông báo</Link>

          <Link to="/admin/reports" className="list-group-item">📊 Báo cáo tổng hợp</Link>

          <Link Link to="/admin/chat" className="list-group-item">💬 Chat</Link>

        </>

      ) : (

        <>

          <Link to="/user/dashboard" className="list-group-item">🏠 Dashboard</Link>

          <Link to="/user/profile-update" className="list-group-item">👤 Cập nhật thông tin</Link>

          <Link to="/user/leave-request" className="list-group-item">📌 Xin nghỉ phép</Link>

          <Link to="/user/leave-history" className="list-group-item">📜 Lịch sử nghỉ phép</Link>

          <Link to="/user/attendance" className="list-group-item">⏱ Chấm công</Link>

          <Link to="/user/chat" className="list-group-item">💬 Chat</Link>

          <Link to="/user/report" className="list-group-item">📊 Báo cáo</Link>

          <Link to="/user/salary-management" className="list-group-item">💰 Lương</Link>

          <Link to="/user/performance-review" className="list-group-item">📊 Hiệu suất</Link>

          <Link to="/user/document-management" className="list-group-item">📂 Tài liệu</Link>

          <Link to="/user/work-schedule" className="list-group-item">📅 Lịch làm việc</Link>

          <Link to="/user/notifications" className="list-group-item">🔔 Thông báo</Link>

        </>

      )}

      <button onClick={logout} className="list-group-item list-group-item-action mt-3">

        🚪 Đăng xuất

      </button>

    </div>

  );

}

export default SidebarMenu;

Config/constants.js

export const DEPARTMENTS = ["IT", "HR", "Finance", "Sales", "Marketing"];

export const POSITIONS = ["Nhân viên", "Quản lý", "Trưởng phòng", "Admin"];

export const ROLES = ["employee", "manager", "admin"];

Context/AuthContext.js

import React, { createContext, useState, useContext } from "react";

export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {

  const [user, setUser] = useState(() => {

    const s = localStorage.getItem("user");

    return s ? JSON.parse(s) : null;

  });

  const login = (data) => {

    localStorage.setItem("token", data.token);

    localStorage.setItem("user", JSON.stringify(data.user));

    setUser(data.user);

  };

  const logout = () => {

    localStorage.removeItem("token");

    localStorage.removeItem("user");

    setUser(null);

  };

  return (

    <AuthContext.Provider value={{ user, login, logout }}>

      {children}

    </AuthContext.Provider>

  );

};

// ✅ Thêm phần này

export const useAuth = () => {

  const context = useContext(AuthContext);

  if (!context) {

    throw new Error("useAuth must be used within an AuthProvider");

  }

  return context;

};

Pages

Login.js

import React, { useState, useContext } from "react";

import { useNavigate, Link } from "react-router-dom";

import api from "../api";

import { AuthContext } from "../context/AuthContext";

function Login() {

  const [username, setUsername] = useState("");

  const [password, setPassword] = useState("");

  const { login } = useContext(AuthContext);

  const navigate = useNavigate();

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const res = await api.post("/auth/login", { username, password });

      login(res.data);

      if (res.data.user.role === "admin") navigate("/admin/dashboard");

      else navigate("/user/dashboard");

    } catch (err) {

      alert(err.response?.data?.error || "Sai thông tin đăng nhập");

    }

  };

  return (

    <div className="container mt-5" style={{ maxWidth: 420 }}>

      <h3>Đăng nhập</h3>

      <form onSubmit={handleSubmit}>

        <input className="form-control mb-2" placeholder="Username"

          value={username} onChange={(e) => setUsername(e.target.value)} />

        <input type="password" className="form-control mb-3" placeholder="Password"

          value={password} onChange={(e) => setPassword(e.target.value)} />

        <button className="btn btn-primary w-100">Đăng nhập</button>

      </form>

      <div className="text-center mt-3">

        Chưa có tài khoản? <Link to="/register">Đăng ký</Link>

      </div>

    </div>

  );

}

export default Login;

Register.js

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

import api from "../api";

function Register() {

  const [form, setForm] = useState({

    username: "",

    password: "",

    name: "",

    department: "",

    position: "",

    role: "employee",

  });

  const navigate = useNavigate();

  const onChange = (e) => setForm({ ...form, [e.target.name]: e.target.value });

  const onSubmit = async (e) => {

    e.preventDefault();

    try {

      await api.post("/auth/register", form);

      alert("Đăng ký thành công, mời đăng nhập");

      navigate("/");

    } catch (err) {

      alert(err.response?.data?.error || "Có lỗi xảy ra");

    }

  };

  return (

    <div className="container mt-5" style={{ maxWidth: 520 }}>

      <h3>Đăng ký tài khoản</h3>

      <form onSubmit={onSubmit}>

        <input name="username" className="form-control mb-2" placeholder="Username" onChange={onChange} />

        <input name="password" type="password" className="form-control mb-2" placeholder="Password" onChange={onChange} />

        <input name="name" className="form-control mb-2" placeholder="Họ tên" onChange={onChange} />

        <input name="department" className="form-control mb-2" placeholder="Phòng ban" onChange={onChange} />

        <input name="position" className="form-control mb-3" placeholder="Chức vụ" onChange={onChange} />

        <select name="role" className="form-control mb-3" value={form.role} onChange={onChange}>

          <option value="employee">Nhân viên</option>

          <option value="manager">Quản lý</option>

          <option value="admin">Admin</option>

        </select>

        <button className="btn btn-success w-100">Đăng ký</button>

      </form>

    </div>

  );

}

export default Register;

Pages/admin

AdminDashboard.js

import React from "react";

import SidebarMenu from "../../components/SidebarMenu";

function AdminDashboard() {

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="admin" /></div>

        <div className="col-9">

          <h3>📊 Admin Dashboard</h3>

          <p>Chào mừng Admin! Dùng menu bên trái để quản lý.</p>

        </div>

      </div>

    </div>

  );

}

export default AdminDashboard;

AttendanceManagement.js

// src/pages/admin/AttendanceManagement.js

import React, { useEffect, useState } from "react";

import SidebarMenu from "../../components/SidebarMenu";

import api from "../../api";

function AttendanceManagement() {

  const [list, setList] = useState([]);

  const load = async () => {

    try {

      const res = await api.get("/admin/attendance"); // API admin lấy tất cả chấm công

      setList(res.data);

    } catch (err) {

      console.error("❌ Lỗi tải chấm công:", err);

      alert("Không tải được dữ liệu chấm công");

    }

  };

  useEffect(() => {

    load();

  }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>⏱ Quản lý chấm công</h3>

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Nhân viên</th>

                <th>Ngày</th>

                <th>Giờ check-in</th>

                <th>Giờ check-out</th>

                <th>Tổng giờ</th>

              </tr>

            </thead>

            <tbody>

              {list.map((a) => (

                <tr key={a.\_id}>

                  <td>{a.userId?.username}</td>

                  <td>{new Date(a.date).toLocaleDateString()}</td>

                  <td>{a.checkIn ? new Date(a.checkIn).toLocaleTimeString() : "-"}</td>

                  <td>{a.checkOut ? new Date(a.checkOut).toLocaleTimeString() : "-"}</td>

                  <td>{a.totalHours || 0}</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default AttendanceManagement;

ChatAdmin.js

// frontend/src/pages/admin/ChatAdmin.js

import React, { useEffect, useRef, useState } from "react";

import io from "socket.io-client";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

import { useAuth } from "../../context/AuthContext";

const socket = io(process.env.REACT\_APP\_API\_BASE?.replace("/api", "") || "http://localhost:5000", {

  transports: ["websocket"]

});

export default function ChatAdmin() {

  const { user } = useAuth();

  const me = user?.user;

  const [employees, setEmployees] = useState([]);

  const [rooms, setRooms] = useState([]);

  const [currentRoom, setCurrentRoom] = useState(null);

  const [messages, setMessages] = useState([]);

  const [text, setText] = useState("");

  const bottomRef = useRef();

  useEffect(() => {

    socket.emit("join", { userId: me?.\_id });

    loadRooms();

    loadEmployees();

  }, [me]);

  const loadRooms = async () => {

    const res = await api.get("/messages/rooms");

    const combined = [];

    if (res.data.deptRoom) combined.push(res.data.deptRoom);

    if (res.data.privateRooms) combined.push(...res.data.privateRooms);

    setRooms(combined);

  };

  const loadEmployees = async () => {

    const res = await api.get("/admin/employees");

    setEmployees(res.data || []);

  };

  const loadMessages = async (room) => {

    const res = await api.get(`/messages/${room.\_id}`);

    setMessages(res.data || []);

    setTimeout(() => bottomRef.current?.scrollIntoView({ behavior: "smooth" }), 50);

  };

  const openRoom = async (room) => {

    setCurrentRoom(room);

    await loadMessages(room);

  };

  const startPrivateWith = async (userId) => {

    const res = await api.post("/messages/rooms/private", { otherUserId: userId });

    await loadRooms();

    setCurrentRoom(res.data);

    await loadMessages(res.data);

  };

  const sendMessage = async (e) => {

    e.preventDefault();

    if (!text.trim() || !currentRoom) return;

    const saved = await api.post("/messages", { roomId: currentRoom.\_id, content: text.trim() });

    // Gửi realtime qua socket

    socket.emit("send\_message", {

      type: currentRoom.type === "group" ? "group" : "private",

      department: currentRoom.department,

      roomId: currentRoom.\_id,

      fromUserId: me?.\_id,

      fromUsername: me?.username,

      message: saved.data.content

    });

    setMessages((prev) => [...prev, saved.data]);

    setText("");

    setTimeout(() => bottomRef.current?.scrollIntoView({ behavior: "smooth" }), 50);

  };

  useEffect(() => {

    const handler = (payload) => {

      if (!currentRoom || payload.roomId !== currentRoom.\_id) return;

      setMessages((prev) => [

        ...prev,

        {

          \_id: Math.random().toString(36).slice(2),

          sender: { username: payload.fromUsername },

          content: payload.message,

          createdAt: new Date().toISOString()

        }

      ]);

      bottomRef.current?.scrollIntoView({ behavior: "smooth" });

    };

    socket.on("receive\_message", handler);

    return () => socket.off("receive\_message", handler);

  }, [currentRoom]);

  return (

    <div className="container-fluid mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

          <div className="card mt-3">

            <div className="card-header"><strong>💬 Chat (Admin)</strong></div>

            <div className="card-body p-2">

              <small className="text-muted">Danh sách nhân viên</small>

              <div className="list-group mt-1" style={{ maxHeight: 300, overflowY: "auto" }}>

                {employees.map(e => (

                  <button key={e.\_id}

                    className="list-group-item list-group-item-action"

                    onClick={() => startPrivateWith(e.userId?.\_id)}>

                    {e.name} <span className="text-muted">({e.userId?.username})</span>

                  </button>

                ))}

              </div>

              <hr className="my-2" />

              <small className="text-muted">Các phòng chat</small>

              <div className="list-group mt-1" style={{ maxHeight: 300, overflowY: "auto" }}>

                {rooms.map(r => (

                  <button key={r.\_id}

                    className="list-group-item list-group-item-action"

                    onClick={() => openRoom(r)}>

                    {r.name || r.participants?.map(p => p.username).join(", ")}

                  </button>

                ))}

              </div>

            </div>

          </div>

        </div>

        <div className="col-9">

          <div className="card">

            <div className="card-header">

              {currentRoom

                ? currentRoom.name || `💬 ${(currentRoom.participants || []).map(p => p.username).join(", ")}`

                : "Chọn đoạn chat để bắt đầu"}

            </div>

            <div className="card-body" style={{ height: 520, overflowY: "auto" }}>

              {messages.map(m => (

                <div key={m.\_id} className={`mb-2 ${m.sender?.\_id === me?.\_id ? "text-end" : ""}`}>

                  <div><small className="text-muted">{m.sender?.username} · {new Date(m.createdAt).toLocaleString()}</small></div>

                  <div className={`d-inline-block p-2 rounded ${m.sender?.\_id === me?.\_id ? "bg-primary text-white" : "bg-light"}`}>

                    {m.content}

                  </div>

                </div>

              ))}

              <div ref={bottomRef} />

            </div>

            <div className="card-footer">

              <form onSubmit={sendMessage} className="d-flex gap-2">

                <input className="form-control" placeholder="Nhập tin nhắn..." value={text}

                  onChange={(e) => setText(e.target.value)} />

                <button className="btn btn-primary" disabled={!currentRoom || !text.trim()}>Gửi</button>

              </form>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

}

DocumentManagementAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function DocumentManagementAdmin() {

  const [docs, setDocs] = useState([]);

  const [folders, setFolders] = useState([]); // danh sách thư mục

  const [form, setForm] = useState({

    folder: "",

    department: "general",

    files: [],

  });

  const [newFolder, setNewFolder] = useState("");

  // Tải danh sách tài liệu

  const load = async () => {

    try {

      const res = await api.get("/admin/documents");

      setDocs(res.data);

      // Lấy tất cả các thư mục duy nhất

      const uniqueFolders = [...new Set(res.data.map((d) => d.folder || "Chung"))];

      setFolders(uniqueFolders);

    } catch (err) {

      alert("❌ Lỗi tải danh sách tài liệu");

    }

  };

  useEffect(() => {

    load();

  }, []);

  // ======= TẠO THƯ MỤC =======

  const handleCreateFolder = async (e) => {

    e.preventDefault();

    if (!newFolder.trim()) return alert("⚠️ Vui lòng nhập tên thư mục");

    try {

      await api.post("/admin/documents/folder", { folder: newFolder.trim() });

      alert(`✅ Đã tạo thư mục '${newFolder}'`);

      setNewFolder("");

      setFolders([...folders, newFolder.trim()]);

    } catch {

      alert("❌ Lỗi tạo thư mục");

    }

  };

  // ======= UPLOAD FILES =======

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!form.folder) return alert("⚠️ Vui lòng chọn thư mục");

    if (!form.files || form.files.length === 0) return alert("⚠️ Chưa chọn file");

    try {

      const fd = new FormData();

      fd.append("department", form.department);

      fd.append("folder", form.folder);

      for (let file of form.files) fd.append("files", file);

      await api.post("/admin/documents", fd, {

        headers: { "Content-Type": "multipart/form-data" },

      });

      alert("✅ Upload thành công");

      setForm({ folder: "", department: "general", files: [] });

      load();

    } catch {

      alert("❌ Lỗi upload file");

    }

  };

  // ======= XÓA FILE =======

  const handleDelete = async (id) => {

    if (!window.confirm("Bạn có chắc muốn xóa tài liệu này?")) return;

    try {

      await api.delete(`/admin/documents/${id}`);

      load();

    } catch {

      alert("❌ Lỗi khi xóa file");

    }

  };

  // ======= TẢI FILE =======

  const handleDownload = (id) => {

    window.open(`http://localhost:5000/api/admin/documents/download/${id}`, "\_blank");

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>📂 Quản lý tài liệu</h3>

          {/\* ====== TẠO THƯ MỤC ====== \*/}

          <form onSubmit={handleCreateFolder} className="card p-3 mb-3">

            <h5>📁 Tạo thư mục mới</h5>

            <div className="d-flex">

              <input

                type="text"

                className="form-control me-2"

                placeholder="Tên thư mục..."

                value={newFolder}

                onChange={(e) => setNewFolder(e.target.value)}

              />

              <button className="btn btn-success">➕ Tạo</button>

            </div>

          </form>

          {/\* ====== UPLOAD FILE ====== \*/}

          <form onSubmit={handleSubmit} className="card p-3 mb-3">

            <h5>📤 Upload tài liệu</h5>

            <div className="row">

              <div className="col-md-4 mb-2">

                <label>Phòng ban</label>

                <select

                  className="form-control"

                  value={form.department}

                  onChange={(e) => setForm({ ...form, department: e.target.value })}

                >

                  <option value="general">Chung</option>

                  <option value="IT">IT</option>

                  <option value="HR">HR</option>

                  <option value="Finance">Finance</option>

                </select>

              </div>

              <div className="col-md-4 mb-2">

                <label>Thư mục</label>

                <select

                  className="form-control"

                  value={form.folder}

                  onChange={(e) => setForm({ ...form, folder: e.target.value })}

                >

                  <option value="">-- Chọn thư mục --</option>

                  {folders.map((f, idx) => (

                    <option key={idx} value={f}>

                      {f}

                    </option>

                  ))}

                </select>

              </div>

              <div className="col-md-4 mb-2">

                <label>Chọn file</label>

                <input

                  type="file"

                  className="form-control"

                  multiple

                  onChange={(e) => setForm({ ...form, files: e.target.files })}

                />

              </div>

            </div>

            <button className="btn btn-primary mt-2">📤 Upload</button>

          </form>

          {/\* ====== DANH SÁCH TÀI LIỆU ====== \*/}

          <div className="card p-3">

            <h5>📑 Danh sách tài liệu</h5>

            {docs.length === 0 ? (

              <p>Chưa có tài liệu nào.</p>

            ) : (

              <table className="table table-bordered mt-2">

                <thead>

                  <tr>

                    <th>Thư mục</th>

                    <th>Tiêu đề</th>

                    <th>Phòng ban</th>

                    <th>Người tải lên</th>

                    <th>Hành động</th>

                  </tr>

                </thead>

                <tbody>

                  {docs.map((d) => (

                    <tr key={d.\_id}>

                      <td>{d.folder || "Chung"}</td>

                      <td>{d.title}</td>

                      <td>{d.department}</td>

                      <td>{d.uploadedBy?.username}</td>

                      <td>

                        <button

                          className="btn btn-sm btn-success me-2"

                          onClick={() => handleDownload(d.\_id)}

                        >

                          ⬇️ Tải

                        </button>

                        <button

                          className="btn btn-sm btn-danger"

                          onClick={() => handleDelete(d.\_id)}

                        >

                          🗑 Xóa

                        </button>

                      </td>

                    </tr>

                  ))}

                </tbody>

              </table>

            )}

          </div>

        </div>

      </div>

    </div>

  );

}

export default DocumentManagementAdmin;

HRManagement.js

import React, { useEffect, useState } from "react";

import SidebarMenu from "../../components/SidebarMenu";

import api from "../../api";

import { DEPARTMENTS, POSITIONS, ROLES } from "../../config/constants";

function HRManagement() {

  const [list, setList] = useState([]);

  const [form, setForm] = useState({

    username: "",

    password: "",

    name: "",

    department: "",

    position: "",

    role: "employee",

  });

  const [editId, setEditId] = useState(null);

  const load = async () => {

    try {

      const res = await api.get("/admin/employees");

      setList(res.data);

    } catch (err) {

      alert("❌ Lỗi tải danh sách nhân viên");

    }

  };

  useEffect(() => { load(); }, []);

  // Thêm nhân viên

  const onSubmit = async (e) => {

    e.preventDefault();

    try {

      if (editId) {

        await api.put(`/admin/employees/${editId}`, form);

        alert("✅ Cập nhật nhân viên thành công");

      } else {

        await api.post("/admin/employees", form);

        alert("✅ Thêm nhân viên thành công");

      }

      setForm({ username: "", password: "", name: "", department: "", position: "", role: "employee" });

      setEditId(null);

      load();

    } catch (err) {

      alert(err.response?.data?.error || "❌ Lỗi");

    }

  };

  // Chỉnh sửa nhân viên

  const onEdit = (emp) => {

    setEditId(emp.\_id);

    setForm({

      username: emp.userId?.username,

      password: "", // không show password

      name: emp.name,

      department: emp.department,

      position: emp.position,

      role: emp.userId?.role,

    });

  };

  // Xóa nhân viên

  const onDelete = async (id) => {

    if (!window.confirm("Bạn có chắc chắn muốn xóa nhân viên này?")) return;

    try {

      await api.delete(`/admin/employees/${id}`);

      alert("🗑️ Đã xóa nhân viên");

      load();

    } catch (err) {

      alert("❌ Lỗi khi xóa");

    }

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="admin" /></div>

        <div className="col-9">

          <h3>👨‍💼 Quản lý nhân sự</h3>

          {/\* Form thêm/sửa nhân viên \*/}

          <form onSubmit={onSubmit} className="mb-4 border p-3 rounded">

            <h5>{editId ? "✏️ Chỉnh sửa nhân viên" : "➕ Thêm nhân viên mới"}</h5>

            <div className="row g-2 mt-2">

              <div className="col">

                <input

                  className="form-control"

                  placeholder="Username"

                  value={form.username}

                  disabled={!!editId}

                  onChange={(e)=>setForm({...form, username: e.target.value})}

                  required

                />

              </div>

              {!editId && (

                <div className="col">

                  <input

                    className="form-control"

                    type="password"

                    placeholder="Password"

                    value={form.password}

                    onChange={(e)=>setForm({...form, password: e.target.value})}

                    required

                  />

                </div>

              )}

            </div>

            <div className="row g-2 mt-2">

              <div className="col">

                <input

                  className="form-control"

                  placeholder="Họ tên"

                  value={form.name}

                  onChange={(e)=>setForm({...form, name: e.target.value})}

                  required

                />

              </div>

              <div className="col">

                <select

                  className="form-control"

                  value={form.department}

                  onChange={(e)=>setForm({...form, department: e.target.value})}

                  required

                >

                  <option value="">-- Chọn phòng ban --</option>

                  {DEPARTMENTS.map(d => <option key={d} value={d}>{d}</option>)}

                </select>

              </div>

              <div className="col">

                <select

                  className="form-control"

                  value={form.position}

                  onChange={(e)=>setForm({...form, position: e.target.value})}

                  required

                >

                  <option value="">-- Chọn chức vụ --</option>

                  {POSITIONS.map(p => <option key={p} value={p}>{p}</option>)}

                </select>

              </div>

            </div>

            <div className="row g-2 mt-2">

              <div className="col">

                <select

                  className="form-control"

                  value={form.role}

                  onChange={(e)=>setForm({...form, role: e.target.value})}

                >

                  {ROLES.map(r => <option key={r} value={r}>{r}</option>)}

                </select>

              </div>

              <div className="col">

                <button className="btn btn-primary w-100">

                  {editId ? "Cập nhật" : "Thêm nhân viên"}

                </button>

              </div>

              {editId && (

                <div className="col">

                  <button

                    type="button"

                    className="btn btn-secondary w-100"

                    onClick={() => {

                      setEditId(null);

                      setForm({ username: "", password: "", name: "", department: "", position: "", role: "employee" });

                    }}

                  >

                    Hủy

                  </button>

                </div>

              )}

            </div>

          </form>

          {/\* Danh sách nhân viên \*/}

          <h5>📋 Danh sách nhân viên</h5>

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Username</th>

                <th>Role</th>

                <th>Họ tên</th>

                <th>Phòng ban</th>

                <th>Chức vụ</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {list.map(e => (

                <tr key={e.\_id}>

                  <td>{e.userId?.username}</td>

                  <td>{e.userId?.role}</td>

                  <td>{e.name}</td>

                  <td>{e.department}</td>

                  <td>{e.position}</td>

                  <td>

                    <button className="btn btn-sm btn-warning me-2" onClick={()=>onEdit(e)}>Sửa</button>

                    <button className="btn btn-sm btn-danger" onClick={()=>onDelete(e.\_id)}>Xóa</button>

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default HRManagement;

LeaveApproval.js

import React, { useEffect, useState } from "react";

import SidebarMenu from "../../components/SidebarMenu";

import api from "../../api";

function LeaveApproval() {

  const [list, setList] = useState([]);

  const load = async () => {

    try {

      const res = await api.get("/admin/leave-requests");

      setList(res.data);

    } catch (err) {

      console.error("❌ Lỗi load nghỉ phép:", err.response?.data || err.message);

      alert("❌ Không tải được danh sách nghỉ phép");

    }

  };

  useEffect(() => { load(); }, []);

  const act = async (id, type) => {

    try {

      await api.put(`/admin/leave-requests/${id}/${type}`);

      load();

    } catch (err) {

      alert("❌ Lỗi cập nhật trạng thái");

    }

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="admin" /></div>

        <div className="col-9">

          <h3>📌 Phê duyệt nghỉ phép</h3>

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Nhân viên</th>

                <th>Từ ngày</th>

                <th>Đến ngày</th>

                <th>Lý do</th>

                <th>Trạng thái</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {list.map(l => (

                <tr key={l.\_id}>

                  <td>{l.userId?.username}</td>

                  <td>{new Date(l.startDate).toLocaleDateString("vi-VN")}</td>

                  <td>{new Date(l.endDate).toLocaleDateString("vi-VN")}</td>

                  <td>{l.reason}</td>

                  <td>

                    {l.status === "pending" && <span className="badge bg-warning">⏳ Chờ duyệt</span>}

                    {l.status === "approved" && <span className="badge bg-success">✅ Đã duyệt</span>}

                    {l.status === "rejected" && <span className="badge bg-danger">❌ Từ chối</span>}

                  </td>

                  <td>

                    {l.status === "pending" && (

                      <>

                        <button className="btn btn-success btn-sm me-2" onClick={()=>act(l.\_id, "approve")}>Duyệt</button>

                        <button className="btn btn-danger btn-sm" onClick={()=>act(l.\_id, "reject")}>Từ chối</button>

                      </>

                    )}

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default LeaveApproval;

NotificationsAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function NotificationsAdmin() {

  const [list, setList] = useState([]);

  const [employees, setEmployees] = useState([]);

  const [departments, setDepartments] = useState(["IT", "HR", "Finance", "Sales"]);

  const [editId, setEditId] = useState(null);

  const [form, setForm] = useState({

    title: "",

    message: "",

    target: "all",   // all | employee | department

    targetValue: "",

  });

  const load = async () => {

    try {

      const res = await api.get("/admin/notifications");

      setList(res.data);

      const empRes = await api.get("/admin/employees");

      setEmployees(empRes.data);

    } catch (err) {

      console.error("❌ API Error:", err.response?.data || err.message);

      alert("Không tải được dữ liệu");

    }

  };

  useEffect(() => {

    load();

  }, []);

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      if (editId) {

        await api.put(`/admin/notifications/${editId}`, form);

        alert("✅ Cập nhật thông báo thành công");

      } else {

        await api.post("/admin/notifications", form);

        alert("✅ Gửi thông báo thành công");

      }

      setForm({ title: "", message: "", target: "all", targetValue: "" });

      setEditId(null);

      load();

    } catch (err) {

      console.error("❌ Lỗi lưu:", err.response?.data || err.message);

      alert("❌ Không thể lưu thông báo");

    }

  };

  const handleDelete = async (id) => {

    if (!window.confirm("Bạn có chắc muốn xóa?")) return;

    try {

      await api.delete(`/admin/notifications/${id}`);

      alert("🗑️ Đã xóa thông báo");

      load();

    } catch (err) {

      console.error("❌ Lỗi xóa:", err.response?.data || err.message);

      alert("❌ Không thể xóa thông báo");

    }

  };

  const startEdit = (n) => {

    setEditId(n.\_id);

    setForm({

      title: n.title,

      message: n.message,

      target: n.target,

      targetValue: n.targetValue || "",

    });

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>🔔 Quản lý thông báo</h3>

          {/\* Form \*/}

          <form onSubmit={handleSubmit} className="mb-3">

            <div className="mb-2">

              <label className="form-label">Tiêu đề</label>

              <input

                type="text"

                className="form-control"

                value={form.title}

                onChange={(e) => setForm({ ...form, title: e.target.value })}

                required

              />

            </div>

            <div className="mb-2">

              <label className="form-label">Nội dung</label>

              <textarea

                className="form-control"

                value={form.message}

                onChange={(e) => setForm({ ...form, message: e.target.value })}

                required

              />

            </div>

            <div className="mb-2">

              <label className="form-label">Gửi đến</label>

              <select

                className="form-control"

                value={form.target}

                onChange={(e) => setForm({ ...form, target: e.target.value, targetValue: "" })}

              >

                <option value="all">Tất cả nhân viên</option>

                <option value="employee">Một nhân viên</option>

                <option value="department">Một phòng ban</option>

              </select>

            </div>

            {form.target === "employee" && (

              <div className="mb-2">

                <label className="form-label">Chọn nhân viên</label>

                <select

                  className="form-control"

                  value={form.targetValue}

                  onChange={(e) => setForm({ ...form, targetValue: e.target.value })}

                  required

                >

                  <option value="">-- Chọn nhân viên --</option>

                  {employees.map((emp) => (

                    <option key={emp.\_id} value={emp.userId?.\_id}>

                      {emp.name} ({emp.userId?.username})

                    </option>

                  ))}

                </select>

              </div>

            )}

            {form.target === "department" && (

              <div className="mb-2">

                <label className="form-label">Chọn phòng ban</label>

                <select

                  className="form-control"

                  value={form.targetValue}

                  onChange={(e) => setForm({ ...form, targetValue: e.target.value })}

                  required

                >

                  <option value="">-- Chọn phòng ban --</option>

                  {departments.map((d) => (

                    <option key={d} value={d}>

                      {d}

                    </option>

                  ))}

                </select>

              </div>

            )}

            <button className="btn btn-primary mt-2">

              {editId ? "💾 Lưu thay đổi" : "➕ Gửi thông báo"}

            </button>

            {editId && (

              <button

                type="button"

                className="btn btn-secondary mt-2 ms-2"

                onClick={() => {

                  setEditId(null);

                  setForm({ title: "", message: "", target: "all", targetValue: "" });

                }}

              >

                Hủy

              </button>

            )}

          </form>

          {/\* Danh sách \*/}

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Tiêu đề</th>

                <th>Nội dung</th>

                <th>Đối tượng</th>

                <th>Ngày gửi</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {list.map((n) => (

                <tr key={n.\_id}>

                  <td>{n.title}</td>

                  <td>{n.message}</td>

                  <td>

                    {n.target === "all"

                      ? "Tất cả"

                      : n.target === "employee"

                      ? `Nhân viên: ${n.targetValue}`

                      : `Phòng ban: ${n.targetValue}`}

                  </td>

                  <td>{new Date(n.createdAt).toLocaleString("vi-VN")}</td>

                  <td>

                    <button className="btn btn-warning btn-sm me-2" onClick={() => startEdit(n)}>

                      Sửa

                    </button>

                    <button className="btn btn-danger btn-sm" onClick={() => handleDelete(n.\_id)}>

                      Xóa

                    </button>

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default NotificationsAdmin;

PerformanceReviewAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function PerformanceReviewAdmin() {

  const [reviews, setReviews] = useState([]);

  const [employees, setEmployees] = useState([]);

  const [editId, setEditId] = useState(null);

  const [form, setForm] = useState({

    userId: "",

    tasksCompleted: 0,

    communication: 0,

    technical: 0,

    attitude: 10,

    feedback: "",

  });

  // Load dữ liệu

  const load = async () => {

    try {

      const res = await api.get("/admin/performance");

      setReviews(res.data);

      const empRes = await api.get("/admin/employees");

      setEmployees(empRes.data);

    } catch (err) {

      console.error("❌ API Error:", err.response?.data || err.message);

    }

  };

  useEffect(() => {

    load();

  }, []);

  // Thêm hoặc cập nhật

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      if (editId) {

        await api.put(`/admin/performance/${editId}`, form);

        alert("✅ Cập nhật đánh giá thành công");

      } else {

        await api.post("/admin/performance", form);

        alert("✅ Đã thêm đánh giá");

      }

      setForm({

        userId: "",

        tasksCompleted: 0,

        communication: 0,

        technical: 0,

        attitude: 10,

        feedback: "",

      });

      setEditId(null);

      load();

    } catch (err) {

      console.error("❌ Lỗi thêm/cập nhật:", err.response?.data || err.message);

      alert("❌ Không thể lưu đánh giá");

    }

  };

  // Xóa

  const handleDelete = async (id) => {

    if (!window.confirm("Bạn có chắc muốn xóa đánh giá này?")) return;

    try {

      await api.delete(`/admin/performance/${id}`);

      alert("🗑️ Đã xóa đánh giá");

      load();

    } catch (err) {

      console.error("❌ Lỗi xóa:", err.response?.data || err.message);

      alert("❌ Không thể xóa đánh giá");

    }

  };

  // Sửa

  const startEdit = (review) => {

    setEditId(review.\_id);

    setForm({

      userId: review.userId?.\_id || "",

      tasksCompleted: review.tasksCompleted || 0,

      communication: review.communication || 0,

      technical: review.technical || 0,

      attitude: review.attitude || 10,

      feedback: review.feedback || "",

    });

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>📊 Đánh giá hiệu suất</h3>

          {/\* Form \*/}

          <form className="card p-3 mb-3" onSubmit={handleSubmit}>

            {/\* Nhân viên \*/}

            <label className="form-label">Nhân viên</label>

            <select

              className="form-control mb-2"

              value={form.userId}

              onChange={(e) => setForm({ ...form, userId: e.target.value })}

              required

            >

              <option value="">-- Chọn nhân viên --</option>

              {employees.map((emp) => (

                <option key={emp.\_id} value={emp.userId?.\_id}>

                  {emp.name} ({emp.userId?.username})

                </option>

              ))}

            </select>

            {/\* Hoàn thành nhiệm vụ \*/}

            <label className="form-label">Hoàn thành nhiệm vụ (số lượng)</label>

            <input

              type="number"

              className="form-control mb-2"

              value={form.tasksCompleted}

              onChange={(e) =>

                setForm({ ...form, tasksCompleted: Number(e.target.value) })

              }

              min="0"

              required

            />

            {/\* Giao tiếp \*/}

            <label className="form-label">Kỹ năng giao tiếp</label>

            <select

              className="form-control mb-2"

              value={form.communication}

              onChange={(e) =>

                setForm({ ...form, communication: Number(e.target.value) })

              }

            >

              <option value="0">Không biết (0)</option>

              <option value="2">Kém (2)</option>

              <option value="4">Có thể giao tiếp (4)</option>

              <option value="6">Khá (6)</option>

              <option value="8">Tốt (8)</option>

              <option value="10">Giỏi (10)</option>

            </select>

            {/\* Kỹ thuật \*/}

            <label className="form-label">Kỹ năng kỹ thuật</label>

            <select

              className="form-control mb-2"

              value={form.technical}

              onChange={(e) =>

                setForm({ ...form, technical: Number(e.target.value) })

              }

            >

              <option value="0">Không biết (0)</option>

              <option value="2">Kém (2)</option>

              <option value="4">Cơ bản (4)</option>

              <option value="6">Khá (6)</option>

              <option value="8">Tốt (8)</option>

              <option value="10">Giỏi (10)</option>

            </select>

            {/\* Thái độ \*/}

            <label className="form-label">Thái độ / tinh thần</label>

            <select

              className="form-control mb-2"

              value={form.attitude}

              onChange={(e) =>

                setForm({ ...form, attitude: Number(e.target.value) })

              }

            >

              <option value="0">Tiêu cực (0)</option>

              <option value="10">Tích cực (10)</option>

            </select>

            {/\* Nhận xét \*/}

            <label className="form-label">Nhận xét</label>

            <textarea

              className="form-control mb-2"

              value={form.feedback}

              onChange={(e) => setForm({ ...form, feedback: e.target.value })}

            />

            <button className="btn btn-primary">

              {editId ? "💾 Lưu thay đổi" : "➕ Thêm đánh giá"}

            </button>

            {editId && (

              <button

                type="button"

                className="btn btn-secondary ms-2"

                onClick={() => {

                  setEditId(null);

                  setForm({

                    userId: "",

                    tasksCompleted: 0,

                    communication: 0,

                    technical: 0,

                    attitude: 10,

                    feedback: "",

                  });

                }}

              >

                Hủy

              </button>

            )}

          </form>

          {/\* Bảng danh sách \*/}

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Nhân viên</th>

                <th>Nhiệm vụ hoàn thành</th>

                <th>Giao tiếp</th>

                <th>Kỹ thuật</th>

                <th>Thái độ</th>

                <th>Nhận xét</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {reviews.map((r) => (

                <tr key={r.\_id}>

                  <td>{r.userId?.username}</td>

                  <td>{r.tasksCompleted}</td>

                  <td>{r.communication}</td>

                  <td>{r.technical}</td>

                  <td>{r.attitude}</td>

                  <td>{r.feedback}</td>

                  <td>

                    <button

                      className="btn btn-warning btn-sm me-2"

                      onClick={() => startEdit(r)}

                    >

                      Sửa

                    </button>

                    <button

                      className="btn btn-danger btn-sm"

                      onClick={() => handleDelete(r.\_id)}

                    >

                      Xóa

                    </button>

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default PerformanceReviewAdmin;

ReportAdmin.js

// src/pages/admin/ReportAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function ReportAdmin() {

  const [reports, setReports] = useState([]);

  const [loading, setLoading] = useState(true);

  const [err, setErr] = useState("");

  const loadReports = async () => {

    setLoading(true);

    try {

      const res = await api.get("/reports");

      setReports(res.data);

    } catch (e) {

      console.error("❌ Lỗi tải báo cáo:", e);

      setErr(e.response?.data?.error || "Không tải được dữ liệu báo cáo.");

    } finally {

      setLoading(false);

    }

  };

  useEffect(() => {

    loadReports();

  }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>📊 Báo cáo tổng hợp</h3>

          {loading && <p>⏳ Đang tải dữ liệu...</p>}

          {err && <div className="alert alert-danger">{err}</div>}

          {!loading && !err && (

            <table className="table table-bordered">

              <thead>

                <tr>

                  <th>Nhân viên</th>

                  <th>Phòng ban</th>

                  <th>Chức vụ</th>

                  <th>Tổng giờ làm</th>

                  <th>Tổng ngày nghỉ</th>

                  <th>Hiệu suất TB</th>

                </tr>

              </thead>

              <tbody>

                {reports.map((r, idx) => (

                  <tr key={idx}>

                    <td>{r.name} ({r.username})</td>

                    <td>{r.department}</td>

                    <td>{r.position}</td>

                    <td>{r.totalHours?.toFixed?.(2) || 0}</td>

                    <td>{r.totalLeaves}</td>

                    <td>{r.avgScore}</td>

                  </tr>

                ))}

              </tbody>

            </table>

          )}

        </div>

      </div>

    </div>

  );

}

export default ReportAdmin;

SalaryManagementAdmin.js

// frontend/src/pages/admin/SalaryManagementAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function SalaryManagementAdmin() {

  const [salaries, setSalaries] = useState([]);

  const [editId, setEditId] = useState(null);

  const [form, setForm] = useState({ hourlyRate: "" });

  const load = async () => {

    try {

      const res = await api.get("/admin/salary");

      setSalaries(res.data);

    } catch (err) {

      console.error("❌ API Error:", err.response?.data || err.message);

      alert("❌ Không tải được dữ liệu lương");

    }

  };

  useEffect(() => { load(); }, []);

  const startEdit = (s) => {

    setEditId(s.\_id);

    setForm({ hourlyRate: s.hourlyRate });

  };

  const saveEdit = async (id) => {

    try {

      await api.put(`/admin/salary/${id}`, form);

      alert("✅ Cập nhật lương thành công");

      setEditId(null);

      setForm({ hourlyRate: "" });

      load();

    } catch (err) {

      console.error("❌ Update Error:", err.response?.data || err.message);

      alert("❌ Lỗi cập nhật lương");

    }

  };

  const cancelEdit = () => {

    setEditId(null);

    setForm({ hourlyRate: "" });

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="admin" /></div>

        <div className="col-9">

          <h3>💰 Quản lý lương (admin)</h3>

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Nhân viên</th>

                <th>Tổng giờ làm</th>

                <th>Tiền giờ (VND)</th>

                <th>Lương (VND)</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {salaries.map((s) => (

                <tr key={s.\_id}>

                  <td>{s.name} ({s.userId?.username})</td>

                  <td>{s.totalHours}</td>

                  <td>

                    {editId === s.\_id ? (

                      <input

                        className="form-control"

                        value={form.hourlyRate}

                        onChange={(e) =>

                          setForm({ ...form, hourlyRate: e.target.value })

                        }

                      />

                    ) : (

                      s.hourlyRate.toLocaleString("vi-VN")

                    )}

                  </td>

                  <td>{s.amount.toLocaleString("vi-VN")}</td>

                  <td>

                    {editId === s.\_id ? (

                      <>

                        <button

                          className="btn btn-success btn-sm me-2"

                          onClick={() => saveEdit(s.\_id)}

                        >

                          Lưu

                        </button>

                        <button

                          className="btn btn-secondary btn-sm"

                          onClick={cancelEdit}

                        >

                          Hủy

                        </button>

                      </>

                    ) : (

                      <button

                        className="btn btn-warning btn-sm"

                        onClick={() => startEdit(s)}

                      >

                        Sửa

                      </button>

                    )}

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default SalaryManagementAdmin;

WorkScheduleAdmin.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function WorkScheduleAdmin() {

  const [list, setList] = useState([]);

  const [employees, setEmployees] = useState([]);

  const [editId, setEditId] = useState(null);

  const [form, setForm] = useState({

    task: "",

    department: "",

    assignedTo: "",

    startDate: "",

    endDate: "",

  });

  // Load lịch + nhân viên

  const load = async () => {

    try {

      const res = await api.get("/admin/work-schedule");

      setList(res.data);

      const empRes = await api.get("/admin/employees");

      setEmployees(empRes.data);

    } catch (err) {

      console.error("❌ API Error:", err.response?.data || err.message);

      alert("❌ Không tải được dữ liệu");

    }

  };

  useEffect(() => {

    load();

  }, []);

  // Khi chọn nhân viên → tự động lấy phòng ban

  const handleEmployeeChange = (e) => {

    const userId = e.target.value;

    const emp = employees.find((emp) => emp.userId?.\_id === userId);

    setForm({

      ...form,

      assignedTo: userId,

      department: emp?.department || "",

    });

  };

  // Thêm / cập nhật lịch

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      if (editId) {

        await api.put(`/admin/work-schedule/${editId}`, form);

        alert("✅ Cập nhật lịch thành công");

      } else {

        await api.post("/admin/work-schedule", form);

        alert("✅ Thêm lịch thành công");

      }

      setForm({ task: "", department: "", assignedTo: "", startDate: "", endDate: "" });

      setEditId(null);

      load();

    } catch (err) {

      console.error("❌ Lỗi thêm/cập nhật:", err.response?.data || err.message);

      alert("❌ Không thể lưu lịch");

    }

  };

  // Xóa

  const handleDelete = async (id) => {

    if (!window.confirm("Bạn chắc chắn muốn xóa?")) return;

    try {

      await api.delete(`/admin/work-schedule/${id}`);

      alert("🗑️ Đã xóa lịch");

      load();

    } catch (err) {

      console.error("❌ Lỗi xóa:", err.response?.data || err.message);

      alert("❌ Không thể xóa");

    }

  };

  // Sửa

  const startEdit = (w) => {

    setEditId(w.\_id);

    setForm({

      task: w.task || "",

      department: w.department || "",

      assignedTo: w.assignedTo?.\_id || "",

      startDate: w.startDate ? w.startDate.split("T")[0] : "",

      endDate: w.endDate ? w.endDate.split("T")[0] : "",

    });

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="admin" />

        </div>

        <div className="col-9">

          <h3>📅 Quản lý lịch làm việc</h3>

          {/\* Form thêm/sửa \*/}

          <form onSubmit={handleSubmit} className="mb-3">

            <div className="mb-2">

              <label className="form-label">Nhiệm vụ</label>

              <input

                type="text"

                className="form-control"

                value={form.task || ""}

                onChange={(e) => setForm({ ...form, task: e.target.value })}

                required

              />

            </div>

            <div className="mb-2">

              <label className="form-label">Nhân viên</label>

              <select

                className="form-control"

                value={form.assignedTo || ""}

                onChange={handleEmployeeChange}

                required

              >

                <option value="">-- Chọn nhân viên --</option>

                {employees.map((emp) => (

                  <option key={emp.\_id} value={emp.userId?.\_id}>

                    {emp.name} ({emp.userId?.username})

                  </option>

                ))}

              </select>

            </div>

            <div className="mb-2">

              <label className="form-label">Phòng ban</label>

              <input

                type="text"

                className="form-control"

                value={form.department || ""}

                readOnly

              />

            </div>

            <div className="row">

              <div className="col">

                <label className="form-label">Ngày bắt đầu</label>

                <input

                  type="date"

                  className="form-control"

                  value={form.startDate || ""}

                  onChange={(e) => setForm({ ...form, startDate: e.target.value })}

                  required

                />

              </div>

              <div className="col">

                <label className="form-label">Ngày kết thúc</label>

                <input

                  type="date"

                  className="form-control"

                  value={form.endDate || ""}

                  onChange={(e) => setForm({ ...form, endDate: e.target.value })}

                  required

                />

              </div>

            </div>

            <button className="btn btn-primary mt-3">

              {editId ? "💾 Lưu thay đổi" : "➕ Thêm lịch"}

            </button>

            {editId && (

              <button

                type="button"

                className="btn btn-secondary mt-3 ms-2"

                onClick={() => {

                  setEditId(null);

                  setForm({ task: "", department: "", assignedTo: "", startDate: "", endDate: "" });

                }}

              >

                Hủy

              </button>

            )}

          </form>

          {/\* Danh sách \*/}

          <table className="table table-bordered">

            <thead>

              <tr>

                <th>Nhiệm vụ</th>

                <th>Phòng ban</th>

                <th>Nhân viên</th>

                <th>Bắt đầu</th>

                <th>Kết thúc</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {list.map((w) => (

                <tr key={w.\_id}>

                  <td>{w.task}</td>

                  <td>{w.department}</td>

                  <td>{w.assignedTo?.username}</td>

                  <td>{new Date(w.startDate).toLocaleDateString("vi-VN")}</td>

                  <td>{new Date(w.endDate).toLocaleDateString("vi-VN")}</td>

                  <td>

                    <button className="btn btn-warning btn-sm me-2" onClick={() => startEdit(w)}>

                      Sửa

                    </button>

                    <button className="btn btn-danger btn-sm" onClick={() => handleDelete(w.\_id)}>

                      Xóa

                    </button>

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default WorkScheduleAdmin;

Pages/user

Attendance.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function Attendance() {

  const [records, setRecords] = useState([]);

  const [todayRecord, setTodayRecord] = useState(null);

  const [loading, setLoading] = useState(true);

  // === Load dữ liệu chấm công ===

  const load = async () => {

    try {

      const res = await api.get("/attendance");

      setRecords(res.data);

      const today = new Date().toISOString().split("T")[0];

      const todayRec = res.data.find((r) => r.date === today);

      setTodayRecord(todayRec || null);

    } catch (err) {

      console.error("❌ Lỗi load chấm công:", err);

      alert("Lỗi tải dữ liệu chấm công!");

    } finally {

      setLoading(false);

    }

  };

  // === Check-in ===

  const checkIn = async () => {

    try {

      await api.post("/attendance/check-in");

      alert("✅ Check-in thành công");

      load();

    } catch (err) {

      alert(err.response?.data?.error || "❌ Lỗi check-in");

    }

  };

  // === Check-out ===

  const checkOut = async () => {

    try {

      await api.post("/attendance/check-out");

      alert("✅ Check-out thành công");

      load();

    } catch (err) {

      alert(err.response?.data?.error || "❌ Lỗi check-out");

    }

  };

  useEffect(() => {

    load();

  }, []);

  // === UI ===

  return (

    <div className="container mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="employee" />

        </div>

        {/\* Nội dung chính \*/}

        <div className="col-9">

          <h3>🕒 Chấm công</h3>

          {/\* Trạng thái hôm nay \*/}

          {todayRecord ? (

            <div className="alert alert-info">

              <strong>Hôm nay:</strong> {todayRecord.date} <br />

              ⏰ Check-in:{" "}

              {todayRecord.checkIn

                ? new Date(todayRecord.checkIn).toLocaleTimeString()

                : "—"}

              <br />

              🕓 Check-out:{" "}

              {todayRecord.checkOut

                ? new Date(todayRecord.checkOut).toLocaleTimeString()

                : "—"}

              <br />

              ⏳ Tổng giờ:{" "}

              {todayRecord.totalHours ? todayRecord.totalHours.toFixed(2) : 0} giờ

            </div>

          ) : (

            <p>📅 Bạn chưa chấm công hôm nay.</p>

          )}

          {/\* Nút chấm công \*/}

          <div className="mb-3">

            <button

              className="btn btn-success me-2"

              onClick={checkIn}

              disabled={!!todayRecord?.checkIn}

            >

              ✅ Check-in

            </button>

            <button

              className="btn btn-danger"

              onClick={checkOut}

              disabled={!todayRecord?.checkIn || !!todayRecord?.checkOut}

            >

              ⏹ Check-out

            </button>

          </div>

          {/\* Bảng lịch sử \*/}

          {loading ? (

            <p>⏳ Đang tải dữ liệu...</p>

          ) : records.length === 0 ? (

            <p>Không có dữ liệu chấm công.</p>

          ) : (

            <table className="table table-bordered">

              <thead>

                <tr>

                  <th>Ngày</th>

                  <th>Giờ vào</th>

                  <th>Giờ ra</th>

                  <th>Tổng giờ</th>

                </tr>

              </thead>

              <tbody>

                {records.map((r) => (

                  <tr key={r.\_id}>

                    <td>{r.date}</td>

                    <td>

                      {r.checkIn ? new Date(r.checkIn).toLocaleTimeString() : "—"}

                    </td>

                    <td>

                      {r.checkOut ? new Date(r.checkOut).toLocaleTimeString() : "—"}

                    </td>

                    <td>{r.totalHours?.toFixed(2) || 0}</td>

                  </tr>

                ))}

              </tbody>

            </table>

          )}

        </div>

      </div>

    </div>

  );

}

export default Attendance;

Chat.js

// frontend/src/pages/user/Chat.js

import React, { useEffect, useMemo, useRef, useState } from "react";

import io from "socket.io-client";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

import { useAuth } from "../../context/AuthContext";

const socket = io(process.env.REACT\_APP\_API\_BASE?.replace("/api", "") || "http://localhost:5000", {

  transports: ["websocket"],

});

export default function Chat() {

  const { user } = useAuth(); // { user: {\_id, username, role, ...} }

  const me = user?.user;

  const [deptRoom, setDeptRoom] = useState(null);

  const [privateRooms, setPrivateRooms] = useState([]);

  const [employees, setEmployees] = useState([]);

  const [chatOutsideDept, setChatOutsideDept] = useState(false);

  const [currentRoom, setCurrentRoom] = useState(null);

  const [messages, setMessages] = useState([]);

  const [text, setText] = useState("");

  const bottomRef = useRef();

  const myDepartment = useMemo(() => me?.department || null, [me]);

  // 🧩 Khi user đăng nhập → join phòng socket riêng và phòng ban

  useEffect(() => {

    if (me?.\_id) {

      socket.emit("join", { userId: me.\_id, department: me.department });

    }

  }, [me]);

  // 🗂 Lấy danh sách phòng ban + private rooms

  const loadRooms = async () => {

    try {

      const res = await api.get("/messages/rooms");

      setPrivateRooms(res.data.privateRooms || []);

      setDeptRoom(res.data.deptRoom || null);

    } catch (err) {

      console.error("❌ Lỗi load rooms:", err);

    }

  };

  // 👥 Lấy danh sách nhân viên (cùng phòng hoặc toàn công ty)

  const loadEmployees = async () => {

    try {

      const scope = chatOutsideDept ? "all" : "dept";

      const res = await api.get(`/employees/peers?scope=${scope}`);

      setEmployees(res.data || []);

    } catch (err) {

      console.error("❌ Lỗi load danh sách nhân viên:", err);

      setEmployees([]);

    }

  };

  // 💬 Lấy tin nhắn trong phòng

  const loadMessages = async (room) => {

    try {

      const res = await api.get(`/messages/${room.\_id}`);

      setMessages(res.data || []);

      setTimeout(() => bottomRef.current?.scrollIntoView({ behavior: "smooth" }), 50);

    } catch (err) {

      console.error("❌ Lỗi load tin nhắn:", err);

      setMessages([]);

    }

  };

  useEffect(() => {

    loadRooms();

  }, []);

  useEffect(() => {

    loadEmployees();

  }, [chatOutsideDept]);

  // 🔄 Socket realtime: nhận tin nhắn

  useEffect(() => {

    const handler = (payload) => {

      if (!currentRoom) return;

      if (payload.roomId && payload.roomId !== currentRoom.\_id) return;

      setMessages((prev) => [

        ...prev,

        {

          \_id: Math.random().toString(36).slice(2),

          sender: { username: payload.fromUsername || "N/A" },

          content: payload.message,

          createdAt: new Date().toISOString(),

        },

      ]);

      bottomRef.current?.scrollIntoView({ behavior: "smooth" });

    };

    socket.on("receive\_message", handler);

    return () => socket.off("receive\_message", handler);

  }, [currentRoom]);

  const openRoom = async (room) => {

    setCurrentRoom(room);

    await loadMessages(room);

  };

  const startPrivateWith = async (otherUserId) => {

    try {

      const res = await api.post("/messages/rooms/private", { otherUserId });

      await loadRooms();

      setCurrentRoom(res.data);

      await loadMessages(res.data);

    } catch (err) {

      console.error("❌ Lỗi tạo private room:", err);

    }

  };

  const sendMessage = async (e) => {

    e.preventDefault();

    if (!text.trim() || !currentRoom) return;

    try {

      // Lưu DB

      const saved = await api.post("/messages", { roomId: currentRoom.\_id, content: text.trim() });

      // Emit realtime

      if (currentRoom.type === "group") {

        socket.emit("send\_message", {

          type: "group",

          department: currentRoom.department,

          roomId: currentRoom.\_id,

          fromUserId: me?.\_id,

          fromUsername: me?.username,

          message: saved.data.content,

        });

      } else {

        const others = (currentRoom.participants || [])

          .map((p) => p.\_id)

          .filter((id) => id !== me?.\_id);

        const toUserId = others?.[0];

        socket.emit("send\_message", {

          type: "private",

          toUserId,

          roomId: currentRoom.\_id,

          fromUserId: me?.\_id,

          fromUsername: me?.username,

          message: saved.data.content,

        });

      }

      setMessages((prev) => [...prev, saved.data]);

      setText("");

      bottomRef.current?.scrollIntoView({ behavior: "smooth" });

    } catch (err) {

      console.error("❌ Lỗi gửi tin nhắn:", err);

    }

  };

  return (

    <div className="container-fluid mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="user" />

          <div className="card mt-3">

            <div className="card-header"><strong>Chat</strong></div>

            <div className="card-body p-2">

              <div className="form-check form-switch mb-2">

                <input

                  className="form-check-input"

                  type="checkbox"

                  checked={chatOutsideDept}

                  onChange={() => setChatOutsideDept((v) => !v)}

                  id="switchOutside"

                />

                <label htmlFor="switchOutside">Chat ngoài phòng ban</label>

              </div>

              {/\* Phòng ban \*/}

              {deptRoom && (

                <div className="mb-3">

                  <div className="d-flex justify-content-between align-items-center">

                    <strong>👥 {deptRoom.name}</strong>

                    <button

                      className="btn btn-sm btn-outline-primary"

                      onClick={() => openRoom(deptRoom)}

                    >

                      Mở

                    </button>

                  </div>

                </div>

              )}

              {/\* Danh sách nhân viên \*/}

              <hr />

              <small className="text-muted">

                Nhân viên {chatOutsideDept ? "toàn công ty" : "trong phòng ban"}

              </small>

              <div className="list-group mt-1" style={{ maxHeight: 320, overflowY: "auto" }}>

                {employees.length === 0 ? (

                  <div className="text-muted small px-2">Không có nhân viên nào</div>

                ) : (

                  employees

                    .filter((e) => e.userId !== me?.\_id)

                    .filter((e) => chatOutsideDept ? true : e.department === myDepartment)

                    .map((e) => (

                      <button

                        key={e.\_id}

                        className="list-group-item list-group-item-action"

                        onClick={() => startPrivateWith(e.userId)}

                      >

                        {e.name || "Chưa đặt tên"}{" "}

                        <span className="text-muted">({e.username})</span>

                      </button>

                    ))

                )}

              </div>

              {/\* Private rooms \*/}

              <hr className="my-2" />

              <small className="text-muted">Đoạn chat gần đây</small>

              <div className="list-group mt-1" style={{ maxHeight: 200, overflowY: "auto" }}>

                {privateRooms.map((r) => (

                  <button

                    key={r.\_id}

                    className="list-group-item list-group-item-action"

                    onClick={() => openRoom(r)}

                  >

                    {r.participants?.map((p) => p.username).join(" , ")}

                  </button>

                ))}

              </div>

            </div>

          </div>

        </div>

        {/\* Khung chat \*/}

        <div className="col-9">

          <div className="card">

            <div className="card-header">

              {currentRoom

                ? currentRoom.type === "group"

                  ? `👥 ${currentRoom.name}`

                  : `💬 ${(currentRoom.participants || [])

                      .map((p) => p.username)

                      .join(" , ")}`

                : "Chọn 1 đoạn hội thoại"}

            </div>

            <div className="card-body" style={{ height: 520, overflowY: "auto" }}>

              {messages.map((m) => (

                <div

                  key={m.\_id}

                  className={`mb-2 ${m.sender?.\_id === me?.\_id ? "text-end" : ""}`}

                >

                  <div>

                    <small className="text-muted">

                      {m.sender?.username} · {new Date(m.createdAt).toLocaleString()}

                    </small>

                  </div>

                  <div

                    className={`d-inline-block p-2 rounded ${

                      m.sender?.\_id === me?.\_id ? "bg-primary text-white" : "bg-light"

                    }`}

                  >

                    {m.content}

                  </div>

                </div>

              ))}

              <div ref={bottomRef} />

            </div>

            <div className="card-footer">

              <form onSubmit={sendMessage} className="d-flex gap-2">

                <input

                  className="form-control"

                  placeholder="Nhập tin nhắn..."

                  value={text}

                  onChange={(e) => setText(e.target.value)}

                />

                <button

                  className="btn btn-primary"

                  disabled={!currentRoom || !text.trim()}

                >

                  Gửi

                </button>

              </form>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

}

DocumentManagementUser.js

// frontend/src/pages/user/DocumentManagementUser.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

export default function DocumentManagementUser() {

  const [docs, setDocs] = useState([]);

  const [file, setFile] = useState(null);

  const [title, setTitle] = useState("");

  const [loading, setLoading] = useState(false);

  // 📦 Lấy danh sách tài liệu

  const load = async () => {

    try {

      const res = await api.get("/employees/documents"); // backend lọc theo phòng ban

      setDocs(res.data || []);

    } catch (err) {

      console.error("❌ Lỗi tải danh sách tài liệu:", err);

      alert(err.response?.data?.error || "Không thể tải danh sách tài liệu!");

    }

  };

  useEffect(() => {

    load();

  }, []);

  // ⬇️ Tải tài liệu (gửi token kèm)

  const downloadFile = async (id, title) => {

    try {

      const token = localStorage.getItem("token");

      const res = await fetch(

        `${process.env.REACT\_APP\_API\_BASE || "http://localhost:5000/api"}/employees/documents/download/${id}`,

        {

          headers: { Authorization: `Bearer ${token}` },

        }

      );

      if (!res.ok) throw new Error("Không thể tải file");

      const blob = await res.blob();

      const url = window.URL.createObjectURL(blob);

      const a = document.createElement("a");

      a.href = url;

      a.download = title || "Tai\_lieu";

      a.click();

      window.URL.revokeObjectURL(url);

    } catch (err) {

      console.error("❌ Lỗi tải file:", err);

      alert("Không thể tải tài liệu (có thể do quyền hạn hoặc lỗi token).");

    }

  };

  // ⬆️ Upload tài liệu

  const uploadFile = async (e) => {

    e.preventDefault();

    if (!file || !title.trim()) return alert("Vui lòng nhập tiêu đề và chọn file!");

    const formData = new FormData();

    formData.append("title", title);

    formData.append("file", file);

    try {

      setLoading(true);

      const res = await api.post("/admin/documents", formData, {

        headers: { "Content-Type": "multipart/form-data" },

      });

      alert("✅ Tải tài liệu lên thành công!");

      setFile(null);

      setTitle("");

      await load();

    } catch (err) {

      console.error("❌ Lỗi tải lên:", err);

      alert(err.response?.data?.error || "Không thể tải lên tài liệu!");

    } finally {

      setLoading(false);

    }

  };

  // 🗑️ Xóa tài liệu (chỉ tài liệu của chính mình)

  const deleteFile = async (id) => {

    if (!window.confirm("Bạn có chắc muốn xóa tài liệu này không?")) return;

    try {

      await api.delete(`/admin/documents/${id}`);

      alert("🗑️ Đã xóa tài liệu");

      await load();

    } catch (err) {

      console.error("❌ Lỗi xóa tài liệu:", err);

      alert(err.response?.data?.error || "Không thể xóa tài liệu!");

    }

  };

  return (

    <div className="container mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="user" />

        </div>

        {/\* Nội dung chính \*/}

        <div className="col-9">

          <h3>📂 Tài liệu phòng ban của tôi</h3>

          {/\* Upload form \*/}

          <form className="card p-3 mb-3" onSubmit={uploadFile}>

            <h5 className="mb-3">⬆️ Tải tài liệu lên</h5>

            <input

              type="text"

              className="form-control mb-2"

              placeholder="Nhập tiêu đề tài liệu..."

              value={title}

              onChange={(e) => setTitle(e.target.value)}

              required

            />

            <input

              type="file"

              className="form-control mb-2"

              onChange={(e) => setFile(e.target.files[0])}

              required

            />

            <button className="btn btn-primary" disabled={loading}>

              {loading ? "Đang tải lên..." : "📤 Tải lên"}

            </button>

          </form>

          {/\* Bảng danh sách \*/}

          <table className="table table-bordered align-middle">

            <thead className="table-light">

              <tr>

                <th>Tiêu đề</th>

                <th>Phòng ban</th>

                <th>Người tải lên</th>

                <th>Hành động</th>

              </tr>

            </thead>

            <tbody>

              {docs.length > 0 ? (

                docs.map((d) => (

                  <tr key={d.\_id}>

                    <td>{d.title}</td>

                    <td>{d.department}</td>

                    <td>{d.uploadedBy?.username}</td>

                    <td>

                      <button

                        className="btn btn-sm btn-success me-2"

                        onClick={() => downloadFile(d.\_id, d.title)}

                      >

                        ⬇️ Tải

                      </button>

                      {/\* Chỉ cho xóa nếu là người đã tải lên \*/}

                      {d.uploadedBy?.\_id === JSON.parse(atob(localStorage.getItem("token").split(".")[1]))?.id && (

                        <button

                          className="btn btn-sm btn-danger"

                          onClick={() => deleteFile(d.\_id)}

                        >

                          🗑️ Xóa

                        </button>

                      )}

                    </td>

                  </tr>

                ))

              ) : (

                <tr>

                  <td colSpan={4} className="text-center text-muted">

                    Chưa có tài liệu

                  </td>

                </tr>

              )}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

LeaveHistory.js

import React, { useEffect, useState } from "react";

import SidebarMenu from "../../components/SidebarMenu";

import api from "../../api";

function LeaveHistory() {

  const [list, setList] = useState([]);

  useEffect(() => {

    api.get("/leave-requests/me").then(res => setList(res.data));

  }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="user" /></div>

        <div className="col-9">

          <h3>📜 Lịch sử nghỉ phép</h3>

          <table className="table table-bordered">

            <thead><tr><th>Từ</th><th>Đến</th><th>Lý do</th><th>Trạng thái</th></tr></thead>

            <tbody>

              {list.map(l => (

                <tr key={l.\_id}>

                  <td>{new Date(l.from).toLocaleDateString()}</td>

                  <td>{new Date(l.to).toLocaleDateString()}</td>

                  <td>{l.reason}</td>

                  <td>{l.status}</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default LeaveHistory;

LeaveRequest.js

import React, { useState } from "react";

import SidebarMenu from "../../components/SidebarMenu";

import api from "../../api";

function LeaveRequest() {

  const [form, setForm] = useState({ from: "", to: "", reason: "" });

  const onSubmit = async (e) => {

    e.preventDefault();

    await api.post("/leave-requests", form);

    alert("Đã gửi đơn nghỉ phép");

    setForm({ from: "", to: "", reason: "" });

  };

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="user" /></div>

        <div className="col-9">

          <h3>📌 Xin nghỉ phép</h3>

          <form onSubmit={onSubmit}>

            <div className="row g-2">

              <div className="col">

                <label>Từ ngày</label>

                <input type="date" className="form-control" value={form.from}

                  onChange={(e)=>setForm({...form, from: e.target.value})} />

              </div>

              <div className="col">

                <label>Đến ngày</label>

                <input type="date" className="form-control" value={form.to}

                  onChange={(e)=>setForm({...form, to: e.target.value})} />

              </div>

            </div>

            <label className="mt-2">Lý do</label>

            <textarea className="form-control mb-3" value={form.reason}

              onChange={(e)=>setForm({...form, reason: e.target.value})} />

            <button className="btn btn-primary">Gửi</button>

          </form>

        </div>

      </div>

    </div>

  );

}

export default LeaveRequest;

NotificationsUser.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function Notifications() {

  const [notis, setNotis] = useState([]);

  const [loading, setLoading] = useState(true);

  const load = async () => {

    try {

      const res = await api.get("/notifications");

      setNotis(res.data);

    } catch (err) {

      console.error("❌ Lỗi load thông báo:", err);

      alert("Không thể tải thông báo.");

    } finally {

      setLoading(false);

    }

  };

  useEffect(() => {

    load();

  }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="employee" />

        </div>

        {/\* Nội dung \*/}

        <div className="col-9">

          <h3>📢 Thông báo</h3>

          {loading ? (

            <p>⏳ Đang tải dữ liệu...</p>

          ) : notis.length === 0 ? (

            <p>Không có thông báo nào.</p>

          ) : (

            <div className="list-group">

              {notis.map((n) => (

                <div key={n.\_id} className="list-group-item">

                  <h5 className="mb-1">{n.title}</h5>

                  <p className="mb-1">{n.message}</p>

                  <small>

                    👤 {n.createdBy?.username || "Hệ thống"} —{" "}

                    {new Date(n.createdAt).toLocaleString()}

                  </small>

                </div>

              ))}

            </div>

          )}

        </div>

      </div>

    </div>

  );

}

export default Notifications;

PerformanceReviewUser.js

// frontend/src/pages/user/PerformanceReviewUser.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

export default function PerformanceReviewUser() {

  const [list, setList] = useState([]);

  const [loading, setLoading] = useState(false);

  // 🔹 Hàm tải dữ liệu đánh giá của chính nhân viên

  const load = async () => {

    setLoading(true);

    try {

      const res = await api.get("/employees/performance"); // ✅ endpoint đúng cho user

      setList(res.data || []);

    } catch (err) {

      console.error("❌ Lỗi tải dữ liệu hiệu suất:", err);

      alert(err.response?.data?.error || "Không thể tải dữ liệu hiệu suất!");

      setList([]);

    } finally {

      setLoading(false);

    }

  };

  useEffect(() => {

    load();

  }, []);

  // 🔹 Hàm tính tổng điểm trung bình

  const calcScore = (r) => {

    const values = [r.tasksCompleted, r.communication, r.technical, r.attitude].filter(

      (v) => typeof v === "number"

    );

    if (values.length === 0) return 0;

    return (values.reduce((a, b) => a + b, 0) / values.length).toFixed(1);

  };

  return (

    <div className="container mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="user" />

        </div>

        {/\* Nội dung chính \*/}

        <div className="col-9">

          <h3 className="mb-3">📊 Đánh giá hiệu suất của tôi</h3>

          {loading ? (

            <div className="text-center text-muted p-3">Đang tải dữ liệu...</div>

          ) : (

            <table className="table table-bordered table-hover align-middle">

              <thead className="table-light">

                <tr>

                  <th style={{ width: "12%" }}>Tổng điểm</th>

                  <th style={{ width: "12%" }}>Nhiệm vụ</th>

                  <th style={{ width: "12%" }}>Giao tiếp</th>

                  <th style={{ width: "12%" }}>Kỹ thuật</th>

                  <th style={{ width: "12%" }}>Thái độ</th>

                  <th style={{ width: "32%" }}>Nhận xét</th>

                  <th style={{ width: "18%" }}>Ngày đánh giá</th>

                </tr>

              </thead>

              <tbody>

                {list.length > 0 ? (

                  list.map((r) => (

                    <tr key={r.\_id}>

                      <td className="text-center fw-bold text-primary">{calcScore(r)}</td>

                      <td className="text-center">{r.tasksCompleted ?? "—"}</td>

                      <td className="text-center">{r.communication ?? "—"}</td>

                      <td className="text-center">{r.technical ?? "—"}</td>

                      <td className="text-center">{r.attitude ?? "—"}</td>

                      <td>{r.feedback || "Không có nhận xét"}</td>

                      <td>

                        {new Date(r.createdAt).toLocaleString("vi-VN", {

                          hour12: false,

                        })}

                      </td>

                    </tr>

                  ))

                ) : (

                  <tr>

                    <td colSpan={7} className="text-center text-muted">

                      Chưa có đánh giá hiệu suất nào

                    </td>

                  </tr>

                )}

              </tbody>

            </table>

          )}

        </div>

      </div>

    </div>

  );

}

ProfileUpdate.js

// frontend/src/pages/user/ProfileUpdate.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

export default function ProfileUpdate() {

  const [profile, setProfile] = useState(null);

  const [name, setName] = useState("");

  const [oldPassword, setOldPassword] = useState("");

  const [newPassword, setNewPassword] = useState("");

  const [loading, setLoading] = useState(false);

  // 🧩 Lấy thông tin hiện tại

  const loadProfile = async () => {

    try {

      const res = await api.get("/employees/me"); // ✅ route chính xác

      setProfile(res.data);

      setName(res.data.name || "");

    } catch (err) {

      console.error("❌ Lỗi tải hồ sơ:", err);

      alert("Không thể tải thông tin cá nhân!");

    }

  };

  useEffect(() => {

    loadProfile();

  }, []);

  // 💾 Cập nhật tên hiển thị

  const saveName = async (e) => {

    e.preventDefault();

    if (!name.trim()) return alert("Tên hiển thị không được để trống!");

    setLoading(true);

    try {

      await api.put("/employees/profile", { name });

      alert("✅ Cập nhật tên hiển thị thành công!");

      await loadProfile();

    } catch (err) {

      console.error("❌ Lỗi cập nhật:", err);

      alert(err.response?.data?.error || "Không thể cập nhật thông tin!");

    } finally {

      setLoading(false);

    }

  };

  // 🔐 Đổi mật khẩu

  const changePassword = async (e) => {

    e.preventDefault();

    if (!oldPassword || !newPassword) return alert("Vui lòng nhập đầy đủ mật khẩu!");

    setLoading(true);

    try {

      await api.put("/employees/change-password", { oldPassword, newPassword });

      alert("✅ Đổi mật khẩu thành công!");

      setOldPassword("");

      setNewPassword("");

    } catch (err) {

      console.error("❌ Lỗi đổi mật khẩu:", err);

      alert(err.response?.data?.error || "Không thể đổi mật khẩu!");

    } finally {

      setLoading(false);

    }

  };

  if (!profile) return <div className="p-3 text-center">Đang tải thông tin...</div>;

  return (

    <div className="container-fluid mt-3">

      <div className="row">

        {/\* Sidebar \*/}

        <div className="col-3">

          <SidebarMenu role="user" />

        </div>

        {/\* Main content \*/}

        <div className="col-9">

          <h3 className="mb-3">👤 Cập nhật thông tin cá nhân</h3>

          {/\* Chỉnh sửa tên hiển thị \*/}

          <div className="card mb-4">

            <div className="card-header bg-light">

              <strong>Đổi tên hiển thị</strong>

            </div>

            <div className="card-body">

              <form onSubmit={saveName}>

                <div className="mb-3">

                  <label className="form-label">Tên hiển thị</label>

                  <input

                    type="text"

                    className="form-control"

                    value={name}

                    onChange={(e) => setName(e.target.value)}

                    required

                  />

                </div>

                <button className="btn btn-primary" disabled={loading}>

                  {loading ? "Đang lưu..." : "💾 Lưu thay đổi"}

                </button>

              </form>

            </div>

          </div>

          {/\* Đổi mật khẩu \*/}

          <div className="card mb-4">

            <div className="card-header bg-light">

              <strong>Đổi mật khẩu</strong>

            </div>

            <div className="card-body">

              <form onSubmit={changePassword}>

                <div className="mb-3">

                  <label className="form-label">Mật khẩu cũ</label>

                  <input

                    type="password"

                    className="form-control"

                    value={oldPassword}

                    onChange={(e) => setOldPassword(e.target.value)}

                    required

                  />

                </div>

                <div className="mb-3">

                  <label className="form-label">Mật khẩu mới</label>

                  <input

                    type="password"

                    className="form-control"

                    value={newPassword}

                    onChange={(e) => setNewPassword(e.target.value)}

                    required

                  />

                </div>

                <button className="btn btn-warning" disabled={loading}>

                  {loading ? "Đang cập nhật..." : "🔐 Đổi mật khẩu"}

                </button>

              </form>

            </div>

          </div>

          {/\* Thông tin tài khoản \*/}

          <div className="card">

            <div className="card-header bg-light">

              <strong>Thông tin tài khoản</strong>

            </div>

            <div className="card-body">

              <p>

                <strong>Tên đăng nhập:</strong> {profile.userId?.username}

              </p>

              <p>

                <strong>Chức vụ:</strong> {profile.userId?.role}

              </p>

              <p>

                <strong>Phòng ban:</strong> {profile.department || "—"}

              </p>

              <p>

                <strong>Vị trí:</strong> {profile.position || "—"}

              </p>

              <p>

                <strong>Ngày tạo:</strong>{" "}

                {new Date(profile.createdAt).toLocaleString("vi-VN")}

              </p>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

}

Report.js

SalaryManagement.js

// frontend/src/pages/user/SalaryManagement.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

const res = await api.get("/salary/me");  // user chỉ xem lương của mình

function SalaryManagement() {

  const [list, setList] = useState([]);

  const load = async () => {

    try {

      const res = await api.get("/salary/me");

      setList(res.data);

    } catch (err) {

      console.error(err);

      alert("❌ Lỗi tải dữ liệu lương");

    }

  };

  useEffect(() => { load(); }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="employee" /></div>

        <div className="col-9">

          <h3>💰 Lương của tôi</h3>

          <table className="table table-striped">

            <thead>

              <tr>

                <th>Tổng giờ làm</th>

                <th>Tiền giờ (VND)</th>

                <th>Lương (VND)</th>

              </tr>

            </thead>

            <tbody>

              {list.map((s) => (

                <tr key={s.\_id}>

                  <td>{s.totalHours || 0}</td>

                  <td>{s.hourlyRate?.toLocaleString()}</td>

                  <td>{(s.amount || 0).toLocaleString()}</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      </div>

    </div>

  );

}

export default SalaryManagement;

UserDashboard.js

import React from "react";

import SidebarMenu from "../../components/SidebarMenu";

function UserDashboard() {

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3"><SidebarMenu role="user" /></div>

        <div className="col-9">

          <h3>🏠 User Dashboard</h3>

          <p>Xin chào! Dùng menu bên trái để thao tác.</p>

        </div>

      </div>

    </div>

  );

}

export default UserDashboard;

WorkScheduleUser.js

// src/pages/user/WorkScheduleUser.js

import React, { useEffect, useState } from "react";

import api from "../../api";

import SidebarMenu from "../../components/SidebarMenu";

function WorkScheduleUser() {

  const [schedules, setSchedules] = useState([]);

  const [loading, setLoading] = useState(true);

  const [err, setErr] = useState("");

  const load = async () => {

    setLoading(true);

    setErr("");

    try {

      const res = await api.get("/work-schedule");

      setSchedules(Array.isArray(res.data) ? res.data : []);

    } catch (e) {

      console.error("❌ Lỗi tải lịch làm việc:", e);

      setErr(e.response?.data?.error || "Không tải được lịch làm việc.");

    } finally {

      setLoading(false);

    }

  };

  useEffect(() => {

    load();

  }, []);

  return (

    <div className="container mt-3">

      <div className="row">

        <div className="col-3">

          <SidebarMenu role="employee" />

        </div>

        <div className="col-9">

          <h3>📅 Lịch làm việc của bạn</h3>

          {loading && <p>⏳ Đang tải dữ liệu...</p>}

          {!loading && err && (

            <div className="alert alert-danger">{err}</div>

          )}

          {!loading && !err && (

            <>

              {schedules.length === 0 ? (

                <p>Không có lịch làm việc nào.</p>

              ) : (

                <table className="table table-bordered">

                  <thead>

                    <tr>

                      <th>Nhiệm vụ</th>

                      <th>Phòng ban</th>

                      <th>Người được giao</th>

                      <th>Ngày bắt đầu</th>

                      <th>Ngày kết thúc</th>

                    </tr>

                  </thead>

                  <tbody>

                    {schedules.map((s) => (

                      <tr key={s.\_id}>

                        <td>{s.task || "—"}</td>

                        <td>{s.department || "—"}</td>

                        <td>{s.assignedTo?.username || "—"}</td>

                        <td>

                          {s.startDate

                            ? new Date(s.startDate).toLocaleDateString("vi-VN")

                            : "—"}

                        </td>

                        <td>

                          {s.endDate

                            ? new Date(s.endDate).toLocaleDateString("vi-VN")

                            : "—"}

                        </td>

                      </tr>

                    ))}

                  </tbody>

                </table>

              )}

            </>

          )}

        </div>

      </div>

    </div>

  );

}

export default WorkScheduleUser;

Src

api.js

// frontend/src/api.js

import axios from "axios";

const api = axios.create({

  baseURL: process.env.REACT\_APP\_API\_BASE || "http://localhost:5000/api",

});

// 🧩 Thêm token tự động cho mọi request

api.interceptors.request.use(

  (config) => {

    const token = localStorage.getItem("token"); // nơi bạn lưu JWT khi login

    if (token) {

      config.headers.Authorization = `Bearer ${token}`;

    }

    return config;

  },

  (error) => Promise.reject(error)

);

export default api;

App.js

import React from "react";

import { BrowserRouter, Routes, Route } from "react-router-dom";

import { AuthProvider } from "./context/AuthContext";

import Protected from "./components/Protected";

// Public

import Login from "./pages/Login";

import Register from "./pages/Register";

// Admin

import AdminDashboard from "./pages/admin/AdminDashboard";

import HRManagement from "./pages/admin/HRManagement";

import LeaveApproval from "./pages/admin/LeaveApproval";

import AttendanceManagement from "./pages/admin/AttendanceManagement";

import SalaryManagementAdmin from "./pages/admin/SalaryManagementAdmin";

import NotificationsAdmin from "./pages/admin/NotificationsAdmin";

import PerformanceReviewAdmin from "./pages/admin/PerformanceReviewAdmin";

import DocumentManagementAdmin from "./pages/admin/DocumentManagementAdmin";

import WorkScheduleAdmin from "./pages/admin/WorkScheduleAdmin";

import ChatAdmin from "./pages/admin/ChatAdmin";

// User

import UserDashboard from "./pages/user/UserDashboard";

import ProfileUpdate from "./pages/user/ProfileUpdate";

import LeaveRequest from "./pages/user/LeaveRequest";

import LeaveHistory from "./pages/user/LeaveHistory";

import Attendance from "./pages/user/Attendance";

import Chat from "./pages/user/Chat";

import Report from "./pages/user/Report";

import SalaryManagement from "./pages/user/SalaryManagement";

import NotificationsUser from "./pages/user/NotificationsUser";

import PerformanceReviewUser from "./pages/user/PerformanceReviewUser";

import DocumentManagementUser from "./pages/user/DocumentManagementUser";

import WorkScheduleUser from "./pages/user/WorkScheduleUser";

import ReportAdmin from "./pages/admin/ReportAdmin";

function App() {

  return (

    <BrowserRouter>

      <AuthProvider>

        <Routes>

          {/\* Public \*/}

          <Route path="/" element={<Login />} />

          <Route path="/register" element={<Register />} />

{/\* ================= ADMIN ================= \*/}

<Route

  path="/admin/chat"

  element={<Protected roles={["admin"]}><ChatAdmin /></Protected>}

/>

<Route

  path="/admin/dashboard"

  element={<Protected roles={["admin"]}><AdminDashboard /></Protected>}

/>

<Route

  path="/admin/hr-management"

  element={<Protected roles={["admin"]}><HRManagement /></Protected>}

/>

<Route

  path="/admin/leave-approval"

  element={<Protected roles={["admin"]}><LeaveApproval /></Protected>}

/>

<Route

  path="/admin/attendance-management"

  element={<Protected roles={["admin"]}><AttendanceManagement /></Protected>}

/>

<Route

  path="/admin/notifications"

  element={<Protected roles={["admin"]}><NotificationsAdmin /></Protected>}

/>

<Route

  path="/admin/salary-management"

  element={<Protected roles={["admin"]}><SalaryManagementAdmin /></Protected>}

/>

<Route

  path="/admin/performance-review"

  element={<Protected roles={["admin"]}><PerformanceReviewAdmin /></Protected>}

/>

<Route

  path="/admin/document-management"

  element={<Protected roles={["admin"]}><DocumentManagementAdmin /></Protected>}

/>

<Route

  path="/admin/work-schedule"

  element={<Protected roles={["admin"]}><WorkScheduleAdmin /></Protected>}

/>

<Route

  path="/admin/reports"

  element={<Protected roles={["admin"]}><ReportAdmin /></Protected>}

/>

          {/\* ================= USER ================= \*/}

          <Route path="/user/dashboard" element={<Protected roles={["employee","manager"]}><UserDashboard /></Protected>} />

          <Route path="/user/profile-update" element={<Protected roles={["employee","manager"]}><ProfileUpdate /></Protected>} />

          <Route path="/user/leave-request" element={<Protected roles={["employee","manager"]}><LeaveRequest /></Protected>} />

          <Route path="/user/leave-history" element={<Protected roles={["employee","manager"]}><LeaveHistory /></Protected>} />

          <Route path="/user/attendance" element={<Protected roles={["employee","manager"]}><Attendance /></Protected>} />

          <Route path="/user/chat" element={<Protected roles={["employee","manager"]}><Chat /></Protected>} />

          <Route path="/user/report" element={<Protected roles={["employee","manager"]}><Report /></Protected>} />

          <Route path="/user/notifications" element={<Protected roles={["employee","manager"]}><NotificationsUser /></Protected>} />

          <Route path="/user/salary-management" element={<Protected roles={["employee","manager"]}><SalaryManagement /></Protected>} />

          <Route path="/user/performance-review" element={<Protected roles={["employee","manager"]}><PerformanceReviewUser /></Protected>} />

          <Route path="/user/document-management" element={<Protected roles={["employee","manager"]}><DocumentManagementUser /></Protected>} />

          <Route path="/user/work-schedule" element={<Protected roles={["employee","manager"]}><WorkScheduleUser /></Protected>} />

        </Routes>

      </AuthProvider>

    </BrowserRouter>

  );

}

export default App;

index.js

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

import "bootstrap/dist/css/bootstrap.min.css";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<App />);